Language-guided 4D Object Reconstruction From Videos

Xiao Hu

Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Doctor of Philosophy degree in Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Creating 4D assets from real-world data is a crucial problem in computer vision, with applications in Augmented Reality (AR), Virtual Reality (VR), and animation. However, this task is often cost-prohibitive, as it requires both detailed object 3D structure and accurate motion within a complex 3D environment. This thesis proposes a framework that converts object-centric monocular videos into 4D representations. The two primary challenges we aim to tackle are: 1) key object extraction from the video, and 2) dynamic object capture and representation.

For the first challenge, Referring Video Object Segmentation (RVOS) is employed, using language prompts to identify target objects while ignoring distracting backgrounds. The limitations of existing RVOS methods are first analyzed, particularly in terms of temporal consistency, and then enhanced by improving the understanding of temporal context. Additionally, we adapted previous frameworks into online methods capable of processing video frames in real-time, which significantly increases the usability.

For the second task, 3D Gaussian Splatting (3DGS), a popular method for 3D scene representation, serves as the baseline for dynamic object representation. One main limitation in 3DGS is identified, that large object motions cannot be well handled during reconstruction. To address this, a motion-deformation-decoupled dynamic 3DGS structure is designed to estimate the object's large overall motion and the local deformation separately, which can represent highly dynamic objects better. Additionally, monocular casual video has an inherent limitation on spatial-temporal observations. Only a limited part of the object can be observed at each point in time. This leads to undesired artifacts in weakly observed areas. To overcome this limitation, a novel framework is proposed to leverage the prior of the 2D generative model to import additional constraints over the weakly observed spatial-temporal areas.

In summary, this thesis takes a casually captured monocular casual video, along with a descriptive sentence pointing to the desired object as input and produces a complete 4D object with accurate 3D structure and corresponding motion, which advances the usability of 4D reconstruction in practical applications.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Representing our physical world in computing is a long-standing issue in the computer vision community. World representations can be used not only for realistic scene rendering in consumer applications like AR/VR [1], animation and gaming, but also in world understanding for other downstream tasks, including autonomous driving, embodied AI, and multi-modal world models. We are living in a 4D world that consists of 3D space and a temporal dimension. Thus, a 4D representation of the world is more complete than 3D and a valuable format of the computer vision field.

Visual data representations can be organized by the richness of information they preserves. At the simplest level, images can be summarized using hand-crafted features [2, 3], which capture key patterns or structures without storing full pixel data. Representations that retain pixel intensities from all channels provide denser descriptions, enabling algorithms to perform more sophisticated vision tasks. Beyond 2D, formats such as point clouds [4], meshes [5], and occupancy grids [6] encode the 3D structure of a scene directly, allowing geometric reasoning. Extending this further, incorporating the temporal dimension yields 4D data, which capture both spatial and motion information, enabling dynamic

scene understanding.

Capturing a scene in various data formats is another challenge. 2D images and 3D videos (2D images with a temporal dimension) can be easily captured by current consumer devices such as digital cameras and smartphones. However, 4D scene recording is usually complicated. Fig. 1.1 shows two examples of how current 4D datasets are recorded. The top row shows the data capture system used to acquire the CMU Panoptic Dataset [7], which can capture human centric 4D scenes. The bottom half shows the system used for the DiVa360 [8] dataset, which is smaller and can only record hand-level 4D objects. A traditional 4D capture system usually consists of multiple cameras with optionally other sensors as a sensor matrix to capture the 4D scene from various view points simultaneously. The background needs to be as simple as possible, so that it does not interfere with the foreground objects. Such a system is expensive as it contains a number of sensors, and complicated since it requires a lot of effort during system setup and calibration. Although current systems can provide good 4D capture, they are not practical for use in real world consumer applications.

This thesis aims at the question whether there is an easy way to obtain a 4D representation of the desired object. To solve this task, we designed a work flow to reconstruct the complete 4D object from a video casually recorded by a monocular camera.

We define the task as the 4D object reconstruction by monocular casual videos. Monocular casual video is a video that is recorded by a handheld camera moving around the object without a fixed moving trajectory or pattern, which is the most easily obtained video type in real-world applications. There are minimal requirements for recording devices and skills. As illustrated in Fig. 1.2, a monocular object centric casual video usually focuses only on the desired object, and the motion of the object is continuous. A long range motion can be usually split into multiple motion units. As shown in Fig. 1.2, the video contains three motion units which can be processed independently. However, there are several key challenges in this task, which are carefully analyzed and handled by the thesis.

First, the video recorded in the real-world usually contains more objects than desired. The irrelevant objects and background may affect the processing of the target object. Also, a single video may contain multiple target objects which need to be processed separately. To solve this challenge, we leverage language, a common type of human prompt, as a guide to segment the desired object described by the language clue. This is quite user-friendly since the user can speak out for which object the video is recorded for during the recording itself. This sub-task is defined as Referring Video Object Segmentation [10] (RVOS). We pointed out the weakness of previous methods that lack temporal connection, and then

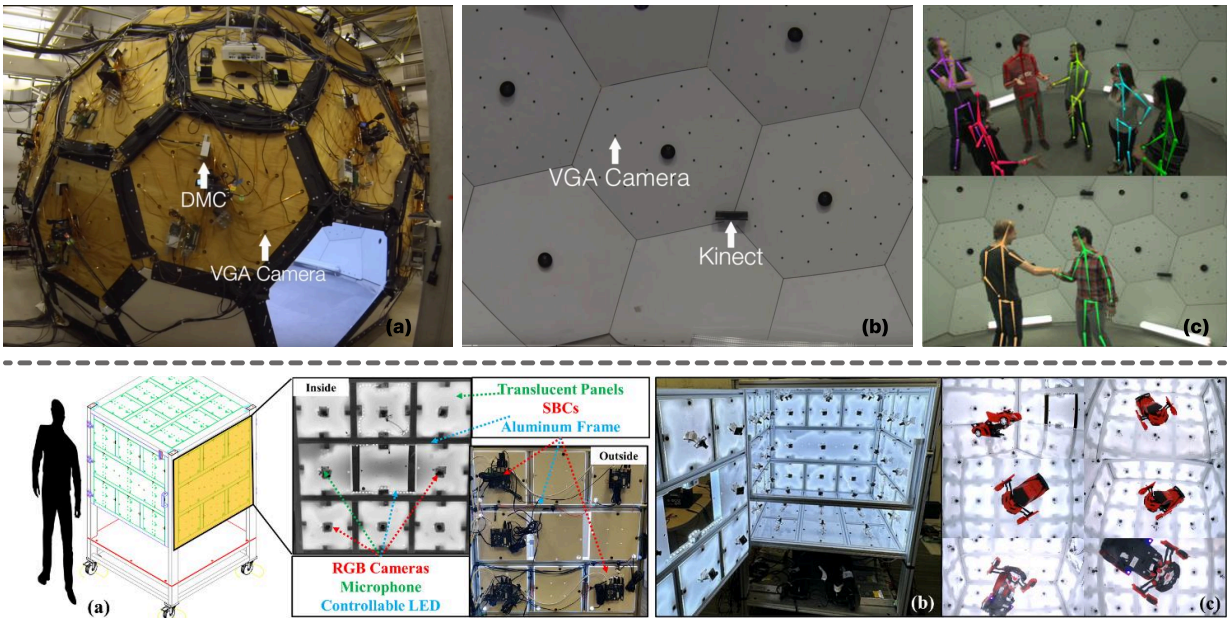


Figure 1.1: Some 4D recording systems. The upper half shows the recording system for CMU Panoptic Dataset [7], and the lower half shows the system for DiVa360 [8]. In each row, (a) shows the overall system setup, (b) shows the setup of the capture devices, and (c) shows samples of recording 4d scenes.

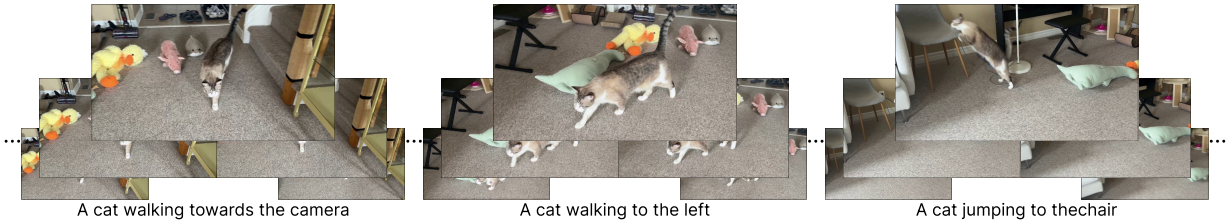


Figure 1.2: An example of an object centric video [9]. An object centric video can consist of a series of actions of the object. This particular example shows three actions involved in cat walking.

propose our RVOS framework to enhance such connection to boost the overall performance. Casual videos are usually in various length, the ability to extract the object from arbitrary length video is also a key challenge. We designed a filtering based framework to convert previous offline into semi-online methods to process the video with arbitrary length.

Second, converting a sequence of frames into a 4D representation is another sub-goal of this thesis. There are plenty of data structures which can be used to represent a 3D object and that can be potentially extended into a 4D representation. 3D Gaussian Splatting (3D-GS) is selected as the baseline structure for our work since it generates high-fidelity results with considerable rendering speed. After exploring current dynamic 3D-GS methods, we found that they can only handle an object with local deformations but not an object with large motion. To fix this, we proposed a motion deformation decoupled dynamic 3D-GS approach to estimate the object local deformation and large motion separately, which greatly improves the reconstruction result in large motion scenarios. We further investigate the problem of 4D object representation by using a single camera. A complete 4D object cannot be fully reconstructed by a single monocular video since it only provides 3D information (2D images with a temporal dimension). Observations are sampled along the camera path, leaving much of the space-time region unconstrained. To overcome this limitation, we leverage the prior knowledge of a diffusion-based generative model to first learn the appearance and attributes of the desired object and refine the visual representation from the out of distribution area.

1.2 Task Analysis & Current Challenges

In this section, the entire task workflow is first analyzed and decoupled. Then, the challenges and limitations for each step are identified as the motivation of our thesis.

The input of the entire pipeline consists of a full monocular video along with its corresponding narrations for the object and actions. The narration can be derived from the audio language or the subtitles of the video. The final output includes the 3D models for all relevant objects and their corresponding motions. As shown in Fig. 1.3, the parallelograms at both ends represent the input and output. The colored boxes denote different procedural steps, and the uncolored boxes represent the intermediate data between the steps.

The complete monocular video, along with its narration, is divided first into several pairs of video narration, each pair describing only a single action (Fig. 1.3 (a)). This task is

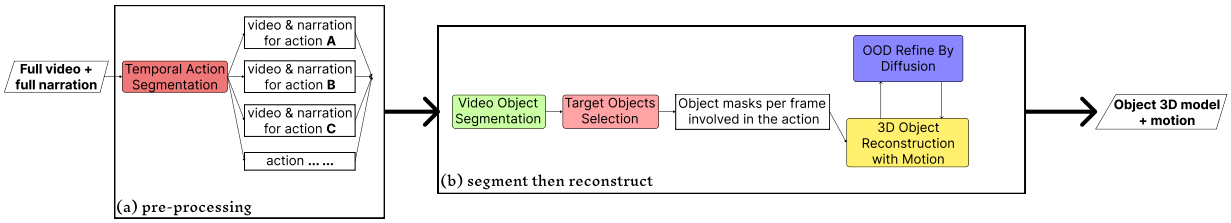


Figure 1.3: The overall workflow for language guided reconstructing a complete 4D object from a monocular video system is illustrated. The parallelograms represent the input/output. (a) shows the pre-processing to segment the entire video into different steps. (b) represents our designed pipeline.

described by video temporal action segmentation [11] (shown in red box in Fig. 1.3). Each video clip contains the objects mentioned in the narration but also unrelated background scenes. After the action segmentation of the video clip, the next step is to reconstruct the object for each video segment. As shown in Fig. 1.3 (b) The objects in each frame are first spatially segmented using video-based object segmentation methods to generate mask candidates (shown in the green box in Fig. 1.3). Because video frames often contain not only the target object but also irrelevant objects, a method is needed to select the appropriate object masks (shown in the red box in Fig. 1.3). In a prompt-based segmentation task [12] the desired object is masked based on a given prompt. These prompts can be reference points, a scribble, or a descriptive sentence. Language narration is a common format for interactive prompting and can be easily recorded together with the video. Language-based prompting is also named referring video object segmentation [13]. In the next step, 3D models of the segmented objects along with the corresponding motions for each action need to be reconstructed (shown in the yellow box in Fig. 1.3). A monocular video cannot fully constrain a complete 4D object reconstruction. A diffusion model is used to refine the 4D object representation for unobserved areas at a certain time (shown in blue box in Fig. 1.3).

Action segmentation, Step (a), has been extensively studied in recent research [11, 14]. Since each video usually only contains tens of elementary actions, it is also relatively straightforward for a human to manually split the full video by inserting the appropriate breaking points. Thus, the primary challenges arise from the remaining steps in (b), which are discussed in the following sections.

1.2.1 Referring Video Object Segmentation

Referring video object segmentation (RVOS) [13] aims at extracting the object mentioned in the narration from the video. This is a relatively new task partially enabled by powerful hardware with large memory that allows the video to be processed at the same time. A straightforward approach is extending previous Referring Image Object Segmentation (RIOS) methods into a video segmentation by adding the temporal dimension [15–18]. Alternatively, some approaches integrate linguistic features into the video object segmentation task [13, 19–21]. However, we have observed that existing methods often struggle with the understanding of the temporal information between frames, and process each frame independently during the decoder stage. This limitation can result in inconsistencies in object segmentation masks across frames. Additionally, current methods process entire video frames simultaneously, which restricts the video length to the memory capacity of the processing device, making it challenging to handle longer or continuous video sequences.

1.2.2 3D Reconstruction with Motion

3D reconstruction creates a 3D model of an object through a series of images taken around the object from different viewpoints. This task is also known as structure from motion (SFM) or 3D modeling [22]. Traditional methods for representing 3D models include point clouds, meshes, occupation grids, or polygons. Recently, neural rendering based approaches [23, 24] have been introduced which use a neural network to store the 3D structure. However, these methods typically struggle with dynamic objects, as object motion violates the static-scene assumption required for accurate reconstruction. Some follow-up research has aimed to address these weaknesses, but issues due to large object motion and occlusions continue to pose significant challenges.

For motion representation, a common approach is to estimate the 6-DoF of the rigid object motion [25]. Non-rigid object motion, however, involves both deformation and transformation, which are often coupled with each other and challenging to estimate independently. Some efforts have been made to estimate the motion of deformable objects. But, they require additional input information, such as the original object model [26], physics-based simulations [27], or extra sensors to scan per-frame point clouds [28].

As discussed in Sec. 1.1, using additional sensors is often costly and complex. For practicality and ease of deployment, this thesis relies solely on a monocular camera, which

is more challenging. A 3D scene can be fully constrained by a monocular video if the video covers enough viewpoints. However, a monocular video cannot fully constrain a 4D scene. A monocular camera can only provide one observation at each time step, which can be treated as a sparse sampling in the entire spatial-temporal space. Only a small portion of the object can be observed at each time, and the rest part would not be constrained during the reconstruction. This leads to an incomplete object model.

1.3 Thesis Statement

This thesis focuses on converting monocular videos into 4D scene representations. Four critical challenges are addressed. First, enhancing temporal understanding in the Referring Video Object Segmentation (RVOS) task to ensure consistent object tracking across frames. Second, deploying the RVOS framework for long and ongoing videos while maintaining temporal consistency and avoiding segmentation drift. Third, representing and storing dynamic 3D object models and motions for large dynamic range objects. Lastly, reconstructing complete object models from single monocular scanning videos.

To tackle the outlined tasks, we develop various methods and corresponding datasets to overcome each limitation in 3D scene representation from casual scanning monocular videos. Initially, we design a temporal context-enhanced structure for the RVOS framework, aimed at improving temporal feature aggregation in video understanding. Following this, we introduce a semi-online structure derived from previous offline methods to enhance temporal consistency for long and ongoing videos. For 3D object and motion modeling, we propose a motion-decoupled deformable 3D Gaussian Splatting (M5D-GS) approach that simultaneously estimates the object model, local deformation, and overall motion. To overcome the limitations of using monocular scanning video, which lacks full spatial-temporal constraints, we design a novel framework to utilize the strength of generative models to provide additional constraints on the unseen part of the object.

To summarize, the main contributions of this thesis are as follows:

- We design a temporal context-enhanced RVOS structure that incorporates a frame token fusion encoder and an instance query transformer decoder to enhance the temporal understanding in the RVOS task. Detailed testcase scenarios are classified to fully demonstrate the improvement achieved by the proposed method.

- We redesign the RVOS task as a stochastic optimization problem and employ a filtering approach to integrate previous offline RVOS models to achieve a semi-online solution. The designed method not only enhances accuracy, but also offers greater flexibility compared to existing state-of-the-art approaches. By simply adjusting the chunk size, the proposed method can seamlessly operate in online, semi-online and offline modes without requiring additional training. This adaptability enables the model to effectively balance the trade-off between accuracy, memory usage, and processing speed.

- We introduce a 3D-GS based framework that decouples the estimation of the object model, object local deformation, and object overall motion within the same optimization period. This approach enables accurate reconstruction of both the 3D object model and its motion. We further generate a new dataset with both synthetic and real-world scenes to show the improvement of the proposed method.

- We investigated the limitations associated with the lack of spatio-temporal constraints when a monocular camera is the only input to the 3D deformable object reconstruction task. We then designed a framework to reconstruct complete 4D objects from monocular casual videos by leveraging the strength of the diffusion model. A new benchmark is created to support the evaluation in this task.

This thesis focuses on methodological research contributions rather than full system engineering. Therefore, an end-to-end automated implementation of the pipeline was not developed. Instead, the designed pipeline was manually validated by executing each stage sequentially and verifying the outputs. Experiments confirm that the overall workflow performs as intended.

1.4 Proposed Solutions

The previous section (Sec. 1.2) outlined the limitations and challenges of current studies. This thesis aims to overcome the limitations and simplify the language-guided conversion of monocular videos into 4D scene representations.

This thesis covers two topics, and two problems are addressed for each topic. Fig. 1.4 shows the summarized structure of the covered problems. In this section, we define the key problems to be solved, corresponding to the above limitations. First, we will explore

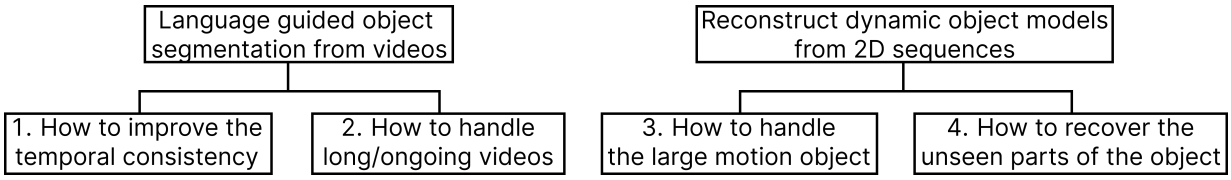


Figure 1.4: The topics and problems targeted by this thesis. Two topics are covered (top row), and two problems of each topic are researched (bottom row).

the temporal consistency of Referring Video Object Segmentation (RVOS) system. Next, we will analyze RVOS for long or ongoing videos. Then, we will address 3D reconstruction methods that incorporate motion. Finally, 3D reconstruction from sparse viewpoints will be discussed to explore the limitations of current monocular video 3D reconstruction.

1.4.1 Temporal Consistent RVOS

Compared with Referring Image Object Segmentation (RIOS), RVOS also considers the temporal information present in both the video clip and the narration. This allows the referring expression to describe not only the object's attributes, but also its behavior over time. Moreover, the RVOS task also requires the association of data between different frames to track the referred to object.

Although RVOS is a relatively new task, RIOS has been studied extensively due to its lower computational resource requirements. A common strategy is to extend RIOS into RVOS by adding a temporal dimension. This extension leads to two different kind of approaches: online and offline approaches. Online approaches [10, 29] process each frame in the video independently and pass meta features for temporal propagation. In contrast, offline approaches [13, 21] enhance image-based encoders, such as ResNet [30], with video-based encoders, e.g. the Video Swin Transformer [31]. However, we found that previous methods primarily focus on either text and visual information aggregation or image-based feature learning, but pay less attention to inter-frame temporal understanding. This leads to temporal inconsistency in the final segmented results.

To address this issue, we propose a temporal context-enhanced RVOS method [32] to improve the inter-frame temporal consistency. In general, if the narration adequately describes the object in each frame and the object is well-observed, applying RIOS to each frame independently can already generate a satisfactory result. However, RVOS offers

the advantage of leveraging the additional temporal dimension, allowing frames with poor object visibility to benefit from better-observed frames. Thus, we analyze how temporal connections influence prediction accuracy in various challenging scenarios, including but not limited to occlusion, motion, and crowded scenes. This work was a collaboration with Basavaraj Hampiholi, Prof. Heiko Neumann and my supervisor Prof. Jochen Lang. I was responsible for the core research contributions, experimental design, and implementation. The coauthors provided valuable feedback and assisted with reviewing and proofreading the manuscript.

1.4.2 RVOS for Long & Ongoing Video

As mentioned above, there are two different approaches of RVOS, online and offline approaches. State-of-the-art methods [32–34] are typically engineered as offline methods, as they can access the global temporal context across all frames. These approaches dominate current benchmarks [10, 35, 36], which only consist of short videos (roughly 100 frames per video sample). However, processing all the frames simultaneously requires significant GPU memory, and this requirement increases as the video length grows. When the video length exceeds the available GPU capacity, offline methods are forced to divide the video into smaller chunks. Based on our observation, this "cut short" strategy introduces segmentation inconsistencies between different video chunks. On the other hand, online methods [10, 29] process each frame independently, and propagate meta features between frames. These methods are not constrained by video length, making them more scalable. However, they tend to suffer from feature drift as the video becomes longer, where the propagated features gradually deviate from the true object features. Addressing these trade-offs is essential for achieving both scalability and accuracy in RVOS.

To address this limitation, we propose a semi-online approach [37] that processes a chunk of video with multiple frames at each time, while optimizing the propagated intermediate feature using a filtering framework. We redefine RVOS as a stochastic optimization problem, and apply a predict-then-update mechanism to propagate intermediate features. This approach not only enables handling long/ongoing videos with high performance, but also dynamically balances accuracy, processing time, and memory with only one training session. This work was undertaken jointly with Prof. Heiko Neumann and my supervisor Prof. Jochen Lang. I was responsible for most of the research tasks, including problem formulation, methodology, and experimental evaluation, while the coauthors provided feedback and manuscript support.

1.4.3 Model Reconstruction for Dynamic Object

3D reconstruction of dynamic objects using a monocular camera remains a significant challenge. A monocular camera can only capture one visible aspect of the object at any given time, making it impossible to directly observe or reconstruct deformations on the occluded or back-facing parts. Traditional methods using point clouds, meshes, or occupancy grids, struggle with deformable objects reconstruction. Recent neural rendering techniques have shown promise in modeling deformations, they struggle with handling large-scale object motions, such as translations and rotations, which are common in real-world recorded videos. Thus, our goal is to design a method capable of simultaneously estimating and reconstructing the object 3D model, local deformation, and overall transformation.

We select a neural rendering based framework, 3D Gaussian Splatting (3D-GS), as the baseline to address this challenge. We redesign the 3D-GS rendering pipeline, introducing motion handling as part of the reconstruction process. To demonstrate both the limitations of existing approaches and the advantages of our method, we create a new benchmark dataset consisting of synthetic and real-world scenes that feature objects with large motions. This benchmark provides a comprehensive testbed for evaluating performance in different scenarios. This work [9] was conducted in collaboration with Dr. Libo Long and my supervisor Prof. Jochen Lang. I was responsible for most of the research and implementation, while the coauthors contributed through discussions and proofreading.

1.4.4 Model Reconstruction from Sparse Views

3D model reconstruction from a visual camera usually relays on the multi-view geometry. A complete object model requires the object to be observed from many different views. Using only sparse views to constrain the 3D reconstruction would lead to the model overfitting to the provided views, and performing poorly on the unconstrained Out-Of-Distribution (OOD) area. This problem is more severe in 4D reconstruction because the additional temporal dimension makes the training views even sparser. Only a limited spatial-temporal portion of the object is observed at each time, and the motion of other parts of the object are unknown.

However, different parts of the object will not move independently. An object usually has a certain motion pattern. Understanding the motion pattern and the object attribute can help us to reconstruct a more realistic object model. According to this, we designed a pipeline to leverage the strength of a diffusion-based generative model as prior to refine

the 4D representation in the OOD area. The diffusion model is first finetuned by a limited number of training views to learn the appearance of the object, and then used to generate images of the object from the unavailable viewpoints at a given time. This work was guided by my supervisor Prof. Jochen Lang, who contributed through discussions and proofreading.

1.5 Assumptions and Problem Scope

This thesis addresses the problem of reconstructing complete 4D dynamic object models from monocular scanning videos with language guidance. Although the proposed framework aims to be generally applicable to practical capture settings, real-world scenarios involve many unpredictable factors that cannot be fully modeled or considered in a single study. Consequently, a set of assumptions is adopted to clearly define the scope of the problem and to ensure that the research objectives remain tractable and well-defined.

Training dataset. The framework assumes that the target object is reasonably well represented in the training data of the segmentation and diffusion models. For objects with highly unusual appearance or unseen categories, the quality of RVOS segmentation and generative priors may decrease. In contrast, the 3DGS-based reconstruction itself does not rely on semantic knowledge of the object and therefore remains largely unaffected by this assumption.

Video Quality. The framework assumes that the input video provides reasonably clear and informative observations of the target object. Significant motion blur or poor image quality can degrade both segmentation and reconstruction performance.

Object Coverage. Our temporal enhanced pipeline is able to handle visual clutter and temporal occlusion in the video. However, the object referred to by the language prompt should be one of the major subjects over the complete recording and visible in most frames; otherwise, the video may lack sufficient useful information. In addition, the camera trajectory should cover most of the object's surface over time. Regions that remain entirely unobserved can only be inferred through generative priors, which may result in unrealistic reconstructions.

The assumptions described above are introduced to clearly define the scope of this thesis and to make the problem tractable within the current framework. Nevertheless, many of these limitations are not fundamental and can be alleviated through future research

and complementary techniques. For example, improved video acquisition methods, such as deblurring or super-resolution preprocessing, could enhance robustness to low-quality inputs. These directions provide promising opportunities to extend the proposed framework beyond the current assumptions and toward more fully unconstrained real-world applications.

1.6 Thesis Overall Structure

In this chapter, we have provided a comprehensive introduction to the thesis. The pipeline for language-guided conversion of a monocular video into a 4D scene representation is analyzed, and the problems in each step are diagnosed. In Chapter. 2, we discuss related research that lays the foundation for our work. In Chapter. 3, we address the temporal consistency challenges within the RVOS task. In Chapter. 4, we then enhance the existing RVOS methods to effectively process long and ongoing videos using a filtering framework. To accurately capture dynamic 3D objects experiencing large motions, we introduce a motion deformation decoupled 3D-GS in Chapter. 5. In Chapter. 6, we design a novel pipeline to leverage a generative model to ease the lack of constraints in the monocular camera dynamic object representation task. Finally, in Chapter. 7, we provide the conclusion of the thesis which highlights the contributions and discuss limitations. A future research plan to guide subsequent investigations in this area is also included.

2

Background & Related Work

In this chapter, we focus on two key processing steps in the designed pipeline. Firstly, Sec. 2.1 provides an overview of recent studies related to the referring video object segmentation (RVOS) tasks with different setups. Secondly, Sec. 2.2 provides an overview of 3D reconstruction research, covering both traditional approaches and neural network-based methods to offer a comprehensive background review.

2.1 Referring Video Object Segmentation

Referring video object segmentation (RVOS) aims to segment the object referred by the given expression from all video frames. This task is relatively new and was first defined in 2018 by Gavrilyuk *et al.* [35], possibly, because the development of computing resources now allows videos with multiple image frames to be processed together. The RVOS task requires not only processing multiple frames, but also multi-modal aggregation of text and image features, which further increases the computational pressure. However, there are two related tasks that have been researched much longer: Video Object Segmentation (VOS) [38, 39], and Referring Image Object Segmentation (RIOS) [40]. These two tasks

correspond to video understanding and multi-modal understanding, which are the two key components of the RVOS task.

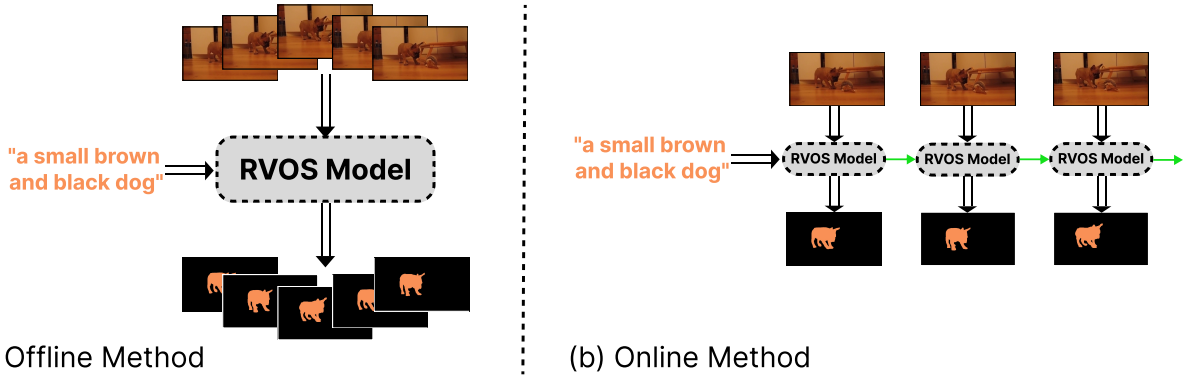


Figure 2.1: The structural comparison between different RVOS structures. The double arrow represents the input/output, the green arrows represent the inter-frame meta value propagation.

Early RVOS methods were usually designed based on one of RIOS and VOS. This leads to two research directions. One type of approaches [10, 29] extend the image-based RIOS with an additional temporal dimension and are usually designed as an online structure. Other approaches [13, 21] integrate language understanding structures into the VOS framework as additional cues guiding the video segmentation, which usually leads to an offline structure. The online methods process each video frame independently with inter-frame feature propagation, while offline methods process the entire video at the same time. Fig. 2.1 shows the structural comparison and the differences between these two types of methods. Offline methods [32–34, 41, 42] usually achieve higher accuracy than online methods, since the inference of each frame can benefit from both temporal directions, while online methods only have forward temporal feature propagation. However, offline methods cannot process long/ongoing videos due to GPU memory limits. Online methods, which process each frame separately, have constant memory usage regardless of video length. However, the feature drift problem becomes more severe for longer videos. In the following sections, RVOS methods with different frameworks will be discussed separately. The widely used datasets will also be reviewed. With the development of Vision Large Language Model (VLLM), some recent works [43–45] use VLLM or modified Large Language Model (LLM) as the foundation model for the RVOS task. Related works and widely used benchmarks are discussed in the following sections.

2.1.1 RVOS Approaches

The method by Gavrilyuk *et al.* [35] can handle not only a fixed set of vocabulary, but also natural language sentences thanks to the inclusion of a language model. The expression is in an actor-and-action format, describing both the attribute (spatial aspect) and the action (temporal aspect) of the object. They employ World2Vec [46] as a text encoder, and I3D [47] as a visual encoder. The text features are used to generate dynamic filters used in the CNN processing of the visual feature maps. This CNN with dynamic filters allows the features from different modalities to be aggregated to generate the final segmentation result. They also contribute a widely used RVOS dataset, the A2D dataset, which will be introduced in the following sections.

Early on, Khoreva *et al.* [48] proposed to extend RIOS into RVOS. The method first uses Mask R-CNN [49] to generate the proposed object bounding boxes in each frame. Then, DBNet [50] and MattNet [51] are used to select the bounding boxes that match the expression with high confidence. Temporal consistency across different frames is used to re-rank the selected bounding boxes from each frame based on temporal motion cues, and only top-ranked bounding box is kept for each frame. An object segmentation approach is applied to each bounding box to obtain the final segmentation mask. However, this multi-stage structure requires different pre-training for each sub-block, which increases the difficulty of using it.

RefVOS [52] argues that the referring expressions in current benchmarks are too trivial, which causes the networks trained on the benchmarks to not generalize well to new videos. The research extended previous benchmarks with more non-trivial expressions and designed a new framework to improve performance. BERT [53] is used as the textual encoder. The proposed method achieved comparable performance on RIOS, and state-of-the-art on RVOS.

The structure of the Transformer [54] provides a unified feature space for different modalities, which makes the aggregation of multi-modal features more efficient. Thus, later works mostly use the attention structure in Transformers for feature aggregation.

Urvos [10] is a framework for online processing where each frame is processed separately with guidance from memory of the previous frames. ResNet [30] is employed as the basic structure. The memory features from previous frames and the textual features from the expression are aggregated with the target frame features at different steps. Another main contribution of this work is the first large-scale referring video object segmentation dataset, named Refer-YouTube-VOS.

PMINet [55] was designed for the 2021 Refer-YouTube-VOS challenge, and achieved second place. The method proposes a cross-modal excitation modulation to conduct interactions between different modalities. Liang *et al.*[56] achieved the best accuracy in the challenge. The proposed method first generates a number of mask candidates in each frame, and then uses a video mask propagation model to get the object tracklets. Textual features are used to select the most relevant tracklet to generate the final output.

Multi-step training methods are not only complicated to train, but also not straightforward for optimization. Recent works design methods for end-to-end training. MTTR [21] extends a video object segmentation framework, VisTR [20], with additional textual feature aggregation. The text query and the video frames are processed by different backbones to extract features, and fused by a multi-modal encoder structure. The rest of the model is similar to VisTR and the image based segmentation model DETR [57]. ReferFormer [13] shares a similar idea to MTTR. The main difference is that the object queries used for the attention decoder are generated from the textual feature instead of being randomly initialized. After multiple layers of attention operations, the object query carries the information for a certain object in each frame. Thus, the object query is used to generate a dynamic kernel for dynamic convolution. Such an object query is also used for object class and bounding box prediction. The Hungarian algorithm is utilized to optimize the object matching across different frames. MTTR and ReferFormer not only unify the RVOS structure into an end-to-end training approach, but also significantly improve accuracy on all benchmarks. Thus, these two methods are widely selected as the baseline methods in later works. However, some limitations still remain in their frameworks. The temporal connection is too loose, only the video-based feature extractor and the final matching stage contain the inter-frame connection. Also, the feature processing for different modalities is not balanced, and the importance of the textual feature is not well considered.

TempCD [41] built a cross frames query connection method to select the queries from different frames that are guided by the textual features. The method first generates a global referent token to summarize the features from all frames, and then distributes it to each frame again to get a specified feature for each frame. The proposed method improves the temporal connection in the decoder, but summarizing all frames into one token may decrease the accuracy when the video gets longer.

OnlineRefer [29] extends ReferFormer into an online approach with feature propagation between frames. The object query generated from the previous frame is used to initialize the query in the next iteration. The position of the bounding box is also passed to the next iteration to provide more spatial cues.

VLT [58] argues that using one query for the entire sentence may not be appropriate. Different words in a sentence may have different levels of importance in different videos. Thus, the method generates various textual queries for a single sentence by focusing on different words. The masks generated from different textual features are then selected to provide the final output.

R^2VOS [59] stipulates that current datasets contain only positive text-visual pairs, which may lead to a bias during training. Thus, they extend the original Refer-YouTube-VOS dataset with negative pairs and achieve better results. The negative samples can benefit the contrastive constraints in their method during training.

With the development of multimodal extensions of large language models, research utilizes the strong multi-modal feature perception ability of the large model to handle the RVOS task. VideoLISA [43] utilizes a powerful multi-modal vision language large model (VLLM), LLaVA [60] with 3.8B parameters, as the backbone to make use of the strength of the large language model. VISA [44] employs large vision-language model, Chat-UniV-7B/13B [61], as a foundation model to jointly handle different tasks. SAMWISE [45] leverages the strength of large visual backbones, like SAM2 [62], to form a powerful RVOS model.

At the time of developing our first work (TCE-RVOS), existing methods paid limited attention to temporal consistency across video frames. Most approaches relied on video-based backbones, but the critical encoder and decoder components still processed each frame independently. This design led to inconsistencies in temporal features and reduced segmentation stability across frames. To address this weakness, TCE-RVOS was proposed to explicitly enhance temporal connections within the RVOS framework.

In addition, prior RVOS methods were primarily designed for offline settings, where all video frames are processed together to maximize accuracy. While effective for short video clips, this offline structure is impractical for long or streaming videos, as it cannot scale efficiently and limits real-world usability. Motivated by this limitation, we developed FF-RVOS, a flexible framework that supports online, semi-online, and offline inference within a single training session. This design improves adaptability and enables deployment on resource-constrained devices.

2.1.2 Dataset

In this section, the widely used RVOS benchmarks are introduced and compared. The benchmarks are listed in the order of their published date.

RefCOCO/+/g [63] is a referring image dataset that only contains image-expression pairs. Although it is not originally designed for the RVOS task, it is widely used for the pre-training of RVOS approaches. Thus, we also include this dataset here. RefCOCO contains over 140,000 referring expressions for 50,000 objects across 20,000 images sourced from the MSCOCO [64] dataset. The referring expressions vary in complexity, some expressions based on spatial relationships, attributes, or activities to the referred object. RefCOCO contains three subsets, RefCOCO, RefCOCO+, and RefCOCOg. RefCOCO focuses on relatively simple referring expressions, where the object being referred to is usually identified by its appearance or location. RefCOCO+ excludes references to object location, encouraging models to focus more on visual attributes such as color, shape, and size to locate the target object. RefCOCOg is a larger and more complex variant of the dataset, where the referring expressions tend to be longer and more descriptive, often requiring an understanding of the entire scene.

A2D Dataset [35] is the first benchmark designed for the RVOS task. The dataset focuses on the interaction between actors (the subjects) and actions (i.e., the tasks being performed). A2D dataset contains annotations for 8 different actors (e.g., adult, child, cat, dog, car, bird) and 8 types of actions (e.g., climbing, crawling, walking, flying, running). 3,782 videos collected from YouTube that span a wide variety of real-world scenarios.

Ref-DAVIS17 [65] is an extension of the DAVIS 2017 dataset [66], designed specifically for the task of referring video object segmentation (RVOS). It combines the rich video segmentation annotations of DAVIS17 with natural language referring expressions, making it a key dataset for research at the intersection of computer vision and natural language processing. However, Ref-DAVIS17 is a small dataset that only contains 90 video sequences.

JHMDB-Sentences [48] is an extended version of the JHMDB [67], specifically designed for RVOS and human action understanding. It incorporates natural language descriptions (sentences) for the actions depicted in the JHMDB videos, linking visual actions with corresponding linguistic expressions. Compared with the A2D dataset, this benchmark contains more types of actions (21 action categories), but with fewer video samples (928 videos).

Refer-YouTube-VOS [10] is one of the most widely used benchmark at this time, since it contains 3,978 high-resolution videos with 9,238 unique objects and corresponding

referring expressions. This dataset extends the YouTube-VOS dataset, which is a widely used benchmark for video object segmentation, by adding referring expressions to describe target objects in video sequences. Unlike previous RVOS datasets with shorter clips, Refer-YouTube-VOS includes longer videos, increasing the difficulty of maintaining segmentation over extended sequences.

MeViS [36] is a recent RVOS benchmark with longer video length and more complicated scenes. The MeViS dataset emphasizes memory-augmented techniques for RVOS. This means that models are encouraged to use temporal memory to track and segment the referred object across frames, especially in scenarios where objects undergo occlusions, changes in appearance, or reappear after being hidden. The dataset shares the same data structure with Refer-YouTube-VOS, but with more challenging testcases. However, this dataset is relatively new and requires more computing resources to process longer videos, thus the performance of current approaches is relatively weak. In other words, there is still a gap between current research and real-world scenarios.

