Karlsruher Institut für Technologie

TOWARDS SCALABLE AND INTERACTION-EFFICIENT VIDEO OBJECT SEGMENTATION IN UNCONSTRAINED SCENARIOS

zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

STÉPHANE VUJASINOVIĆ

Tag der mündlichen Prüfung:	07. November 2025
Hauptreferent:	**Prof. Dr.-Ing. Rainer Stiefelhagen** Karlsruhe Institute of Technology
Korreferent:	**Prof. Dr. Matej Kristan** University of Ljubljana

Stéphane Vujasinović: *Towards Scalable and Interaction-Efficient Video Object Segmentation in Unconstrained Scenarios*

In loving memory to my Baka, Josefina Pavić.
1937 – 2019

ABSTRACT

Video Object Segmentation (VOS) is a fundamental and challenging problem in Computer Vision (CV), where the goal is to delimit the spatial and temporal presence of a collection of objects in a given video sequence at pixel level. Several approaches tackle VOS, namely automatic, semi-automatic, and interactive, each tailored to specific use cases and requirements. Here, semi-automatic (sVOS) and interactive (iVOS) approaches provide the necessary flexibility to segment arbitrary objects by leveraging user cues (to various degrees), balancing automation and adaptability.

However, their applicability is predominantly limited to short, pre-recorded sequences due to their inherent design and the necessary user workload (*e.g.*, reviewing and annotation effort). This bias makes sVOS and iVOS approaches impractical for unconstrained video segmentation (or tracking) applications. We consider a video sequence unconstrained when it is not pre-recorded (*e.g.*, live-streamed), has no fixed length, and shows unpredictable content such as appearance changes, clutter, or occlusions. In addition, the robustness of these models is inherently constrained by their training data. The data distribution learned during training is unlikely to encompass all possible real-world scenarios, particularly when dealing with unusual applications or challenging conditions.

We propose through this work to extend VOS approaches to robustly track arbitrary objects in unconstrained videos via an efficient human-in-the-loop strategy, where we simultaneously focus on minimizing the associated user effort. To reduce this user workload, we design a proactive framework that monitors its predictive uncertainty and requests user corrections (*i.e.*, clicks) on-the-fly when the method's uncertainty is high. Our main idea is to involve the user strategically to resolve challenging or ambiguous situations where the system lacks confidence by leveraging the user's availability already at hand in iVOS and sVOS tasks, but in an active manner. Furthermore, we develop a diversity-driven memory management module that enables segmentation over unconstrained sequences, overcoming the limitations of previous VOS methods, tailored for short videos.

We validate each design choice across various datasets and display consistent improvement in segmentation robustness and in decreasing the user's workload. Through this research, we provide a foundation for proactive, scalable, and user-efficient human-in-the-loop video segmentation adapted for applications geared toward unconstrained videos.

ZUSAMMENFASSUNG

Die Segmentierung von Objekten in Videos (Video Object Segmentation, VOS) ist ein grundlegendes und herausforderndes Problem der Computer Vision (CV), bei dem das Ziel darin besteht, die räumliche und zeitliche Abgrenzung eines Objekts (oder einer Objektgruppe) auf Pixel-Ebene in einem gegebenen Video vorzunehmen. Verschiedene Ansätze adressieren VOS: automatische, semi-automatische oder interaktive Verfahren, die jeweils auf spezifische Anwendungsfälle und Anforderungen zugeschnitten sind. In diesem Rahmen bieten semi-automatische (sVOS) und interaktive (iVOS) Ansätze die notwendige Flexibilität, beliebige Objekte zu segmentieren, indem sie Benutzereingaben in unterschiedlichem Ausmaß nutzen, um Automatisierung und Anpassungsfähigkeit auszubalancieren.

Ihre Anwendbarkeit ist jedoch überwiegend auf kurze, vorab aufgezeichnete Sequenzen beschränkt, bedingt durch ihren grundsätzlichen Aufbau und den damit verbundenen Benutzeraufwand (z.B. Überprüfung, Annotierungsaufwand). Diese Einschränkungen machen sVOS- und iVOS-Ansätze für die Segmentierung (oder Verfolgung) von nicht eingeschränkten Videos weitgehend unpraktisch. Wir betrachten eine Videosequenz als "nicht eingeschränkt", wenn sie nicht vorab aufgenommen wurde (z.B. Livestreams), keine feste Länge aufweist und unvorhersehbare Inhalte wie Erscheinungsänderungen, Unordnung oder Verdeckungen zeigt. Zusätzlich ist die Robustheit dieser Modelle inhärent durch ihre Trainingsdaten begrenzt. Die während des Trainings erlernte Datenverteilung deckt in der Regel nicht alle möglichen realen Szenarien ab, insbesondere bei ungewöhnlichen Anwendungen oder herausfordernden Bedingungen.