2.2 3D/4D Object Representation

3D object representation [68] involves capturing the shape and appearance of physical objects or environments and recreating them as three-dimensional digital models. The dynamic 3D object (4D) representation also considers the motion and deformation of the object. In order to capture the 3D object representation from the real world, various sensors are used, including LiDAR[69], structured light scanners [70, 71], stereo cameras [72], and monocular cameras (photogrammetry) [73]. Given that our task uses one monocular camera only, the following discussion will focus on methods employing monocular cameras.

Traditional 3D representation and reconstruction tasks usually follow the frameworks of Simultaneous Localization and Mapping (SLAM) [74] and Structure from Motion (SfM) [68]. In these methods, a 3D scene is reconstructed from multiple frames based on epipolar geometry. More recently, deep learning–based methods have been introduced to optimize 3D representations using neural networks or to directly generate 3D structures from images in an end-to-end manner. The following introduction of related methods is divided into traditional approaches (Sec. 2.2.1), deep learning approaches(Sec. 2.2.2), and the widely used benchmarks(Sec. 2.2.3).

2.2.1 Traditional Approaches

Traditional SFM is an optimization problem. The objective is to optimize the positions of landmarks (points, meshes, voxels, etc) across different viewpoints. In an ideal scenario, solving this optimization is straightforward, since only a few pairs of points are needed to calibrate the camera pose [75], which further supports the 3D reconstruction. However, real-world applications introduce complexity. Factors like image blur, low-quality feature extraction, and incorrect feature matching, which introduce errors into the system. To enhance robustness, Fisher *et al.*[76] introduced Random Sample Consensus (RANSAC) to filter out outliers during optimization. This strategy published in 1981 became fundamental for robust estimation in noisy environments. Later on, RANSAC was first utilized in SFM by Hartley *et al.* [68] in 2000.

Even after obvious outlier removal, remaining data still contain small errors that impact the final accuracy. To address this, Bundle Adjustment (BA) [77] was published and later formalized for computer vision by Triggs [78] to optimize all the variables together, which became the most famous optimization algorithm used in SFM and SLAM tasks. Filtering based optimization [79, 80] is another popular algorithm used in the SLAM task. Compared with global optimization, filtering based method only optimize the current frame with one (or a sliding window of) previous frame(s). This allows the application to use less memory, but it usually comes at the cost of reduced performance, especially when the image sequences are longer. Traditional optimization relies on constraints such as epipolar geometry and triangulation.

Optimization algorithms function as the backend of a SFM workflow. The frontend typically involves feature extraction, feature matching, and 3D representation. A popular format of 3D representation is point-based [68]. Point-based methods contain various advantages. First, the 3D attributes of a point cloud only contains the xyz coordinate, which is very memory efficient. Second, the image frames are already rasterized into pixels, which does not need extra processing for 2D point generation. Third, the sparsity of the point cloud allows them to be optimized in the early stage with low computational resources.

ORB-SLAM [81] (Oriented FAST and Rotated BRIEF-SLAM) is one of the popular SLAM baseline systems. It is designed for real-time operation in various environments, including monocular, stereo, and RGB-D (stereo and RGB-D are implemented in later versions) settings. ORB-SLAM is particularly known for utilizing ORB features, which offers a good balance of accuracy and computational efficiency. These features are fast

to extract, and invariant to rotation and scale change, which makes them well-suited for real-time feature extraction and matching. ORB-SLAM utilizes Bundle Adjustment for backend optimization. ORB-SLAM also supports loop closure, which significantly improves the accuracy in large scenes. In the second version, ORB-SLAM2 [82] introduced multi-sensor fusion, integrating IMU, stereo, and RGB-D cameras into the pipeline. In the latest version, ORB-SLAM3 [83] adds support for multi-map operations, dividing the scene into sub-maps where only the active sub-map is optimized. This setup increases the processing speed and reduces memory cost in large environment.

COLMAP [22] is a widely used framework for the SFM task. Because COLMAP is compiled for different operating system with a well designed interface, it is widely used even for applications beyond SFM tasks. COLMAP utilizes SIFT [84] feature for feature extraction and matching. While SIFT is slower than ORB in terms of computation, it is significantly more robust, which makes it ideal for SFM, where accuracy is prioritized over real-time performance. For back-end optimization, COLMAP also employs Bundle Adjustment. Additionally, COLMAP supports the conversion of sparse point cloud representations into dense point clouds and textured meshes to improve 3D geometry reconstruction. When using stereo cameras, it can also capture the absolute scale of the scene.

While feature point-based approaches dominate the field, some methods leverage different geometries, which are also very insightful. LSD-SLAM [85] is a direct visual odometry method, which directly calibrates two images without feature extraction. The framework utilizes the photometric error to constrain the optimization. Compared with feature point-based approaches, this framework can generate a denser scene, since all the points are reconstructed instead of only the extracted feature points. However, the photometric error is sensitive to changes in light conditions and large camera motion, which makes LSD-SLAM less robust.

StructSLAM [86] is an indoor SLAM system that uses building structure lines as features and landmarks. Compared with points, a line segment contains more attributes (two points with a connection), which increases the capability of the entire system. However, the drawbacks are also apparent. The additional attributes require more computational resources. Unlike points, which can either be visible or not visible in an image, a line segment can also be partially visible, which increases the difficulty of feature matching. Therefore, this work only uses structure lines, which are the line segments aligned with the xyz axis. Although the proposed method outperforms all competitors on their proposed dataset, it is still of limited use in general scenes. PL-SLAM [87] combines point features with the line features into one framework for optimization. By adding extra line constraints, PL-SLAM

improves the performance over the state-of-the-art methods at that time.

KinectFusion [88] uses a moving low-cost depth camera to reconstruct indoor scenes. The 3D structure is represented by a voxel-based truncated signed distance function (TSDF), which fuses multiple depth images into a continuous surface. This method is highly influential in modern dense volumetric SfM.

To incorporate object motion into consideration, SFM under dynamic scenes has also been widely studied. Most research in this field involves deep learning blocks for motion estimation or semantic segmentation. However, there are some earlier approaches that do not rely on deep learning. Tomasi *et al.* [89] introduced the factorization method for recovering 3D structure and camera motion from a sequence of images, laying the foundation for later dynamic SfM methods by addressing multi-view reconstruction. Dai *et al.* [90] published work that addressed the problem of non-rigid SfM by introducing a method to handle deformable objects. This method utilizes motion factorization to enable SfM in dynamic scenes where objects undergo non-rigid transformations.

Torresani *et al.* [91] presented a solution for flow-based tracking and 3D reconstruction of deformable objects in monocular sequences. The method utilized rank constraints to model the deformation of objects over time with the assumption that even non-rigid motions often exhibit low-rank structure. By combining factorization methods with these rank constraints, the authors develop an approach to simultaneously track non-rigid objects and estimate their 3D shape from 2D image sequences.

Fitzgibbon *et al.* [92] proposed a technique that simultaneously estimates the motion of the camera and each moving object, which uses multibody factorization to separate the trajectories of different objects. This method groups image points corresponding to the same object by analyzing their motion patterns, enabling accurate reconstruction of both the camera motion and the 3D structure of each object. RANSAC is used iteratively to classify the motion patterns of the feature points.

Christoph *et al.* [93] provided a 3D scene flow estimation that aims to jointly recover dense geometry and 3D motion from stereoscopic image sequences. The method generalizes classical disparity and 2D optical flow estimation. Unlike traditional scene flow methods that assume smooth motion across the scene, this approach segments the scene into multiple rigid components, allowing for more accurate estimation of motion in environments with independently moving objects. The algorithm jointly estimates both 3D structure and motion, ensuring consistency between the two, and uses an energy minimization framework to optimize the segmentation and motion estimation.

In addition to geometry-driven approaches like SLAM and SfM, a significant body of work explores image-based rendering (IBR) and light field techniques, which generate novel views directly from images without explicitly reconstructing full 3D geometry.

Early IBR methods include view interpolation [94] and later extensions in image warping [95]. These approaches exploit pixel correspondences and image warps to synthesize in-between views. While effective for certain settings, they depend heavily on dense correspondences and are sensitive to occlusions or viewpoint changes.

Light field methods [96], also known as plenoptic rendering, reinterpret captured images as slices of a 4D radiance function. This representation enables view synthesis from arbitrary viewpoints by resampling the light field directly. However, classical light field methods require very dense sampling of viewpoints, which limits their practicality in sparse capture scenarios.

To reduce sampling demands, geometry-assisted rendering methods were introduced. For example, view-dependent texture mapping [97] projects image textures onto geometric proxies (meshes or coarse geometry) and renders novel views using view-dependent appearance. While more efficient than pure IBR, these methods are sensitive to geometric inaccuracies and require precise proxies.

Hybrid and layered representations such as layered depth images [98], multi-plane images (MPI) [99], and billboard impostors [100] offer compromises between geometric accuracy and rendering flexibility. These techniques help manage occlusion and view interpolation efficiency, though often at the cost of artifacts or limited viewpoint coverage.

Later on, researchers began to use deep learning approaches to replace the processing steps in the entire pipeline. Recently, several deep learning based methods were published to reconstruct the 3D scene in an end-to-end manner.

2.2.2 Deep Learning Approaches

With the development of deep learning techniques, most computer vision tasks now utilize deep learning approaches. As mentioned earlier, the SLAM/SFM framework consists of various components. One common strategy is to use deep networks to replace one or more steps in the entire pipeline. Another strategy involves using deep networks to generate extra modalities as input to provide more constraints.

CNN-SLAM [101] utilizes a CNN network to predict the depth and semantic segmentation of the input image. Then it uses the predicted information along with the original image for subsequent processing.

When a deep network processes an image, the first step is usually encoding the image from *RGB* color space into a high dimensionial embedding space. This step naturally extracts features for each pixel. So, some research utilizes these deep features to replace previously hand-coded features in the framework. LIFT-SLAM [102] proposes a Learned Invariant Feature Transform (LIFT) descriptor to represent the feature points extracted from the image. The overall pipeline still follows ORB-SLAM with monocular images as input.

One drawback of traditional approaches is that the reconstructed 3D geometry lacks global information. Each point, line segment, or voxel only contains the feature for itself. By using deep neural networks, semantic feature can be predicted, which enhances the classification of 3D geometry. This benefits motion estimation, since the 3D geometries from the same object usually follow similar motion patterns.

DynaSLAM [103] first utilizes Mask R-CNN [49] to segment the image frames, then masks out all the dynamic objects in the scene and only uses the static background for reconstruction. This approach reduces the optimization error by removing non-static feature points. DynaSLAM II [104] utilizes an RGB-D camera as input, allowing the method to also estimate the motion of moving objects in the scene. However, this work was tested on a street view dataset that included only moving cars.

DOE-SLAM [105] first uses semantic segmentation to segment all feature points in the frame. Then, it clusters the objects and the background based on the estimated motions. The background static features are utilized to estimate the camera pose. When there are enough background points to optimize the camera pose, the motion of all moving objects is estimated. When most of the image frame is covered by moving objects, the system can recover the camera pose from the moving features and the predicted object motions. However, this work does not consider non-rigid motion.

Thanks to the increased computing resource and ability, several end-to-end approaches are published which dominate the SFM task. DUSt3R [106] takes a pair of unconstrained images as input and output a pointmap for each image expressed in the same coordinate frame of the first image. However, the proposed deep neural network can only process two frames at a time. Multiple images still need to be handled by traditional optimization strategy. MUSt3R [107] employs an additional latent memory block to represent the features of

the global scene, which allows the system to process multiple frames. MV-DUSt3R [108] extends the original DUSt3R to support multi-frames processing. VGGT [109] uses a strong pretrained visual encoder, DINO [110], to extract the features, global attention layers to aggregate features from all the frames, and finally output the camera matrix, depth maps, and point clouds together. Its end-to-end multi-task learning method which is trained on large scaled datasets allows VGGT to achieve SOTA performance.

Although the above research embedded deep networks into the workflow, the 3D scene is still represented by discrete geometries. Some continuous 3D scene representations have been proposed that fit deep learning well and achieve better performance. A Signed Distance Field [111] (SDF) is a scalar field where each point in space holds the signed distance to the closest surface or object boundary. The sign of the distance indicates whether the point is inside or outside the surface. Zhang *et al.* [112] introduced a deep learning approach to reconstruct 3D surfaces from multi-view images by learning a SDF. Instead of relying on explicit geometric representations such as point clouds or meshes, the method leverages a neural network to model the SDF directly from image inputs, enabling a continuous and compact 3D surface representation.

Neural Radiance Fields [23] (NeRF) and related volume rendering methods have fundamentally redefined IBR by using neural networks to encode the 5D light field and density implicitly, overcoming the dependency on explicit geometry and dense sampling that plagued classical IBR and light field approaches. The method models a scene as a continuous 3D function that outputs the color and density of points in space, allowing for realistic and detailed rendering of complex 3D structures from sparse 2D images. The most significant contribution of NeRF is the differentiable rendering function. To compute the color C of a pixel in NeRF, the following integral is used:

$$C = \int_{t_{\text{near}}}^{t_{\text{far}}} T(t) \cdot \sigma(t) \cdot \mathbf{c}(t)\, dt \tag{2.1}$$

where:

- $T(t)$ is the accumulated transmittance from t_{near} to t,
- $\sigma(t)$ is the volume density at point t,
- $\mathbf{c}(t)$ is the RGB color at point t.

The accumulated transmittance $T(t)$ is given by:

$$T(t) = \exp\left(-\int_{t_{\text{near}}}^{t} \sigma(t')\, dt'\right) \tag{2.2}$$

The above equations match each 2D pixel to a 3D ray with a differentiable combination, which allows the deep network to optimize the entire pipeline in an end-to-end manner. The 3D representing quality of NeRF is significantly higher than that of previous approaches, since the scene is represented densely by a continues function. When provided with enough training images, the method can represent the scene at arbitrary scales. In contrast, the rendering of such a dense representation is very time consuming. Additionally, the original NeRF is not suitable for dynamic scenes, as object motion affects the calculation of Eq. (2.1).

Several follow up works focus on upgrading the original NeRF to handle dynamic scenes. A common strategy is maintaining a canonical 3D space and a deformation model simultaneously. The canonical space represents the overall static 3D structure, while the deformation model predicts the deformation of the 3D scene according to temporal variations.

NeRFies [113] builds on the strengths of NeRF but adapts it for situations where scene structure, viewpoint, or object deformation change over time. NeRFies incorporates a deformation model that warps points from a static reference scene to account for these changes, such as object motion or shape deformation. It also uses a latent code to capture per-frame dynamics like lighting changes and object movement.

HyperNeRF [114] is an extension of NeRFies designed to handle complex, dynamic scenes with non-rigid deformations. HyperNeRF goes further by incorporating a space-time deformation field that allows it to model complex geometric changes across both space and time. This capability allows HyperNeRF to represent scenes with more intricate movements, such as human body articulation or subtle deformations that NeRF and NeRFies struggle with.

K-Planes [115] follows a similar strategy by maintaining a canonical space and a deformation field. The main contribution lies in introducing a novel space-time feature representation. K-Planes encodes the entire scene by not only the xyz spatial planes but also the temporal coordinate. Each position is treated as the intersection of multiple planes. Their representation enriches the correlation of the position with its surrounding environments.

As mentioned before, one main limitation of NeRF is the rendering speed. Adding a new deformation model further increases the complexity of the entire process. Given

that a large portion of the scene is empty, researchers are exploring ways to only represent the occupied area. Kerbl *et al.* [24] combined splatting [116] with the neural rendering approach from NeRF to introduce 3D Gaussian Splatting (3D-GS), which has been widely adopted. Each 3D Gaussian $\mathcal{G}_i$ contains a mean vector $\mathcal{X}_i \in \mathbb{R}^3$ to represent the center point, a covariance matrix $\Sigma_i \in \mathbb{R}^{3\times3}$, a opacity α_i, and a view-dependent color defined by spherical harmonic (SH) coefficients $sh_i \in \mathbb{R}^k$ (k is a hyperparameter for the number of SH functions). Because the shape of a Gaussian splat is an ellipsoid, it can be treated as an upgraded version of the point cloud. These additional parameters increase the capability of the 3D scene representation. The 3D-GS rendering equation is as follow:

$$c_i = \mathcal{G}_i^{2D} sh_i \alpha_i, \mathcal{C} = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2.3}$$

3D-GS can achieve comparable accuracy to NeRF while significantly reducing the rendering time. The original 3D-GS is also designed for static scene representation. Therefore, the strategies used by dynamic NeRF can be integrated into 3D-GS to handle dynamic scenes as well.

4DGS [117] directly extends the 3D Gaussian into 4D Gaussian with one additional time dimension. The 4D Gaussian Splats are first projected into 3D space according to the time, and the subsequent rendering procedures follow the original 3D-GS. However, directly adding one more dimension significantly increases the computational costs.

Another 4DGS [118] designed by Wu *et al.*follows the canonical space and deformation field setup. The method first constructs a space of static canonical 3D Gaussian Splats. Then, each 3D Gaussian Splat is encoded using the same multi-planes strategy as K-Planes to predict the 3D offset over time.

Deformable 3D-GS (D3D-GS) [119] also follows the two step approach. The main difference from 4DGS is a novel space-time encoding strategy. Each dimension is embedded independently and directly concatenated into the final feature. This feature is then passed to the deformation model to predict the time-related offset.

However, all previous methods share the common assumption that all objects in the scene are undergoing only minor motions. In other words, there are no large transformations and only local deformations. The capability of the deformation model is small to prevent overfitting, but this assumption also reduces the ability to predict large motions. In reality, the 3D Gaussian Splats from the same object should follow similar motion pat-

tern. However, previous methods process the offset independently, without considering the semantic correlation. Both object-level large motion and semantic correlation are critical in instructional videos.

In order to increase the robustness of the dynamic 3D-GS in large motion range scenarios, we proposed M5D-GS to represent the object overall motion and local deformation separately, which can also be potentially used for object pose estimation.

In the task of 3D/4D reconstruction from video, the image quality, and the diversity of the camera poses also affect the final performance. Low quality images (e.g., motion blur, low light condition) result in inaccurate features. Low diversity of camera poses can lead to overfitting to the given viewpoints during training, and hence the model does not generalize to other views. In recent years, generative models, including Variational Autoencoder (VAE) [120], Diffusion models [121], and flow matching [122], show the strength of generating realistic novel images based on given constraints. Many researchers integrate generative models into 3D reconstruction frameworks to provide additional information.

3DGS-Enhancer [123] is the first method to integrate a generative model into a 3D-GS pipeline. The method first interpolates novel camera poses between the given camera poses, and renders the novel view images accordingly. The interpolated video is fed into a video diffusion block to refine frame sequences that include clean and blurry images.

ViewCrafter [124] is a point cloud based method. The initial point cloud built from the reference frame is first re-projected to a novel viewpoint. Then the re-projected low quality image is refined by a point-conditioned video diffusion model to enhance the quality conditioned by the reference frame.

ViewExtrapolator [125] adopts a similar idea as ViewCrafter, but proposed a novel way to make use of the generative model. The work uses Stable Video Diffusion (SVD) [126] as the base model. The method does not need to finetune/retrain the generative model but directly modifies the direction of gradient from the blurry image to the clean image.

DIFIX3D+ [127] implements a diffusion model trained on image pairs to denoise blurry inputs. A blurry image together with a clean reference image but from a different viewpoint are fed into the generative model. The output contains the corresponding clean images for the two inputs. However, the method assumes that there is a clean reference image that captures a similar view although from another viewpoint.

Some methods directly use generative models to refine the quality of the input images. DiET-GS [128] uses a diffusion prior to deblur images due to fast camera motion by using

the event camera. S2Gaussian [129] increases the input image resolution with a generative model to improve the final representation quality.

However, all of these approaches are designed for static scenes with limited performance on dynamic scenes. The presence of dynamic elements introduces temporal ambiguities that significantly complicate the denoising process, making current generative-enhanced methods ineffective for dynamic 4D reconstruction. To overcome this limitation, our MoSc-GS is proposed to combine dynamic 4D Gaussian Splatting with a generative model. The 4DGS component represents the dynamic object in 4D space, while the generative model compensates for the unconstrained regions of the object that cannot be fully observed due to the limitations of a monocular camera.

2.2.3 Dataset

Figure 2.2: Some examples of the widely used dataset. From the left to the right, D-NeRF [130], HyperNeRF [114], and NeRF-DS [131]

The dataset used for dynamic 3D scene representations can be divided into synthetic scenes and real world scenes. Synthetic scenes can be generated by 3D rendering software such as Blender [132]. The advantages of synthetic scenes include more accurate variables (such as camera pose and timestamps) and the ability to generate training and testing sets simultaneously but from different viewpoints. In addition, object segmentation masks are easy to obtain if needed. In contrast, real-world scenes present various deviations, such as camera calibration error, image blur, and complicated lighting effects. However, real-world scenes are more important since the goal is to apply the designed method to capture physical objects. Some widely used dynamic scene representation datasets are discussed below with examples shown in Fig. 2.2.

Unbiased4D [133] is a simple monocular dynamic object representation dataset. It consists of two real-world recorded scenes and three synthetic scenes. Each scene contains

a single camera trajectory, and both the training and test sets are sampled from this trajectory. The deformation range of the recorded object is also limited.

NeRFies [113] & **HyperNeRF** [114] are two widely used monocular dynamic objects reconstruction datasets, which were both published by the same research group. These datasets focus on the local deformation of object surfaces. Most of the scenes are designed to concentrate on human face and expressions. HyperNeRF also contains some general scenes, which extends the scope of the dataset's applications. However, the test cases in these datasets cover only a portion of the object. Both the training and test sets focus on the same region. The reconstructed object is not complete, since the unseen part of the object is not unconstrained.

N3DV [134], also known as DyNeRF, is a real dataset for dynamic scene representation and new view synthesis. The dataset contains 20 different scenes recorded for 10 seconds at 30 FPS. The data set is human-centered, and most dynamic elements are associated with human body movements. However, the range of motion is relatively small, as only the upper bodies move in most scenes. This dataset is widely used because it is recorded with simultaneous camera views from 18 different cameras.

NeRF-DS [131] is specifically designed for Dynamic Specular (DS) objects. The focus is on objects made of materials like metals and polished surfaces, which can cause specular reflections. A stereo camera is used to record the scene, allowing one camera's data to be used as the training set and the other as the test set.

D-NeRF [130] is a monocular camera based synthetic dataset that contains dynamic objects. The dataset contains eight different scenes involving non-rigid motions. The synthetic scenes are generated using 3D rendering software, with the camera circling around the object. Based on our analysis, this dataset also only contains local object deformations without significant object level transformations. Since the dataset is synthetically generated using rendering software, the camera poses are randomly distributed around the object. The camera trajectory is not continuous over time, which contradicts real-world capture conditions.

There are also several multi-view dynamic scene reconstruction datasets, such as DyNeRF [134] and the DeepView Video dataset [135]. These datasets are not discussed in detail here, as this thesis focuses on the monocular camera scenario.

Existing 4D reconstruction datasets are limited in their ability to evaluate dynamic object representations comprehensively. Most available datasets contain simple motions, or partial object coverage. In particular, they often assume limited camera trajectories

to focus on a portion of the object or provide only synthetic views without realistic motion continuity, which restricts their applicability for large-motion or monocular scenarios. To address these limitations, two new datasets were developed in this thesis. The first, introduced in M5D-GS, focuses on objects with large and complex motions to evaluate robustness under high-dynamic conditions. The second, proposed in MoSc-GS, extends this effort to monocular scanning videos, simulating realistic data acquisition where the object is observed from sparse and non-continuous viewpoints. These datasets not only provide more diverse and challenging benchmarks for dynamic object reconstruction but also serve as valuable resources for advancing research toward real-world 4D reconstruction.

2.3 Summary

In this chapter, we introduced related research for language-guided object reconstruction from monocular videos. Two different tasks are discussed, Referring Video Object Segmentation (RVOS) and 3D scene representations of dynamic environment. For each task, we first inspect related studies, and then highlight the weaknesses in current research which limit the reconstruction ability of the framework of this thesis. The widely used benchmarks for each task are presented to thoroughly investigate the domain gap between current methods and the goals of this thesis. To be specific, this thesis focuses on four different problems, 1) improving the temporal consistency for RVOS task; 2) designing a RVOS framework for long/ongoing video processing; 3) handling large object level motion during the dynamic 3D scene representation; (4) reducing the lack of constraints in dynamic object representation based on monocular scanning video.

3

Temporal Context Enhanced Referring Video Object Segmentation

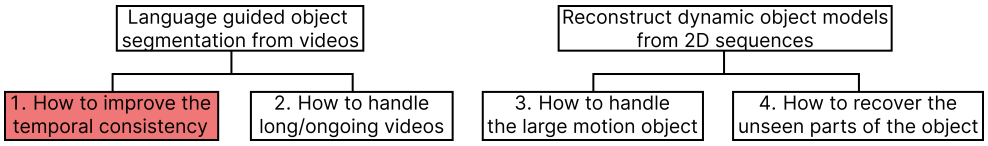


Figure 3.1: The overall tasks of the thesis, this chapter focuses on the task highlighted with red.

In this chapter, we focus on the task of improving the temporal consistency in the Referring Video Object Segmentation. The goal of Referring Video Object Segmentation is to extract an object from a video clip based on a given expression. While previous methods have utilized the transformer's multi-modal learning capabilities to aggregate information from different modalities, they have mainly focused on spatial information and paid less attention to temporal information. To enhance the learning of temporal information, we propose temporal context enhanced RVOS (TCE-RVOS) with a novel frame token fusion

(FTF) structure and a novel instance query transformer (IQT). Our technical innovations maximize the potential information gain of videos over single images. Our contributions also include a new classification of two widely used validation datasets for investigation of challenging cases. The experimental results demonstrate that TCE-RVOS effectively captures temporal information and outperforms the previous state-of-the-art methods by increasing the $J\&F$ score by 4.0 and 1.9 points using ResNet-50 and VSwin-Tiny as the backbone on Ref-YouTube-VOS, respectively, and +2.0 mAP on A2D-Sentences dataset by using VSwin-Tiny backbone, at the time of this work was published.

This work was a collaboration with Basavaraj Hampiholi, Prof. Heiko Neumann and my supervisor Prof. Jochen Lang. I was responsible for the core research contributions, experimental design, and implementation. The coauthors provided valuable feedback and assisted with reviewing and proofreading the manuscript. The work in this chapter has been published [32] and the code is available at `https://github.com/haliphinx/TCE-RVOS`

3.1 Introduction

Video understanding[31, 136] has great potential, as it draws upon visual spatio-temporal information along with audio and language information. Various video-based tasks have been put forward, including but not limited to video classification [137], temporal video action segmentation [138], and video object detection [35]. The transformer structure [54, 139] has shown strong ability in both visual and language understanding, and most importantly, to serve a unified structure for different data formats. All these efforts have led to the task of Referring Video Object Segmentation (RVOS). RVOS is a cross-modal task that takes a video clip with a text expression as input and segments the referred object in all the video frames. Compared with the Referring Image Object Segmentation task (RIOS), the referring expression in RVOS can describe not only an object in space but also motion in the spatio-temporal dimension. Furthermore, RVOS methods also require data association to track the referred object across multiple frames.

Initially, researchers utilized complicated model structures with multi-step training strategies [140–142]. Recently, benefiting from the transformer structure, various end-to-end learning structures [10, 13, 21] have been published that have achieved state-of-the-art performance on various benchmarks. However, previous methods focused on either text and visual information aggregation or image-based feature learning, but paid less attention

to the inter-frame temporal understanding. This leads to a problem that the segmentation performs well on most frames individually but lacks the ability to fuse information across frames. As a consequence, these methods have limitations in handling motion blur and occlusions. To overcome these limitations and fully exploit the inter-frame temporal information in the video, we present a novel approach called Temporal Context Enhanced Referring Video Object Segmentation (TCE-RVOS). Various experiment results show that TCE-RVOS outperforms previous state-of-the-art methods by handling some challenging scenarios (e.g., occlusion, and motion) better.

Our main contributions are as follows:

- We designed a frame token fusion (FTF) module as encoder to aggregate features between frames in the video clip using memory tokens. The memory tokens first distill information for each frame independently and then enrich the overall encoding with information from other frames.
- We propose an instance query transformer (IQT) module in the decoding stage to directly aggregate queries about the same object in different frames. This overcomes issues caused by insufficient visual information in the current frame due to, e.g., occlusion or motion blur.
- We further organize the Refer-YouTube-RVOS [10] validation set into subcategories including occlusion, motion, crowded, interaction between objects, ambiguous queries, and object presence to investigate how our method improves the SOTA.

Our proposed method improves the *J&F* by a large margin of 4.0 and 1.9 points over the previous SOTA ReferFormer [13] using spatial backbone ResNet-50 [30], and spatio-temporal backbone Video Swin Transformer tiny [31], respectively.

3.2 Related Work

The overall background research is provided in Sec. 2.1. This section primary focuses on the detailed related work up to the date when this work was published to emphasize the contribution of this work at that time.

Video Object Segmentation is commonly solved using two different model types: offline and online models. The offline model solves all frames at once [20, 143, 144], while

the online model segments the object in the first frame then propagates to the rest [145–147]. Solving all frames at once has a large temporal receptive field by processing the whole video clip simultaneously. However, it requires large memory size for computing and is weak in data association. VisTR [20] builds upon the image-based object segmentation structure DETR [148] by processing the object queries from all the frames together. IFC [143] shares the similar idea of an image-based object segmentation Mask2Former [149] for video-based segmentation and utilizes a token-based inter-frame communication to overcome the data association problem. On top of IFC, VITA [144] fully tokenizes the frame features to distill the visual information. The online model IDOL [147] shows that the inter-frame data association is the weakness in current video instance segmentation tasks. IDOL utilizes contrastive learning to enhance the data association across frames.

Referring Video Object Segmentation is a relatively new task that was first introduced by Gavrilyuk *et al.* [35] in 2018 to segment the actors and actions in video clips. Since RVOS involves video object segmentation, neural language processing, and cross-modal learning, early research [56, 140–142] typically combined models from different tasks, resulting in complex structures that are difficult to train end-to-end. A straightforward approach is to extend image-based methods [15–18] to process each frame of the video clip separately. However, this approach neglects temporal information. To incorporate temporal information, spatio-temporal backbones such as I3D [47], and VSwin Transformer [31] have been used. Nonetheless, the processing after the backbone still treats each frame independently. URVOS [10] splits the task into an image-based referring object segmentation and a mask propagation task. Recurrent neural networks and memory mechanisms are used in [141, 142] to provide an online strategy. However, this approach loses the ability to leverage information after the current frame. Another direction is to integrate linguistic features into the video object segmentation task [19, 20]. MTTR [21], and ReferFormer [13] are two state-of-the-art methods that borrow ideas from a VOS structure called VisTR [20]. However, cross-frame information exchange only occurs in the backbone stage (if a spatio-temporal backbone is used), and the instance sequence matching at the final stage. The encoder and decoder stages still process each frame independently. MTTR also requires un-referred instance masks during training, which increases the annotation workload. Some researches stipulated that the imbalance between the language pipeline and the vision pipeline would affect the model performance. VLT [58] generates a number of language features for a single sentence to close the gap between the two types of feature. R^2-VOS [59] first enriched the original dataset with mismatched video-text pairs, then proposed a contrastive learning structure to filter out the mismatched pairs to help the model understand the language feature better.

3.3 Method

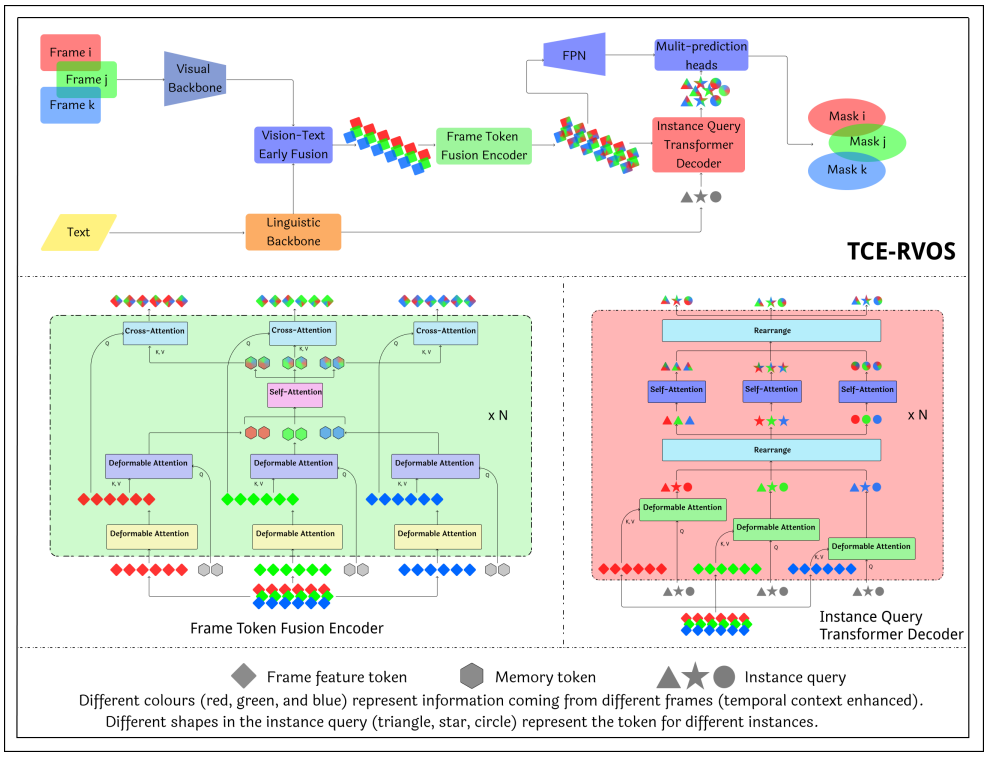


Figure 3.2: Overview of the TCE-RVOS framework. The sub-graph on the top shows the overall structure. The bottom left and bottom right sub-graphs present the frame token fusion encoder and instance query transformer decoder respectively. The attention blocks in the same level with the same background color share the same weights.

Our proposed TCE-RVOS method is based on ReferFormer [13], which is the previous state-of-the-art model. The model follows an offline framework with backbone, encoder, decoder and post-processing structure, where the entire video clip and corresponding text narration are input to the model. The binary mask of the referred object in each frame is predicted. After analyzing the framework of previous works [13, 21], we found that the temporal context aggregation only happens during the feature extraction when using a spatio-temporal backbone like Video Swin Transformer, and in the post processing stage. However, the encoder and decoder stages are adapted from the image based segmentation

model DETR and handle each frame independently. TCE-RVOS utilizes a newly designed frame token fusion encoder and an instance query transformer decoder to enhance communication across video frames. The TCE-RVOS model structure is presented in Fig. 3.2. The model contains four main stages: (1) Backbone and early fusion (Sec. 3.3.1), (2) Frame Token Fusion encoder (Sec. 3.3.2), (3) Instance Query Transformer Decoder (Sec. 3.3.3), and (4) Post-processing and Prediction (Sec. 3.3.4). The second and third stages contain our newly proposed structure, and the other two stages are adapted and upgraded from ReferFormer. We note that the same baseline structures are also used in various other RVOS models (e.g., MTTR [21]). In Fig. 3.2, the sub-graph on the top shows the overall framework. The sub-graphs on the bottom left and bottom right present the detailed structure of frame token fusion encoder and instance query transformer decoder. Frames from the video clip and the text expression are passed through the visual backbone and linguistic backbone separately. Once the visual and linguistic features are fused by the vision-text early fusion block, the aggregated features are sent into the frame token fusion encoder to further process the information as well as communicate between frames. The temporally enhanced features are used to guide the instance queries to distill the instance-related features in the instance query transformer decoder. Finally, each instance query is used to predict the instance mask, bounding box, and reference score.

Given a video clip $V = \{I_i\}_{i=1}^{T}, I_i \in \mathbb{R}^{D \times H \times W}$, with T frames and a text expression $E = \{t_i\}_{i=1}^{L}$ with L words. D represents the number of frame channels, H and W represent the height and width of the frame. The proposed model will predict T frames of binary segmentation masks of the referred object, $M = \{m_i\}_{i=1}^{T}, m_i \in \mathbb{R}^{H \times W}$.

3.3.1 Backbone & Early Fusion

Visual Backbone. Since the proposed model can easily adapt to different backbone structures, various types of visual backbones are tested in our model. ResNet [30] is utilized to provide 2D image-based spatial features. The Video Swin Transformer [31] is tested to provide the spatio-temporal feature of the whole input video clip. The output last few layers of the backbone is used to provide multi-scale features. The generated features are flattened from 2D feature maps into 1D feature tokens for processing by subsequent multi-head attention blocks. As the result, the output for a given video clip is $F_{vis} = \{v_i\}_{i=1}^{T}, v_i \in \mathbb{R}^{S \times C}, S = \sum_l H_l \times W_l$. C is the channel size of the feature. l represents the set of feature layers from the backbone.

Linguistic Backbone. We follow ReferFormer [13] and use RoBERTa [150] as a

backbone to extract features of the text expression. RoBERTa returns a set of the word based features corresponding to each word separately, $F_{word} = \{f_i\}_{i=1}^{L}, f_i \in \mathbb{R}^C$, and a sentence based feature $F_{sentence} \in \mathbb{R}^C$ to generate the initial object query in the decoding stage.

Vision-Text Early Fusion. This block is a vanilla multi-head cross-attention model. The visual feature F_{vis} is added to a fixed 3D positional encoding as the *query*. The *key* and *value* are based on F_{word} . This stage allows the resultant features to carry both vision and language context. A simple feedforward network (FFN) is utilized to further process the output feature.

3.3.2 Frame Token Fusion Encoder

Previous works [13, 21] encode frames in a batch by treating each frame individually during encoding. The features between different frames are not aggregated and thus the temporal context is not well encoded. Inspired by IFC [143], a video instance segmentation framework, we have designed an efficient yet simple structure as shown in the bottom left of Fig. 3.2 to enhance the temporal context aggregation between frames in the encoding stage. Compared with the inter-frame communication used in IFC, which directly concatenates memory tokens to the end of each feature token query and performs self-attention, the proposed structure not only saves memory but also incorporates multi-scale features that benefit the visual understanding.

The deformable self-attention model [57] is used to reduce the overall memory usage and support multi-scale feature processing. The features from each frame are first processed by an identical deformable attention block to independently aggregate the spatial information in each frame (shown in the yellow blocks in Fig. 3.2). A set of randomly initialized trainable memory tokens, denoted by $F_{memory} = \{j_i\}_{i=1}^{T}, j_i \in \mathbb{R}^{N \times C}$ are utilized to distill and represent information for each frame. The hyperparameter N determines the number of memory tokens used for each frame. This step also uses the deformable cross-attention model, F_{memory} as *query*, and F_{vis} as *key* and *value*, so that the memory tokens obtain multi-scale information from the frame features (shown as the dark blue blocks in Fig. 3.2). The resulting memory tokens, which now contain information from all frames independently, are forwarded as input to a self-attention block (shown as the pink block in Fig. 3.2) for communication between frames. Finally, we update frame feature tokens F_{vis} corresponding to the memory tokens using a naive cross-attention block. Each attention block is followed by a simple FFN, which is not shown in the graph for clarity. The output

features are now carrying the information from both the corresponding frame, and other frames.

Compared with the inter-frame communication used in IFC which directly concatenates memory tokens to the end of each feature token query and performs self-attention, the proposed structure not only saves memory but also incorporates multi-scale features that benefit the visual understanding.

3.3.3 Instance Query Transformer Decoder

Previous models MTTR [21] and ReferFormer [13] extend the image based segmentation model DETR [148] into a video-based model by using the same decoder for each frame in the video in parallel, and hence independently. The data association is hardcoded based on the query order (i.e., the first instance query from each frames belongs to the same instance). This strategy does not incorporate instance query communication between frames. The advantages of video over single image are not fully exploited. To enhance the learning of temporal information, we design the Instance Query Transformer Decoder block as shown in the bottom right of Fig. 3.2 to enhance temporal information learning by operating multi-head attention between the features of an object from all frames.

The proposed decoder follows a DETR-like structure that initializes a number of instance queries, $F_{instance} = \{h_i\}_{i=1}^{T}, h_i \in \mathbb{R}^{Q \times C}$, to generate Q instance candidates for each frame, guided by the frame feature tokens output from the encoder. However, different from previous models, the newly designed decoder establishes a two step feature aggregation structure to enhance the temporal context learning. The instance query first aggregates the spatial features from each frame independently, then temporal features from all frames belonging to the same instance are fused. The instance queries are initialized by the sentence feature $F_{sentence}$ and a fixed 2D positional embedding. Then deformable cross-attention is used to extract information from the frame feature tokens output from the encoder (shown as the green blocks in Fig. 3.2). The instance queries from different frames are rearranged and combined into groups based on the corresponding instance (shown as the light blue blocks in Fig. 3.2), and processed by a self-attention block to communicate between them (shown as the dark blue blocks in Fig. 3.2), which provides the temporal context aggregation. The functionality of the rearrange block is the transpose of the first two dimension of the vector, and can be described as $Rearrange(\mathbb{R}^{T \times Q \times C}) = \mathbb{R}^{Q \times T \times C}$. Deformable cross-attention is used to let the frame query extract information from the frame feature token (shown in green blocks). The instance queries from different frames are then

rearranged so that the queries for the same instance but different frames are combined into one group. Each query group is processed by a self-attention block to communicate between each other. By doing so, the instance from different frames with different viewpoint can benefit the instance segmentation especially when the referred instance is occluded or has motion blur in some of the frames. The Instance Query Transformer Decoder block output the processed instance queries for the Multi-tasks prediction.

3.3.4 Post-processing and Prediction

The post-processing stage in the proposed method is similar to that used in MTTR and ReferFormer. However, we found that the cross-modal feature pyramid network (CM-FPN) used in ReferFormer takes a large amount of memory but accuracy improvements are limited. We use instead a classic Feature Pyramid Network (FPN) reducing the model size but with the same overall performance. Three outputs are predicted for each instance query sequence: The bounding box, denoted as $P_{bbox} \in \mathbb{R}^{4\times T\times Q}$; the reference score, denoted as $P_{ref} \in \mathbb{R}^{T\times Q}$, which indicates the confidence of each instance query being referred to by the sentence, and the segmentation mask, denoted as $P_{mask} \in \mathbb{R}^{H\times W\times T\times Q}$ for each instance query in each frame. To generate the P_{bbox} and P_{ref} outputs, two feed-forward networks with three layers are used separately for each instance query. The mask for each instance query is predicted by first generating conditional convolution kernels [151] using the instance query and the dense feature map, and then applying convolution with these kernels to the dense feature map. The instance sequence with the highest overall P_{ref} score is selected as the final output prediction for the video.

3.3.5 Loss Functions

The overall loss consists of three terms as follows

$$\mathcal{L}_{overall} = \lambda_{bbox}\mathcal{L}_{bbox} + \lambda_{ref}\mathcal{L}_{ref} + \lambda_{mask}\mathcal{L}_{mask}. \tag{3.1}$$

The bounding box prediction loss $\mathcal{L}_{bbox}$ is composed of the L1 loss and the generalized IoU (GIoU) loss [152]. $\mathcal{L}_{ref}$ is a focal loss to supervise whether the prediction is the referred instance. $\mathcal{L}_{mask}$ consists of the DICE loss and per-pixel focal loss [153]. $\lambda_k, k \in \{bbox, ref, mask\}$, represents the corresponding coefficients to balance different losses (see Sec. 3.4.1 for their settings).

3.4 Experimental Evaluation

3.4.1 Experimental Setup

Datasets. The proposed method was evaluated on Ref-YouTube-VOS [10], A2D-Sentences [35] and Ref-DAVIS17 [48] datasets to compare the performance with state-of-the-art methods. The Ref-YouTube-VOS dataset is a large-scale benchmark that contains 3978 high-resolution videos from YouTube, with 15K language expressions and 131K high-quality manual annotations. The dataset is split into 3471 training videos, 202 validation videos and 305 test videos. Since the test video set is not publicly available, all models were evaluated on the validation set for fair comparison. Ref-DAVIS17 is built from DAVIS17 [66] by adding text expressions to the VOS dataset. Although the dataset only contains 90 videos, it is still widely used in the R-VOS task. A2D-Sentences dataset extends the original A2D video object segmentation dataset with text expressions. A2D-Sentences contains 3,782 video samples with 3-5 frames annotations per sample.

Evaluation Metrics. The standard evaluation metrics for Ref-YouTube-VOS and Ref-DAVIS17 are Jaccard index (J) for region similarity, contour accuracy $F1$ score (F), and their average ($J\&F$). The (J) score focuses on the overall segmentation quality, and the (F) score focuses more on the segmentation details (boundary accuracy).

To evaluate the proposed method and compare with previous works on A2D-Sentences dataset, we adopt precision@K ($K \in [0.5, 0.6, 0.7, 0.8, 0.9]$), overall & mean IoU, and mean average precision (mAP) over 0.50:0.05:0.95. Overall IoU computes the ratio between the total intersection and the total union area over all the test samples. Mean IoU is the averaged IoU over all the test samples.

Implementation Details. Various backbones were evaluated in our model. The outputs from the last three layers of the backbone are used to generate multi-scale features with spatial down-sampling rates of {8, 16, 32} respectively. The number of memory tokens used in the Frame Token Fusion Encoder is $N = 8$. All the experiments were conducted using 4 Nvidia V100 GPUs with 32GB memory.

In order to compare with state-of-the-art methods, we use a similar hyperparameter settings than [13]. In particular, both the encoder and decoder are 4 layers. The dimension for both visual and linguistic features is $C = 256$. Batch size is set to 1. The number of instance queries for each frame in the Instance Query Transformer Decoder is $Q = 5$. We use AdamW [154] with an initial learning rate $1e-4$ as the optimizer. The coefficients for

the loss function are $\lambda_{bbox} = 2$, $\lambda_{ref} = 2$, and $\lambda_{mask} = 5$. By default, the input video frame number for training is $T = 5$, and the model is pretrained on the image based referring object segmentation dataset Ref-COCO [63]. The video downsampling strategy follows the previous work ReferFormer. First, a middle frame (the 3rd frame) is randomly selected. Then, two frames (the 2nd and 4th frames) are randomly sampled within a short temporal range before and after the middle frame. Finally, two additional frames (the 1st and 5th frames) are randomly sampled from a longer temporal range on both sides of the middle frame. This strategy increases the diversity of the training set. All the experiments are conducted with the competitors published up to the date when this work was published.

3.4.2 Comparison Results

Tab. 3.1 presents a comparison between the proposed TCE-RVOS and state-of-the-art methods on Ref-YouTube-VOS dataset. The upper half of the table shows the comparison of methods using a spatial backbone. TCE-RVOS outperforms all other methods with a ResNet-50 backbone, with an improvement of at least 4.0 points in the $J\&F$ index. Moreover, our model performs better than previous models with the larger backbone ResNet-101 by at least 2.3 points. By using ResNet-101, our method achieves state-of-the-art. The bottom half of the table shows the comparison for all models using a spatio-temporal backbone. TCE-RVOS outperforms other models using the same VSwin-Tiny backbone with at least 1.9 points improvement, and even outperforms a model using the large-scale backbone VSwin-Small with a 1.2 points increase. Fig. 3.3 shows two prediction examples. In each sub-figure, the top, middle, and bottom sequences represent the result from MTTR, ReferFormer, and TCE-RVOS, respectively. Two challenging scenarios are selected as the examples. In Fig. 3.3a, the person is partially occluded by the window in some of the video frames, but never fully occluded. In Fig. 3.3b, the parachute is fully occluded by a person in the front in several frames at the start. The result shows that our proposed method increases the capability of handling occluded instance segmentation to predict more accurate masks, and has the ability to verify if the referred instance is visible in the frame by using the information from other frames.

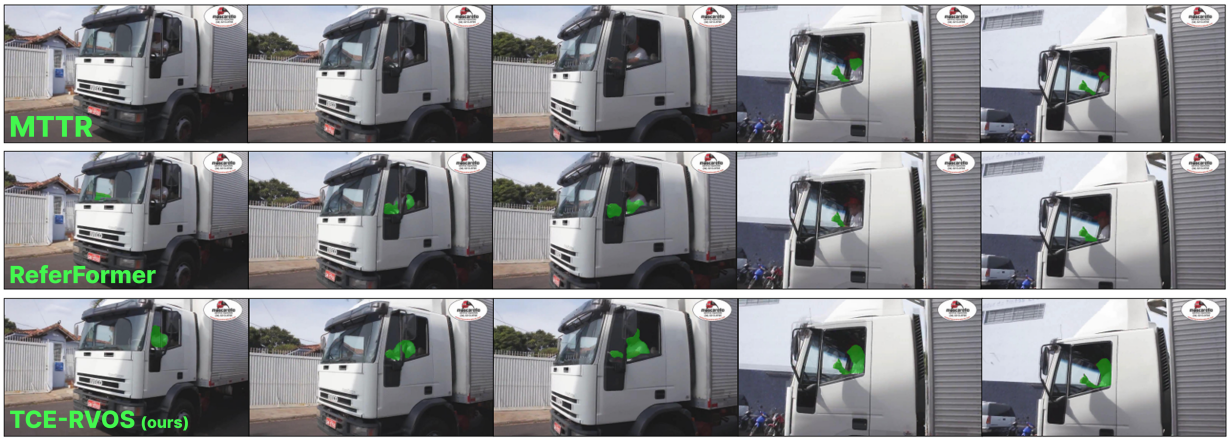
Tab. 3.2 shows a comparison result on Ref-Davis17 dataset for various methods by using the ResNet-50 backbone. Since Ref-Davis17 is a small dataset with only 90 videos in total, we directly evaluate the model trained on Ref-YouTube-VOS dataset on Ref-Davis17 without fine-tuning. The result shows that our model has a good generalization which achieves similar accuracy on both dataset, and outperforms all the competitors.

Method	Backbone	*J&F*	*J*	*F*
Spatial Backbone				
CMSA [155]	ResNet-50	34.9	33.3	36.5
CMSA+RNN [155]	ResNet-50	36.4	34.8	38.1
URVOS [10]	ResNet-50	47.2	45.3	49.2
PMINet [55]	ResNet-101	48.2	46.7	49.6
PMINet + CFBI [55]	ResNet-101	53.0	51.5	54.5
ReferFormer [13]	ResNet-50	55.6	54.8	56.5
ReferFormer [13]	ResNet-101	57.3	56.1	58.4
TCE-RVOS (ours)	ResNet-50	<u>59.6</u>	<u>58.3</u>	<u>60.8</u>
TCE-RVOS (ours)	ResNet-101	**60.8**	**59.4**	**62.2**
Spatio-Temporal Backbone				
MTTR [21]	VSwin-Tiny	55.3	54.0	56.6
ReferFormer [13]	VSwin-Tiny	59.4	58.0	60.9
ReferFormer [13]	VSwin-Small	<u>60.1</u>	<u>58.6</u>	<u>61.6</u>
TCE-RVOS (ours)	VSwin-Tiny	**61.3**	**59.8**	**62.7**

Table 3.1: Comparison with state-of-the-art methods on Ref-Youtube-VOS [10]. The top portion shows models with spatial backbone, and the bottom portion shows models with spatio-temporal backbone. The best results are in bold, and the second best results are underlined.

Method	*J&F*	*J*	*F*
CMSA [155]	34.7	32.2	37.2
CMSA+RNN [155]	40.2	36.9	43.5
URVOS [10]	51.5	47.3	56.0
ReferFormer [13]	<u>58.5</u>	<u>55.8</u>	<u>61.3</u>
TCE-RVOS (ours)	**59.4**	**56.5**	**62.4**

Table 3.2: Comparison with state-of-the-art methods by using ResNet-50 as a backbone on Ref-Davis17 [48]. The results for ReferFormer and TCE-RVOS are obtained with a model trained on Ref-Youtube-VOS datset without finetuneing.



(a) The prediction result for the expression "a person wearing a white shirt is driving a white truck moving down the road", shown in green masks.

(b) The prediction result for the expression "a white and red parachute blowing in the wind", shown in blue masks.

Figure 3.3: Comparison between qualitative result of MTTR (top sequence), ReferFormer (middle sequence), and TCE-RVOS (bottom sequence) from Ref-YouTube-RVOS dataset [10]. (a) Partial occlusion, and (b) Complete occlusion.

Method	Backbone	Precision					IoU		mAP
		P@0.5	P@0.6	P@0.7	P@0.8	P@0.9	Overall	Mean	
Hu *et al.*[40]	VGG-16	34.8	23.6	13.3	3.3	0.1	47.4	35.0	13.2
Gavrilyuk *et al.*[35]	I3D	47.5	34.7	21.1	8.0	0.2	53.6	42.1	19.8
CMSA+CFSA [156]	ResNet-101	48.7	43.1	35.8	23.1	5.2	61.8	43.2	-
ACAN [157]	I3D	55.7	45.9	31.9	16.0	2.0	60.1	49.0	27.4
RefVOS [52]	ResNet-101	57.8	-	-	-	9.3	67.2	49.7	-
CSTM [142]	I3D	65.4	58.9	49.7	33.3	9.1	66.2	56.1	39.9
CMPC-V [141]	I3D	65.5	59.2	50.6	34.2	9.8	65.3	57.3	40.4
ClawCraneNet [158]	ResNet-50/101	70.4	67.7	61.7	48.9	17.1	63.1	59.9	-
MTTR(w=8) [21]	VSwin-Tiny	72.1	68.4	60.7	45.6	16.4	70.2	61.8	44.7
MTTR(w=10) [21]	VSwin-Tiny	75.4	71.2	63.8	48.5	16.9	72.0	64.0	46.1
TCE-RVOS (ours)	ResNet-50	80.3	77.1	70.1	53.3	18.2	75.6	67.5	51.2
ReferFormer [13]	VSwin-Tiny	82.8	79.2	72.3	55.3	19.3	77.6	69.6	52.8
ReferFormer [13]	VSwin-Small	82.6	79.4	73.1	57.4	21.1	77.7	69.8	53.9
TCE-RVOS (ours)	VSwin-Tiny	83.0	79.9	73.6	56.7	20.5	77.5	69.9	54.8
ReferFormer [13]	VSwin-Base	<u>83.1</u>	<u>80.4</u>	<u>74.1</u>	<u>57.9</u>	<u>21.2</u>	**78.6**	<u>70.3</u>	<u>55.0</u>
TCE-RVOS (ours)	VSwin-Base	**83.3**	**80.6**	**74.6**	**58.6**	**22.2**	<u>78.4</u>	**70.5**	**56.0**

Table 3.3: Comparison with state-of-the-art Methods on the A2D Dataset [35]. The best results are in bold, and the second best results are underlined.

Tab. 3.3 shows the comparison result of TCE-RVOS with previous methods. Only using the spatial backbone (ResNet-50), the proposed model already outperforms most of the previous methods. When a spatio-temporal backbone (VSwin-Tiny) is used, TCE-RVOS outperforms all the previous methods using the backbone no larger than VSwin-Tiny with at least 2.0 mAP improvement, and also outperforms the model using a larger backbone VSwin-Small with 0.9 mAP increment. When the large spatio-temporal backbone VSwin-Base is used, our proposed method outperforms all others, and achieves the state-of-the-art.

3.4.3 Ablation Study For Model Structure

To fully investigate where and how our model improves the previous SOTA method, some ablation studies are made including a model component analysis, different scenarios observations, and a study of the impact of temporal window size. All the experiments in this section are using ResNet-50 as backbone and are trained on Ref-YouTube-VOS dataset if nothing else is specified.

Model Components. In order to show that the designed Frame Token Fusion Encoder (Sec. 3.3.2) and Instance Query Transformer Decoder (Sec. 3.3.3) benefit the model performance, ReferFormer is selected as the baseline since it is the previous state-of-the-art model and shares similar overall structure with TCE-RVOS. The comparison is done by changing the encoder and decoder. Tab. 3.4 shows the ablation study result for the model components. Both Frame token Fusion Encoder and Instance Query Transformer Decoder benefit the model performance. Notably the Frame Token Encoder improves the contour accuracy (F) score more than the region similarity (J) score. The state-of-the-art method ReferFormer performs well on the overall segmentation quality in common scenarios. However, some challenging scenarios (i.e., occlusion, and motion blur) will change the shape of the object and decrease the boundary prediction accuracy (F). By adding our newly designed structures to enhance the temporal context understanding in the network, the segmentation information from other frames will benefit the prediction in challenging frames.

Length of Temporal Window. We conduct an experiment to understand the impact of the temporal window size on the final performance. In Tab. 3.6, we observe that the performance of the model improves as the number of input frames are increased. These results imply that our proposed method is able to capture temporal context and thereby contributing to the enhancement of model performance.

	J&F	*J*	*F*
ReferFormer	55.6	54.8	56.5
+FTF Encoder	56.3 (+0.7)	54.9 (+0.1)	57.6 (+1.1)
+IQT Decoder	58.1 (+2.5)	57.1 (+2.3)	59.0 (+2.5)
TCE-RVOS	59.6 (+4.0)	58.3 (+3.5)	60.8 (+4.3)

Table 3.4: Ablation study. Both, our frame token fusion (FTF) encoder and instance query transformer (IQT) decoder benefit the model. ReferFormer [13] is the baseline of our method.

	Occlusion			**Presence**		**Crowded**		**Interaction**		**Ambiguity**		**Motion**		
	No	Partial	Full	Full	Partial	No	Yes	No	Yes	No	Yes	No	Slow	Fast
Samples	325	453	56	688	156	402	432	541	293	718	111	203	275	356
ReferFormer	63.3	52.9	33.2	60.0	34.9	63.7	48.2	61.9	44.1	59.3	31.9	51.9	62.8	52.2
+FTF Encoder	63.7	53.5	35.8	59.7	40.2	65.0	48.1	63.6	42.8	59.3	36.8	51.6	64.1	52.9
+IQT Decoder	64.6	55.9	37.9	61.5	41.8	66.0	50.7	64.6	45.9	61.6	35.3	**56.8**	63.3	54.7
TCE-RVOS	**65.9**	**57.4**	**40.1**	**62.6**	**45.3**	**66.1**	**53.4**	**65.5**	**48.6**	**62.6**	**40.5**	56.7	**67.0**	**55.5**

Table 3.5: Ablation study for testcase scenarios. Results are obtained with the ResNet-50 backbone on Ref-Youtube-RVOS [10]. The best results are in bold, and the second best results are underlined. Our frame token fusion (FTF) encoder and instance query transformer (IQT) decoder are most effective in categories where temporal information is relevant. ReferFormer [13] is the baseline of our method.

No. of frames	*J&F*	*J*	*F*
T=3	57.7	56.5	58.9
T=4	58.7	57.3	60.0
T=5	**59.6**	**58.3**	**60.8**

Table 3.6: Ablation study for the impact of the temporal windows size. We use Resnet-50 as backbone on Ref-Youtube-RVOS [10]. The accuracy increases as the number of input frames are increased.

Above experiments show the improvement of TCE-RVOS over state-of-the-art models by not only increasing the overall accuracy, but also handling challenging scenarios better. Both, the novel proposed encoder and decoder benefit model performance by enhancing the temporal context through all video frames. TCE-RVOS shares a similar model size with the ReferFormer model (e.g., 178M vs. 177M parameters when using the VSwin-Base backbone).

3.4.4 Ablation Study For Testcase Scenarios

In RVOS datasets, the text expression describes a single instance but the instance may not be in view in all the frames of a video. It may also be in the field-of-view but occluded by other objects in the scene. The instance segmentation is of varying difficulty because the instance may be stationary or moving relative to the camera, or because there may be many different instances of the same class of objects visible in the video. The different text expressions are also of varying quality and may describe the instance in absolute terms or only relative to other scene objects. Sometimes even multiple objects fit a given text expression. Therefore, we classify the validation set based on the testcase scenarios according to the following categories:

1. Occlusion: We categorize videos into no occlusion, partial occlusion, and full occlusion. If the referred instance is fully occluded in any of the frames of the video clip, we classify the clip as fully occluded. A video is classified as showing partial occlusion if the referred instance is partially occluded in at least one frame but never fully occluded in any frame.

2. Presence: We categorize videos into partial presence and full presence of the referred instance. Presence is lost if the referred instance completely exits or hasn't entered the field-of-view in one of the video frames, which is different from occlusion.

3. Object Motion: We categorize object motion based on the differences of the center and size (height, width) of the bounding box in the video frames. We use categories of no, slow and fast motion of the referred instance.

4. Crowded: We categorize a video to show crowding if there are multiple instances of objects with the same class than the referred instance (e.g., referring to one person in a group of people). The category is split into crowded and not crowded.

5. Interaction: We categorize videos whether the text expression describes the instance by an attribute related to other objects in the frame (e.g., the person near a tree). This category is divided into no interaction and interaction.

6. Ambiguity: We define a sample as ambiguous if there are multiple instances in the video clip which satisfy the text expression. The samples are divided into ambiguous and unambiguous.

Above categories can be clustered into two groups for different evaluations. (1) Temporal relationship: occlusion, presence, and motion can benefit from enhancing the temporal information because the poor visibility of the instance is limited to part of the frames. The segmented object from other frames can guide the prediction in frames with poor visibility. (2) Vision-text aggregation: crowded, interaction, and ambiguity are closely related to the textual description of the referred to instance but can benefit little from temporal relationships. Some visualized examples and detailed discussions for different challenging scenarios are shown as below.

Presence. In the category presence, we identify videos and text expressions where the referred instance is not in the field-of-view in some of the frames. This scenarios is due to the relative motion between the camera and the referred instance. Fig. 3.4 shows an example of the scenario when the referred instance is not present in some frames of the video. The referred instance "white toilet" is not in the camera field-of-view in the first several frames, and appears only later due to the camera motion. TCE-RVOS (the bottom sequence in Fig. 3.4) outperforms other methods by not only detecting that the referred instance is not present in the first two frames shown but also by generating more accurate masks in the remaining frames. The presence category is semantically different from occlusion as an occluded object is in the field-of-view of the camera but the view is (partially) obstructed by other foreground objects.

Object Motion. We use a simple yet effective script to classify object motion by evaluating the change in the bounding box location and size of the referred instance between frames. The pseudo-code is shown in algorithm 1.

Since Ref-YouTube-RVOS dataset [10] does not provide the ground truth for the validation set, we conduct experiments on the A2D dataset [35] with two different thresholds for the bounding box center τ_c and size change τ_s. We use $\tau_c = \tau_s = 25$ and 50 pixels, respectively. A comparison of the results is shown in Tab. 3.7. The results show that the improvement of TCE-RVOS over ReferFormer is mainly coming from the challenging

Figure 3.4: Comparison between qualitative result of MTTR [21] (top sequence), ReferFormer [13] (middle sequence), and TCE-RVOS (bottom sequence) from the Ref-YouTube-RVOS dataset [10]. The expression is "the white toilet is between the white tub and green cabinet", and results are shown in purple masks. The example shows a partial presence scenario.

Algorithm 1 Instance Motion Classification Script

Input: τ_c, τ_s: the pre-defined thresholds for bounding box center and size changes; w_i, h_i, c_i: the width, height, and center of the instance bounding box in frame i
Output: The classified motion status for each instance
$\hat{w} \leftarrow w_i - w_{i-1}$
$\hat{h} \leftarrow h_i - h_{i-1}$
$\hat{c} \leftarrow \text{distance}(c_i, c_{i-1})$
if $\hat{c} \geq \tau_c$ **then**
    The instance is in fast motion
else
    if $\hat{w} \geq \tau_s$ **or** $\hat{h} \geq \tau_s$ **then**
        The instance is in slow motion
    else
        The instance is not in motion
    end if
end if

	Motion		
	No	Slow	Fast
$\tau_c = \tau_s = 25$ *pixels*			
Samples	220	264	811
ReferFormer	47.2	59.3	55.1
TCE-RVOS (ours)	47.2 (+0.0)	60.8 (+1.5)	56.0 (+0.9)
$\tau_c = \tau_s = 50$ *pixels*			
Samples	473	350	472
ReferFormer	51.3	60.6	54.0
TCE-RVOS (ours)	52.0 (+0.7)	61.8 (+1.2)	55.0 (+1.0)

Table 3.7: Effect of different thresholds for motion classification (see Algorithm 1). Results are with the VSwin-Base backbone on the A2D validation dataset [35].

scenarios. The incremental improvement for the slow and fast motion classes are higher than for the no motion class. A threshold of $\tau_c = \tau_s = 25$ makes this clearer in the case of the A2D dataset.

Interaction. In some of the video samples, the text expression describes the instance by its relationship with other objects in the scene. This scenario increases the difficulty of inference, since the model not only needs to detect multiple instances, but also model the relationship between them. Fig. 3.5 shows an example of an interaction scenario. The target instance is referred by "an adult seal to the left of another adult seal". The expression "left" is related to another seal on the right, and the expression "adult" is related to the baby seal in the front. As shown in Fig. 3.5, all three compared methods understand the expression "left" and hence generate masks on one of the two seals on the left. MTTR (top sequence in Fig. 3.5) cannot utilize the expression "adult", and generates the mask on the baby seal in the front. ReferFormer (middle sequence in Fig. 3.5) is able to model the interaction between the different instances. However, the motion and occlusion of the referred instance leads to poor quality of the mask prediction. The result shows that TCE-RVOS not only better models the relative interaction between instances, but also generates more accurate masks compared with the two competitors.

Ambiguity. The Ref-YouTube-RVOS dataset contains various complicated scenes in which the object is hard to describe with a single text expression. Fig. 3.6 shows a good example. The supplied text expression is "a black bird flying among other birds to the

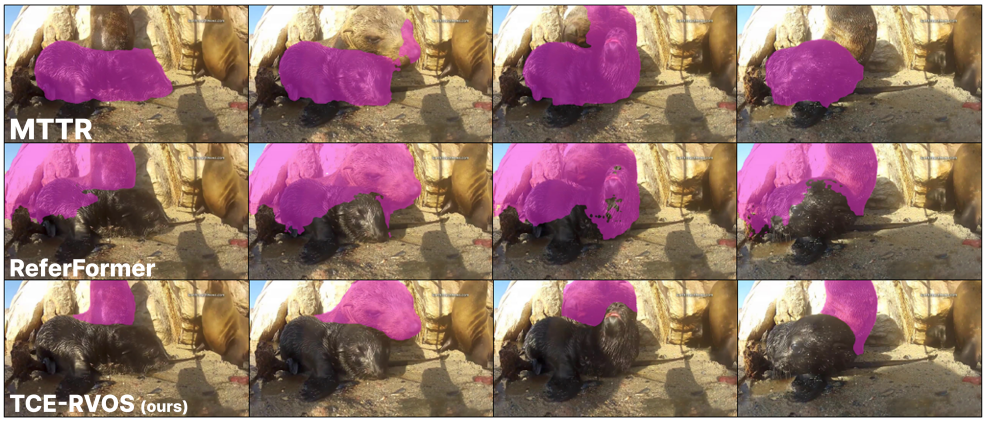


Figure 3.5: Comparison between qualitative result of MTTR (top sequence), ReferFormer (middle sequence), and TCE-RVOS (bottom sequence) from Ref-YouTube-RVOS dataset [10]. The expression is "an adult seal to the left of another adult seal", and results are shown in purple masks. The example shows an interaction scenario.

left". This text expression is not able to uniquely identify a specific bird from all the birds in the video clip. Multiple instances satisfy the expression in the video clip. In Fig. 3.6 only the original frames from the Ref-YouTube-RVOS dataset are shown because the ground truth is not provided.

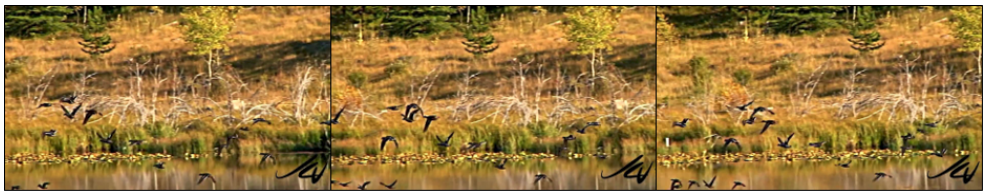


Figure 3.6: An example frame sequence with the expression "a black bird flying among other birds to the left". The example shows a scenario with an ambiguous text expression.

Tab. 3.5 shows the results for the testcase scenarios study. Since the ground truth for the Ref-YouTube-VOS validation set is not accessible, the motion status is classified empirically. We also classified the A2D validation set by two different thresholds to study the influence of thresholds on results for different motion categories. For temporal related categories, the improvement depends on sub-classes. For example in Tab. 3.5, comparing with the baseline model under the occlusion category, the improvements are 2.6, 4.5, and

6.9 points in the *J&F* index for no occlusion, partial occlusion, and full occlusion, respectively. TCE-RVOS improves the accuracy more for the occluded than the non-occluded instances. In the vision-text aggregation categories, the improvements are more equal with improvements for no interaction of 3.6 points and interaction of 4.5 points in the *J&F* index, respectively. This observation shows that our proposed novel encoder and decoder structures improve the model performance by handling the temporal related challenges well. When looking at the vision-text aggregation, similar improvements across sub-classes can be observed. This is expected as our model does not make any improvements specific to the vision-text aggregation structure.

3.5 Summary

In this chapter, we inspect the current RVOS methods, and find out that their structures are all lacking in the understanding of the temporal aspect. In order to improve the temporal consistency for the RVOS framework, We proposed TCE-RVOS, an end-to-end referring video object segmentation approach. We first analyzed that the temporal context in previous work is weak. The data association between the same object in different frames limits the model performance in challenging scenarios. To overcome this weakness, our method enhances the temporal context learning in the model by a novel frame token fusion encoder with an instance query transformer decoder, and achieves state-of-the-art results with a clear margin (4.0 points in the *J&F* index on Ref-YouTube-VOS using ResNet50). We classified the Ref-YouTube-VOS validation dataset and A2D Dataset to investigate the performance in challenging scenarios. The ablation study shows that our improved structure achieves the goal of better handling the temporal context in challenging scenarios.

However, our method, together with previous SOTA methods, are all operate in an offline manner. All the video frames are required to be processed together. This limits the model to handle long/on going videos, which makes current methods not flexible for real-world application. In the next chapter, we focus on releasing such weakness by designing a novel Semi-online RVOS structure.

4

A Filtering Framework for Semi-online Referring Video Object Segmentation

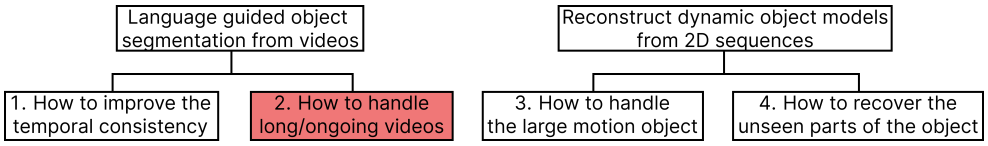


Figure 4.1: The overall tasks of the thesis, this chapter focuses on the task highlighted with red.

As we discussed, previous Referring Video Object Segmentation (RVOS) frameworks are mostly designed in offline manner to dominate the widely used benchmarks which only contain short video clips. In this chapter, we focus on the RVOS method for long/ongoing videos which cannot be handled by offline setups due to the memory limitation. Referring video object segmentation (RVOS) aims to extract objects from videos based on provided text narrations. Previous approaches typically employ offline processing to work on all video frames simultaneously, which is not always possible. Offline processing becomes also

ineffective for long videos, as directly cutting the video into short clips to fit memory limits would result in the loss of temporal consistency. To make RVOS more applicable to real-world ongoing/long video scenarios, we introduce a Filtering Framework for RVOS (FF-RVOS), the first model capable of operating in online, semi-online, and offline modes with just one training session. We redesign RVOS as a stochastic optimization problem, and leverage the filtering method to optimize object states across temporal sequences. Our method enhances temporal consistency when a long or ongoing video needs to be processed in shorter clip sequences due to memory limitations. FF-RVOS demonstrates superior performance compared to previous state-of-the-art methods on public benchmarks with clear improvements, especially when the video is too long to be processed at once. Our framework can also be embedded into different offline methods to boost the temporal consistency.

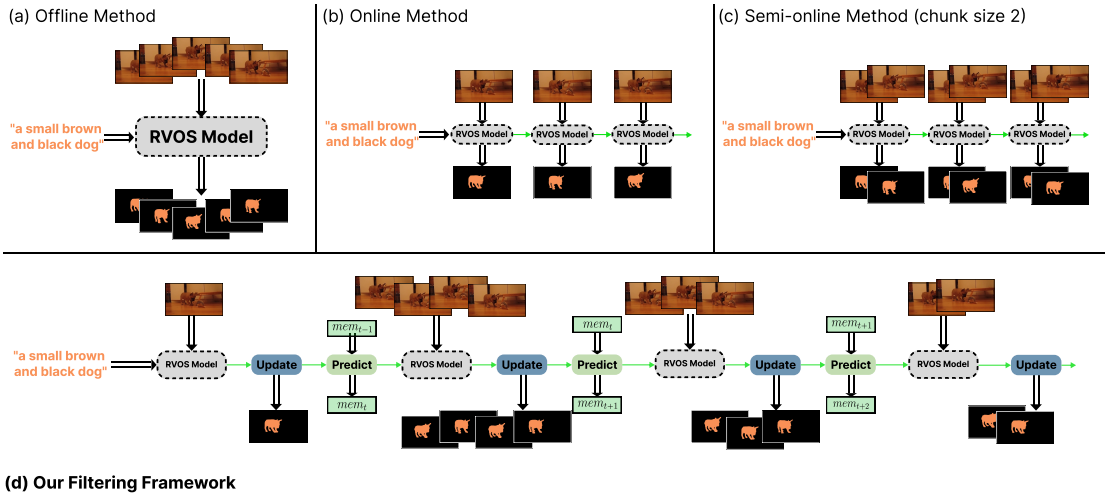
This work [37] was carried out jointly with Prof. Heiko Neumann and my supervisor Prof. Jochen Lang. I was responsible for most of the research tasks, including problem formulation, methodology, and experimental evaluation, while the coauthors provided valuable feedback and manuscript support. The work in this chapter has been presented at ACM Multimedia 2025 and the code is available at `https://github.com/haliphinx/FF-RVOS`

4.1 Introduction

The Artificial Intelligence community has shown heightened interest in video understanding [31, 136]. With the current powerful computing resources capable of processing multiple image frames simultaneously, various image-based tasks have transitioned into video-based domains, including but not limited to video classification [159], video object detection [160], video object segmentation [161] and video temporal segmentation [162]. In addition to image understanding, video-based tasks require comprehension of spatio-temporal information and potentially language cues. Thus, several multi-modal tasks have been established to integrate visual information with audio or video captions, such as audio-visual segmentation [163], video grounding [164], and referring video object segmentation [32]. The research in this chapter still centers on the Referring Video Object Segmentation (RVOS) task, which involves taking a video with a textual expression as input and segmenting the referred object across all video frames.

State-of-the-art (SOTA) RVOS methods are typically engineered as offline models [32–

34] to get a global temporal context. Such approaches are effective for current benchmarks [10, 35, 36] that only involve short videos (roughly 100 frames per video sample). However, in long/ongoing video and real-world applications, the feasibility of offline methods are limited due to GPU memory constraints. A straight forward strategy for an offline method to handle long/ongoing video is cutting the video into short clips and processing them independently. This semi-online like structure loses the temporal consistency, because of the weak inter-chunk connection. Conversely, online methods [29, 48] that process each frame independently offer the advantage of accommodating long videos and real-world scenarios, but they encounter challenges in effectively associating the referred object in the temporal domain. Semi-online methods keep both the advantages of online and offline methods. However, semi-online method usually has the limitations that the chunk size (window size) needs to be fixed for both training and inference, and the selection of the chunk size effects the final performance a lot.



(d) Our Filtering Framework

Figure 4.2: Structural comparison on current methods. (a) the offline method [34, 41, 42]. (b) the online method [29, 48]. (c) semi-online method shown with the chunk size of 2. (d) our filtering based semi-online method with arbitrary chunk size. The double arrow represents the input/output, the green arrows represent the meta value propagation. Notably, each chunk can have different size in our model.

To overcome above limitations, we propose a Filtering Framework for temporally consistent semi-online RVOS (FF-RVOS), the first model which is capable of seamlessly switching between online, semi-online (chunkwise), and offline modes without additional training.

The size of each chunk is not required to be identical in a single forward passing. This versatility enables our method to process long videos and ongoing videos with high accuracy while balancing the trade-off between accuracy, processing latency and memory usage across different devices with the same model weights. The structural comparison on different methods is shown in Fig. 4.2. Offline methods (Fig. 4.2(a)) process the entire video at once. Online methods (Fig. 4.2(b)) process video frames one by one, with meta feature propagation. Original semi-online methods (Fig. 4.2(c)) process a chunk with a fixed number of frames at a time and propagate meta features between chunks. Such propagation only passes features between two consecutive chunks, which usually leads to feature drift. Our framework (Fig. 4.2 (d)) is inspired by the concept of filtering methods (e.g., particle filter [165], and Kalman filter [166]). We redefine the RVOS task as a stochastic optimization problem of optimizing the hidden state of the referred object across the temporal dimension. Our method treats the segmentation mask as an observation of the hidden state at each time step, and optimizes the hidden state recurrently in a predict-then-update fashion.

To summarize, our main contributions are:

- We redesign the RVOS task as a stochastic optimization problem and utilize a filtering approach to propagate stabilized features for long/ongoing videos. Our proposed method demonstrates performance improvements over previous SOTA methods, especially when the video is too long to be processed at once.
- Our proposed framework is capable of processing videos in offline, online, and semi-online (chunkwise) modes without additional training. By simply modifying the chunk size, our method can effectively balance the trade-off between accuracy, processing time, and memory usage to fit in different devices.
- To address the limitations of current public benchmarks, which only involve short videos, We annotate a new public small dataset consisting of long videos (more than 1000 frames per video).
- Our designed framework can be easily embedded into other offline methods to boost the temporal consistency and convert them into semi-online mode for long/ongoing video processing.

4.2 Related Work

The overall background research is provided in Sec. 2.1, and some detailed discussions are made in Sec. 3.2. In this section, we focus on the more up-to-date researches after our previous TCE-RVOS work.

For the recent video object/instance segmentation researches, GenVIS [167] devises a generalized framework to handle the VIS task in a semi-online mode. DEVA [168] decouples the video object segmentation task into an image-level segmentation and a bi-directional temporal propagation tasks to handle the spatial and temporal domain separately. Segment Anything 2 (SAM2) [62] extends the original SAM to support VIS, which also achieves good accuracy. OneVOS [169] proposes an unified model to process feature extraction, matching, memory management, and object aggregation into one vision transformer. OpenVIS [170] uses InstCLIP to implement an open-vocabulary instance representation. TROY-VIS [171] simplifies the core components of the widely used VIS framework, including text encoder, feature enhancer, and instance decoder to let their method executable in real time.

Video segmentation segments multiple instances in the video, while RVOS only segments the instance described by the text expression. Although they share many features in common, RVOS focuses requires multi-modal aggregation.

Referring video object segmentation [35] (RVOS) aims to localize the object mentioned by a given text expression within the video both, in both spatial and temporal dimensions. In recent RVOS researches, TempCD [41] maintains a set of global referent tokens to improve temporal consistency. SgMg [42] utilizes a spectrum-guided multi-granularity optimization to avoid feature drift in the decoder. SOC [33] uses a video-level object cluster to enhance multi-modal feature matching. HTR [172] uses an inter-frame collaboration block to improve the temporal consistency between frames. TF^2 [173] leverages the idea of object tracking to propagate the features from keyframes to the rest. MUTR [174] employs additional global temporal interaction blocks to further aggregate temporal information. DsHmP [34] decouples the expression into static and dynamic parts to better understand the clue. Combing the RVOS task with large multi-modality models is another direction of research. VideoLISA [43] utilizes a powerful multimodal LLM, LLaVA, as the backbone to make use of the strength of the large language model. VISA [44] employs large vision-language model to jointly handle different tasks. SAMWISE [45] leverages the strength of large general models, like SAM2 [62], to form a powerful RVOS model. Although SAM2 processes the video in a semi-online setup, it requires a global memory pool to store all previous results, the size of which would increase unbounded when the video getting longer.