In dieser Arbeit erweitern wir bestehende VOS-Ansätze, um beliebige Objekte in nicht eingeschränkten Videos robust verfolgen zu können. Dazu entwickeln wir eine effiziente Mensch-im-Loop-Strategie, die den Benutzeraufwand gezielt minimiert. Kernstück unseres Ansatzes ist ein proaktives Framework, das seine prädiktive Unsicherheit kontinuierlich überwacht und bei hoher Unsicherheit während der Segmentierung Benutzerkorrekturen (z.B. Klicks) in Echtzeit anfordert. Auf diese Weise reduzieren wir sowohl den Überwachungs- als auch den Annotierungsaufwand erheblich. Darüber hinaus entwerfen wir ein diversitätsgetriebenes Speichermanagement-Modul, das eine langfristige Segmentierung ermöglicht und die bisherigen Einschränkungen semi-automatischer Methoden überwindet. Unsere zentrale Idee ist es, den Benutzer strategisch einzubeziehen, um schwierige oder mehrdeutige Situationen aufzulösen, wenn das System wenig Vertrauen in seine Vorhersagen hat. Wir nutzen die Präsenz des Benutzers, der auch in modernen sVOS- und iVOS-Szenarien üblicherweise bereits vorhanden ist, um initiale Hinweise zu geben, finale Ergebnisse zu überprüfen und nachträgliche Korrekturen vorzuschlagen.

Wir validieren jede Designentscheidung auf verschiedenen Datensätzen und zeigen konsistente Verbesserungen hinsichtlich der Segmentierungsrobustheit sowie einer Reduktion des Benutzeraufwands. Mit dieser Arbeit schaffen wir eine Grundlage für eine proaktive, skalierbare und benutzerfreundliche Mensch-im-Loop-Videosegmentierung, die für Anwendungen auf nicht eingeschränkten Videoinhalten ausgelegt ist.

ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. Dr.-Ing. Rainer Stiefelhagen for accepting me into his lab as an external PhD student and providing me with continuous support and guidance throughout this journey, especially towards the completion of this thesis. I am very thankful for the opportunity and privilege he gave me to work under his guidance, benefiting from the reading group sessions and the PhD workshops.

I would also like to sincerely thank Prof. Dr. Matej Kristan for agreeing to serve as my second examiner. His expertise in the field is greatly appreciated and will ensure a highly relevant assessment of my work. My sincere thanks also go to the members of my examination committee for their valuable time and expertise in evaluating this dissertation.

This work was conducted at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) in the Object Recognition Department, where I had the opportunity to engage in cutting-edge research. Here, I would like to extend my gratitude to Dr. Michael Arens and Dr. Norbert Scherer-Negenborn for the valuable discussions and the freedom they allowed me in my research (and naturally, the financial support). Special thanks go to Sebastian and Stefan, who provided helpful input and feedback during the early stages of my thesis and who granted me significant freedom in the later stages to independently explore research directions that "boggled" my mind. In addition, I'd like to also thank all my colleagues at the IOSB, and especially Eckart, Francisco, Leon, Valentin, Ann-Kristin, Jens, Dominik and José for the valuable discussions, be it formal or informal.

Furthermore, I'm grateful for the brief but insightful exchanges I had with the members of the Computer Vision for Human-Computer Interaction Lab at the Karlsruhe Institute of Technology during the diverse reading groups and especially the lab's PhD workshops, which where always a source of inspiration and motivation for me. A special mention goes to Saquib for the chess games!

Now, I would also like to express my deepest gratitude to all my previous professors and mentors who believed in me and ignited my passion for research during my undergraduate studies. Among them, I would particularly like to acknowledge Prof. Dr. Thierry de Larochelambert, Prof. Dr. Olivier Piccin, Prof. Dr. Olivier Schecker, and Prof. Dr. Ferdinand Olawsky. Lastly, special thanks go to M. Daval, who was the first to inspire me to pursue a path in science.

Naturally, I would like to thank my sister, my father, and especially my mother, who always believed in me and provided unwavering support during moments of doubt. Also, my friends, who helped me forget about this thesis for much-needed vacations, thus reminding me that life (sometimes) extends beyond research. So thank you all, Stevi

(Vujakon), Miham, Le K, Corti, Mate, Clément, Fred, the Lutz brothers, Nathan, Xavier, les Baleines (Marly, Mathias and Gaëtan) and "les CINQuantes nuances de bleu" . A special thank you and dedication goes to my late grandmother, who always shared the wisdom, "škola prije cure", a principle I strove to follow to the best of my ability!