Our method can be converted between online, semi-online, and offline modes without retraining. The size of different chunk can be different, which increases the flexibility and balances the trade-off between accuracy and cost. We believe that the semi-online method is better suited for the long video RVOS task than the online method. The visual information in the video is sparse because consecutive frames are usually highly similar in appearance. Processing each frame independently as the online method would be inefficient. The ability of dynamically control the chunk size during the forward processing allows our method to better process long/ongoing videos.

4.3 Method

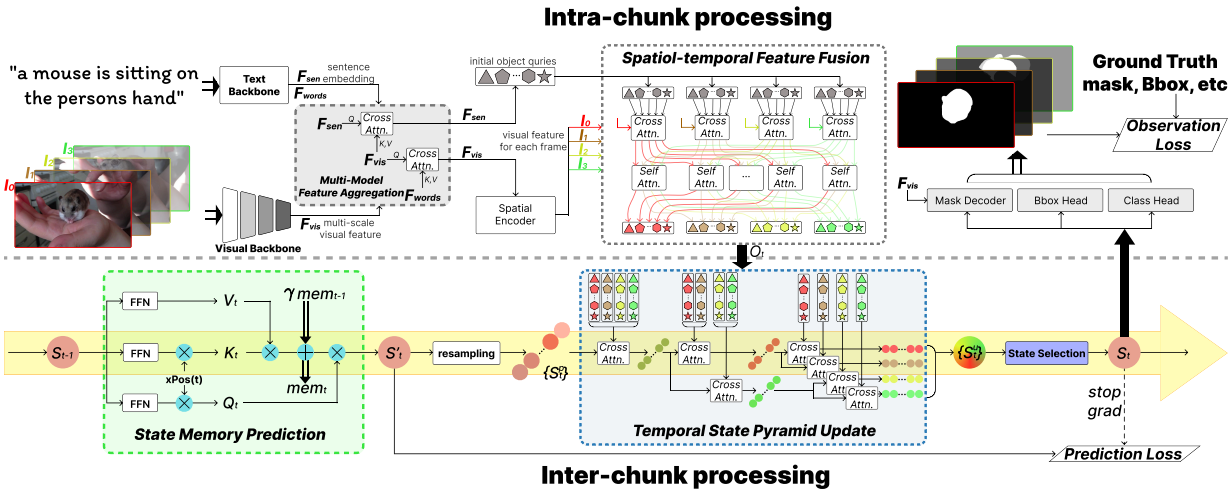


Figure 4.3: The structure of FF-RVOS. Intra-chunk processing handles the feature aggregation within each chunk, and inter-chunk processing propagates the features across chunks. The yellow arrow shows the filtering flow of the hidden state.

4.3.1 Overview

The goal is to segment the object referred to by a sentence within a given video and predict a sequence of binary masks corresponding to all frames in the video. In this section, we use the following definitions of key variables.

- $\mathbf{C}$: the hidden channel size; $\mathbf{h}, \mathbf{w}$: the height and width of the image frame; $\mathbf{t}$: the chunk index; $\mathbf{m_t}$: the number of frames in the chunk $\mathbf{t}$.
- $\mathbf{S_t}, \mathbf{S'_t}$: the actual and predicted hidden states of the referred object in the chunk $\mathbf{t}$.
- $\{\mathbf{S^p_t}\}, \{\mathbf{S^u_t}\}$: the set of possible states after the prediction and update block.
- $\mathbf{O_t}$. The observed features of the object for the chunk $\mathbf{t}$.

4.3.2 Task Setup as Filtering Problem

The objective of the filtering problem is to estimate the posterior density of the state given the observation. We define the RVOS task as a non-linear stochastic filtering problem, where the state of the referred object is hidden to be optimized, and the object mask serves as the observation of the hidden state. Compared with object masks, object hidden states provide a more robust connection across temporal space. This is motivated by the fact that the object hidden state describes the real state of the object, including but not limited to the object pose and shape. This hidden state is based on the entire scene and independent of the image frame and hence not affected by occlusions or visibility. The observation can be seen as the projection of the hidden state into the image space. Let $P(.)$ be a probability distribution, then the nonlinear filtering equation is

$$P(S_{t-1}|O_0, \ldots, O_{t-1}) \overset{prediction}{\longrightarrow} P(S_t|O_0, \ldots, O_{t-1}) \overset{update}{\longrightarrow} P(S_t|O_0, \ldots, O_t) \tag{4.1}$$

Since the probability densities cannot be calculated in closed form, we adopt the particle filter approach [165]. The densities are approximated by a set of particles sampled from the distributions, which can be expressed as:

$$P(S_t|O_0, \ldots, O_{t-1}) \approx \{S^p_t\}, P(S_t|O_0, \ldots, O_t) \approx \{S^u_t\} \tag{4.2}$$

To mimic sequential importance resampling, only the object state with the highest confidence (text-object similarity) will be resampled and propagated to the next chunk. Therefore, the recursion equation is transferred as:

$$S_{t-1} \overset{prediction}{\longrightarrow} S'_t \overset{resampling}{\longrightarrow} \{S^p_t\} \overset{update}{\longrightarrow} \{S^u_t\} \overset{selection}{\longrightarrow} S_t \tag{4.3}$$

Fig. 4.3 shows the detailed structure of the designed framework. The entire model is divided into two parts: intra-chunk and inter-chunk processing. The intra-chunk processing operates with multi-modal features in both spatial and temporal dimensions. The new observation O_t is used to update the hidden state. The inter-chunk block serves as the hidden state filtering pipeline. The hidden state propagation flow (the yellow arrow in Fig. 4.3) follows Eq. (4.3). Unlike previous online methods [10, 29, 141], which focus solely on optimizing the spatial mask, our framework also optimizes the hidden state S of the object. This enhances the stability of inter-chunk propagation. Our method can convert between online, semi-online, and offline modes using the same weight, and the chunk sizes do not need to be uniform for each chunk.

4.3.3 Intra-chunk Processing

Our method focuses on improving the temporal consistency when the offline method cannot handle the entire video at once and requires splitting the video into chunks for separate processing. A robust intra-chunk understanding structure ensures reliable observations, thereby enhancing the accuracy of state updates. Inspired by the previous SOTA offline method MUTR [174], the intra-chunk processing block is designed as illustrated at the top of Fig. 4.3. Pretrained visual and textural backbones are utilized to extract initial visual and textual features. A multi-modal feature aggregation block is employed to fuse the features across different modalities. The textural features are used to initialize a set of object queries, while the visual features are further processed by a deformable transformer-based spatial encoder. We incorporate a spatial-temporal feature fusion block from TCE-RVOS [32] to enhance temporal feature aggregation within the chunk. The resulting output queries are served as the observation O_t of the current chunk, which acts as input to the filtering pipeline. The updated state queries are subsequently used for output generation by feeding them into various output decoder heads.

RVOS Baseline. The designed intra-chunk processing block achieves SOTA performance. To fully demonstrate the generalization capability of our inter-chunk processing structure, two recent offline RVOS frameworks (TCE-RVOS, and SOC [33]) are evaluated by directly replacing our intra-chunk block. The experimental results show that our filtering framework can be seamlessly integrated into other query-based offline structures to enhance their ability to process long videos.

4.3.4 Inter-chunk Processing

The inter-chunk processing block (bottom of Fig. 4.3) propagates the state memory through time and forms the filtering framework (Eq. (4.3)). In order to allow the model to seamlessly switch between online, offline, and semi-online modes, the inter-chunk processing block should be able to propagate the state between consecutive chunks with arbitrary chunk sizes. At the same time, performance should not be affected by differences in chunk sizes between training and inference setups.

State Memory Prediction. Given the previous state S_{t-1}, the model predicts and samples a set of potential states $\{S_t^p\}$ for the current chunk, which propagates the object state between chunks.

Inspired by the state space modeling structure RetNet [175], a state memory mem_t, is utilized to enrich the temporal feature across chunks. The shape of the state memory block depends only on the hidden channel sizes, so that each chunk can be of different size. We propose our prediction block in Fig. 4.3 (sub-block in the green box at the bottom). The object state for the previous chunk S_{t-1} is passed into this block. Three independent linear feed-forward networks (FFN) are employed to generate query Q_t, key K_t, and value V_t from S_{t-1}, respectively. During the training phase, only a subset of frames is sampled from each video, however, during inference, all frames are processed. To account for this, xPos [176], a relative position embedding, is used to represent relative positions between frames rather than absolute positions. γ is a fixed memory decay factor ($\gamma \in (0, 1)$) to ensure more focus in memory on recent chunks. The online format of memory updates and the prediction of S'_t are calculated as:

$$mem_t = \gamma mem_{t-1} + (K_t\ xPos(t))^T V_t, \tag{4.4}$$

$$S'_t = (Q_t\ xPos(t))mem_t \tag{4.5}$$

The initial state is generated from the textural features, and the memory is initialized as all zero. Thus, Eq. (4.4) and Eq. (4.5) can be converted into offline format as:

$$\begin{aligned} M_{ij} &= \begin{cases} \gamma^{i-j}, & i \geq j \\ 0, & i < j \end{cases} \\ S'_t &= (Q_t\ xPos(t))(K_t\ xPos(t))^T \odot MV_t \end{aligned} \tag{4.6}$$

The matrix M (with elements M_{ij}) is the combination of the attention mask and the memory decay factor. $\odot$ represents the element-wise multiplication. In particular, the different formats share the same trainable weight. Therefore, the block can process different chunk sizes without retraining. Because consecutive chunks may have different chunk sizes, only the latest state with the highest confidence will be passed to the resampling block in the sequential importance resampling. A repeat function combined with a learnable positional embedding acts as a sampling function that approximates the probability $P(S_t|O_{0:t-1})$ with a set of n samples $\{S_t^p\}$. This can be described as $\{S_t^p\} = pos\_embed(repeat(S_t', n))$.

Temporal State Pyramid Update. The purpose of this component is to update the predicted object states using the observation of the current chunk. The observation O_t for the current chunk contains features from multiple frames. Directly duplicating the predicted state to match the chunk size is suboptimal, especially for large chunks, as the same object may exhibit significant variations over a longer time span. To address this, a temporal state pyramid update structure is designed, incorporating L cross attention layers to refine the predicted state, progressively transforming chunkwise global features into frame-specific features. The structure of 3 layers ($L = 3$) is illustrated in Fig. 4.3 (sub-block in the blue box at the bottom). The predicted state is gradually duplicated and O_t is reshaped to fit the number of states. A duplication factor d is defined to control the repetition ratio, which is determined by the number of layers L, and the chunk size m_t (Eq. (4.7)).

$$d = \arg\min_{d, d\in\mathbb{Z}^+}(d^{(L-1)} \geq m_t) \tag{4.7}$$

The operation for $layer_i$ ($i \in (1, \ldots, L)$), where the initial $layer_0 = \{S_t^p\}, \in \mathbb{R}^{1\times n\times C}$, can be described as shown in Eq. (4.8). The granularity of the temporal feature for each layer is denoted as $g_i = d^{(i-1)}$. When $g_i = 1$, the entire chunk is processed together, whereas when $g_i = m_t$, each frame is processed separately.

$$\begin{aligned} q_i &= repeat(layer_{i-1}, g_i), \in \mathbb{R}^{g_i\times n\times C}, \\ k_i = v_i &= reshape(O_t, g_i), \in \mathbb{R}^{g_i\times (m_t g_i^{-1} n)\times C} \\ layer_i &= cross\_atten(q_i, k_i, v_i) \end{aligned} \tag{4.8}$$

$repeat(.)$ is an interleaving repeat operation applied on the first dimension. $reshape(.)$ function modifies the first two dimensions. Together, these operations ensure that each layer satisfies the required granularity. This proposed structure extracts frame-specific

features without losing the chunkwise global information.

State Selection. Our model produces three outputs from each object state $\hat{S}_t \in \{S_t^u\}$: the object mask $mask_t \in \mathbb{R}^{m_t \times h \times w}$, the object bounding box $bbox_t \in \mathbb{R}^{m_t \times 4}$, and the confidence score $conf_t \in \mathbb{R}^{m_t \times 1}$. $bbox_t$ and $conf_t$ are generated by two simple multi-layer perceptrons (MLP), while the mask is predicted by a conditional convolutional filter [151] generated by the object state. Only the object state with the lowest observation loss $\mathcal{L}_{ob}$ is selected as the final state S_t. Such selection procedure can be described as:

$$\begin{aligned} \mathcal{L}_{ob}(\hat{S}_t) &= \sum_i \lambda_i \mathcal{L}_i(\hat{S}_t), i \in \{bbox, conf, mask\} \\ S_t &= \arg \min_{\hat{S}_t \in \{S_t^u\}} \mathcal{L}_{ob}(\hat{S}_t) \end{aligned} \tag{4.9}$$

The bounding box similarity score $\mathcal{L}_{bbox}$ comprises the L1 loss and the generalized IoU (GIoU) loss [152]. $\mathcal{L}_{conf}$ is a focal loss designed to supervise the text-object similarity. $\mathcal{L}_{mask}$ consists of the DICE loss and per-pixel focal loss [153]. λ_i represents the corresponding coefficients to balance the different similarity scores. A lower $\mathcal{L}_{ob}$ indicates an object state sample with a higher probability to be the real state. During inference, the object state with the highest confidence $conf_t$ is selected as the final state.

Loss Functions. As described in Eq. (4.1), the entire model contains two main stages: a prediction and an update stage. The loss is designed to simultaneously constrain these components. The prediction loss focuses on a realistic object state prediction, while the update loss evaluates the accuracy of observation. The detailed loss function is:

$$\mathcal{L}_{overall} = \lambda_{pred}\, \mathcal{L}_{sim}(sg(S_t), S_t') + \mathcal{L}_{ob}(S_t), \tag{4.10}$$

The prediction loss is formulated as a contrastive loss $\mathcal{L}_{sim}$ that constrains the cosine similarity between the predicted state S_t' and the updated state S_t. $sg(.)$ indicates a stop-gradient operation, so that only the predicted state would be optimized. Such loss encourages the prediction block to predict the object state as close to the final state as possible. The update loss is the same as the $\mathcal{L}_{ob}$, which directly constrains the quality of the generated object masks. The overall loss in training the model is the combination of the above two losses.

4.3.5 Embedding into Other Offline Structures

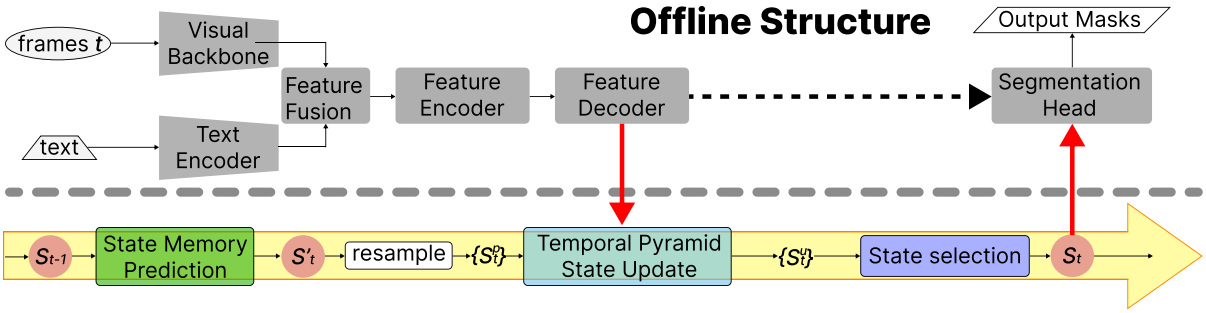


Figure 4.4: Structure of embedding the FF into other offline structures. The upper half shows the original offline structure. To embed our framework, the dashed arrow is removed, and two red arrows are added to bridge the two parts.

As mentioned above, our filtering framework can be embedded into other query-based offline structures to enhance temporal consistency and enable conversion into semi-online methods. Fig. 4.4 illustrates how other models can be integrated into our framework. Most previous offline RVOS methods [32–34, 174] can be generalized to the structure depicted in the upper half of Fig. 4.4. The main differences between these methods lie in the specific design of each functional block, while their overall pipelines remain similar. In our approach, the features from the decoder are passed into the FF pipeline, and the updated state is fed back to generate the final mask output. As shown in the diagram, the original connection between the decoder and the segmentation head (dashed arrow) is severed, and two red arrows are added to connect the offline structure to our FF.

4.4 Experimental Evaluation

4.4.1 Experiments Setup

Datasets. The experiments are carried out on four benchmarks. Refer-YouTube-VOS [10], Refer-Davis$_{17}$ [48] are two popular benchmarks that are widely used in the field of RVOS. However, they only contain short videos. MeViS [36] is a recent dataset that prioritizes longer and more intricate testcases. We also annotated a novel small dataset with four

long videos containing thousands of frames per video sample for evaluation on long videos. One-fiftieth of the frames are manually labeled. Comparisons on Refer-YouTube-VOS and MeViS are presented in this section. The evaluation metrics consist of the region similarity Jaccard index ($\mathcal{J}$), the contour accuracy $F1$ score ($\mathcal{F}$), and the average $\mathcal{J}\&\mathcal{F}$.

Implementation Details. All the training procedures with our approach are conducted using 4 Nvidia A100 GPUs with 40GB memory. For a fair comparison, we adopt hyperparameter settings similar to [32, 174]. The coefficients for the loss function are $\lambda_{bbox} = 2$, $\lambda_{conf} = 2$, $\lambda_{mask} = 5$, and $\lambda_{pred} = 2$. Various backbones are evaluated in the experiments. The outputs from the last three layers of the respective backbone are used to provide multi-scale features with spatial down-sampling rates of $\{8, 16, 32\}$. The spatial encoder and the spatial-temporal feature fusion block are concatenated $N = 4$ times. The temporal state pyramid update block is concatenated $N = 3$ times. The initial learning rate for the visual backbone and the main model is $2e - 5$ and $1e - 4$. The model is fine-tuned on different benchmarks for 6 epochs with the learning rate being reduced at the third and fifth epochs. Furthermore, the model is pre-trained on the image-based referring object segmentation dataset Ref-COCO [63]. The object state sampling rate is set to $n = 5$ enabling a fair comparison with previous state-of-the-art (SOTA) methods. Due to limited compute resources, our model is only trained with 4 GPUs instead of 8 as in previous methods [13, 21, 29, 33, 34, 174]. However, our method still outperforms previous SOTA methods. The batch size is set to 1 for all benchmarks. During training, the input video clip is divided into three chunks of arbitrary sizes, as our model does not require identical chunk sizes for every chunk. For the integration of SOC [33] and TCE-RVOS [32], we first freeze the baseline model and train only our framework for 4 epochs, then finetune the whole structure together for 2 epochs.

4.4.2 Comparison Results

Refer-YouTube-VOS. Fig. 4.5 presents the comparison on the Refer-YouTube-VOS dataset. The figure on the left displays the accuracy of our FF-RVOS compared to previous methods, using various backbones. FF-RVOS achieves SOTA performance with the highest $J\&F$ accuracy. Compared to the previous SOTA methods MUTR and SAMWISE, our method demonstrates greater stability, as indicated by flatter curves across different chunk sizes. In contrast, the accuracy of MUTR drops significantly as chunk size is reduced. The graph on the right of Fig. 4.5 highlights the generalization of the proposed framework. By integrating our FF into off-the-shelf offline RVOS methods, both the overall accuracy and

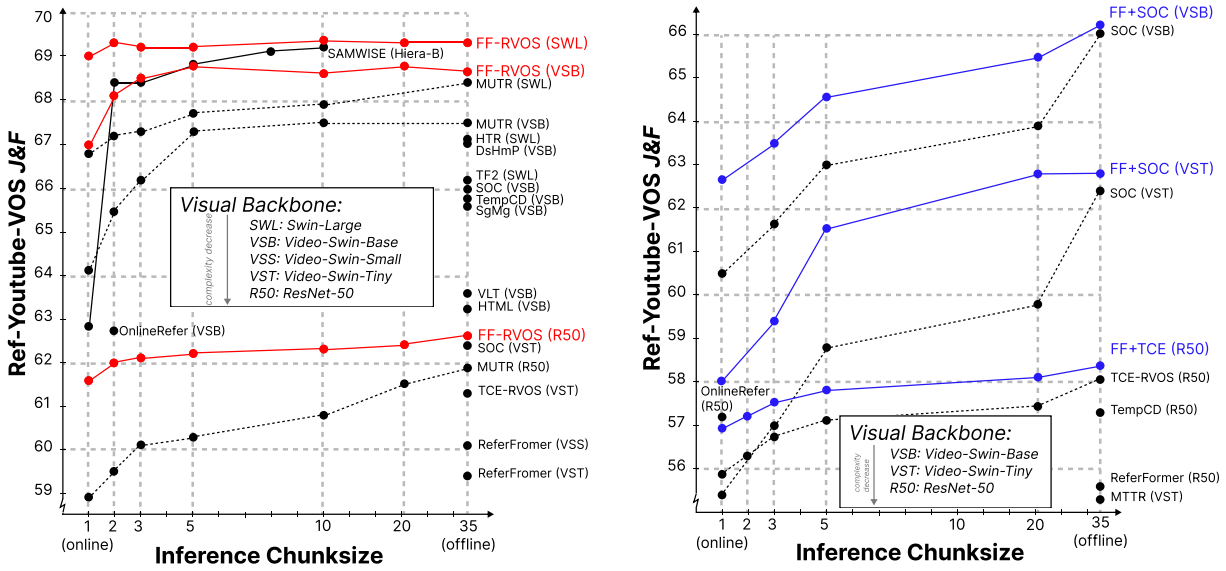


Figure 4.5: Comparison result on the Ref-YouTube-VOS dataset [10]. The left graph shows the comparison between our FF-RVOS and previous SOTA methods with different backbones. The right graph shows the improvement of integrating our FF into previous offline methods. The dashed curve indicates methods not originally designed for semi-online mode, for which we manually cut the video into clips only for testing.

stability are improved. In particular, with image-based backbones (e.g., Swin Transformer and ResNet-50) the accuracy curve remains flat across all chunk sizes. Performance of video-based backbones slightly declines when the chunk size approaches 1. This occurs because the video-based backbone (Video Swin Transformer) suffers from the absence of sufficient temporal features. Although the proposed FF-RVOS achieves SOTA accuracy, the primary goal of this work is not to increase accuracy in an offline setup.

Tab. 4.1 presents the quantitative comparison between our approach and previous SOTA methods, evaluated using different backbones and processing modes. The results of our methods are highlighted, and the best results are in bold. The comparison result demonstrates that our proposed method leads to large increases in performance over the previous SOTA. Furthermore, integrating our framework into other offline methods also improves their performance.

MeViS. Although Ref-YouTube-VOS is widely used for the RVOS task, it has several limitations. First, the video length is short with a maximum of 35 frames per video

Figure 4.6: Qualitative results from Refer-YouTube-VOS (top), and MeViS (middle), and Long Video dataset (bottom). Various challenging scenes are presented, including occlusion, presence, and crowding.

sample. Second, the expressions primarily describe the attributes of the object rather than its behavior, reducing the challenges in temporal understanding. To address these limitations, the MeViS dataset has been introduced, featuring longer videos and more complex expressions. Since MeViS is a new dataset, many previous methods have not been evaluated on it. Therefore, we evaluate models trained on Ref-YouTube-VOS directly on MeViS with various chunk sizes. Tab. 4.3 shows the comparison of different methods using different chunk sizes. The results of our proposed method are highlighted. The result demonstrates that our FF framework improves performance while handling challenging videos with chunks.

Tab. 4.2 presents the comparisons on the MeViS dataset. The table is split into three parts. The upper part shows the zero-shot evaluation with the models trained on Ref-YouTube-VOS, and directly evaluated on MeViS. This zero-shot evaluation avoids unfair advantages for some models, since different models usually require different training strategies for different datasets, and it can also evaluate the ability of models to generalize. The results illustrate that integrating the proposed FF into other offline models can significantly improve accuracy in a dataset with long videos and motion expressions. The middle part

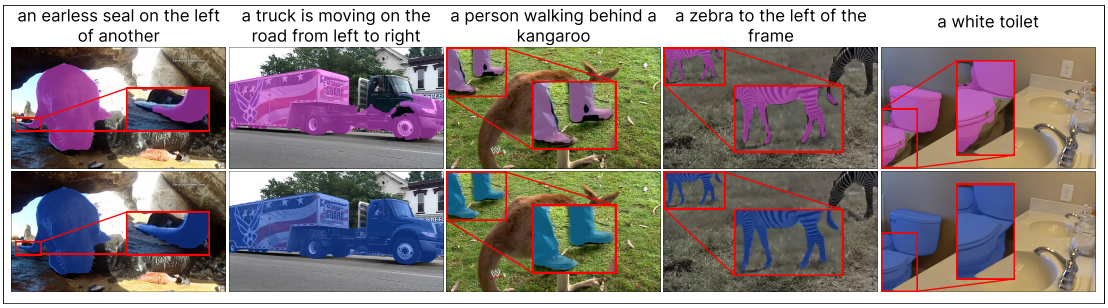


Figure 4.7: Qualitative results generated by directly decoding the predicted state S'_t (top row) and the results decoded from the final updated state S_t (bottom row). The prediction block can already generate a good overall mask, and the update block refines the detail of the segmentation masks.

of Tab. 4.2 shows the models trained on MeViS. FF-RVOS achieves the highest accuracy demonstrating the strength of our approach. The bottom part shows the recent research using large visual language models pretrained on other large scale datasets.

Refer-DAVIS$_{17}$. Tab. 4.4 presents a comparative analysis between our proposed FF-RVOS and SOTA methods across the Refer-DAVIS$_{17}$ benchmarks. Different backbones are evaluated to demonstrate the generalization. Our proposed method is marked by a dark background, with the best results highlighted in bold. FF-RVOS, operating as a semi-online method, achieves SOTA results compared with other methods with similar backbones, regardless of using online or offline modes. SOC [33] is another state-of-the-art offline method which utilizes contrastive learning for better multi-modal feature matching. Our framework can be embedded into other offline methods to improve the temporal consistency across different chunks. So we directly use SOC as the baseline to show the improvement by our framework. By adding our framework, the previous SOTA offline method can be converted into a semi-online method with increased accuracy.

Long Video Benchmarks. Despite MeViS being the most intricate RVOS dataset available at the time of our research, the video samples (13 seconds per video on average) are still shorter than what can be expected in real applications. To comprehensively show the capabilities of our method, we manually annotated four long videos sourced from [179, 180] with object masks for every fiftieth frame and with expressions. The average number of frames is 2022. We selected three SOTA offline methods, TCE-RVOS, SOC, MUTR, the semi-online method SAMWISE and the online method OnlineRefer as competitors. Tab. 4.5 presents the results on the long video dataset. The main challenge is segmen-

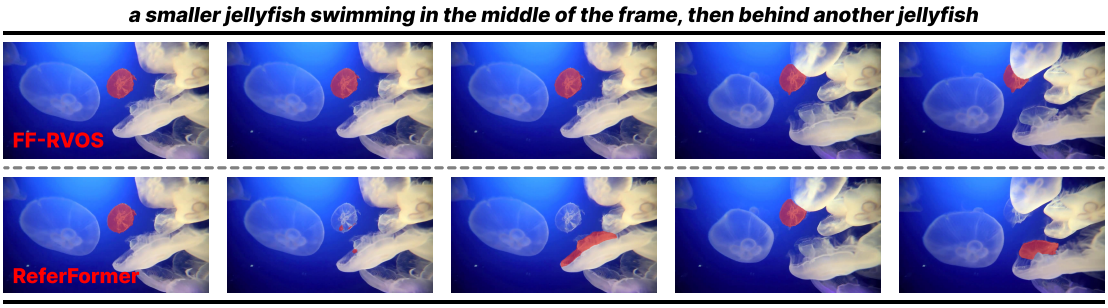
tation consistency across long videos. A ResNet-50 backbone is used for all competitors except SAMWISE which uses Hiera-B. The offline method performs less effectively than the semi-online methods when it has to cut the video into clips for inference due to limited memory. Our method achieves substantially better results than the competitors, clearly demonstrating that our proposed architecture can handle long videos better.

Qualitative Result. Fig. 4.6 shows the visualized results from the Ref-YouTube-VOS dataset (top) and the MeViS dataset (bottom). The color of each expression corresponds to the color of its mask. The results show that the proposed method effectively handles various challenging scenes. Extra qualitative results are provided in Fig. 4.8 and Fig. 4.9. Fig. 4.8 shows the comparison between our proposed FF-RVOS and various state-of-the-art online/offline methods. The visualization demonstrates that our method provides a better segmentation consistency over time. Fig. 4.9 provides the result on FF-RVOS by using different sizes of chunks for inference. We can see that by using a different chunk size, the segmentation quality remains similar with some changes in details. This supports the goal that our model can generate results of similar quality using different chunk sizes.

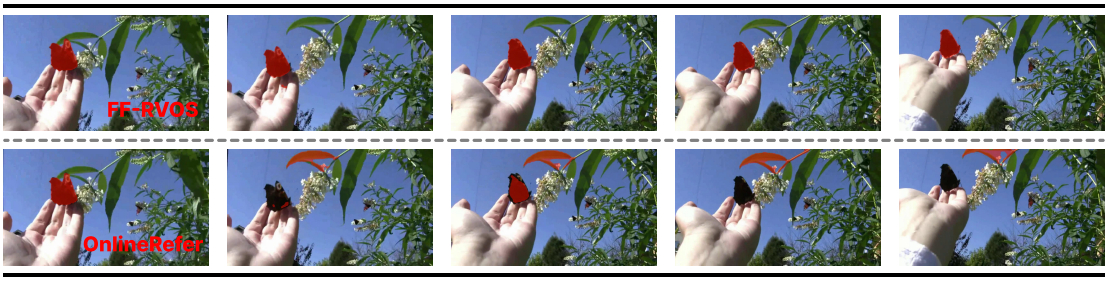
4.4.3 Ablation Study

A series of extensive ablation studies are conducted to investigate the effects of key components in our method.

Inference Costs. The inference costs are key metrics to evaluate the practicality of a method. We provide an ablation study of the inference costs, including Frames Per Second (FPS), inference latency, and peak GPU memory usage, conducted on a consumer device (RTX 3090). Tab. 4.6 shows a comparison of different chunk sizes. The results are generated using the Refer-YouTube-VOS dataset. FPS measures the overall inference speed. However, an offline method with high FPS does not necessarily indicate real-time operation, since the result will be output when all the video frames are processed. We also compare the average latency (the time between inputting a frame and receiving the output) for different chunk sizes. Peak GPU memory usage represents the maximum GPU memory required during inference, which is more critical than average GPU memory usage when selecting a suitable device. This evaluation demonstrates that our method can be adjusted to balance performance and cost, which enables our model to be deployed on various devices without additional training. Tab. 4.7 shows the comparison of inference costs. All the competitors are using V-Swin-Base as backbone except SAMWISE which



(a) Example results of FF-RVOS (ours) and ReferFormer, shown in red masks.



(b) Example results of FF-RVOS (ours) and OnlineRefer, shown in red masks.

(c) Example results of FF-RVOS (ours) and TCE-RVOS, shown in purple masks.

Figure 4.8: Qualitative comparison between results of the purposed FF-RVOS (ours) versus ReferFormer (top sequence), OnlineRefer (middle sequence), and TCE-RVOS (bottom sequence) on the Ref-YouTube-RVOS dataset [10].

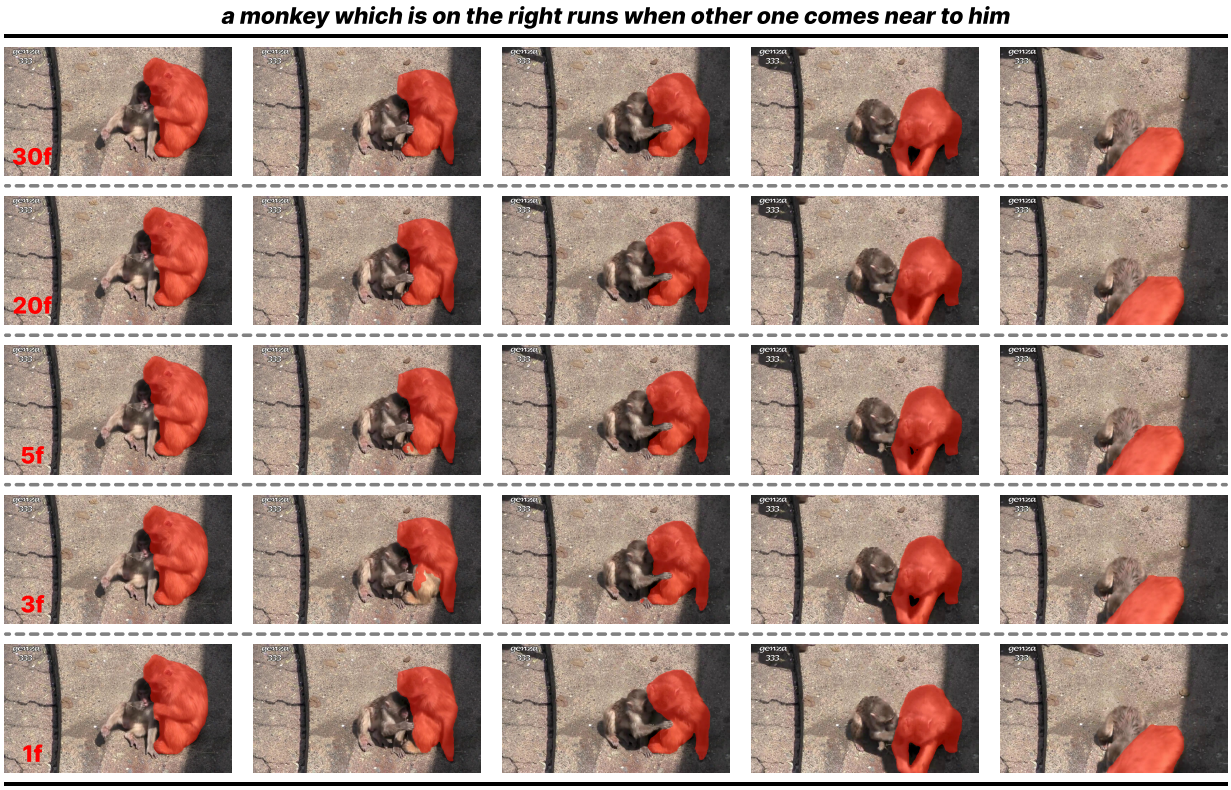


Figure 4.9: Qualitative results from Refer-YouTube-VOS generated by FF-RVOS (ours) using different chunk sizes. The chunk sizes are 30 frames, 20 frames, 5 frames, 3 frames, and 1 frame from the top to the bottom rows, respectively. The prediction results are shown in red masks.

uses Hiera-B. The results demonstrate that our method balances accuracy and efficiency well.

Importance of the prediction loss. The overall loss is composed of a prediction loss and an observation loss. The observation loss constrains the output masks which directly corresponds to the evaluation metrics. The prediction loss is a self-supervised contrastive loss, which does not directly impact object mask generation. We conducted an ablation study to investigate the impact of using the prediction loss. The results (Tab. 4.8) demonstrate that the prediction loss constrains the prediction block to output a more accurate object hidden state improving model performance. The results are obtained with a ResNet-50 backbone and a chunk size of 5. Fig. 4.7 illustrates qualitative results of the predicted state S'_t and the final updated state S_t. The first row shows the results generated by directly feeding the predicted state S'_t into the decoder heads. The second row shows the results of decoding the updated state S_t, which is also the final output of our model. The predicted state S'_t is generated only based on previous chunks, without access to images from the current chunk. However, S'_t can already generate a good overall mask, missing only some fine details. The proposed temporal state pyramid update block refines the state using information from the current chunk, resulting in a detailed mask.

Ablation study for the Temporal State Pyramid Update. Since the predicted state may differ in size from the current chunk size, a strategy to align the sizes of states is necessary for the subsequent decoder heads. A naive approach is to directly duplicate the predicted state to match the chunk size. However, this method risks losing fine-grained details for each frame. We address this with our structure of the temporal state pyramid update block that gradually duplicates the state size, extracting features from the entire chunk for each frame step-by-step. Tab. 4.9 presents the ablation study of this proposed block, where different chunk sizes are evaluated. In Tab. 4.9, a ✓ indicates the use of the temporal pyramid block, while a × represents the naive approach. The results show that our proposed state update block significantly improves accuracy by a clear margin.

Impact of the chunk size. Our FF-RVOS can switch between online, semi-online, and offline modes without additional training. We evaluate the performance when using different numbers of frames for each chunk in terms of accuracy and inference time (Fig. 4.10). In general, the accuracy and processing time increase as the chunk size increases. However, the accuracy stabilizes when the chunk size exceeds 5 frames, while the processing time keeps increasing. This ablation study shows the flexibility and efficiency of our approach.

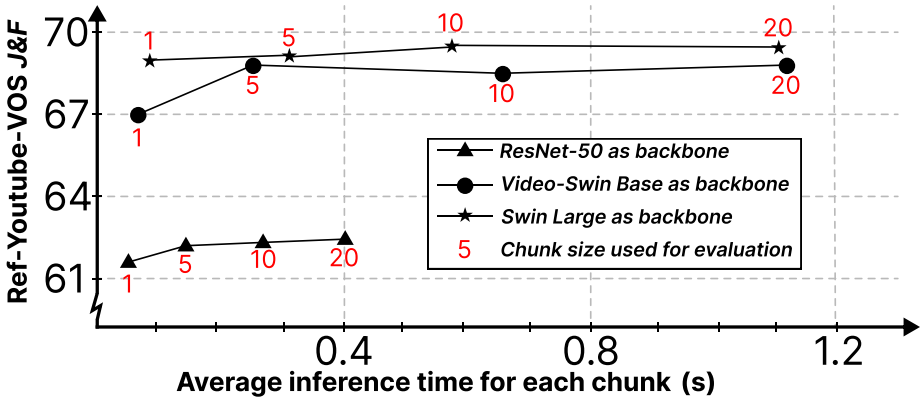


Figure 4.10: Ablation study on the impact of the chunk size. The inference time (x-axis) and $\mathcal{J}\&\mathcal{F}$ accuracy (y-axis) are compared by using different chunk sizes.

4.5 Summary

In this chapter, we focus on the problem that using RVOS for long and streaming videos, which cannot be well handled by previous offline frameworks. We introduced FF-RVOS, a unified RVOS framework, for processing long videos and ongoing videos. Our method is capable to be converted between online, semi-online, and offline modes with robust performance without necessitating additional training. In addition, our framework is feasible to be embedded into other offline methods to boost the temporal consistency when the video need to be cut into short clips for memory limitation. We redefined the RVOS task as a stochastic optimization problem and devised our novel method in a filtering based framework. Our proposed method outperforms previous state-of-the-art methods across widely used public benchmarks with a clear increment. Several ablation studies further validate that our framework is feasible to balance the trade-off between the performance and the cost by controlling the chunk size. However, there remains room for improvement. Long videos and ongoing videos often encounter the challenge of expression ambiguity. The expression may only be suitable for a portion of the video, as the object keeps changing appearance over time. Our method does not prioritize optimizing the language feature representation, which could be a potential area for future work.

From the next chapter, we start to discuss the tasks related to another topic, dynamic object 4D reconstruction.

Method	Backbone	Mode	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
VideoLISA [43]	Large Model	Offline	63.7	61.7	65.7
SAMWISE [45]		Semi	69.2	67.8	70.6
URVOS [10]	ResNet-50	online	47.2	45.2	49.1
OnlineRefer [29]		online	57.3	55.6	58.9
TCE-RVOS [32]		Offline	58.1	57.1	59.0
FF+TCE		Semi	58.4	57.3	59.5
MUTR [174]		Offline	61.9	60.4	63.4
FF-RVOS (ours)		Semi	**62.6**	**61.0**	**64.1**
ReferFormer [13]	Video-Swin Base	Offline	62.9	61.3	64.6
OnlineRefer [29]		Semi	62.9	61.0	64.7
HTML [177]		Offline	63.4	61.5	65.2
VLT [58]		Offline	63.8	61.9	65.6
SgMg [42]		Offline	65.7	63.9	67.4
TempCD [41]		Offline	65.8	63.6	68.0
SOC [33]		Offline	66.0	64.1	67.9
DEVA [168]		Offline	66.0	-	-
FF+SOC		Semi	66.2	64.0	68.3
HTR [172]		Offline	66.7	64.9	68.6
DsHmP [34]		Offline	67.1	65.0	69.1
MUTR [174]		Offline	67.5	65.4	69.6
FF-RVOS (ours)		Semi	**68.8**	**66.7**	**70.9**
OnlineRefer [29]	Swin-Large	Online	63.5	61.6	65.5
TF^2 [173]		Offline	66.2	64.0	68.4
HTR [172]		Offline	67.1	65.3	68.9
MUTR [174]		Offline	68.4	66.4	70.4
FF-RVOS (ours)		Semi	**69.4**	**67.0**	**71.8**

Table 4.1: Comparison for offline performance on Refer-Youtube-VOS benchmark. Our approaches are highlighted, and the best results are in **bold**. This table can not show the main contribution of our work since our method does not focus on the offline setup.

Method	Backbone	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Zero-Shot				
TCE-RVOS [32]	ResNet-50	32.7	30.6	34.9
FF+TCE	ResNet-50	37.8	35.6	40.0
SOC [33]	Video-Swin-B	40.9	37.6	44.1
FF+SOC	Video-Swin-B	42.9	40.0	45.9
MUTR [174]	Swin-Large	**43.4**	-	-
Fine-Tuned				
URVOS [10]	ResNet-50	27.8	25.7	29.9
LBDT [178]	ResNet-50	29.3	27.8	30.8
MTTR [21]	Video-Swin-T	30.0	28.8	31.2
ReferFormer [13]	Video-Swin-B	31.0	29.8	32.2
OnlineRefer [29]	Swin-Large	32.3	31.5	33.1
VLT-TC [58]	Darknet-56	35.5	33.6	37.3
LMPM [36]	Video-Swin-T	37.2	34.2	40.2
DsHmP [34]	Swin-Tiny	46.4	43.0	49.8
FF-RVOS (ours)	Swin-Large	**47.3**	**44.0**	**50.6**
Large Visual Language Model				
VISA [44]	Chat-UniVi-13B	41.8	47.1	44.5
VideoLISA [43]	LLaVA-Phi-3-V	44.4	41.3	47.6
SAMWISE [45]	Segment Anything 2	**49.5**	**46.6**	**52.4**

Table 4.2: Comparison for offline performance on the MeViS dataset [36]. The upper part shows the models trained on Refer-Youtube-VOS, and zero-shot evaluated on MeViS. The middle half shows the models fine-tuned on MeViS. The bottom part shows the method using pretrained large models. This table can not show the main contribution of our work since our method does not focus on the offline setup.

	Chunk Size			
	1 (Online)	5	10	20
TCE-RVOS	37.2	37.7	38.2	38.9
FF+TCE	38.6(+1.4)	39.8(+1.9)	40.3(+2.1)	40.7(+1.8)
SOC	36.4	39.9	40.3	40.9
FF+SOC	38.4(+2.0)	41.2(+1.3)	42.1(+1.8)	42.9(+2.0)
MUTR	41.1	41.4	41.3	41.3
FF-RVOS	41.6(+0.5)	42.5(+0.9)	42.8(+1.5)	43.1(+1.8)

Table 4.3: Zero-shot comparison on the MeViS with different chunk sizes. All competitors are using the V-Swin Base as the visual backbone.

Method	Backbone	Mode	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
SOC	VSwin-T	Offline	63.5	60.2	66.7
FF+SOC		Semi	63.6	60.0	67.1
ReferFormer	VSwin-B	Offline	61.1	58.1	64.1
VLT		Offline	61.6	58.9	64.3
HTML		Offline	62.1	59.2	65.1
OnlineRefer		Semi	62.4	59.1	65.6
SgMg		Offline	63.3	60.6	66.0
SOC		Offline	64.2	61.0	67.4
FF+SOC		Semi	64.6	60.8	68.4
TempCD		Offline	64.6	61.6	67.6
DsHmp		Offline	64.9	61.7	68.1
DEVA		Offline	66.3	-	-
MUTR		Offline	66.4	62.8	70.0
FF-RVOS (ours)		Semi	**67.1**	**63.6**	**70.5**
ReferFormer	Swin-L	Offline	60.5	57.6	63.4
OnlineRefer		Online	64.8	61.6	67.7
MUTR		Offline	68.0	64.8	71.3
FF-RVOS (ours)		Semi	**68.2**	**65.0**	**71.4**

Table 4.4: Comparison for offline performance on Refer-DAVIS$_{17}$ benchmark. Our approaches are highlighted. This table can not show the main contribution of our work since our method does not focus on the offline setup.

Method	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
SOC [33]	80.7	80.1	81.3
TCE-RVOS [32]	80.8	79.8	81.8
OnlineRefer [29]	82.2	81.2	83.2
MUTR [174]	82.2	81.7	82.8
SAMWISE [45]	83.0	83.8	82.0
FF-RVOS (ours)	**85.2**	**83.9**	**86.6**

Table 4.5: Comparison on the Long video dataset. The best results are in bold. All the models are trained on Refer-Youtube-VOS, and zero-shot evaluated on the long video dataset.

	Backbone	Chunk Size			
		1	5	10	20
FPS↑	ResNet-50	15.9	30.4	33.3	34.4
Latency↓		0.063	0.156	0.285	0.519
GPU↓		2.64	5.21	8.39	14.7
FPS↑	Video-Swin Base	12.4	13.9	14.4	15.8
Latency↓		0.081	0.264	0.669	1.126
GPU↓		2.95	5.46	10.47	21.20
FPS↑	Swin-Large	10.2	15.2	16.0	16.2
Latency↓		0.098	0.313	0.589	1.102
GPU↓		3.37	6.01	9.32	17.25

Table 4.6: The comparison of the inference speed with different chunk sizes and backbones. The Frames Per Second (FPS↑), the Latency↓ (in seconds) between inputting a frame and getting the result, and the peak GPU↓ memory usage (in GB) are evaluated.

Method	Chunk Size	FPS↑	Latency↓	GPU↓	$\mathcal{J}\&\mathcal{F}$ ↑
OnlineRefer	2	16.15	0.917	4.8	62.9
FF-RVOS		16.13	0.131	3.69	68.1
SAMWISE*	5	3.78	3.047	13.61	68.8
FF-RVOS		13.9	0.264	5.46	68.8
SOC	20	26.8	0.987	22.98	64.0
MUTR		16.3	1.058	21.19	67.5
FF-RVOS		15.8	1.126	21.20	68.8

Table 4.7: The comparison of the inference costs with different methods. * SAMWISE uses Hiera-B backbone and others use Video Swin Base. The results of our method are highlighted.

Prediction Loss	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Without	60.9	59.3	62.6
With	62.2 (+1.3)	60.5 (+1.2)	64.0 (+1.4)

Table 4.8: Ablation study on the prediction loss.

Chunk size	Temporal Pyramid	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
1	×	60.0	58.4	61.6
	✓	61.6 (+1.6)	59.9 (+1.5)	63.3 (+1.7)
10	×	60.4	58.6	62.3
	✓	62.3 (+1.9)	60.6 (+2.0)	63.9 (+1.6)
20	×	60.9	59.3	62.6
	✓	62.4 (+1.5)	60.7 (+1.4)	64.0 (+1.4)

Table 4.9: Ablation study for the effect of temporal pyramid state update block.

5

Motion Decoupled 3D Gaussian Splatting for Dynamic Object Representation

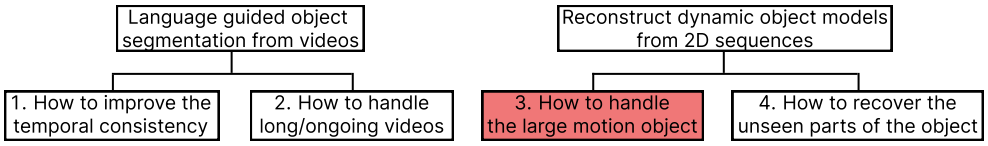


Figure 5.1: The overall tasks of the thesis, this chapter focuses on the task highlighted with red.

In this chapter, we discuss a new field, 3D reconstruction for dynamic objects. To be specified, the object contains not only the local deformation, but also the overall transformation. Dynamic object modeling is a critical challenge in 3D scene reconstruction. Previous methods typically maintain a canonical space to represent the object model, and a deformation field to express the object motion. However, this approach fails when the object undergoes large motions. The position variation caused by significant motion not only complicates the establishment of a canonical space, but also misleads the interpretation

of the deformation field. To overcome the above weaknesses, we propose Motion Decoupled Dynamic 3D Gaussian Splatting (M5D-GS), the first 3D-GS model that separates motion and deformation modeling for dynamic object representation with large motion from a monocular camera. M5D-GS increases the practicality of 3D-GS, as it is common for objects to move, rotate, and deform simultaneously. Current datasets only contain object deformations with slight motions. We introduce a pipeline to reuse current datasets by adding large motions into the scene. To fully demonstrate our improvements, we also introduce a new benchmark featuring several new synthetic scenes with complex motions, some scenes augmented from previous datasets, and some real world recorded scenes. Our M5D-GS significantly increases the accuracy under large motion scenarios while maintaining high rendering speed, which makes it suitable for dynamic object representation tasks including 4D novel view synthesis and real-time rendering.

This work [9] was conducted in collaboration with Dr. Libo Long and my supervisor Prof. Jochen Lang. I was responsible for most of the research and implementation, while the coauthors contributed through discussions and proofreading. The work in this chapter was presented at AAAI 2025, and the project code can be found at `https://github.com/haliphinx/M5D-GS`.

5.1 Introduction

3D scene representation is the foundation of many 3D computer vision tasks, including but not limited to novel view synthesis [97], 3D modeling [22], and motion/animation rendering. These tasks are commonly involved in augmented reality (AR), virtual reality (VR), and potentially autonomous driving, where real-time rendering and dynamic objects are two critical challenges. Traditional methods usually use point clouds [181], meshes [182], occupation grids [183], or polygons [184] to represent the 3D scene. Although these representations may provide additional physical attributes, they are either expensive to acquire or lacking precision especially in dynamic scenes.

Neural Radiance Fields (NeRF) [23] introduced the volume rendering technique to bridge 3D scenes with projected 2D images. This differentiable rendering strategy allows NeRF to implicitly represent a 3D scene with a function based on the view pose, which can be fitted by a deep learning model. NeRF represents the scene continuously, enabling high-precision 3D scene generation. However, the main challenge of NeRF is the training and rendering costs. The original NeRF requires hours or even days to be trained for a

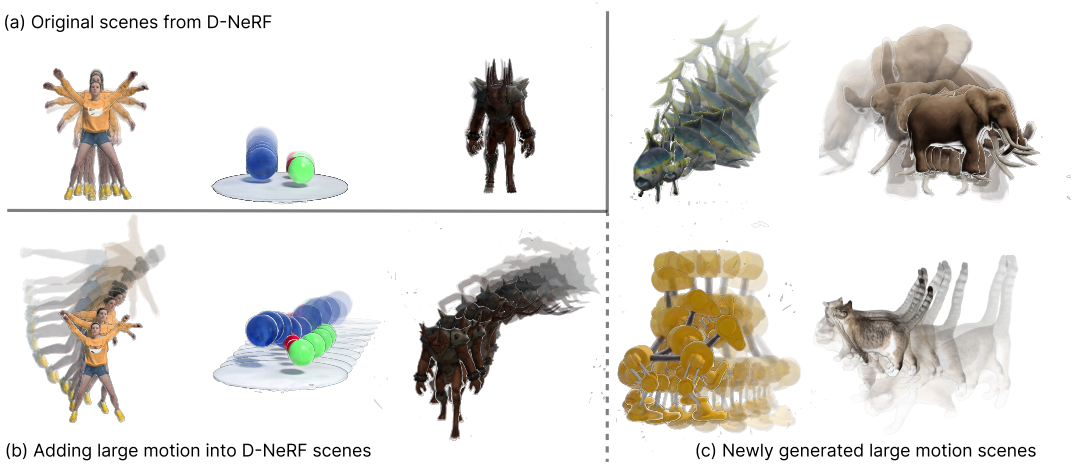


Figure 5.2: Visualization of motion for the scenes. Each scene is represented by a stack of 10 frames uniformly sampled from the temporal space. Greater overlap indicates lighter motion within the scene. (a) Scenes from the D-NeRF dataset, where the frames are highly overlapped, indicating minimal motion. (b) Scenes with large motions added to the original scenes in (a). (c) Newly generated and real world scenes.

single scene. Multiple variants of NeRF [185–188] have been published to accelerate the training speed, but it is still challenging to find the balance between speed and accuracy. 3D Gaussian Splatting (3D-GS) [24] represents a 3D scene with a set of 3D Gaussian points and utilizes a differentiable 3D Gaussian splatting technique to project the 3D scene into a 2D image. This approach not only saves memory by describing only the occupied area but also represents the scene explicitly, making it easier for visualization and editing. 3D-GS achieves comparable accuracy with NeRF but reduces time complexity significantly, allowing for real-time rendering.

Various follow-up works to NeRF [189, 190] explore using NeRF for dynamic scenes. Different 3D graphic structures, including voxel-grids [191, 192] and planes [115, 193], have been used to boost the training speed and provide extra spatial constraints. However, they still suffer from slow processing speeds, and low accuracy. Some research for dynamic 3D-GS [117–119, 194, 195] have been published, incorporating deformation fields into static scenes. These approaches largely share the fundamental idea of maintaining a static canonical space to represent the 3D structure and a time-related deformation field to represent motion. Both components are optimized together to achieve state-of-the-art performance,

which leads to potential problems in large motion scenarios. First, large motion increases the difficulty of establishing a canonical space. Second, this joint optimization strategy can be ambiguous in understanding large motions and easily converge to local minima.

To overcome the limitations of 3D-GS in large motion scenes, we propose Motion Decoupled Dynamic 3D Gaussian Splatting (M5D-GS), which separates the learning of object motion and deformation in dynamic environments. We investigate the motion range in a widely used benchmark D-NeRF [130]. As shown in Fig. 5.2 (a), the motion in each scene is minimal. To address this, we introduce a pipeline that adds object-level motions to existing benchmarks, allowing us to reuse previous datasets without introducing additional bias (Fig. 5.2 (b)). We also generate new scenes containing large motions and real world recorded scenes (Figure 5.2 (c)) to effectively evaluate the performance under scenarios involving complicated motion. Our proposed M5D-GS (Fig. 5.3) decouples the representation of 3D structure, object level motion, and per-Gaussian local deformation, which significantly improves performance over previous state-of-the-art methods in large motion scenes. In summary, our main contributions are:

- We propose M5D-GS, a 3D-GS-based framework designed to handle complex motions for dynamic object representation with high-fidelity and real-time rendering.
- We design a framework to explicitly estimate the object motion and the object deformation separately, and a coarse-to-fine matching strategy to handle large motion initialization.
- We introduce an easy-to-use pipeline that converts existing benchmarks into large motion scenarios, facilitating the study of complex motion representation.
- We generate a new dataset that includes both novel scenes with complex motions and scenes converted from existing benchmarks. Several real world recorded testcases are included as well to fully evaluate the practicality in real world applications. Both the dataset and the source files used for its creation will be open-sourced to the public, allowing the community to further investigate severe motion understanding.

5.2 Related Work

In Sec. 2.2, the research history of the 3D reconstruction, together with the widely used 3D representation data structures are introduced. In this section, we only focus on the

works of NeRF and 3D-GS for dynamic scene representation.

5.2.1 Dynamic Neural Rendering

Neural Radiance Fields (NeRF) [23] have demonstrated promising performance in 3D scene representation and high-quality novel view synthesis. However, traditional NeRF suffers from high training and inference times and struggles with dynamic scenes. One strategy is to extend the 3D scene with a temporal dimension to create a 4D scene [134, 196], but this straightforward extension increases computational costs. Another approach direction [113, 130, 197–200] maintains a static canonical space with an deformation field to represent motion. This spilt structure eases the problem of insufficient constraints in dynamic scenes, leading to improved accuracy. Other methods enhance overall quality by incorporating additional modalities. [199, 201] use object segmentation to separate static background from moving objects. [202] employs depth maps to provide extra constraints. Additionally, 2D CNNs [203, 204] are utilized to enrich spatial understanding.

Some efforts aim at reducing the computational expenses in dynamic NeRF. [134] utilizes keyframes to compress visual features, capitalizing on redundant information in videos. Combining implicit neural rendering with explicit 3D structures significantly reduces the time complexity. For instance, [205] utilizes voxel grids to represent the 3D scene, while [115] uses planes to provide additional spatial constraints. Despite these advancements, current NeRF-based methods still struggle with either accuracy or high computational demands in dynamic scene.

5.2.2 Dynamic 3D Gaussian Splatting

3D Gaussian Splatting (3D-GS) represents a 3D scene with a set of 3D Gaussian points and uses a differentiable neural rendering procedure for image rendering. By representing the scene with sparse points, 3D-GS significantly reduces computation time while achieving comparable accuracy to NeRF. This explicit representation also enables post-training manipulation of the 3D scene and reduces the memory usage of the deep neural network. Extending 3D-GS to dynamic scenes follows a pathway similar to that of NeRF. 4DGS [206] extends the 3D Gaussian distribution to 4D Gaussian by incorporating a temporal dimension. Some methods [118, 119, 194, 195] use separate sub-models to represent the static canonical space and the deformation field. Although this approach works well for scenes

with slight motion and deformation, it struggles with large motions. The model can become ambiguous about whether the re-projection error is coming from the deformation or the canonical space structure change. To address this, we propose our M5D-GS, which fully decouples the static 3D structure, object level motion, and local deformation. Our method significantly improves accuracy in large motion scenario while maintaining comparable time efficiency.

5.3 Method

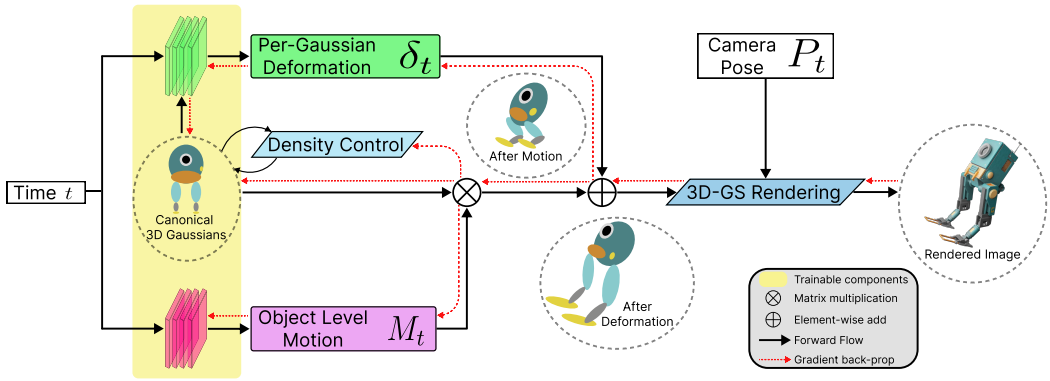


Figure 5.3: The overall workflow of M5D-GS. The inputs are the timestamp t, and the camera pose P_t. The output is the rendered image at the time t viewed from P_t. Three trainable components are included (highlighted with a yellow background): an MLP for predicting object level motion (highlighted in pink), an MLP for predicting local deformation (highlighted in green), and a set of 3D Gaussians representing the canonical space. The 3D Gaussian points are first transformed by the overall motion M_t, then deformed by the local deformation δ_t, and finally projected into the camera frame.

5.3.1 Preliminaries

3D Gaussian Splatting (3D-GS) is an explicit 3D representation. The 3D scene is represented as a set of 3D Gaussians $\mathcal{G}$. Each 3D Gaussian $\mathcal{G}_i$ contains a mean vector $\mathcal{X}_i \in \mathbb{R}^3$ to represent the center point, and a covariance matrix $\Sigma_i \in \mathbb{R}^{3\times3}$. The distribution of $\mathcal{G}_i$ can be described as $\mathcal{G}_i = e^{-\frac{1}{2}\mathcal{X}_i^T\Sigma_i^{-1}\mathcal{X}_i}$.

In order to optimize the covariance matrix in a differentiable back-propagation pass, the matrix is decomposed into a rotation matrix R and a scaling matrix S as $\Sigma = RSS^TR^T$. The 3D covariance matrix and the center point can be projected into a 2D camera plane by using the camera extrinsic matrix P and the Jacobian matrix of the affine approximation of the projective transformation J as:

$$\mathcal{X}^{2D} = JP\mathcal{X}, \Sigma^{2D} = JP\Sigma P^T J^T \tag{5.1}$$

The rotation matrix R is represented by a quaternion $q \in \mathbf{SO}(3)$, and the scale matrix S is expressed as a vector $s \in \mathbb{R}^3$. The projected 2D Gaussian $\mathcal{G}_i^{2D}(\mathcal{X}^{2D}, \Sigma^{2D})$, the opacity α_i, and the view-dependent color defined by spherical harmonic (SH) coefficients $sh_i \in \mathbb{R}^k$ (k is a hyperparameter for the number of SH functions) determine how a Gaussian point affects the color of a pixel $\mathcal{C}$ by:

$$c_i = \mathcal{G}_i^{2D} sh_i \alpha_i, \mathcal{C} = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{5.2}$$

N is the set of the ordered 3D Gaussians overlapping the pixel C projected into the camera. We now have the full attributes of a 3D Gaussian as $\mathcal{G}_i : (\mathcal{X}_i, q_i, s_i, sh_i, \alpha_i)$ which will be optimized during the training and stored for inference.

5.3.2 M5D-GS Framework

To handle large motions in the dynamic scene representation task, we introduce M5D-GS, which decouples the object level motion and per-Gaussian deformation δ_t. D3D-GS [119] is selected as our baseline, since it achieves both good accuracy and robustness in slight motion scenes. The overall framework is shown in Fig. 5.3. The inputs of the framework contain a timestamp t and the camera pose P_t. The output is the rendered 2D image. The whole framework contains three trainable parts (emphasized with yellow background in Fig. 5.3): a set of 3D Gaussians as canonical space $\mathcal{G}$, an MLP for object level motion prediction M_t, and an MLP for per-Gaussian deformation prediction δ_t. The canonical 3D Gaussians are first transformed to the destined pose at time t by the predicted object level M_t to handle the large motion. Then, each Gaussian is fed into another MLP, along with t, to predict the local deformation. The canonical space is dynamically controlled by

various thresholds, including gradient, Gaussian scale, and opacity, to be either pruned or densified.

Motion & Deformation Representation. The entire motion of the scene is decoupled into an object level motion for large translation and rotation, and a per-Gaussian deformation to represent local deformation. The object level motion is predicted by an MLP, Θ_{motion} (shown in red in Fig. 5.3). The input to this block is the timestamp t, and the output is the overall motion M_t, which includes a rotation quaternion and a translation vector. The 3D Gaussians in the canonical space are first transformed by M_t to align with the 3D scene at time t in a rigid manner. The per-Gaussian deformation is predicted by another MLP, Θ_{deform} (shown in green in Fig. 5.3). The output, δ_t, contains a set of transformations for each Gaussian, including rotation, translation, and scaling. This deformation allows the 3D Gaussian point cloud to capture local deformations that cannot be addressed by the object level motion alone. Since the global motion has already been excluded by the object level motion M_t, δ_t are typically very small (close to 0). To avoid vanishing gradients, δ_t is added as an offset to the 3D Gaussians rather than being multiplied. The final 3D Gaussians $\mathcal{G}_t$ at time t can be calculated as:

$$\begin{aligned} M_t = \Theta_{motion}(t), \delta t = \Theta_{defrom}(\mathcal{G}, t) \\ \mathcal{G}_t = M_t \mathcal{G} + \delta_t \end{aligned} \tag{5.3}$$

Coarse-to-Fine Matching. Training monocular dynamic 3D scene representation often lacks constraints because there is only one frame available at each time. Different motions can result in the same observation. For instance, moving toward the camera and enlarging the scale can produce similar images. We assume that the object level motion involves only translation and rotation, without significant scale changes, while minor scaling can be handled by per-Gaussian deformation.

The initialization of the canonical space is a core step in the framework. A low-quality canonical space would also negatively affect the subsequent per-Gaussian deformation. Thus, it is a common strategy to first warm up the training by only optimizing the canonical space to an initial scene. This strategy assumes that even if the scene is dynamic, there is still some overlap between the predicted and actual 3D Gaussians. This assumption allows the multi-view geometry to estimate the 3D Gaussian point during the warm up stage. However, in scenes with large motion, this assumption is usually broken, resulting in the warm up initialization generating an empty scene. Fig. 5.4 illustrates a simplified 2D

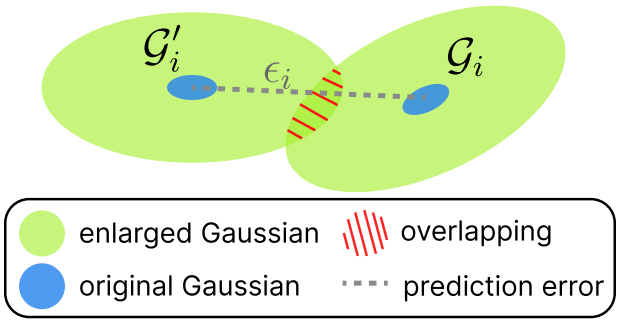


Figure 5.4: A simplified 2D example of the coarse-to-fine matching. Two blue ellipses represent the predicted distribution $\mathcal{G}'_i$ and the actual distribution $\mathcal{G}_i$ of a same Gaussian point at time t_i at real scale (fine matching). The gray dashed line represents the predicted translation error ϵ_i. The two green ellipses are the enlarged Gaussian points for coarse matching, with the red shadow represents the overlap.

example. $\mathcal{G}'_i$ and $\mathcal{G}_i$ represent the predicted and actual distributions of the same Gaussian point at time t_i. ϵ_i denotes the predicted translation error. The blue ellipses depict the distributions at real scale for fine matching, while the green ellipses represent the enlarged Gaussian points for coarse matching. To statistically analyze this task, Bhattacharyya distance (D_B) [207] is employed to measure the Overlap Coefficient (OC) between $\mathcal{G}'_i$ and $\mathcal{G}_i$ as $OC = exp(-D_B)$. The definition of D_B is as follows:

$$D_B(\mathcal{G}_i, \mathcal{G}'_i) = \frac{1}{8}(\mathcal{X}'_t - \mathcal{X}_t)^T(\frac{\Sigma'_t+\Sigma_t}{2})^{-1}(\mathcal{X}'_t - \mathcal{X}_t) + \frac{1}{2}ln(\frac{det(\frac{\Sigma'_t+\Sigma_t}{2})}{\sqrt{det(\Sigma'_t)det(\Sigma_t)}}) \tag{5.4}$$

$0 \leq OC \leq 1$, and $OC = 1$ means fully overlapping. The correlation between OC and each variable can be simplified as follows:

$$OC \propto D_B^{-1} \propto \epsilon_i^{-1} S_i \tag{5.5}$$

In Eq. (5.5), S_i represents the scale of the Gaussian point, and $\propto$ indicates a positive correlation. A step-by-step proof is provided in Sec. 5.4.5. In large motion scenes, ϵ_i is typically very large at the beginning. This leads to a low OC, which causes the model

to treat the Gaussian point as an outlier. To address this issue, we also train the object level motion block during the warm up stage. During this stage, each Gaussian point is uniformly zoomed up in its local coordinates to increase OC. After the warm up stage, ϵ_i is reduced to an acceptable range, and the scale factor is gradually reduced to 1 by the time when the warm up stage is complete. Our proposed coarse-to-fine matching improves the initialization robustness and quality in large motion scenes.

Optimization Loss. The main constraints for the proposed M5D-GS still follow the original 3D-GS without additional loss for motion estimation. The overall constraint includes a per-pixel $\mathcal{L}_1$ loss and a D-SSIM loss [24] $\mathcal{L}_{D-SSIM}$. The loss function is as follow, λ is the loss coefficient $\mathcal{L}_{img} = \mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}$.

5.3.3 New Benchmarks with Complex Motion

Current benchmarks [130, 131, 134] for dynamic scene representation usually contain only slight motion and deformation. Fig. 5.2 qualitatively shows the range of motion for different scenes. Each scene is represented as a stack of 10 frames uniformly sampled from the temporal space with the camera pose fixed. The more overlap between different frames, the less motion involved in the scene. Fig. 5.2 (a) shows the scenes from the original D-NeRF dataset, most of the frame contents overlap, especially in the latter two scenes. To fairly evaluate previous methods under large motion scenarios, we propose a pipeline to add large rigid motions to the existing scenes while using the same dataset images. Assume I_t is the frame image at time t, J is the camera intrinsic matrix, $P_t^{w\to c}$ and $P_t^{c\to w}$ represent the world-to-camera and camera-to-world transformation matrices, and O_t represents the object pose in the world coordinate. The goal is to add a designed motion T_t into O_t, while keeping I_t the same. The object pose is an implicit variable that cannot be modified directly without regenerating the whole scene. Therefore, we design the following equations to add the object motion indirectly by modifying the camera pose:

$$I_t = JP_t^{w\to c}O_t = JP_t^{w\to c}(T_t^{-1}T_t)O_t = J(P_t^{w\to c}T_t^{-1})(T_tO_t) = J(T_tP_t^{c\to w})^{-1}(T_tO_t) \quad (5.6)$$

Eq. (5.6) proves that transforming the camera extrinsic matrix by the inverse of the designed motion and using the same image is equivalent to transforming the object by the same motion matrix. This allows us to reuse previous dataset images by only manipulating the camera extrinsic matrix. Fig. 5.2 (b) shows the scenes after adding complex motions.

We also generate 5 new synthetic scenes involving complex motion and 2 novel real world recorded scenes. Several examples are shown in Fig. 5.2 (c).

5.4 Experimental Evaluation

5.4.1 Experiment Setup

We select five state-of-the-art 3D representation methods as the competitors. D3D-GS [119], SC-GS [195], and 4D-GS [118] are three state-of-the-art Gaussian Splatting based methods. 4D-GS shares a similar structure with D3D-GS but is different in the description of the deformation field. SC-GS is developed on top of D3D-GS by adding sparse control points and additional local deformation constraints. Two state-of-the-art NeRF-based methods K-Planes [115], and TiNeuVox [191], are also incorporated. NeRF-based methods represent the scene densely and are usually more stable but perform worse than the 3D-GS based methods. The evaluation metrics follow the previous public benchmarks [130, 198]. Specifically, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and VGG-based Learned Perceptual Image Patch Similarity (LPIPS) [208] are used.

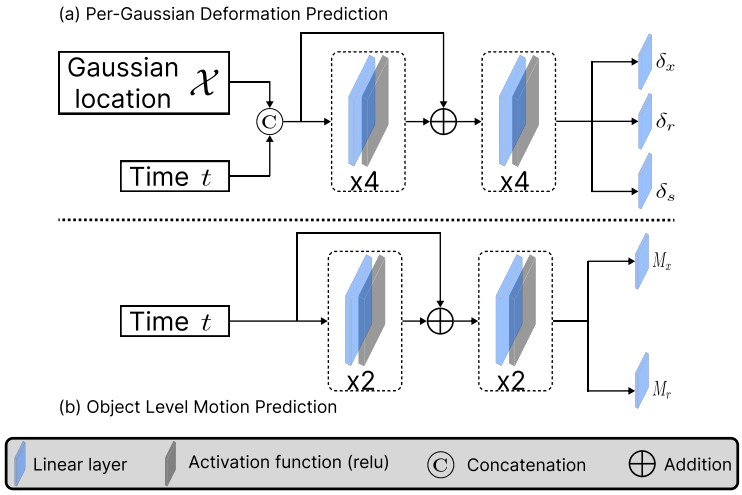


Figure 5.5: Detailed structure for the per-Gaussian deformation and object level motion prediction.

Framework Details. The detailed structures for the object level motion prediction and per-Gaussian local deformation prediction are shown in Fig. 5.5. Multilayer Perceptrons (MLPs) with different numbers of layers are used for different predictions. The object level motion focuses on the overall motion, so the model only takes the timestamp as input. The output contains a translation M_x and a rotation M_r. The per-Gaussian deformation model aims at predicting the local deformation, so both the timestamp and the location of each 3D-Gaussian are fed into the model. The output contains the offsets $\{\delta_x, \delta_r, \delta_s\}$ to describe the deformation. An original 3D-Gaussian in the canonical space $\mathcal{G} : \{\mathcal{X}, q, s, sh, \alpha\}$ is transferred to $\mathcal{G}_t : \{\mathcal{X} + M_x + \delta_x, M_r \times q + \delta_r, s + \delta_s, sh, \alpha\}$ at the timestamp t for the following rendering process.

Since the camera pose of the real world testcase is not as precise as the synthetic dataset, an additional motion constraint is utilized during the warm up training stage to prevent the motion prediction drift. This loss is described as $\mathcal{L} = \lambda|M - I|$. M is the predicted motion, I represents the static motion (zero translation with identity rotation), and λ is a coefficient value set to 0.02.

Training Strategies. The channel size for hidden features is set to 256. A full resolution of 800×800 with a black background is used for synthetic datasets. A resolution of 960×540 is used for the real world testcases. Different training strategies are employed to ensure a fair comparison and achieve optimal performance. The MLPs used for object motion prediction and per-Gaussian deformation prediction have 4 and 8 layers, respectively. The entire framework is trained end-to-end with 40K iterations. In the first 5K iterations, the per-Gaussian deformation block is frozen, and only the 3D canonical space and the object level motion blocks are trained. This is the warm up initialization stage. After this stage, all components are trained and optimized together. To prevent overfitting, the opacity of each 3D Gaussian is reset to 0 in every 3K iterations. The 3D Gaussian densification process stops at the 20K iteration.

5.4.2 Dataset

Current public datasets [130, 131, 198] only contain scenes with slight motions. To fully evaluate performance in scenes with large motion, we propose a novel dataset that contains both synthetic and real world scenes with objects under complex motions. The proposed dataset contains ten different scenes, three of which are extended from the previous dataset by adding large motions, five brand new scenes generated from scratch, and two real world recorded scenes.

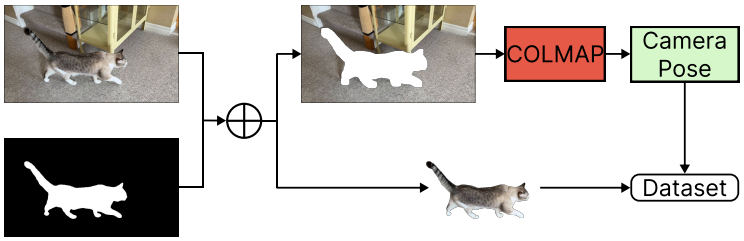


Figure 5.6: The pipeline of the real world testcase generation.

Real World Scenes. The real world testcases are recorded by a smartphone with 60 FPS to avoid motion blur. The image size is downsampled to 960×540 to balance image quality and size. The object mask is generated for each frame to separate the dynamic object from the static background. The background images are used for ground truth camera pose estimation. The foreground dynamic object is used in the dataset. We follow the same processing pipeline to recover the ground truth camera pose as Nerfies [113]. 250 frames are sampled uniformly from each video, with 200 images for training and 50 for testing. Two real-world scenes are generated: one featuring a cat (first row in 5.7) and another featuring a pillow (first row in 5.10). These two objects are selected because they represent fundamentally different motion patterns. The cat undergoes both movement and deformation, while the pillow mainly exhibits rotation and deformation. Additional scenes could be generated using the provided pipeline.

Synthetic Scenes. The pipeline of generating synthetic scenes follows the original NeRF [23]. 200 timestamps are uniformly sampled from the entire capture duration for training image generation. 50 timestamps are randomly sampled from the total 200 timestamps to generate testing images. The camera poses used for image generation are randomly selected focusing on the upper half of the object. More precisely, the location of the camera is higher than the object center and the camera is always looking at the object. The object motion includes general translation, rotation, and realistic animation. The eight synthetic scenes with diverse motion patterns are generated to comprehensively demonstrate the generalization ability of the proposed method, including human body motion, multi-object motion, and articulated motion.

Difference Between Real World & Synthetic Testcases. It is important to emphasis the difference between real world and synthetic testcases to help audiences to better use our novel dataset. First, the data structures are different. Real world and synthetic datasets are generated by different pipelines. To provide a fair comparison with previous

	Number of Scenes	Average frame num (train/test)	Object Motion	Object Deformation	Object Model	Has Real World Scenes
D-NeRF	8	138/20	×	✔	×	×
M5D-GS (Ours)	**10**	**185/41**	✔	✔	✔	✔

Table 5.1: Comparison of our proposed Dataset with D-NeRF [130]. Ours contains both object motion and deformation. The source files, including the 3D models of the objects, are also provided.

datasets, our benchmark follows the same data structure as previous real world and synthetic datasets, resulting in two different data structures. The real world testcases follow NeRF-DS [131], while the synthetic testcases follows D-NeRF [130]. Second, the camera matrices of the real world testcases are not as precise as those of the synthetic testcases. Both, the camera extrinsic and intrinsic matrices are estimated by COLMAP [22], a Structure From Motion (SFM) framework. Third, the camera motion trajectory for the real world testcases are continuous, but the camera poses for the synthetic testcases are randomly sampled at each timestamp independently. This increases the difficulty for the real world testcases, as only part of the object may be visible for a short period. Lastly, the timestamps of the testing set are sampled from the range of the training set in the synthetic dataset, but the timestamps of the training and testing sets in real world datasets do not overlap. Since only one camera is used to record the video in the real world dataset, it cannot capture the same scene at the same timestamp from different camera poses.

Tab. 5.1 shows the comparison between our proposed dataset and the D-NeRF benchmark. Our novel dataset contains more frames with object motions and deformations, including both rigid and non-rigid deformations. Real world recorded scenes are included as well. We also release the source files of the object models used for the synthetic dataset generation, allowing for potential 3D-level evaluations in the future. Several examples of our dataset are shown in Fig. 5.2 (b) and (c).

5.4.3 Results and Comparisons

Comparison on Large Motion Dataset. Tab. 5.2 shows the results of the comparison between our method and competitors on the proposed large motion dataset. The best results are shown in bold, and the second best results are underlined. Two real world testcases are listed in the last column. In each sub-table, methods above the dashed line are NeRF-based, while the GS-based methods are below the dashed line. "Fails" in the table

Method	Jumping Jacks + Motion			Hell Warrior + Motion			Bouncing Balls + Motion			Elephant			Grey Cat		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TiNeuVox	29.22	0.95	0.07	31.18	0.92	0.13	28.71	0.97	0.11	16.37	0.77	0.20	33.28	0.97	0.04
K-Planes	27.05	0.94	0.07	18.69	0.88	0.14	32.60	0.97	0.07	Fails			—		
4D-GS	Fails			12.01	0.84	0.17	16.65	0.88	0.27	15.05	0.87	0.17	17.59	0.89	0.10
SC-GS	Fails			Fails			9.42	0.53	0.37	18.26	0.82	0.15	Fails		
D3D-GS	Fails			32.16	0.93	0.10	30.37	0.97	0.05	29.00	0.96	0.07	Fails		
M5D-GS (ours)	**35.86**	**0.99**	**0.02**	**37.37**	**0.97**	**0.06**	**36.47**	**0.99**	**0.02**	**37.44**	**0.99**	**0.03**	**39.69**	**0.99**	**0.01**
Method	Robot			Pokemon			Fish			Robot Dog			Pillow		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TiNeuVox	24.52	0.89	0.13	14.89	0.75	0.24	27.33	0.93	0.10	12.99	0.71	0.31	27.38	0.93	0.08
K-Planes	17.56	0.84	0.22	Fails			22.32	0.91	0.13	Fails			—		
4D-GS	22.45	0.91	0.11	17.15	0.89	0.20	16.02	0.90	0.13	16.65	0.88	0.17	17.69	0.88	0.13
SC-GS	Fails			21.44	0.83	0.15	Fails			15.08	0.79	0.20	Fails		
D3D-GS	24.32	0.92	0.11	29.84	0.95	0.07	Fails			19.25	0.90	0.12	29.89	0.96	0.06
M5D-GS (ours)	**29.78**	**0.96**	**0.05**	**35.60**	**0.98**	**0.03**	**36.15**	**0.98**	**0.02**	**33.46**	**0.98**	**0.02**	**32.37**	**0.98**	**0.04**

Table 5.2: Quantitative results on our large motion dataset. PSNR, SSIM, and LPIPS(VGG) are evaluated for each competitor. The best result is **bolded**, and the second best is underlined. The last column contains two real world testcases.

indicates that the method fails to reconstruct the 3D scene in the corresponding testcase or generates meaningless noise. The experiment shows that our method significantly improves the accuracy and robustness in large motion scenes. The visualized comparison is shown in Fig. 5.7, and more visualizations can be found in Sec. 5.4.6.

Experiments on Previous D-NeRF Dataset [130] is a widely used synthetic benchmark containing deformable objects with precise camera pose. Tab. 5.3 shows the quantitative results on the D-NeRF dataset. The D-NeRF dataset only contains slight motions. The objects are mostly placed in a static position with some animation involved as deformation. SC-GS uses downsampled images for their experiment and employs restricted constraints designed for local deformation, including As-Rigid-As-Possible (ARAP) loss, linear blend skinning, and k-nearest neighbor (KNN) based control points. These constraints enable SC-GS to perform well on the D-NeRF dataset but reduce robustness in large motion scenes. For all competitors, the best result is shown in bold, and the second best is underlined. The results show that our proposed M5D-GS achieves comparable performance even though the dataset only contains slight motions.

Visualizing the Predicted Motion. The ground truth of the motion is not accessible because object deformation can introduce slight motions as well. For example, in Fig. 5.8, the fish's body movement is reflected as minor fluctuations in the trajectory. To illustrate this, we provide an example in Fig. 5.8 that qualitatively shows the estimated motion trajectory of the scene. The black line represents the estimated motion trajectory. Five frames are uniformly sampled from the entire timeline. The estimated positions are

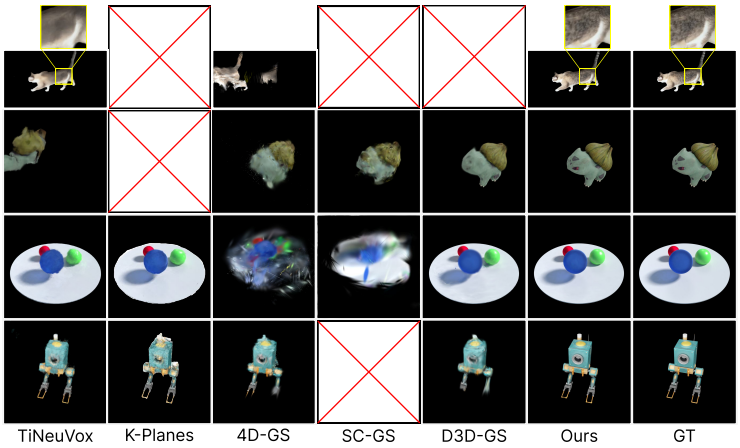


Figure 5.7: Qualitative examples from our large motion dataset for each competitor. The blank box with a red cross indicates that the method cannot handle a particular scene. Zoom in for more detailed views.

indicated by red dots and the orientations are represented by RGB coordinates. This visualization demonstrates that our method successfully estimates the motion, fulfilling our objective.

Comparison of Rendering Speed. Tab. 5.4 shows the rendering speed for all 3D-GS based competitors. NeRF-based methods are significantly slower than 3D-GS based methods, so only four 3D-GS based methods are compared. The average processing Frame Per Seconds (FPS) of four testcases where all the 3D-GS based methods succeeded are evaluated. The best result is bold, and the second best is underlined. The total rendering time depends not only on the method complexity but also on the total number of Gaussian points in the scene. D3D-GS achieves the highest FPS, and our M5D-GS achieves the second best. However, we can observe that the 3D scenes generated by D3D-GS are blurrier (shown in Fig. 5.7) with lower PSNR accuracy.

5.4.4 Ablation Study

Motion & deformation prediction. As shown in the framework structure (Fig. 5.3), M5D-GS represents the change of the entire scene with an overall motion model to predict the object level translation and rotation, and a local deformation model for local deformation prediction. It is inappropriate to evaluate the rendering accuracy for each part

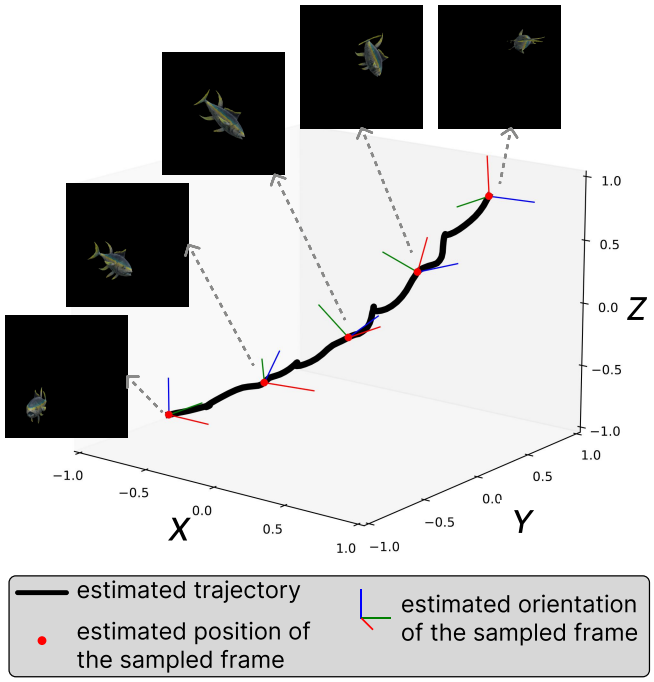


Figure 5.8: An example of the estimated object motion. The black line represents the estimated trajectory over the entire timeline. Five samples are depicted with the corresponding image frames shown around them. The red dot indicates the estimated position, and the RGB coordinate represents the estimated orientation.

independently since both parts are working collaboratively. Therefore, we provide a set of qualitative results, as shown in Fig. 5.9. The three rows represent three different types of deformations. In the first row, the elephant undergoes a non-rigid deformation; the robot in the middle row undergoes a rigid deformation; and, multiple objects with different motions are included in the last row. Object rotations and translations are involved in all three scenes as well. The first three columns show the rendered image from the 3D Gaussian points at different stages (corresponding to the three hand-drawn robot pictures in Fig. 5.3), and the last column shows the ground truth image. In each row, the four images are generated from the same camera viewpoint and angle. The framework selects the optimal pose for the scene to reconstruct the canonical space during training, so the 3D Gaussians in the canonical space (Fig. 5.9 first column) are under different orienta-

	Hell Warrior			Mutant			Hook			Bouncing Balls		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D-GS	29.89	0.9155	0.1056	24.53	0.9336	0.0580	21.71	0.8876	0.1034	23.20	0.9591	0.0600
D-NeRF	24.06	0.9440	0.0707	30.31	0.9672	0.0392	29.02	0.9595	0.0546	38.17	0.9891	0.0323
TiNeuVox	27.10	0.9638	0.0768	31.87	0.9607	0.0474	30.61	0.9599	0.0592	40.23	0.9926	0.0416
Tensor4D	31.26	0.9254	0.0735	29.11	0.9451	0.0601	28.63	0.9433	0.0636	24.47	0.9622	0.0437
K-Planes	24.58	0.9520	0.0824	32.50	0.9713	0.0362	28.12	0.9489	0.0662	40.05	0.9934	0.0322
4D-GS	28.71	0.9733	0.0369	37.59	0.9880	0.0167	32.73	0.9760	0.0272	40.62	0.9942	0.0155
D-3DGS	41.54	0.9873	0.0234	42.63	0.9951	0.0052	37.42	0.9867	0.0144	41.01	0.9953	**0.0093**
SC-GS	**42.93**	**0.994**	**0.0155**	**45.19**	**0.999**	**0.0028**	**39.87**	**0.997**	**0.0076**	**44.91**	**0.998**	0.0190
M5D-GS (Ours)	41.13	0.9863	0.0264	43.19	0.9957	0.0044	38.51	0.9871	0.0127	40.69	0.9944	0.0106
	T-Rex			Stand Up			Jumping Jacks			Average		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D-GS	21.93	0.9539	0.0487	21.91	0.9301	0.0785	20.64	0.9297	0.0828	23.40	0.9299	0.0767
D-NeRF	30.61	0.9671	0.0535	33.13	0.9781	0.0355	32.70	0.9779	0.0388	31.14	0.9690	0.0464
TiNeuVox	31.25	0.9666	0.0478	34.61	0.9797	0.0326	33.49	0.9771	0.0408	32.73	0.9715	0.0495
Tensor4D	23.86	0.9351	0.0544	30.56	0.9581	0.0363	24.20	0.9253	0.0667	27.44	0.9421	0.0569
K-Planes	30.43	0.9737	0.0343	33.10	0.9793	0.0310	31.11	0.9708	0.0468	31.41	0.9699	0.0470
4D-GS	34.23	0.9850	0.0131	38.11	0.9898	0.0074	35.42	0.9857	0.0128	35.34	0.9846	0.0185
D-3DGS	38.10	0.9933	**0.0098**	44.62	0.9951	0.0063	37.72	0.9897	0.0126	40.43	0.9918	0.0116
SC-GS	**41.24**	**0.998**	0.0103	**47.89**	**0.999**	**0.0023**	**41.13**	**0.998**	**0.0067**	**43.31**	**0.998**	**0.0091**
M5D-GS (Ours)	37.96	0.9937	0.0099	46.49	0.9963	0.0049	38.05	0.9855	0.0126	40.86	0.9913	0.0115

Table 5.3: Quantitative results on D-NeRF dataset. PSNR, SSIM, and LPIPS(VGG) are evaluated for each competitor. The best result is **bold** and the second best is underlined.

Method	SC-GS	4D-GS	D3D-GS	M5D-GS (Ours)
FPS	37.36	93.90	**267.39**	135.15
PSNR	16.05	16.38	27.12	**35.74**

Table 5.4: Comparison of rendering frame per second (FPS) and rendering quality. The average value across three scenes where all 3D-GS based methods succeeded are compared. The best result is **bolded**, and the second best is underlined.

tions and locations. Once the object level motion is predicted, the 3D Gaussian points are transformed to match the desired object structure (Fig. 5.9 second column). Finally, the time-variant local deformation is predicted and added to the transformed 3D Gaussian points to generate the final 3D Gaussian points for image rendering (Fig. 5.9 third column). This visualization indicates that our framework effectively handles the motion and deformation in two separate steps as designed.

Influence of the Large Motion. Our proposed benchmark contains three scenes that are upgraded from the original D-NeRF dataset with large motion added. These three scenes can be used to evaluate how large motion affects the performance of different

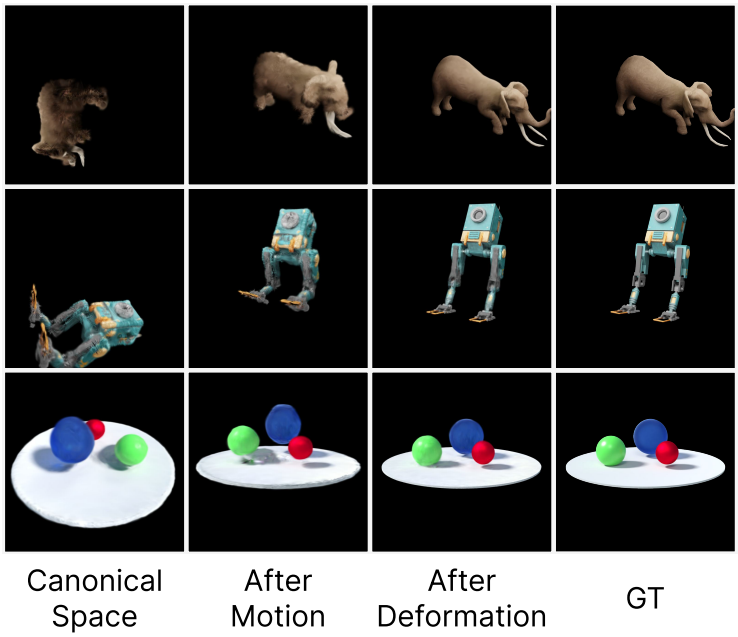


Figure 5.9: Visualization of the reconstructed scenes at three stages: the canonical space, after motion, and after deformation. These three steps correspond to the stages in Fig. 5.3.

	TiNeuVox	K-Planes	4D-GS	D3D-GS	M5D-GS (Ours)
Jumping Jacks	4.27	4.06	-	-	**2.19**
Hell Warrior	7.09	5.89	16.70	9.38	**3.76**
Bouncing Balls	11.52	7.45	23.97	10.64	**3.22**
Ave decrement ↓	7.63	5.80	20.34*	10.01*	**3.06**

Table 5.5: Influence of the large motion. The average decrease after adding large motion is compared. "-" indicates the method fails, and "*" represents the value is average of only two successful scenes.

methods. The PSNR decrement is evaluated for different scenes. SC-GS is not compared since it either fails during training or generates a low-accuracy results in three scenes (the first three scenes in Tab. 5.2). Tab. 5.5 shows how large motion affects different methods. The numbers shown in the table indicate the decrement of PSNR after adding large motion to the original scenes. The smaller the number, the less influence the method experiences by the large motion. The dash means the method fails when processing the large motion

scenes. Our method significantly improves robustness under large motion scenarios with the smallest decrement.

5.4.5 Details of Course-to-Fine Matching

The detailed proof of the proposed course-to-fine matching strategy is provided in this section. The goal of the 3D-GS is to maximize the overlap between the predicted Gaussian points $\mathcal{G}'$ and the real Gaussian points $\mathcal{G}$. Since the Gaussian point is stored in an ordered list, we can easily get the corresponding points from different timestep. Thus, we use $\mathcal{G}$ to represent a single Gaussian point here for simplification. If the predicted $\mathcal{G}'$ is perfectly matching the real $\mathcal{G}$, their probability distributions should be totally overlapped. In other words, the more overlapping between two Gaussian distribution, the more similarity between them. The overlapping between two Gaussian distributions can be described as the Overlap Coefficient (OC):

$$OC(\mathcal{G}, \mathcal{G}') = \int_{\mathcal{X}} \sqrt{\mathcal{G}(x)\mathcal{G}'(x)}dx \tag{5.7}$$

For Gaussian distribution, OC can be calculated through Bhattacharyya distance [207] D_B as $OC = exp(-D_B)$. The defination of D_B is:

$$D_B(\mathcal{G}, \mathcal{G}') = \frac{1}{8}(\mathcal{X}' - \mathcal{X})^T(\frac{\Sigma'+\Sigma}{2})^{-1}(\mathcal{X}' - \mathcal{X}) + \frac{1}{2}ln(\frac{det(\frac{\Sigma'+\Sigma}{2})}{\sqrt{det(\Sigma')det(\Sigma)}}) \tag{5.8}$$

In 3D-GS, the covariance Σ is decoupled into the combination of the rotation matrix R and scaling matrix S as $\Sigma = RSS^TR^T$. R is an orthogonal matrix, and S is a diagonal matrix. We first prove that the second term in Eq. (5.8) is not correlated to S.

$$det(\frac{\Sigma' + \Sigma}{2}) = 2^{-n}det(R'S'S'^TR'^T + RSS^TR^T) \tag{5.9}$$

During the warm up training, the scale S is not changed, so $S' = S$. n is a constant value which equals to the dimension of the matrix $\Sigma' + \Sigma$. Specifically, $n = 3$. Eq. (5.9) can be simplified as:

$$
\begin{aligned}
det(\frac{\Sigma' + \Sigma}{2}) &= 2^{-3} det((R' + R)SS^T(R'^T + R^T)) \\
&= det(S)^2(2^{-3} det((R' + R)(R'^T + R^T))
\end{aligned}
\tag{5.10}
$$

Similarly to the lower part of the second term:

$$
\begin{aligned}
\sqrt{det(\Sigma')det(\Sigma)} =& \sqrt{det(R')det(S')det(S'^T)det(R'^T)} \\
& \sqrt{det(R)det(S)det(S^T)det(R^T)} \\
=& \, det(S)^2
\end{aligned}
\tag{5.11}
$$

Above all, the second term in Eq. (5.8) is simplified as:

$$
\frac{1}{2} ln(\frac{det(\frac{\Sigma' + \Sigma}{2})}{\sqrt{det(\Sigma')det(\Sigma)}}) = \frac{1}{2} ln(2^{-3} det((R' + R)(R'^T + R^T)) \tag{5.12}
$$

Eq. (5.12) shows that the second term of Eq. (5.8) is only correlated with the rotation matrix R, but not the scaling matrix S. For the first term, $\mathcal{X}' - \mathcal{X}$ is the prediction error ϵ.The first term can be simplified as:

$$
\epsilon^T (\frac{2}{(R' + R)SS^T(R'^T + R^T)})\epsilon \tag{5.13}
$$

And Eq. (5.8) can be finally expressed as:

$$
\begin{aligned}
D_B(\mathcal{G}, \mathcal{G}') =& \frac{1}{8} \epsilon^T (\frac{2}{(R' + R)SS^T(R'^T + R^T)})\epsilon \\
& + \frac{1}{2} ln(2^{-3} det((R' + R)(R'^T + R^T))
\end{aligned}
\tag{5.14}
$$

According to Eq. (5.14) the correlation between OC and each variable can be determined as:

$$OC \propto D_B^{-1} \propto \epsilon^{-1} S \tag{5.15}$$

As shown in Eq. (5.14), the rotation matrices appear in both numerator and denominator, so the correlation is not cleared. Thus we zoomed up the scale during the warm up stage to prevent the outlier misclassification due to the low OC value.

5.4.6 More Visualizations

More visualization results are provided in this section to demonstrate the improvement. Fig. 5.10 shows the comparison results for all the competitors. We can see that the proposed method not only more robust than others with no failure case, but also generates more precise results. Fig. 5.11 shows two additional results for the motion estimation. Five frames are uniformly sampled on the entire motion trajectory, and the orientations are shown with RGB coordinates. The result indicates that our method estimates the motion successfully as designed.

5.5 Summary

In this chapter, we propose a 3D-GS based method to reconstruct and represent the dynamic object with local deformation and large transformation. We propose M5D-GS, a framework that decouples motion and deformation prediction, to handle the 3D representation task under large motion scenarios. By processing the object level motion and per-Gaussian local deformation separately, our method not only improves the robustness but also outperforms all the competitors in large motion scenes. To fully evaluate the capacity of different methods under large motion, we introduce a novel large motion dataset combining three scenes from previous dataset with additional large motion, five newly generated synthetic challenging scenes, and two real world recorded scenes. Our method significantly improves the performance in the large motion dataset without losing generalization in previous benchmarks with slight motion. However, some limitations are remained. Current monocular 4D representation datasets usually contains limitations on the spatial temporal sampling since a single camera cannot fully constrain a 4D object. In the next chapter, we would discuss this limitation, and provide a corresponding solution.

Figure 5.10: More visualized comparison results. The blank box with a red cross indicates that the method cannot handle a particular scene.

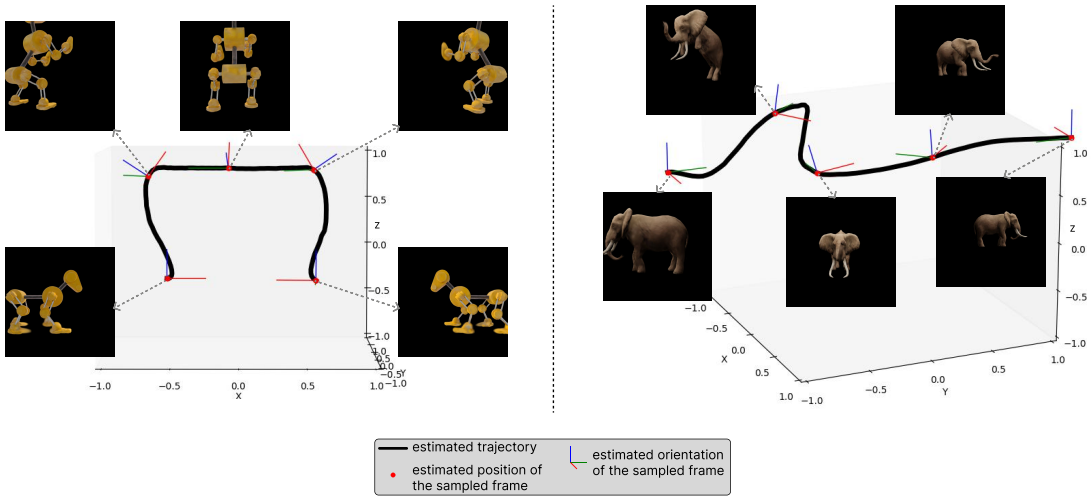


Figure 5.11: More visualization of the estimated motion. The black line represents the estimated trajectory over the entire timeline. Five samples are depicted with the corresponding image frames shown around them. The red dot indicates the estimated position, and the RGB coordinate represents the estimated orientation.

6

MoSc-4DGS: Monocular Scanning Video 4D Gaussian Splatting for Dynamic Object Representation

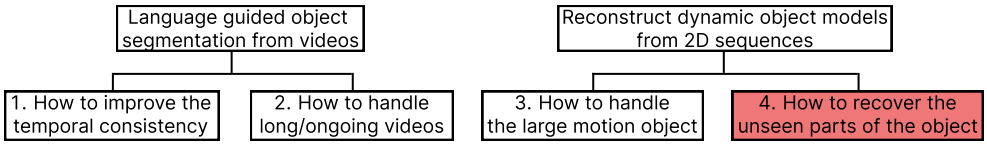


Figure 6.1: The overall tasks of the thesis, this chapter focuses on the task highlighted with red.

Monocular video-based 4D object capture is a challenging task because a single monocular camera captures only 3D information, which cannot fully constrain a 4D representation. Spatiotemporal regions far from the training data tend to be at low quality. Current benchmarks for monocular video-based 4D object representation either use a randomly sampled synthetic camera, or capture only a limited portion of the object. To address these limitations and ease dynamic object capture, we propose Monocular Scanning Video 4D Gaussian Splatting (MoSc-4DGS). We define a scanning video as a monocular video

in which the camera continuously moves around the dynamic object, such that different parts of the object are observed at different time steps, and the entire object is eventually covered over time. We design an enhanced deformable 3D-GS framework that learns from both precise training samples and imprecise generated samples that we obtain with a novel finetuning strategy for diffusion models. Compared to previous state-of-the-art approaches, our method significantly improves reconstruction accuracy and robustness. This work was guided by my supervisor Prof. Jochen Lang, who contributed through discussions and proofreading.

6.1 Introduction

3D object representation forms the foundation of many computer vision tasks, including novel view synthesis [97], object modeling [22], and motion or animation rendering [209]. A variety of data structures have been proposed to represent 3D scenes, including point clouds, meshes, occupancy grids, and polygons. More recently, Neural Radiance Fields (NeRF) [23] and 3D Gaussian Splatting (3D-GS) [24] have been introduced, offering high-accuracy representations.

To incorporate temporal information, many 3D object representation frameworks have been extended to 4D representations, such as DynaSLAM [103], D-NeRF [130], and D3D-GS [206]. However, a fundamental challenge remains: How to capture videos suitable for 4D object reconstruction. A monocular video provides only 3D information (2D images over time), which is insufficient to fully constrain a 4D representation. Fig. 6.2 illustrates several commonly used capture strategies. The first strategy (Fig. 6.2 (a)) employs a multi-camera system to capture the dynamic object from different viewpoints simultaneously. Multi-camera setups require careful synchronization and calibration, making them expensive and impractical for everyday use. Despite this, the quality of the resulting 4D representation is highly dependent on the density and configuration of the cameras.

For monocular camera setups, a common approach is to capture only a small portion of the object (Fig. 6.2 (b)). As a result, only the observed regions of the object can be reconstructed, leading to an incomplete representation. Another strategy, applicable only to synthetic datasets, involves randomly sampling camera viewpoints around the object at each time step to form a discontinuous camera trajectory (Fig. 6.2 (c)). The resulting sequence can be treated as a capture by a pseudo multi-camera system with small temporal offsets between the cameras. However, this strategy is not feasible in

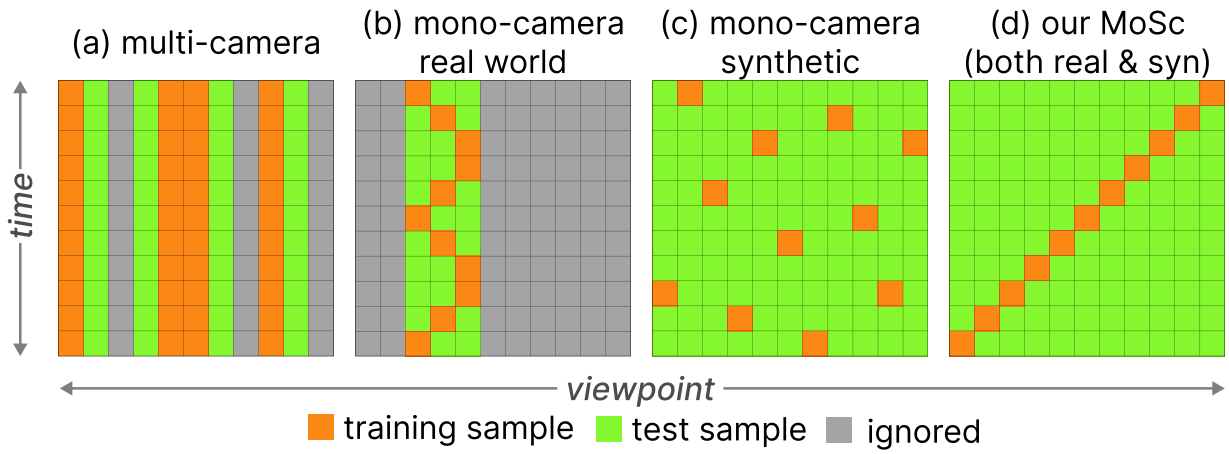


Figure 6.2: Spatiotemporal sampling strategies across different benchmark setups. (a) Multi-camera benchmarks [134, 135, 210] capture dynamic scenes from multiple viewpoints simultaneously. (b) Real-world monocular benchmarks [113, 114, 131] observe only a limited spatial region of the object. (c) Synthetic monocular benchmarks [9, 130] randomly select a camera pose for each timestep, resulting in a discontinuous trajectory. (d) Our Monocular Scanning (MoSc) benchmark uses a continuous monocular camera trajectory to capture the entire object over time.

real-world scenarios because the discontinuous and rapid movement is not possible by physical cameras. Fig. 6.3 visualizes different monocular camera capturing strategies. Solid pyramidal symbols represent training samples, while outlined pyramids indicate test samples. The temporal dimension is indicated using the color of the frames.

The previous strategies are not practical for real-world monocular setups, as they all suffer from significant limitations. Our goal is to reconstruct a complete 4D dynamic object using videos captured by a single monocular camera. To this end, we introduce a new type of video input, termed a Monocular Scanning (MoSc) video. A MoSc video is defined as a continuous monocular video in which the camera casually moves around the object, such that different parts of the object are observed at different time steps, and over time, the entire object is eventually covered. This type of video is more practical for real-world applications and can be used in both synthetic and real-world scenarios. The corresponding spatiotemporal sampling pattern is illustrated in Fig. 6.2 (d), with a visualized example shown in Fig. 6.3 (c). However, monocular scanning videos introduce new challenges. Each part of the object is constrained by limited observation in the spatiotemporal region, and the spatiotemporal distribution between the training and test sets varies significantly. To

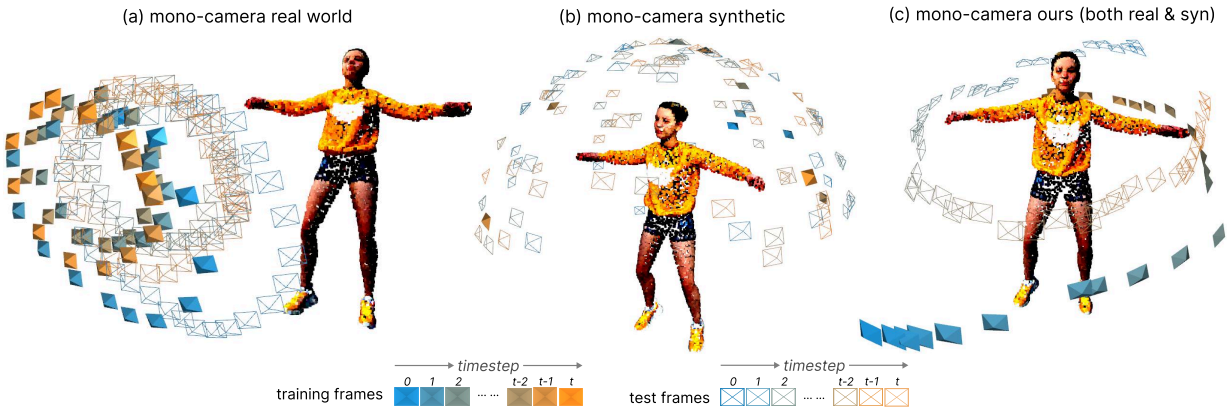


Figure 6.3: Camera pose sampling strategies in monocular video datasets. (a) Real-world dataset [113, 114, 131] trajectories cover only a small portion of the object. (b) Synthetic datasets [9, 130] randomly sample camera poses around the object, simulating a pseudo multi-camera setup. (c) Our monocular casual dataset captures the full object via a single camera moving continuously around it.

address these challenges, we propose MoSc-4DGS, Monocular Scanning Video based 4D Gaussian Splatting. The structure of the object is first represented as a set of deformable 3D Gaussian splats (D3D-GS). To improve performance in weakly constrained regions, we introduce two additional supervisory signals: a shape prior and a diffusion-based texture prior. We propose an object-centered diffusion fine-tuning strategy to enhance the quality of object-specific images. Finally, a novel D3D-GS training strategy is designed to learn effectively from both precise and imprecise (generated) samples. The contributions of this work are:

- We define a new task for dynamic object representation using a more practical input type, monocular scanning videos. A new benchmark is proposed consisting of both real-world and synthetic monocular scanning videos.
- We propose Monocular Scanning Video based 4D Gaussian Splatting, a novel framework for reconstructing complete dynamic objects from monocular scanning videos. MoSc-4DGS enhances Deformable 3D Gaussian Splatting to effectively learn from both accurate training samples and imprecise generated samples, while mitigating the impact of uncertainty.

- We introduce a novel diffusion model finetuning strategy to enhance the quality of rendered images in weakly constrained regions.

6.2 Related Work

The overall discussion of the Gaussian Splatting is discussed in Sec. 2.2, and the detailed research history of 3D-GS is provided in Sec. 5.2. In this section, we focus on some latest researches after our previous work M5D-GS, and some scene reconstruction methods using generative model.

6.2.1 4D Gaussian Splatting

There are many recent works using Gaussian splatting to model the 4D scenes. Deformable 3D-GS [119] maintains a set of static 3D Gaussian splats alongside a separate deformation field to capture temporal changes. E-D3DGS [211] replaces fixed positional encoding with learnable latent features per splat for flexibility. To improve generalization, NPGs [212] first predict a coarse point cloud using a low-rank MLP, which is then refined via 4D Gaussian Splatting. MoDGS [213] introduces depth maps as auxiliary inputs to provide geometric constraints, which are not always available in real-world scenarios. MotionGS [214] incorporates optical flow as a supervisory signal to better capture motion dynamics. HybridGS [215] separates dynamic objects from the static background to improve reconstruction quality. GenMOJO [216] uses a similar static-dynamic separation idea with an additional generative model to refine the result. SC-GS [117] embeds controllable keypoints as anchors into the 3D structure. Gaussian Prediction [217] predicts the motion trajectories of these keypoints to support more accurate motion interpolation. 4D-Rotor [218] proposes a temporal slicing approach that projects 4D splats into 2D frames for efficient rendering. M5D-GS [9] decouples object motion and local deformation to better model complex dynamic behaviors. MoDec-GS [219] uses hierarchical canonical space to handle severe scene changes. BARD-GS [220] uses the depth map to refine the motion blur due to camera and object motions.

Current methods generally rely on strict assumptions about camera trajectories, such as dense view sampling or synchronized multi-camera setups, which are rarely achievable with casual real-world monocular videos. In contrast, our approach is designed to operate

on monocular scanning videos, where the camera casually moves around the object and observes different parts over time, which is easy to realize in practice.

6.2.2 Generative Model Enhanced Scene Reconstruction

Traditional neural rendering methods typically require a large number of frames from diverse viewpoints to provide sufficient spatial constraints. Otherwise, the model tends to overfit to the training set and performs poorly on unseen views. The success of diffusion-based generative model [121] motivates their integration into scene reconstruction pipelines. ViewCrafter [124] leverages a point-conditioned video diffusion model to enhance the quality of point cloud-rendered views generated from the training set. ViewExtrapolator [125] adopts a similar idea but proposes a training-free strategy by directly controlling the gradients of a pre-trained Stable Video Diffusion model. DIFIX3D+ [127] implements a diffusion model trained on image pairs to denoise blurry inputs. DiET-GS [128] uses a diffusion prior to deblur images due to fast camera motion. S2Gaussian [129] increase the input image resolution with a generative model to improve the final representation quality.

However, all of these approaches are designed for static scenes with limited performance on dynamic scenes. The presence of dynamic elements introduces temporal ambiguities that significantly complicate the denoising process, making current generative-enhanced methods ineffective for dynamic 4D reconstruction.

6.3 Method

As shown in Fig. 6.4. The entire pipeline consists of two sub-models with five consecutive steps. A diffusion model (Sec. 6.3.3) is first finetuned to enhance blurry images. Then, an improved D3D-GS model (Sec. 6.3.5) with different setups is used to represent the dynamic scenes with improved tolerance to imprecise training samples. The top row of Fig. 6.4 illustrates the overall workflow, while the detailed structure of each step is shown in the matching colored boxes below. The core idea is to leverage a generative model to enhance D3D-GS performance in the regions that fall outside of the training set distribution.

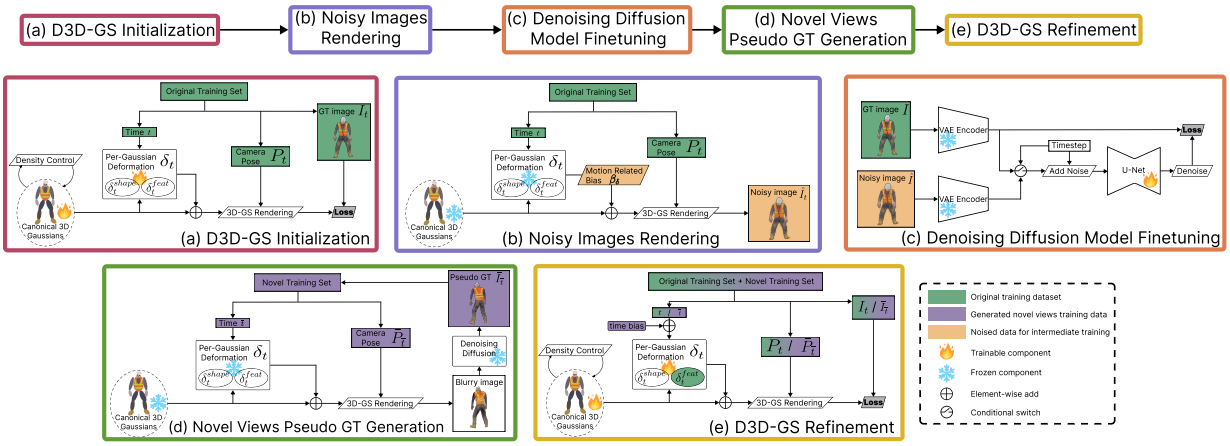


Figure 6.4: The overall workflow contains five steps. (a) D3D-GS initialization, (b) Noisy images rendering, (c) denoising diffusion model finetuning, (d) novel views pseudo GT generation, (e) D3D-GS refinement. The top row shows the work flow, and the blocks below show the detailed structure for each step with the same boundary color.

6.3.1 Preliminaries

Diffusion Model Tuning. A diffusion-based generative model [121] typically consists of three key components: an encoder [120], a U-Net, and a decoder. The encoder and decoder project features between the image space and the latent space, while the U-Net serves as the denoising module that predicts the noise to be removed at each timestep. Some works trained diffusion models [221] on large-scale datasets to provide strong baselines. When finetuning a pretrained diffusion model, it is common practice to update only the U-Net [222]. During training, the model learns to minimize the discrepancy between the added noise ε and the predicted noise ε_θ generated by the U-Net θ as:

$$f_m = \sqrt{\bar{\alpha}_m} f_0 + \sqrt{1 - \bar{\alpha}_m}\varepsilon \tag{6.1}$$

$$\theta = \underset{\theta}{argmin} \parallel \varepsilon - \varepsilon_\theta(f_m, m) \parallel^2 \tag{6.2}$$

f_0 denotes the latent feature of the image extracted by the encoder. Let m be the number of steps of adding noise, f_m the resulting noised feature after m steps, and $\bar{\alpha}_m$ a constant that controls the noise level added at each step. Eq. (6.2) serves as the main optimization target.

6.3.2 Blurry-Clean Image Pairs Rendering

As illustrated in Fig. 6.4 (a) and (b), the clean images are generated by rendering images from an initial D3D-GS model trained on the original training set. To produce the blurry image set, a deformation bias β_δ is added to the Gaussian splat, simulating the appearance of regions outside the training distribution. The uncertainty of a Gaussian splat correlates with its deformation range, where regions with low dynamics are better constrained due to being observed in multiple frames. After adding the bias, Eq. (5.3) can be rewritten as Eq. (6.3), where λ_β is a hyperparameter to control the amount of bias to be added, and $\mathcal{N}$ is a normal distribution.

$$\beta_\delta = (1 + \lambda_\beta \mathcal{N})\delta_t, \ \hat{\mathcal{G}}_t = \mathcal{G} + \beta_\delta \tag{6.3}$$

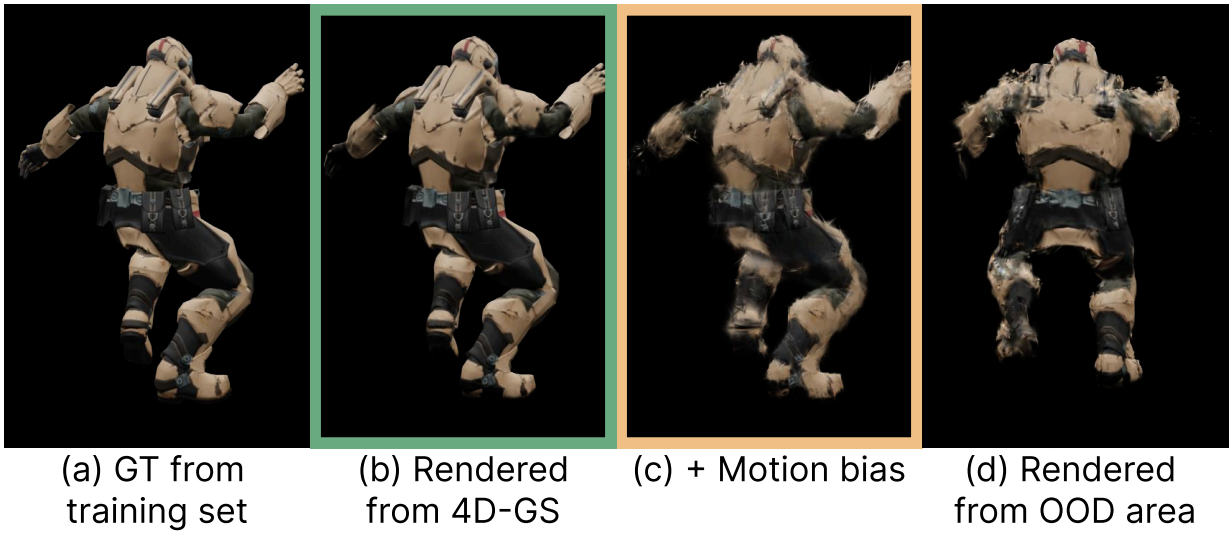


Figure 6.5: The visualization of adding the motion related bias. (a) GT image from the training set, (b) the rendered image from the initialized D3D-GS, (c) the rendered image with motion bias added, (d) the rendered image from the same timestep but different viewpoint. The blurry-clean image pair is formed by (b) and (c).

Fig. 6.5 illustrates the effect of adding motion-related bias. Fig. 6.5 (a) shows an image from the training set, while Fig. 6.5 (b) shows the corresponding image rendered by the D3D-GS model trained on that set. The model can generate high quality images

for viewpoints within the training distribution. In contrast, Fig. 6.5 (d) presents an image rendered at the same timestep but from a different viewpoint not seen during training. This OOD rendering results in noticeable blurring, particularly in highly dynamic regions such as the arms and legs. Fig. 6.5 (c) depicts an image rendered from a training viewpoint with the proposed motion bias applied. The visualization demonstrates that the added motion bias effectively simulates the OOD blurry appearance. Fig. 6.5 (b) and (c) form the blurry-clean image pair used for the diffusion model finetuning in Sec. 6.3.3 (as shown in Fig. 6.4 (c)).

6.3.3 Denoising Diffusion Model Tuning

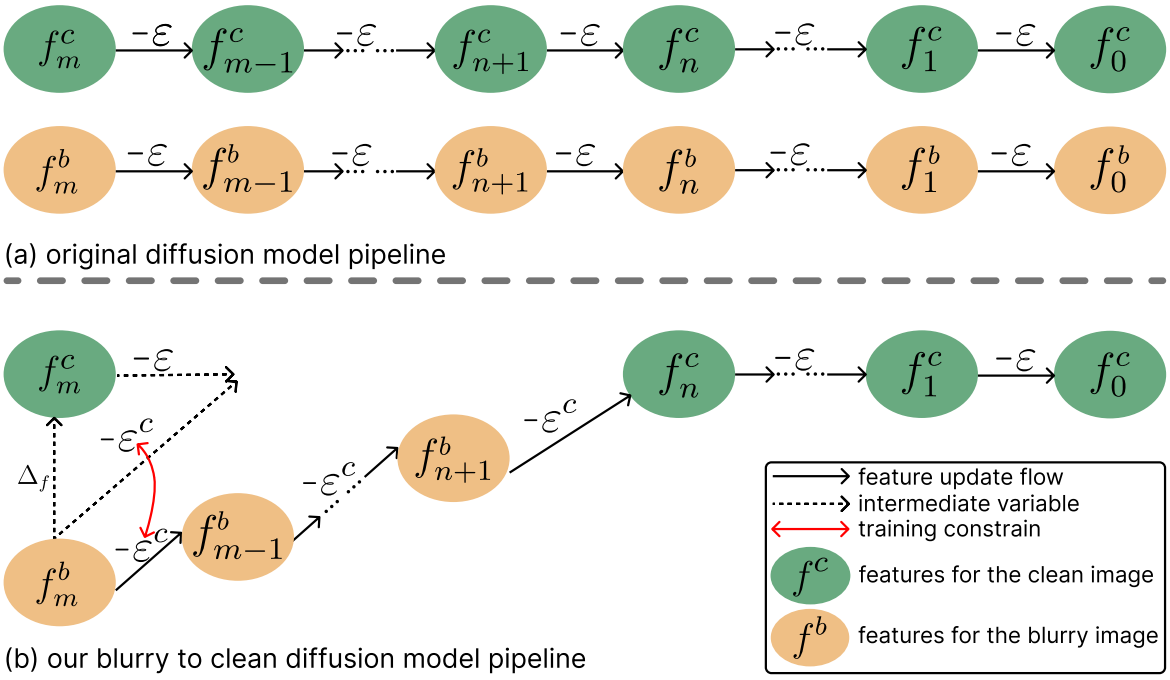


Figure 6.6: The comparison of the original and our blurry to clean diffusion reverse process. The variable above each arrow indicate the gradient. (a) The original diffusion reverse process. The diffusion of clean and blurry features are independent. (b) The pipeline of our blurry to clean diffusion pipeline. The gradient of the feature update depends on both, the noise and the feature difference.

Previous deblurring methods usually focused on the motion [223] or defocus blur [224], which are different from the blur type in this aspect. Finetuning their models are impractical since each object scene usually contains only a few hundreds of image pairs. Moreover, previous methods [225–227] usually aim at learning the pixel level refinement, which consider less of the object attribute. To overcome above limitations, we propose a novel deblurry framework for object specified applications. Our approach leverage a pretrained diffusion model to provide a strong image generation ability, which is finetuned by both the clean image to learn the appearance of the object, and the blurry image to learn the concept of deblurring.

The original training objective in Eq. (6.3) is not suitable for this task, as the standard U-Net is designed to predict only the added noise ε. Fig. 6.6 (a) shows the reverse process of the original diffusion procedure. The pipeline for clean (f^c) and blurry (f^b) images are independent without any interaction. Thus the difference between clean and blurry features is not considered.

As in Fig. 6.6 (b), we adapt the pretrained U-Net to predict not only the noise ε but also the difference Δ_f between the latent features of the blurry image f_m^b and the clean image f_m^c at the same noise level m. In the beginning, the reverse process denoises f_m^b by ε^c at each step to push it closer to f_m^c. According to Eq. (6.1), the noise ε^c can be calculated as:

$$\Delta_{f_m} = \sqrt{\bar{\alpha}_m}(f_0^c - f_0^b), \varepsilon^c = \varepsilon - \Delta_{f_m} \tag{6.4}$$

$$\varepsilon = \begin{cases} \varepsilon & m \leq n \\ \varepsilon^c & m > n \end{cases} \tag{6.5}$$

However, training involves a single noise addition step, but inference proceeds through gradual denoising across multiple steps. In other words, Δ_{f_m} is a constant value during training, but it is optimized during the inference. As a result, for certain timesteps $0 < n < m$, the latent features f_n^b and f_n^c may already be close enough. Continuing to use ε^c as the target would import undesired bias. To address this, we introduce a timestep-controlled switching function that dynamically selects the denoising target, as formulated in Eq. (6.5). n controls the point at which we switch from the blurry-to-clean noise ε^c to the original clean-to-clean noise ε. This adaptive strategy defines the structure of our denoising diffusion model finetuning, as illustrated in Fig. 6.4 (c).

6.3.4 Generating Novel Views Pseudo GT Images

Image rendering is determined by two inputs, the time t and the camera pose P_t, a novel view image must use a time and camera pose combination different from the training set. The distribution of the training set is illustrated on the right of Fig. 6.2 where novel views are sampled from the green areas representing unseen combinations. Therefore, our strategy is to rematch the timesteps and camera poses from the training to form novel view combinations as:

$$\bar{t} = t_i, \bar{P}_{\bar{t}} = P_{t_j}, i \neq j \tag{6.6}$$

t_i and P_{t_j} are sampled from the training set at different timestep. $\bar{t}$ and $\bar{P}_{\bar{t}}$ denote the samples used for novel view image rendering. These new samples are first passed through the initial D3D-GS (Fig. 6.4 (a)) to render initial blurry images, which are then refined by the finetuned diffusion model from Fig. 6.4 (c). The full pipeline for generation of novel views is illustrated in Fig. 6.4 (d). The rendered image $\bar{I}$, along with $\bar{t}$ and $\bar{P}$, are then incorporated into the pseudo ground truth set. The novel views in the pseudo ground truth set support training in the OOD regions.

6.3.5 Learning from Inaccurate Samples

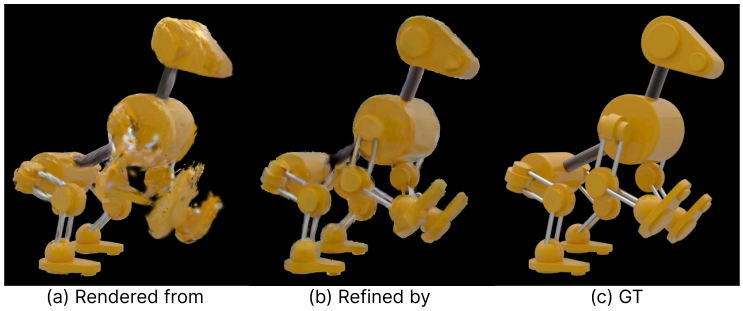


Figure 6.7: The visualization of the denoising diffusion result. (a) the original image rendered from OOD set, (b) the image refine by the denoising diffusion. (c) the ground truth image.

The training sets of the 4D representation dataset usually contain a few hundred samples, which limits the number of image pairs available for diffusion model finetuning. As

a result, the denoised images, while improved, may still not match the accuracy of those from the original training set. Since the OOD regions are typically under-constrained, the estimated object poses in these areas may be less reliable. Therefore, directly incorporating the novel view pseudo GT set into the training may introduce undesirable artifacts. Fig. 6.7 shows a comparison between the original OOD image, the diffusion model refined image and the ground truth. Although the refined image is better than the original, it still does not exactly match the ground truth. This discrepancy is due to the inherent uncertainty in the diffusion model and the limited size of the finetuning dataset.

To effectively learn from inaccurate samples, we upgrade the D3D-GS structure as illustrated in Fig. 6.4 (e). First, due to the limited control in the diffusion and the imprecise initial object pose in $\bar{I}_{\bar{t}}$, the output from the diffusion model may correspond to a slightly different timestep, denoted as $\bar{I}_{\bar{t}+\epsilon}$. To address this, we introduce a trainable time bias to estimate the time shift ϵ for each novel generated sample. Second, the refined image may exhibit color inconsistency (compared between Fig. 6.7 (b) and (c)). To mitigate this, we fix the color and opacity of the Gaussian splats during training for novel samples, updating only the position, rotation, and scale to constrain the physical shape of the object. Finally, we reduce the loss weight for the generated samples to avoid instability caused by their potentially inaccurate constraints. For the original training samples, we add an extra alpha channel loss to better constrain the object boundaries, as the input images are masked using object segmentation.

6.3.6 Monocular Scanning Video Dataset

To fully illustrate the capability of our method, we generate a 4D object representation dataset, named MoSc dataset, from a scanning video that follows the camera pose sampling strategy illustrated in Fig. 6.3 (c). The dataset includes six synthetic and two real-world recorded test scenes. A key challenge in scanning video 4D object representation is the difference between the spatial-temporal distributions of the training and test sets. To quantify this, we evaluate the Root Mean Squared Error (RMSE) of the normalized location and orientation over a temporal interval between the training and test set.

Fig. 6.8 compares the estimated distribution differences between the training and test sets for our dataset and five widely used monocular 4D representation datasets. The location difference, shown on the left of Fig. 6.8, indicates that our dataset has a higher location disparity at small time intervals, demonstrating that the training and test sets follow different distributions. As the time interval approaches 1, which means the entire

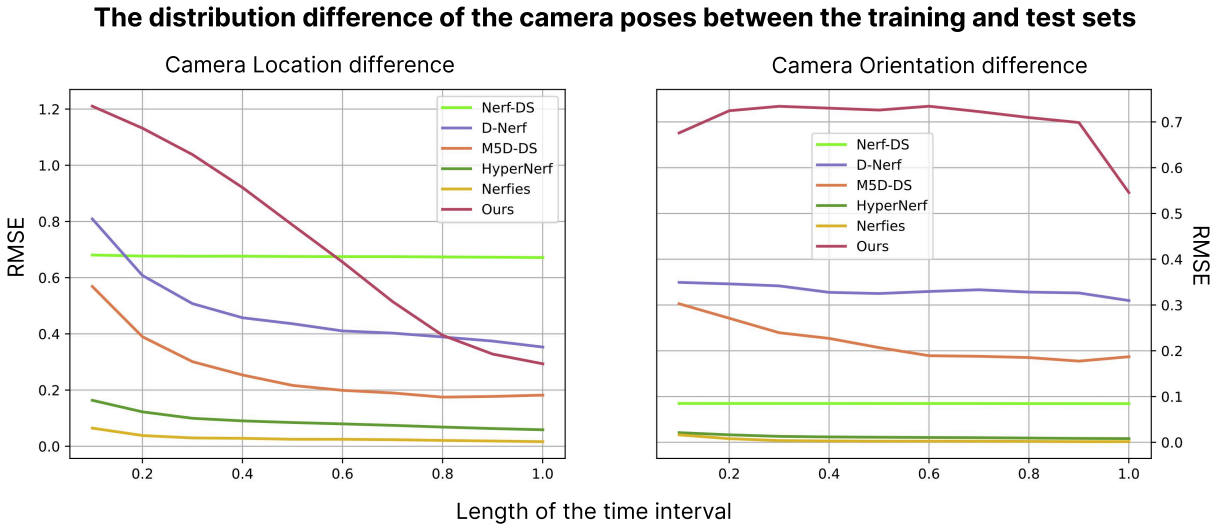


Figure 6.8: The distribution difference between the training and test set. The location difference is on the left, and the orientation difference is on the right.

training and test sets are averaged, the location difference decreases. This is because both the training and test sets eventually cover the entire object, resulting in similar average camera positions near the object center. In contrast, the orientation difference on the right of Fig. 6.8 remains high in our dataset, indicating a greater variation in viewing angles. This highlights the challenges in reconstructing 4D objects from scanning videos because of the distribution shift between training and test views.

6.4 Experiment

6.4.1 Experiment Setup

We select eight SOTA 4D-GS methods for comparison. Gaussian-flow [228] and MotionGS [214] failed to generate reasonable results for most test cases, and were thus excluded from the final evaluation. The final comparisons are conducted between our method and the following six approaches: D3D-GS [119], SC-GS [117], 4DGS-CVPR [118], 4DGS-ICLR [206], 4D-Rotor [218], and Gaussian Prediction [217]. Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and VGG-based Learned Perceptual Image Patch

Similarity (LPIPS) [208] are used for evaluation. These metrics comprehensively assess image quality from pixel-level accuracy to perceptual similarity.

6.4.2 Training Details

We give training details of the entire workflow in this section. The entire pipeline contains multiple processing steps with different models. Full details of the training setup can be found in the source code.

D3D-GS. We follow the same training setup as the original D3D-GS method [206] in the initialization stage. Specifically, the number of training steps is 40000, the initial learning rate is $1.6e-4$, and the batch size is set to 1. The Gaussian splat density control triggers every 100 training steps from 500 to 15000 steps. The first 3000 steps are used to initialize the Gaussian splat structure without training the deformation field, which works as the warm up stage. During the stage of learning from inaccurate images, the learning rate reduction coefficient for the generated samples is gradually lowered from 0.55 to 0.3. The coefficient for the alpha channel boundary loss is 0.01.

Blurry Image Rendering. As introduced in the main manuscript, we add a motion-related bias into the well trained viewpoints to simulate the appearance of under-constrained viewpoints. The coefficients of bias added for the Gaussian splat's location, orientation, and scales are 0.1, 0.3, and 0.8, respectively.

Diffusion Model. For the finetuning of the diffusion model, we follow the setup of DreamBooth [222]. A prompt "sks" is used to import the new concept of the object. The resolution is set to 800×800, the batch size is set to 1. The entire finetuning process contains 1200 training steps with a fixed learning rate of $1e-5$. Stable Diffusion V1.5 is selected for the initial weights since it is pretrained well on large scale datasets. For inference, the language guidance scale is set to 0.45, the maximum step is set to 1000, and the strength is 0.55.

6.4.3 Dataset Generation

To fully explore the performance on monocular scanning video 4D object representation, we generated our MoSc dataset containing both real-world recordings and synthetic testcases. The basic pipeline is the same as described in 5.4.2. The differences are discussed in this section.

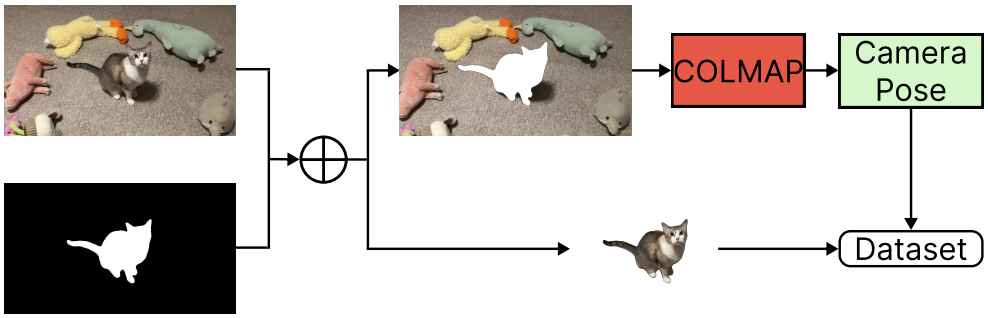


Figure 6.9: The pipeline of generating real-world recorded testcases. The shown image sample is from the real-cat testcase.

Synthetic Testcases. To generate synthetic testcases, the Blender engine is selected since it is widely used in the community and supports Python for coding. The camera in the training set moves around the object in a pre-defined continuous trajectory over time, and the test camera is randomly selected around the object at each step. The individual camera poses are sampled uniformly into 200 viewpoints around the object from the entire trajectory, which are assigned with an increasing count as the timestep.

Real-World Testcases. Two cameras are placed at the opposite side of the object and record the same action for training and for the test set. The original videos are recorded at 60 fps, and then uniformly downsampled to 6 fps. The two videos are manually aligned in the temporal dimension. Fig. 6.9 shows the pipeline for the spatial dimension collaboration. The video frames are first segmented to obtain the target object. COLMAP is used to calibrate the camera poses and optimize the camera intrinsic matrix based on background images without the influence of the dynamic object. The whole testcase is post-processed by the Nerfies [113] script to normalize the camera poses and calculate the scene information.

6.4.4 Results and Comparisons

Main Results. Tab. 6.1 presents the quantitative comparison between our method and competing approaches on the MoSc dataset. The best result is bold and the second best is underlined. "Fails" in the table indicates cases where the method fails to produce meaningful outputs. The two testcases in the last column correspond to real-world captured scenes. Our method not only achieves the best performance across all metrics but

Method	Stand Up			Robot			Dance			Real-Robot		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SC-GS	Fails			Fails			23.146	0.897	0.080	Fails		
4DGS (iclr)	27.382	0.926	0.063	Fails			21.412	0.848	0.136	Fails		
4D-Rotor	17.243	0.830	0.180	14.746	0.732	0.271	17.203	0.791	0.210	Fails		
4DGS (cvpr)	21.966	0.929	0.065	19.488	0.895	0.115	21.226	0.863	0.101	25.799	0.932	0.070
D-3DGS	22.675	0.936	0.056	21.801	0.886	0.116	21.990	0.897	0.106	26.035	0.831	0.061
G-Prediction	26.979	0.953	0.037	26.147	0.939	0.058	22.789	0.907	0.078	26.000	0.935	0.060
Ours	**29.746**	**0.964**	**0.036**	**29.194**	**0.957**	**0.054**	**26.018**	**0.933**	**0.059**	**29.445**	**0.961**	**0.042**
Method	Rob-Dog			Elephant			Dinasor			Real-Cat		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SC-GS	Fails			23.583	0.881	0.096	24.892	0.930	0.055	18.099	0.824	0.139
4DGS (iclr)	20.583	0.849	0.146	22.264	0.840	0.139	23.097	0.881	0.106	21.975	0.869	0.130
4D-Rotor	15.920	0.770	0.229	17.880	0.777	0.201	17.292	0.811	0.186	Fails		
4DGS (cvpr)	20.813	0.905	0.095	19.435	0.899	0.110	22.865	0.921	0.073	16.014	0.905	0.092
D-3DGS	22.865	0.940	0.084	22.469	0.901	0.106	23.761	0.920	0.068	22.461	0.905	0.088
G-Prediction	22.556	0.931	0.065	24.224	0.912	0.086	24.418	0.925	0.054	22.191	0.900	0.088
Ours	**27.547**	**0.955**	**0.046**	**28.448**	**0.943**	**0.076**	**26.457**	**0.936**	**0.059**	**23.812**	**0.916**	**0.081**

Table 6.1: Quantitative results on MoSc dataset. PSNR, SSIM, and LPIPS(VGG) are evaluated for each competitor. The best result is **bold** and the second best is underlined. The two testcases in the last column are the real-world recorded scenes.

also demonstrates strong robustness, successfully handling all test cases without failure. Fig. 6.10 shows qualitative comparisons. Red crosses marks failure cases. The ground truth images are shown in the last column and the results from our MoSc-GS appear in the second last column. Compared to other methods, MoSc-GS consistently renders the highest-quality images in challenging out-of-distribution regions.

	PSNR ↑	SSIM ↑	LPIPS ↓
Initial 4D-GS	24.1925	0.9065	0.0944
Pre-trained Stable Diffusion	23.2798	0.8964	0.1033
Baseline Finetuning	23.6167	0.9099	0.0982
Our Finetuning	**28.4437**	**0.9430**	**0.0684**

Table 6.2: The result for diffusion model.

Evaluation on the Diffusion model. To evaluate the performance of the finetuned diffusion model, a small subset of samples from the OOD region is held out as a validation set. Tab. 6.2 presents the quantitative results, while Fig. 6.11 shows the corresponding visualizations. Although the pretrained Stable Diffusion model produces smoother images, its accuracy (e.g., PSNR) drops. This is attributed to PSNR's sensitivity to pixel-wise color differences. Stable Diffusion often alters both the appearance and structure of the image,

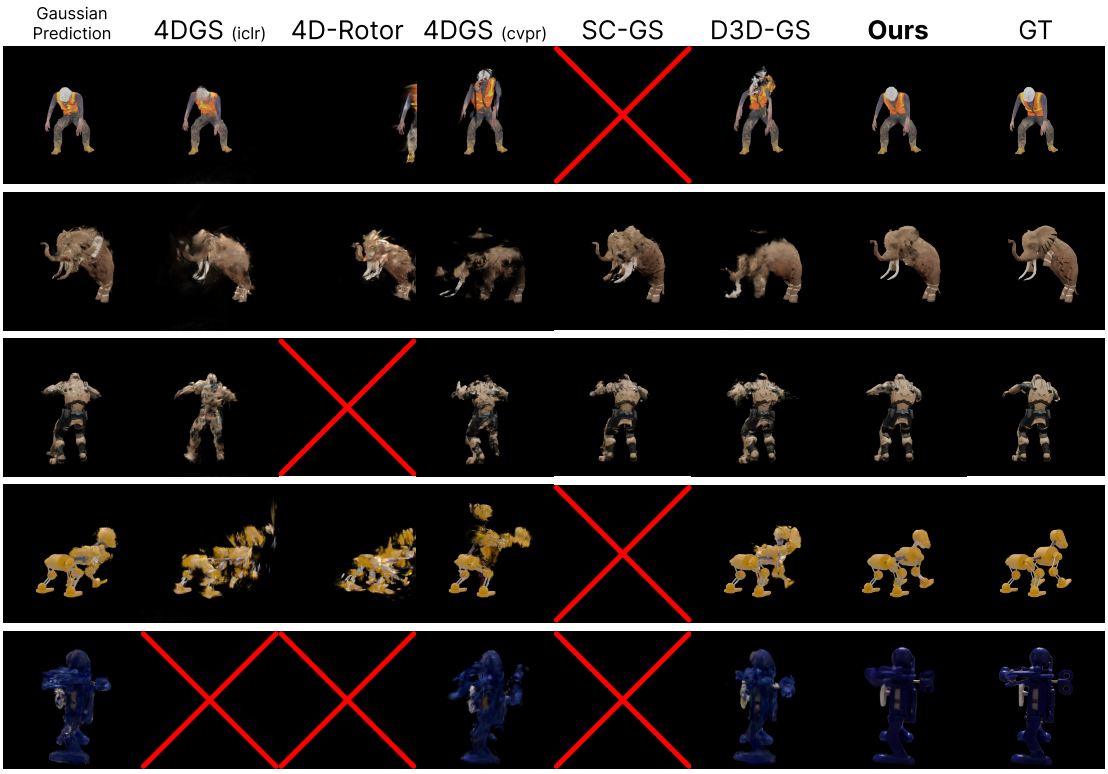


Figure 6.10: Qualitative examples from our MoSc dataset for each competitor. The red cross indicates that the method cannot handle a particular scene. The last row shows a real-world recorded scene. Zoom in for more detailed views.

leading to lower metric scores despite better perceptual quality. A similar trend is observed for the diffusion model finetuned using the baseline training strategy (Eq. (6.2)). In contrast, our proposed finetuning strategy yields a substantial improvement in all metrics, demonstrating its effectiveness in preserving both structural integrity and visual fidelity in OOD regions.

6.4.5 Ablation study

Ablation study for each component. The performance at each stage are evaluated to understand their individual contributions. Fig. 6.12 presents visualized results after each step, along with corresponding PSNR scores. To enhance visibility of improvements, the

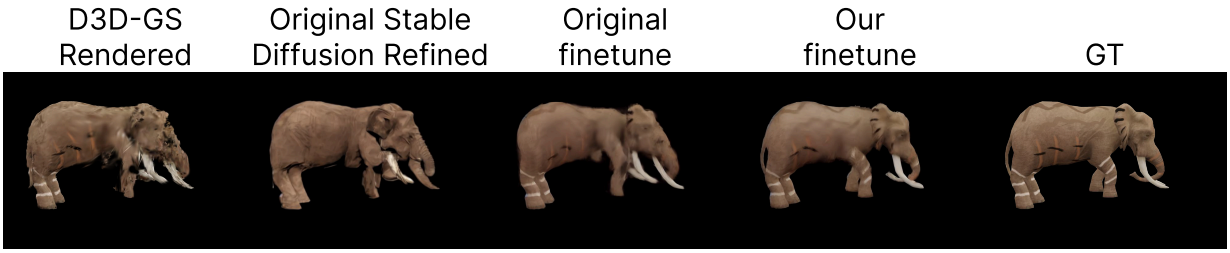


Figure 6.11: Qualitative examples for each block. Zoom in for more detailed views. The PSNR values are presented on each image.

	PSNR ↑	SSIM ↑	LPIPS ↓
baseline	23.007	0.915	0.086
+boundary	24.654	0.921	0.077
+novel view	27.583	0.946	0.058

Table 6.3: Ablation results on MoSc dataset for each component.

background of the RGBA images is set to white (except for the diffusion model output), whereas black backgrounds were used during training. In the baseline method (first column of Fig. 6.12), overfitting to the training samples is common, often resulting in meaningless Gaussian splats around the object that resemble the background color. Introducing an object boundary constraint during training significantly mitigates this issue, producing a more well-defined object structure (second column of Fig. 6.12). This improvement is further enhanced when novel views are incorporated into training (forth column of Fig. 6.12). The third column shows the refined image by feeding the second column into the finetuned Stable Diffusion. While the diffusion process generate perceptually better images, it remains difficult to control and can occasionally produce results with lower PSNR (as seen in the second row of Fig. 6.12). Our proposed learning from inaccurate samples strategy overcomes this limitation by effectively integrating the noisy diffusion output with clean training data. This approach enables the model to recover structural details better than the raw generated image alone. Tab. 6.3 summarizes the average quantitative performance across the MoSc dataset for each pipeline component. The results clearly demonstrate that each designed module contributes meaningfully to improving the overall reconstruction quality.

Evaluation on the entire spatial-temporal space. To further analyze how our

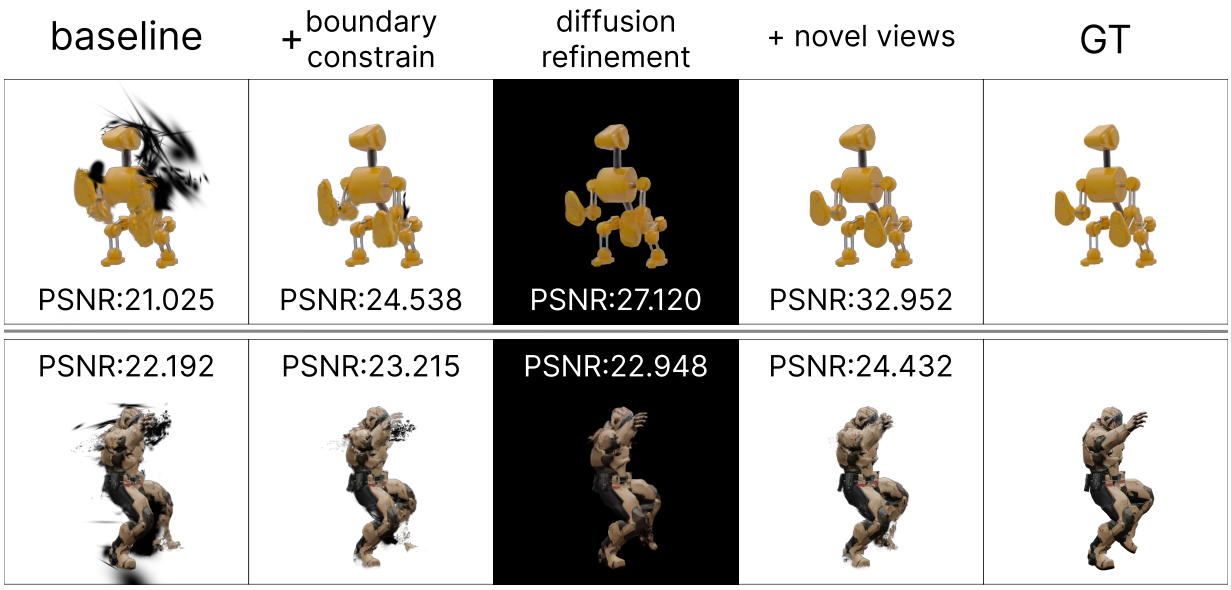


Figure 6.12: Qualitative examples for each block. Zoom in for more detailed views. The PSNR values are presented on each image.

method enhances dynamic object representation in the spatio-temporal domain, we generated a synthetic scene with image frame matrix. The PSNR matrices are visualized in Fig. 6.13. Specifically, 180 time steps are uniformly sampled along the vertical axis, while 180 viewpoints are uniformly sampled around the object along the horizontal axis. The training images are selected from the diagonal. As expected, the PSNR values are high on the diagonal due to direct supervision. As shown in the left panel of Fig. 6.13, the baseline approach exhibits strong overfitting to the training views, with significantly lower PSNR values in off-diagonal (unseen) regions. Introducing the object boundary constraint (middle panel) eases this overfitting, resulting in improved generalization. Finally, by incorporating the generated novel views into training (right panel), our method substantially boosts the overall PSNR across the full spatio-temporal domain, demonstrating improved representation capability for dynamic scenes.

Blurry-clean noise curve. As mentioned in 6.3.2, the original diffusion model can only take one type of image as input from blurry and clean images. Fig. 6.14 shows the noise differences between the predicted and ground truth noise. We can see that there is a clear gap between the clean-clean noise and the blurry-blurry noise. The gap increases when the noise step gets smaller. Our blurry-clean finetuning strategy can work as designed

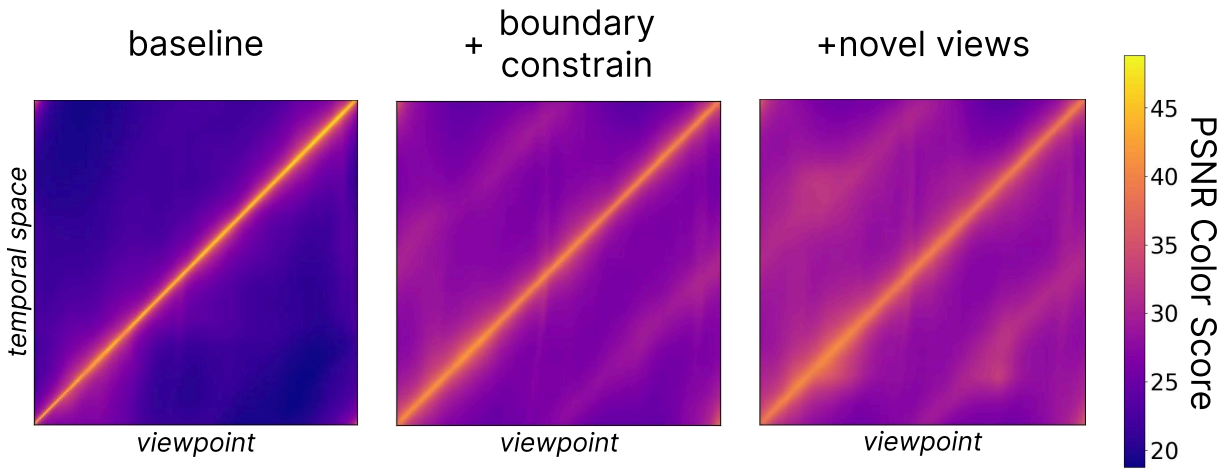


Figure 6.13: The PSNR evaluation of the spatial-temporal space of a synthetic testcase. The horizontal axis is the spatial space, and the vertical axis represents the temporal space. The training samples are selected from the bottom left to top right diagonal.

to refine the blurry image. Specifically, the blurry input image is obtained by first adding noise to a random image (blue curve from left to right). Then, it is denoised by our strategy to generate a clean image (green curve from right to left). The red curve shows how a clean image is processed in the original diffusion model.

Visualized denoising results. Fig. 6.15 shows some additional visualized results of our blurry-clean diffusion model. The first row shows the raw images rendered by the initial D3D-GS model from the out of distribution area of the training set. Artifacts and blur can be observed in the images. The bottom row shows the images refined by the blurry-clean diffusion model. Although the image is not perfect but the object has a reasonable appearance. The image is clearly much better than the blurry image even with cleaned color and boundary.

Comparison with other deblurring methods. As mentioned in 6.3.2, previous image deblurring methods usually focus on motion, defocus, or weather (e.g., image detoriation due to rain and snow). They all consider only pixel level refinement without object appearance. We provide visual comparisons with previous methods as shown in Fig. 6.16. Four widely used baseline methods are compared, including FFformer [229], Stripformer [230], Restormer [231], and MIMO-UNet [226].

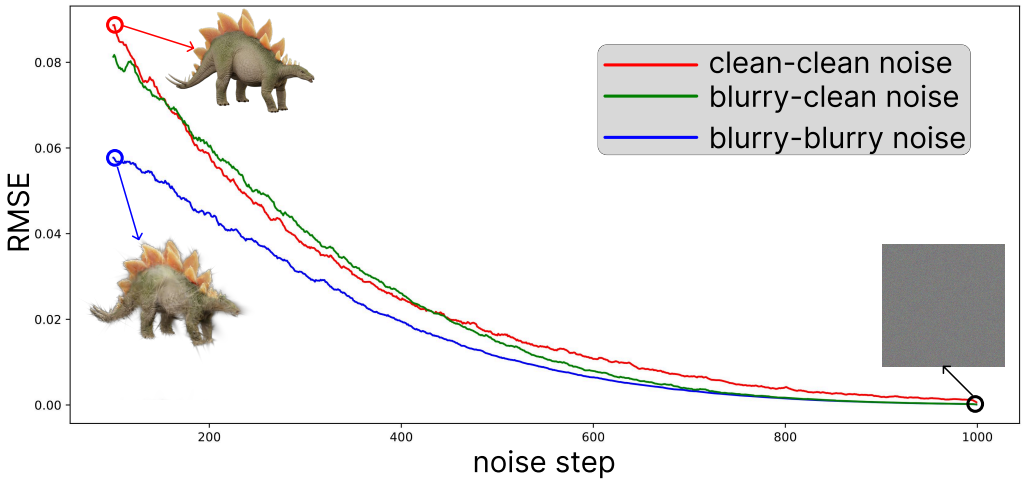


Figure 6.14: The noise curve of the original diffusion model under clean-clean denoising, blurry-blurry denoising, and our finetuned blurry-clean denoising.

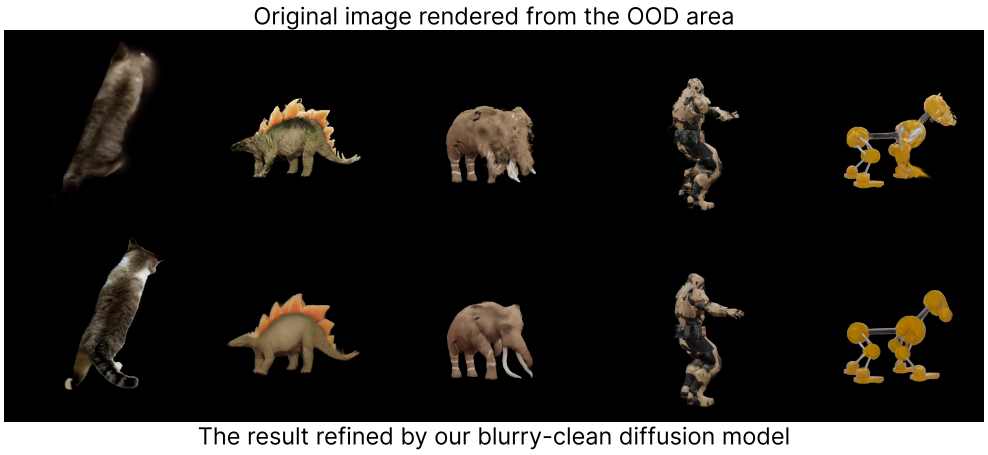


Figure 6.15: Visualization of the blurry-clean diffusion model result. The top row shows the images rendered by the initial D3D-GS. The bottom row shows the images refined by our blurry-clean diffusion model.

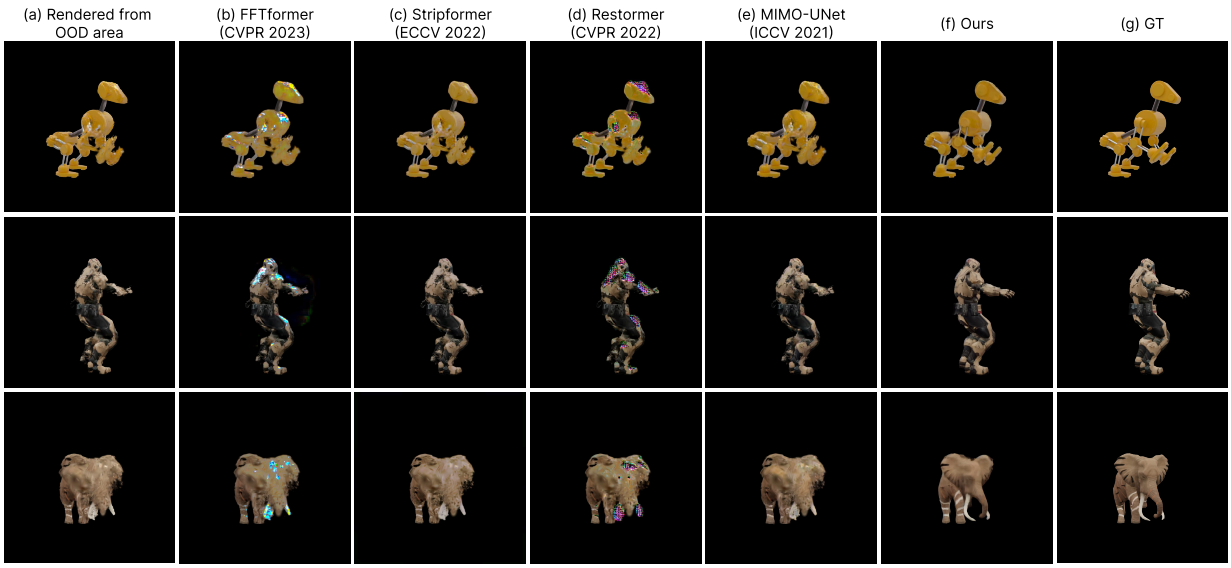


Figure 6.16: Comparison with other deblurring methods.

6.5 Conclusion

In this chapter, we propose MoSc-4DGS, a method that reconstructs complete object geometry and motion from a single scanning sequence. Unlike previous monocular 4D datasets that either capture only partial views or rely on unrealistic camera trajectories, MoSc-4DGS enables full object reconstruction under real-world monocular constraints. The core challenge in this task lies in the significant distribution gap between training and test views. To address this, we introduce a novel diffusion finetuning strategy that enhances semantic understanding and image fidelity. Our novel structure of D3D-GS is designed to render blurry images for diffusion model training, and learn precise features from the imprecise generated samples. To validate our approach, a new monocular scanning video dataset is proposed containing both synthetic and real-world sequences. MoSc-4DGS achieves SOTA results and shows strong generalization across benchmarks. However, some limitations are remained. as potential future work directions. First, our method only finetuned the UNet of the diffusion model, which improves the overall quality but may still not work well for some details. Second, the decoder can also be tuned in the diffusion model, which would generate more accurate images. Last, our method is agnostic to motion types; incorporating explicit motion priors, such as keypoints, could improve handling of

complex dynamics. These directions offer avenues for future.

7

Conclusion

In this thesis, we explore the challenge of reconstructing complete dynamic objects from casually captured monocular scanning videos for real-world applications. We decouple the problem into two tasks: Referring Video Object Segmentation (RVOS) and dynamic object reconstruction. We first investigate the limitations of existing approaches (Sec. 1.4), and in the subsequent chapters, we propose novel methods to address these limitations.

In Chapter. 3, we focus on improving the temporal consistency of the RVOS task. Previous research typically extends image-based object segmentation frameworks to video-based settings without sufficiently considering temporal consistency. This weakness leads to significant drop in accuracy when the object is not well observed across frames. To address these limitations, we introduce a temporal context–enhanced (TCE) framework, in which features continuously communicate across frames. To fully demonstrate the contribution of our TCE-RVOS method, we also analyze and categorize a widely used benchmark into various challenging scenarios (e.g., occlusion, motion). The proposed method achieves significant accuracy improvements, particularly under these challenging conditions.

In Chapter. 4, we further extend the usability of the RVOS framework. Early research typically formulated RVOS as an offline approach, where all video frames are fed into the model and segmentation masks are generated simultaneously. Such offline frameworks

perform well on short video datasets, as each frame has access to both past and future information. However, this approach is impractical for long or ongoing video processing, as it requires a large amount of GPU memory. When the entire video is split into chunks of smaller sizes that fit into memory of the offline model, accuracy drops significantly. To overcome this limitation, we propose a filtering-based RVOS framework that can process videos in offline, online, and semi-online modes with only a single training session. FF-RVOS achieves state-of-the-art accuracy with stable performance across various chunk sizes. Moreover, our filtering framework can be integrated into other offline approaches to enhance their temporal consistency when processing long or ongoing videos.

The remaining two chapters focus on the task of dynamic object representation. In Chapter. 5, we address the challenge of representing objects with large dynamics. Previous methods typically separate an object's static structure and its deformation field into two sub-models, which struggle to handle objects with large dynamic ranges. To overcome this limitation, we designed a novel framework that decouples local object deformations from global large-scale motions. Furthermore, we proposed a new benchmark featuring objects with large motions to demonstrate the advantages of our approach. Our M5D-GS does not only improve the robustness of dynamic object representations but also achieves state-of-the-art performance in both accurate object reconstruction and motion estimation.

In Chapter. 6, we first discuss the limitations of using monocular videos to reconstruct dynamic objects. A single monocular video can only provide partial 3D information, which is insufficient to fully constrain the 4D reconstruction. Previous benchmarks employ certain tricks to mitigate this limitation, but these approaches also restrict their applicability in real-world scenarios. To address this, we generated a novel benchmark for reconstructing dynamic objects from monocular scanning videos. To compensate for the unconstrained spatiotemporal regions of the dynamic objects, we propose a diffusion-based generative model. Additionally, the deformable 3D-GS is enhanced to learn reliable structures from both precise ground-truth views and imprecise generated views. Our MoSc-GS successfully reconstructs complete object models with motion using only casually captured monocular scanning videos.

7.1 Thesis Contributions

This thesis addresses the problem of reconstructing complete 4D dynamic object models from casually captured monocular scanning videos. Two sub-tasks are investigated: Refer-

ring Video Object Segmentation (RVOS) and 4D object reconstruction. Various limitations of previous approaches are addressed in this thesis as the main contributions. In summary, the contributions of this thesis are as follows:

- An offline RVOS framework that enhances temporal consistency between video frames, improving both, overall accuracy on public benchmarks and robustness under various challenging scenarios, including but not limited to object occlusion, motion, and ambiguity (Chapter. 3).
- A semi-online RVOS pipeline achieved by integrating a filtering framework. The proposed framework requires only a single training session but can perform inference in offline, semi-online, and online modes. This allows the framework to be deployed on various devices without retraining (Chapter. 4).
- A novel 4D representation framework that simultaneously reconstructs the object's canonical model, local deformations, and global motion. The proposed method achieves state-of-the-art accuracy in large-motion scenarios and is robust to various types of object motion (Chapter. 5).
- A monocular scanning video–based 4D object representation framework. By integrating a diffusion-based generative model to compensate for unconstrained spatiotemporal regions of the dynamic object, this approach reconstructs complete 4D objects, increasing the usability of 4D representations in real-world applications (Chapter. 6).
- Two 4D reconstruction datasets targeting high-dynamic-range objects and monocular scanning videos, with the dataset generation pipeline and processing scripts publicly released.

These contributions not only improve performance on their respective tasks but also help bridge the gap between 4D representation research and potential applications targeted at the consumer level, such as 4D reconstruction, AR/VR, and animation.

7.2 Limitations and Future Work

This thesis addressed the task of reconstructing complete 4D object models from monocular scanning videos. To achieve this, Referring Video Object Segmentation (RVOS) was

employed to extract objects from video frames, and 4D-GS was developed for dynamic object representation. Despite the fact that this thesis successfully achieves its initial goal of reconstructing and representing complete 4D dynamic objects from casually captured monocular scanning videos, there remains room for improvement, which can serve as directions for future work.

For RVOS, with recent advances, foundation models have achieved remarkable success in segmentation tasks. Many powerful foundation models, such as DINOv3 [232] and Cosmos [233], have been introduced, trained on billions of visual samples for multiple tasks. In this context, designing task-specific models for image segmentation may no longer be an optimal choice, given the scale of available datasets and the generalization ability of foundation models. Future research could instead focus on areas where foundation models still face limitations. First, in long or ongoing videos, a textual expression may only correspond to a subset of frames, as the appearance of the object evolves over time. The adaptation and evolution of multi-modal representations across time is therefore a key direction. Second, temporal-based descriptions, such as object actions and behaviors, cannot be captured from a single image frame.

These limitations point to key directions for future research. One avenue is to develop methods that adapt and evolve multi-modal representations across time, enabling more accurate grounding of text expressions throughout entire video sequences. Another is to design approaches that more effectively integrate information across frames, allowing systems to capture temporal dependencies required for understanding actions and behaviors.

For 4D reconstruction, current approaches remain limited in two main aspects. First, reconstructed models typically lack physical attributes such as elasticity, friction, or density, which restricts their applicability in downstream tasks like physics-based simulation or realistic interaction. Second, while generative models can produce diverse outputs, they often lack controllability and may generate undesired objects, whereas reconstruction methods, though more precise, are constrained by strict requirements such as camera poses and viewing conditions.

These limitations open promising avenues for future research. Incorporating physical properties into reconstructed models would enable seamless integration with physical simulators, supporting more realistic rendering and interaction. Furthermore, combining the strengths of generative and reconstruction paradigms could reduce constraints while maintaining accuracy, allowing for the creation of flexible, controllable, and high-fidelity 4D object models.

As AI technology continues to advance, problems that are currently considered difficult are likely to become tractable. The methods and contributions presented in this thesis address several open challenges in 4D object reconstruction from monocular videos and provide directions for further research. These results highlight both the progress made and the opportunities that remain for advancing the field.

References

1. Sutherland, I. E. *A head-mounted three dimensional display* in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I* (1968), 757–764.
2. Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. *ORB: An efficient alternative to SIFT or SURF* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2011), 2564–2571.
3. Von Gioi, R. G., Jakubowicz, J., Morel, J.-M. & Randall, G. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32,** 722–732 (2008).
4. Levoy, M. & Whitted, T. The use of points as a display primitive (1985).
5. Lorensen, W. E. & Cline, H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics* **21,** 163–169 (1987).
6. Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer* **22,** 46–57 (2002).
7. Joo, H. *et al.* Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).
8. Lu, C.-Y. *et al. DiVa-360: The Dynamic Visual Dataset for Immersive Neural Fields* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024).
9. Hu, X., Long, L. & Lang, J. *Motion Decoupled 3D Gaussian Splatting for Dynamic Object Representation* in *AAAI Conference on Artificial Intelligence* **39** (2025), 3590–3598.
10. Seo, S., Lee, J.-Y. & Han, B. *URVOS: Unified referring video object segmentation network with a large-scale benchmark* in *Proceedings of the European Conference on Computer Vision* (2020), 208–223.
11. Ding, G., Sener, F. & Yao, A. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
12. Kirillov, A. *et al. Segment Anything* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 4015–4026.
13. Wu, J., Jiang, Y., Sun, P., Yuan, Z. & Luo, P. *Language as queries for referring video object segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 4974–4984.

14. Shen, Y. & Elhamifar, E. *Progress-aware online action segmentation for egocentric procedural task videos* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 18186–18197.

15. Ding, H., Liu, C., Wang, S. & Jiang, X. *Vision-Language Transformer and query generation for referring segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 16321–16330.

16. Huang, S. *et al. Referring image segmentation via cross-modal progressive comprehension* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 10488–10497.

17. Luo, G. *et al. Multi-task collaborative network for joint referring expression comprehension and segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 10034–10043.

18. Kamath, A. *et al. MDETR-modulated detection for end-to-end multi-modal understanding* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 1780–1790.

19. Yao, R., Lin, G., Xia, S., Zhao, J. & Zhou, Y. Video object segmentation and tracking: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* **11,** 1–47 (2020).

20. Wang, Y. *et al. End-to-end video instance segmentation with transformers* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 8741–8750.

21. Botach, A., Zheltonozhskii, E. & Baskin, C. *End-to-end referring video object segmentation with multimodal transformers* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 4985–4995.

22. Schönberger, J. L. & Frahm, J.-M. *Structure-from-motion revisited* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2016), 4104–4113.

23. Mildenhall, B. *et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* in *Proceedings of the European Conference on Computer Vision* (2020), 405–421.

24. Kerbl, B., Kopanas, G., Leimkühler, T. & Drettakis, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* **42,** 1–14 (2023).

25. Wang, C. *et al. DenseFusion: 6D object pose estimation by iterative dense fusion* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 3343–3352.

26. Bartoli, A., Gérard, Y., Chadebecq, F., Collins, T. & Pizarro, D. Shape-from-template. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37,** 2099–2118 (2015).

27. Kairanda, N., Tretschk, E., Elgharib, M., Theobalt, C. & Golyanik, V. *f-SFT: Shape-from-template with a physics-based deformation model* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 3948–3958.

28. Schulman, J., Lee, A., Ho, J. & Abbeel, P. *Tracking deformable objects with point clouds* in *IEEE International Conference on Robotics and Automation (ICRA)* (2013), 1130–1137.

29. Wu, D., Wang, T., Zhang, Y., Zhang, X. & Shen, J. *OnlineRefer: A simple online baseline for referring video object segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 2761–2770.

30. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2016), 770–778.

31. Liu, Z. *et al. Video swin transformer* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 3202–3211.

32. Hu, X., Hampiholi, B., Neumann, H. & Lang, J. *Temporal Context Enhanced Referring Video Object Segmentation* in *Winter Conference on Applications of Computer Vision* (2024), 5574–5583.

33. Luo, Z. *et al.* SOC: Semantic-assisted object cluster for referring video object segmentation. *Advances in Neural Information Processing Systems* **36** (2024).

34. He, S. & Ding, H. *Decoupling static and hierarchical motion perception for referring video segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 13332–13341.

35. Gavrilyuk, K., Ghodrati, A., Li, Z. & Snoek, C. G. M. *Actor and action video segmentation from a sentence* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 5958–5966.

36. Ding, H., Liu, C., He, S., Jiang, X. & Loy, C. C. *MeViS: A large-scale benchmark for video segmentation with motion expressions* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 2694–2703.

37. Hu, X., Neumann, H. & Lang, J. *A Filtering Framework for Semi-online Referring Video Object Segmentation* in *Proceedings of the 33rd ACM International Conference on Multimedia* (2025), 3222–3231.

38. Brand, M. & Kettnaker, V. Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22,** 844–851 (2002).

39. Brendel, W. & Todorovic, S. *Video object segmentation by tracking regions* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2009), 833–840.

40. Hu, R., Rohrbach, M. & Darrell, T. *Segmentation from natural language expressions* in *Proceedings of the European Conference on Computer Vision* (2016), 108–124.

41. Tang, J., Zheng, G. & Yang, S. *Temporal collection and distribution for referring video object segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 15466–15476.

42. Miao, B., Bennamoun, M., Gao, Y. & Mian, A. *Spectrum-guided multi-granularity referring video object segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 920–930.

43. Bai, Z. *et al.* One token to seg them all: Language instructed reasoning segmentation in videos. *Advances in Neural Information Processing Systems* **37,** 6833–6859 (2024).

44. Yan, C. *et al. ViSA: Reasoning video object segmentation via large language models* in *Proceedings of the European Conference on Computer Vision* (2024), 98–115.

45. Cuttano, C., Trivigno, G., Rosi, G., Masone, C. & Averta, G. SAMWISE: Infusing wisdom in SAM2 for Text-Driven Video Segmentation. *arXiv preprint arXiv:2411.17646* (2024).

46. Mikolov, T. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

47. Carreira, J. & Zisserman, A. *Quo vadis, action recognition? A new model and the Kinetics dataset* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), 6299–6308.

48. Khoreva, A., Rohrbach, A. & Schiele, B. *Video object segmentation with language referring expressions* in *Asian Conference on Computer Vision* (2019), 123–141.

49. He, K., Gkioxari, G., Dollár, P. & Girshick, R. *Mask R-CNN* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2017), 2961–2969.

50. Zhang, Y. *et al. Discriminative bimodal networks for visual localization and detection with natural language queries* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), 557–566.

51. Yu, L. *et al. MattNet: Modular attention network for referring expression comprehension* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 1307–1315.

52. Bellver, M. *et al.* RefVOS: A closer look at referring expressions for video object segmentation. *arXiv preprint arXiv:2010.00263* (2020).

53. Kenton, J. D. M.-W. C. & Toutanova, L. K. *BERT: Pre-training of deep bidirectional transformers for language understanding* in *Proceedings of NAACL-HLT* **1** (2019), 2.

54. Vaswani, A. *et al.* Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

55. Ding, Z. *et al.* Progressive multimodal interaction network for referring video object segmentation. *The 3rd Large-scale Video Object Segmentation Challenge* **8** (2021).

56. Liang, C. *et al.* Rethinking cross-modal interaction from a top-down perspective for referring video object segmentation. *arXiv preprint arXiv:2106.01061* (2021).

57. Zhu, X. *et al.* Deformable DETR: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159* (2020).

58. Ding, H., Liu, C., Wang, S. & Jiang, X. VLT: Vision-Language Transformer and Query Generation for Referring Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

59. Li, X. *et al.* R^2VOS: Robust Referring Video Object Segmentation via Relational Multimodal Cycle Consistency. *arXiv preprint arXiv:2207.01203* (2022).

60. Liu, H., Li, C., Wu, Q. & Lee, Y. J. Visual instruction tuning. *Advances in Neural Information Processing Systems* **36,** 34892–34916 (2023).

61. Jin, P., Takanobu, R., Zhang, W., Cao, X. & Yuan, L. *Chat-UniVi: Unified visual representation empowers large language models with image and video understanding* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 13700–13710.

62. Ravi, N. *et al.* SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714* (2024).

63. Yu, L., Poirson, P., Yang, S., Berg, A. C. & Berg, T. L. *Modeling context in referring expressions* in *Proceedings of the European Conference on Computer Vision* (2016), 69–85.

64. Lin, T.-Y. *et al. Microsoft COCO: Common objects in context* in *Proceedings of the European Conference on Computer Vision* (2014), 740–755.

65. Khoreva, A., Rohrbach, A. & Schiele, B. Video Object Segmentation with Language Referring Expressions. *arXiv preprint arXiv:1803.08006* (2018).

66. Pont-Tuset, J. *et al.* The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675* (2017).

67. Jhuang, H., Gall, J., Zuffi, S., Schmid, C. & Black, M. J. *Towards understanding action recognition* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2013), 3192–3199.

68. Hartley, R. & Zisserman, A. *Multiple view geometry in computer vision* (Cambridge University Press, 2003).

69. Wehr, A. & Lohr, U. Airborne laser scanning—an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing* **54,** 68–82 (1999).

70. Geng, J. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics* **3,** 128–160 (2011).

71. Zhang, Z. Microsoft Kinect sensor and its effect. *IEEE Multimedia* **19,** 4–10 (2012).

72. Scharstein, D. & Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* **47,** 7–42 (2002).

73. Faugeras, O. *Three-dimensional computer vision: a geometric viewpoint* (MIT Press, 1993).

74. Thrun, S. Probabilistic robotics. *Communications of the ACM* **45,** 52–57 (2002).

75. Longuet-Higgins, H. C. A computer algorithm for reconstructing a scene from two projections. *Nature* **293,** 133–135 (1981).

76. Fischler, M. A. & Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* **24,** 381–395 (1981).

77. Brown, D. *The bundle adjustment-progress and prospect* in *XIII Congress of the ISPRS, Helsinki* (1976).

78. Triggs, B., McLauchlan, P. F., Hartley, R. I. & Fitzgibbon, A. W. *Bundle adjustment—a modern synthesis* in *Vision Algorithms: Theory and Practice* (2000), 298–372.

79. Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* **82,** 35–45 (1960).

80. Gordon, N. J., Salmond, D. J. & Smith, A. F. M. *Novel approach to nonlinear/non-Gaussian Bayesian state estimation* in *IEE Proceedings F (Radar and Signal Processing)* **140** (1993), 107–113.

81. Mur-Artal, R., Montiel, J. M. M. & Tardos, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* **31,** 1147–1163 (2015).

82. Mur-Artal, R. & Tardós, J. D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics* **33,** 1255–1262 (2017).

83. Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M. M. & Tardós, J. D. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics* **37,** 1874–1890 (2021).

84. Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **60,** 91–110 (2004).

85. Engel, J., Schöps, T. & Cremers, D. *LSD-SLAM: Large-scale direct monocular SLAM* in *Proceedings of the European Conference on Computer Vision* (2014), 834–849.

86. Zhou, H. *et al.* StructSLAM: Visual SLAM with building structure lines. *IEEE Transactions on Vehicular Technology* **64,** 1364–1375 (2015).

87. Gomez-Ojeda, R., Moreno, F.-A., Zuniga-Noël, D., Scaramuzza, D. & Gonzalez-Jimenez, J. PL-SLAM: A stereo SLAM system through the combination of points and line segments. *IEEE Transactions on Robotics* **35,** 734–746 (2019).

88. Newcombe, R. A. *et al. KinectFusion: Real-time dense surface mapping and tracking* in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (2011), 127–136.

89. Tomasi, C. & Kanade, T. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision* **9,** 137–154 (1992).

90. Dai, Y., Li, H. & He, M. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision* **107,** 101–122 (2014).

91. Torresani, L., Yang, D. B., Alexander, E. J. & Bregler, C. *Tracking and modeling non-rigid objects with rank constraints* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* **1** (2001), I–I.

92. Fitzgibbon, A. W. & Zisserman, A. *Multibody structure and motion: 3D reconstruction of independently moving objects* in *Proceedings of the European Conference on Computer Vision* (2000), 891–906.

93. Vogel, C., Schindler, K. & Roth, S. 3D scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision* **115,** 1–28 (2015).

94. Chen, S. E. & Williams, L. *View interpolation for image synthesis* in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), 279–288.

95. Seitz, S. M. & Dyer, C. R. *Physically-valid view synthesis by image interpolation* in *Proceedings IEEE Workshop on Representation of Visual Scenes* (1995), 18–25.

96. Levoy, M. & Hanrahan, P. Light Field Rendering. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96),* 31–42 (1996).

97. Debevec, P., Yu, Y. & Borshukov, G. *Efficient view-dependent image-based rendering with projective texture-mapping* in *Rendering Techniques' 98* (1998), 105–116.

98. Shade, J., Gortler, S., He, L.-w. & Szeliski, R. *Layered depth images* in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), 231–242.

99. Zhou, T., Tucker, R., Flynn, J., Fyffe, G. & Snavely, N. Stereo Magnification: Learning View Synthesis using Multiplane Images. *ACM Transactions on Graphics* **37** (2018).

100. Maciel, P. W. C. & Shirley, P. *Visualizing complex environments using motion parallax* in *Proceedings of the 1995 Symposium on Interactive 3D Graphics* (ACM, 1995), 109–116.

101. Tateno, K., Tombari, F., Laina, I. & Navab, N. *CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), 6243–6252.

102. Bruno, H. M. S. & Colombini, E. L. LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing* **455,** 97–110 (2021).

103. Bescos, B., Fácil, J. M., Civera, J. & Neira, J. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters* **3,** 4076–4083 (2018).

104. Bescos, B., Campos, C., Tardós, J. D. & Neira, J. DynaSLAM II: Tightly-coupled multi-object tracking and SLAM. *IEEE Robotics and Automation Letters* **6,** 5191–5198 (2021).

105. Hu, X. & Lang, J. DOE-SLAM: dynamic object enhanced visual SLAM. *Sensors* **21,** 3091 (2021).

106. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B. & Revaud, J. *DUSt3R: Geometric 3D vision made easy* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 20697–20709.

107. Cabon, Y. *et al. MUSt3R: Multi-view network for stereo 3D reconstruction* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 1050–1060.

108. Tang, Z. *et al. MV-DUSt3R+: Single-stage scene reconstruction from sparse views in 2 seconds* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 5283–5293.

109. Wang, J. *et al. VGGT: Visual geometry grounded transformer* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 5294–5306.

110. Caron, M. *et al. Emerging Properties in Self-Supervised Vision Transformers* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021).

111. Curless, B. & Levoy, M. *A Volumetric Method for Building Complex Models from Range Images* in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), 303–312.

112. Zhang, J., Yao, Y. & Quan, L. *Learning signed distance field for multi-view surface reconstruction* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 6525–6534.

113. Park, K. *et al. Nerfies: Deformable neural radiance fields* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 5865–5874.

114. Park, K. *et al.* HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228* (2021).

115. Fridovich-Keil, S., Meanti, G., Warburg, F. R., Recht, B. & Kanazawa, A. *K-Planes: Explicit radiance fields in space, time, and appearance* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 12479–12488.

116. Westover, L. *Footprint evaluation for volume rendering* in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (1990), 367–376.

117. Yang, Z., Yang, H., Pan, Z., Zhu, X. & Zhang, L. *Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting* in *International Conference on Learning Representations* (2024).

118. Wu, G. *et al. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 20310–20320.

119. Yang, Z. *et al. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 20331–20341.

120. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

121. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* **33,** 6840–6851 (2020).

122. Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M. & Le, M. *Flow Matching for Generative Modeling* in *International Conference on Learning Representations* (2023).

123. Liu, X., Zhou, C. & Huang, S. 3DGS-Enhancer: Enhancing unbounded 3D Gaussian splatting with view-consistent 2D diffusion priors. *Advances in Neural Information Processing Systems* **37,** 133305–133327 (2024).

124. Yu, W. *et al.* ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048* (2024).

125. Liu, K., Shao, L. & Lu, S. Novel View Extrapolation with Video Diffusion Priors. *arXiv preprint arXiv:2411.14208* (2024).

126. Blattmann, A. *et al.* Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127* (2023).

127. Wu, J. Z. *et al.* Difix3D+: Improving 3D Reconstructions with Single-Step Diffusion Models. *arXiv preprint arXiv:2503.01774* (2025).

128. Lee, S. & Lee, G. H. *DiET-GS: Diffusion Prior and Event Stream-Assisted Motion Deblurring 3D Gaussian Splatting* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 21739–21749.

129. Wan, Y., Shao, M., Cheng, Y. & Zuo, W. *S2Gaussian: Sparse-View Super-Resolution 3D Gaussian Splatting* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 711–721.

130. Pumarola, A., Corona, E., Pons-Moll, G. & Moreno-Noguer, F. *D-NeRF: Neural Radiance Fields for Dynamic Scenes* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 10318–10327.

131. Yan, Z., Li, C. & Lee, G. H. *NeRF-DS: Neural radiance fields for dynamic specular objects* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 8285–8295.

132. Blender Online Community. *Blender - a 3D modelling and rendering package* Version 3.6. Blender Foundation (2023).

133. Johnson, E., Habermann, M., Shimada, S., Golyanik, V. & Theobalt, C. *Unbiased 4D: Monocular 4D reconstruction with a neural deformation model* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 6598–6607.

134. Li, T. *et al. Neural 3D Video Synthesis from Multi-view Video* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 5521–5531.

135. Broxton, M. *et al.* Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics* **39,** 86–1 (2020).

136. Feichtenhofer, C. *X3D: Expanding architectures for efficient video recognition* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 203–213.

137. Brezeale, D. & Cook, D. J. Automatic video classification: A survey of the literature. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **38,** 416–430 (2008).

138. Lei, P. & Todorovic, S. *Temporal deformable residual networks for action segmentation in videos* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 6742–6751.

139. Dosovitskiy, A. *et al.* An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929* (2020).

140. Ning, K., Xie, L., Wu, F. & Tian, Q. *Polar Relative Positional Encoding for Video-Language Segmentation* in *International Joint Conference on Artificial Intelligence* **9** (2020), 10.

141. Liu, S. *et al.* Cross-modal progressive comprehension for referring segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44,** 4761–4775 (2021).

142. Hui, T. *et al. Collaborative spatial-temporal modeling for language-queried video actor segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 4187–4196.

143. Hwang, S., Heo, M., Oh, S. W. & Kim, S. J. Video instance segmentation using inter-frame communication transformers. *Advances in Neural Information Processing Systems* **34,** 13352–13363 (2021).

144. Heo, M., Hwang, S., Oh, S. W., Lee, J.-Y. & Kim, S. J. VITA: Video instance segmentation via object token association. *arXiv preprint arXiv:2206.04403* (2022).

145. Oh, S. W., Lee, J.-Y., Xu, N. & Kim, S. J. *Video object segmentation using space-time memory networks* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 9226–9235.

146. Yang, Z., Wei, Y. & Yang, Y. *Collaborative video object segmentation by foreground-background integration* in *Proceedings of the European Conference on Computer Vision* (2020), 332–348.

147. Wu, J. *et al. In defense of online models for video instance segmentation* in *Proceedings of the European Conference on Computer Vision* (2022), 588–605.

148. Carion, N. *et al. End-to-end object detection with transformers* in *Proceedings of the European Conference on Computer Vision* (2020), 213–229.

149. Cheng, B., Misra, I., Schwing, A. G., Kirillov, A. & Girdhar, R. *Masked-attention mask transformer for universal image segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 1290–1299.

150. Liu, Y. *et al.* RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

151. Tian, Z., Shen, C. & Chen, H. *Conditional convolutions for instance segmentation* in *Proceedings of the European Conference on Computer Vision* (2020), 282–298.

152. Rezatofighi, H. *et al. Generalized intersection over union: A metric and a loss for bounding box regression* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 658–666.

153. Milletari, F., Navab, N. & Ahmadi, S.-A. *V-Net: Fully convolutional neural networks for volumetric medical image segmentation* in *IEEE International Conference on 3D Vision (3DV)* (2016), 565–571.

154. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

155. Ye, L., Rochan, M., Liu, Z. & Wang, Y. *Cross-modal self-attention network for referring image segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 10502–10511.

156. Ye, L., Rochan, M., Liu, Z., Zhang, X. & Wang, Y. Referring segmentation in images and videos with cross-modal self-attention network. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44,** 3719–3732 (2021).

157. Wang, H., Deng, C., Yan, J. & Tao, D. *Asymmetric cross-guided attention network for actor and action video segmentation from natural language query* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 3939–3948.

158. Liang, C., Wu, Y., Luo, Y. & Yang, Y. ClawCraneNet: Leveraging object-level relation for text-based video segmentation. *arXiv preprint arXiv:2103.10702* (2021).

159. Wang, L., Li, W., Li, W. & Van Gool, L. *Appearance-and-relation networks for video classification* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 1430–1439.

160. Zhu, X., Wang, Y., Dai, J., Yuan, L. & Wei, Y. *Flow-guided feature aggregation for video object detection* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2017), 408–417.

161. Cheng, H. K. & Schwing, A. G. *XMem: Long-term video object segmentation with an Atkinson-Shiffrin memory model* in *Proceedings of the European Conference on Computer Vision* (2022), 640–658.

162. Gao, S.-H. *et al. Global2Local: Efficient structure search for video action segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 16805–16814.

163. Zhou, J. *et al. Audio–visual segmentation* in *Proceedings of the European Conference on Computer Vision* (2022), 386–403.

164. Yang, A., Miech, A., Sivic, J., Laptev, I. & Schmid, C. *TubeDETR: Spatio-temporal video grounding with transformers* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 16442–16453.

165. Djuric, P. M. *et al.* Particle filtering. *IEEE Signal Processing Magazine* **20,** 19–38 (2003).

166. Chui, C. K. & Chen, G. *Kalman Filtering: with Real-Time Applications* 5th (Springer, 2017).

167. Heo, M. *et al. A generalized framework for video instance segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 14623–14632.

168. Cheng, H. K., Oh, S. W., Price, B., Schwing, A. & Lee, J.-Y. *Tracking anything with decoupled video segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 1316–1326.

169. Li, W. *et al. OneVOS: unifying video object segmentation with all-in-one transformer framework* in *Proceedings of the European Conference on Computer Vision* (2025), 20–40.

170. Guo, P. *et al. OpenVIS: Open-vocabulary video instance segmentation* in *AAAI Conference on Artificial Intelligence* **39** (2025), 3275–3283.

171. Yan, B., Sundermeyer, M., Tan, D. J., Lu, H. & Tombari, F. *Towards real-time open-vocabulary video instance segmentation* in *Winter Conference on Applications of Computer Vision* (2025), 1861–1871.

172. Miao, B., Bennamoun, M., Gao, Y., Shah, M. & Mian, A. Temporally consistent referring video object segmentation with hybrid memory. *IEEE Transactions on Circuits and Systems for Video Technology* (2024).

173. Yan, R. *et al. Tracking-forced Referring Video Object Segmentation* in *Proceedings of the 32nd ACM International Conference on Multimedia* (2024), 5356–5364.

174. Yan, S. *et al. Referred by multi-modality: A unified temporal transformer for video object segmentation* in *AAAI Conference on Artificial Intelligence* **38** (2024), 6449–6457.

175. Sun, Y. *et al.* Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621* (2023).

176. Sun, Y. *et al.* A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554* (2022).

177. Han, M. *et al. HTML: Hybrid temporal-scale multimodal learning framework for referring video object segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 13414–13423.

178. Ding, Z. *et al. Language-bridged spatial-temporal interaction for referring video object segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 4964–4973.

179. Liang, Y., Li, X., Jafari, N. & Chen, J. Video object segmentation with adaptive feature bank and uncertain-region refinement. *Advances in Neural Information Processing Systems* **33,** 3430–3441 (2020).

180. Fan, H. *et al.* LaSOT: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision* **129,** 439–461 (2021).

181. Lin, C.-H., Kong, C. & Lucey, S. *Learning efficient point cloud generation for dense 3D object reconstruction* in *AAAI Conference on Artificial Intelligence* **32** (2018).

182. Gao, J. *et al.* Learning deformable tetrahedral meshes for 3D reconstruction. *Advances in Neural Information Processing Systems* **33,** 9936–9947 (2020).

183. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. & Geiger, A. *Occupancy networks: Learning 3D reconstruction in function space* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 4460–4470.

184. Nan, L. & Wonka, P. *PolyFit: Polygonal surface reconstruction from point clouds* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2017), 2353–2361.

185. Chen, A., Xu, Z., Geiger, A., Yu, J. & Su, H. *TensoRF: Tensorial radiance fields* in *Proceedings of the European Conference on Computer Vision* (2022), 333–350.

186. Wang, L. *et al. Fourier plenoctrees for dynamic radiance field rendering in real-time* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 13524–13534.

187. Wang, P. *et al. F2-NeRF: Fast neural radiance field training with free camera trajectories* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 4150–4159.

188. Yu, A. *et al. Plenoctrees for real-time rendering of neural radiance fields* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 5752–5761.

189. Li, Z., Wang, Q., Cole, F., Tucker, R. & Snavely, N. *DynIBaR: Neural dynamic image-based rendering* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 4273–4284.

190. Li, L., Shen, Z., Wang, Z., Shen, L. & Tan, P. Streaming radiance fields for 3D video synthesis. *Advances in Neural Information Processing Systems* **35,** 13485–13498 (2022).

191. Fang, J. *et al. Fast dynamic radiance fields with time-aware neural voxels* in *SIGGRAPH Asia 2022 Conference Papers* (2022), 1–9.

192. Shao, R. *et al. Tensor4D: Efficient neural 4D decomposition for high-fidelity dynamic reconstruction and rendering* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 16632–16642.

193. Cao, A. & Johnson, J. *HexPlane: A fast representation for dynamic scenes* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 130–141.

194. Sun, J. *et al. 3DGStream: On-the-fly training of 3D Gaussians for efficient streaming of photo-realistic free-viewpoint videos* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 20675–20685.

195. Huang, Y.-H. *et al. SC-GS: Sparse-controlled Gaussian splatting for editable dynamic scenes* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 4220–4230.

196. Gao, C., Saraf, A., Kopf, J. & Huang, J.-B. *Dynamic view synthesis from dynamic monocular video* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 5712–5721.

197. Guo, X. *et al. Forward flow for novel view synthesis of dynamic scenes* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 16022–16033.

198. Li, Z., Niklaus, S., Snavely, N. & Wang, O. *Neural scene flow fields for space-time view synthesis of dynamic scenes* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 6498–6508.

199. Tretschk, E. *et al. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 12959–12970.

200. Xian, W., Huang, J.-B., Kopf, J. & Kim, C. *Space-time neural irradiance fields for free-viewpoint video* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 9421–9431.

201. Song, L. *et al.* NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* **29,** 2732–2742 (2023).

202. Attal, B. *et al.* TöRF: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in Neural Information Processing Systems* **34,** 26289–26301 (2021).

203. Lin, H. *et al. Efficient neural radiance fields for interactive free-viewpoint video* in *SIGGRAPH Asia 2022 Conference Papers* (2022), 1–9.

204. Peng, S., Yan, Y., Shuai, Q., Bao, H. & Zhou, X. *Representing volumetric videos as dynamic MLP maps* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 4252–4262.

205. Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P. & Hedman, P. *Zip-NeRF: Anti-aliased grid-based neural radiance fields* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), 19697–19705.

206. Yang, Z., Yang, H., Pan, Z., Zhu, X. & Zhang, L. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. *arXiv preprint arXiv:2310.10642* (2023).

207. Bhattacharyya, A. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics,* 401–406 (1946).

208. Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. *The unreasonable effectiveness of deep features as a perceptual metric* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 586–595.

209. Zhao, L., Lu, X., Fan, R., Im, S. K. & Wang, L. GaussianHand: Real-time 3D Gaussian rendering for hand avatar animation. *IEEE Transactions on Visualization and Computer Graphics* (2024).

210. Sabater, N. *et al. Dataset and pipeline for multi-view light-field video* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2017), 30–40.

211. Bae, J. *et al. Per-Gaussian embedding-based deformation for deformable 3D Gaussian splatting* in *Proceedings of the European Conference on Computer Vision* (2024), 321–335.

212. Das, D., Wewer, C., Yunus, R., Ilg, E. & Lenssen, J. E. *Neural parametric Gaussians for monocular non-rigid object reconstruction* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 10715–10725.

213. Liu, Q. *et al. MoDGS: Dynamic Gaussian Splatting from Casually-captured Monocular Videos with Depth Priors* in *International Conference on Learning Representations* (2024).

214. Guo, X., Zhang, W., Liu, R., Han, P. & Chen, H. *MotionGS: Compact Gaussian splatting SLAM by motion filter* in *2024 7th International Conference on Robotics, Control and Automation Engineering (RCAE)* (2024), 685–692.

215. Lin, J. *et al. HybridGS: Decoupling Transients and Statics with 2D and 3D Gaussian Splatting* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 788–797.

216. Chu, W.-H. *et al. Robust Multi-Object 4D Generation for In-the-wild Videos* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 22067–22077.

217. Zhao, B. *et al. GaussianPrediction: Dynamic 3D Gaussian prediction for motion extrapolation and free view synthesis* in *ACM SIGGRAPH 2024 Conference Papers* (2024), 1–12.

218. Duan, Y. *et al. 4D-Rotor Gaussian splatting: towards efficient novel view synthesis for dynamic scenes* in *ACM SIGGRAPH 2024 Conference Papers* (2024), 1–11.

219. Kwak, S. *et al.* MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting. *arXiv preprint arXiv:2501.03714* (2025).

220. Lu, Y., Zhou, Y., Liu, D., Liang, T. & Yin, Y. *BARD-GS: Blur-aware reconstruction of dynamic scenes via Gaussian splatting* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025), 16532–16542.

221. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. *High-resolution image synthesis with latent diffusion models* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 10684–10695.

222. Ruiz, N. *et al. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 22500–22510.

223. Li, H. *et al. Real-world deep local motion deblurring* in *AAAI Conference on Artificial Intelligence* **37** (2023), 1314–1322.

224. Feng, H., Zhou, H., Ye, T., Chen, S. & Zhu, L. *Residual Diffusion Deblurring Model for Single Image Defocus Deblurring* in *AAAI Conference on Artificial Intelligence* **39** (2025), 2960–2968.

225. Liu, X. *et al.* One-step diffusion model for image motion-deblurring. *arXiv preprint arXiv:2503.06537* (2025).

226. Cho, S.-J., Ji, S.-W., Hong, J.-P., Jung, S.-W. & Ko, S.-J. *Rethinking coarse-to-fine approach in single image deblurring* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), 4641–4650.

227. Kong, L., Dong, J., Ge, J., Li, M. & Pan, J. *Efficient frequency domain-based transformers for high-quality image deblurring* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 5886–5895.

228. Lin, Y., Dai, Z., Zhu, S. & Yao, Y. *Gaussian-Flow: 4D reconstruction with dynamic 3D Gaussian particle* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), 21136–21145.

229. Zhang, D. *et al. FF-Former: Swin Fourier transformer for nighttime flare removal* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), 2824–2832.

230. Tsai, F.-J., Peng, Y.-T., Lin, Y.-Y., Tsai, C.-C. & Lin, C.-W. *Stripformer: Strip transformer for fast image deblurring* in *Proceedings of the European Conference on Computer Vision* (2022), 146–162.

231. Zamir, S. W. *et al. Restormer: Efficient transformer for high-resolution image restoration* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 5728–5739.

232. Siméoni, O. *et al.* DINOv3. *arXiv preprint arXiv:2508.10104* (2025).

233. Agarwal, N. *et al.* Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575* (2025).