Now I would get into trouble if I forgot this one..., so I would like to express my deepest gratitude to my partner in life, Camille. You have been my "pierre angulaire" throughout this journey, providing immense support and being incredibly understanding during what could often feel like a solitary endeavor against the winds. Thank you for your unconditional love, support, and immense understanding.

It is with a mix of multiple sentiments, good and less so, that I reflect upon this journey. This endeavor was not just an academic pursuit, but an opportunity to feed my insatiable curiosity for learning new things and building them. While I initially thought this would be similar to a "Classe Préparatoire" a fond memory I have, the PhD proved to be an entirely different kind of academic landscape to navigate. Beyond the technical knowledge acquired, this journey has also been one of self-discovery, offering invaluable insights through the various challenges encountered along the way. These will undoubtedly prove essential in my future endeavors. This PhD journey has taught me invaluable lessons, not least among them the profound impact an environment can have on one's ability to thrive. My sincere thanks to those whose support, in various forms, helped me to navigate the complexities of this experience.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CV	Computer Vision
HITL	Humain-in-the-loop
VOS	Video Object Segmentation
sVOS	Semi-automatic Video Object Segmentation
VOT	Visual Object Traking
aVOS	Automatic Video Object Segmentation
iVOS	Interactive Video Object Segmentation
SOTA	State-of-the-Art
DL	Deep Learning
DNN	Deep Neural Networks
SAM	Segment Anything Model
SL	Supervised Learning
SGD	Stochastic Gradient Descent
iIOS	Interactive Image Object Segmentation
CNN	Convolutional Neural Network
FOR	Frame of Reference
BCE	Binary Cross Entropy
ziVOS	Lazy interactive Video Object Segmentation
IDI	Interaction Density Index
ACI	Average Correction Interval
NoC	Number of Correction
IoU	Intersection over Union

Part I

BACKGROUND

1 INTRODUCTION

1.1 Context

Video content creation and consumption have considerably increased in recent years, fundamentally reshaping the digital media landscape. This growth is supported by: (1) On-demand streaming platforms (*e.g.*, Netflix, Apple TV+, and Prime Video), which have seen a steady increase in subscribers, with current market predictions estimating that the industry will reach approximately 399.05 billion USD by 2032, compared to today's 113.78 billion USD [1]. (2) Live broadcasting networks (*e.g.*, ESPN, CBS Sports, Eurosport, Twitch), with, as an example ESPN reporting nearly 25 million subscribers in Q1 2025, marking an 18-fold increase over the past five years [2]. (3) Social media platforms (*e.g.*, Instagram, TikTok, YouTube), which have democratized video production and distribution, enabling users worldwide to generate and share content at an unprecedented scale [3].

Benefiting from this trend are video editing tools, *e.g.*, Adobe Premiere Pro[1], DaVinci Resolve[2], iMovie[3]. The growing demand is reflected in future market projections, which is expected to reach approximately 5.13 billion USD by 2032, up from 3.09 billion USD in 2023 [4]. To be more specific, Adobe Premiere Pro's user base has grown from 5 million in 2019 to approximately 30 million in 2024 [5]. Similarly, DaVinci Resolve grew from 900 users in 2009 to around 5 million in 2023 [6]. Furthermore, this trend extends beyond professionals to also include hobbyists [7]. Hence, there is an increasing demand for video editing tools that are more sophisticated and user-friendly to keep up with the massive volume and velocity at which content is produced and distributed today. Ideally, these tools should automate repetitive and time-consuming tasks to reduce user constraints and facilitate content creation and modification.

One of the many pillars supporting these advancements is Video Object Segmentation (VOS). It is a fundamental problem in Computer Vision (CV), where the goal is to delineate the spatial (pixel-level) and temporal (frame-level) extent of an object of interest in a given video sequence.

Note that, beyond entertainment, VOS has significant utility in surveillance and security applications [8, 9], as evidenced by market projections, rising from an evaluation of 19.12 billion USD in 2018 to (an estimated) 33.06 billion USD for 2026 [10]. Modern surveillance systems frequently handle vast volumes of live-streamed video from multi-

1 https://www.adobe.com/products/premiere.html
2 https://www.blackmagicdesign.com/products/davinciresolve
3 https://www.apple.com/ca/imovie/

ple cameras, making continuous human monitoring impractical. Here, an automated pipeline using VOS methods can help reduce the user's workload by tracking objects of interest and detecting uncertain or suspicious events to alert the user proactively. Similarly, livestock and wildlife monitoring [11] can benefit from pixel-level tracking. Farmers and researchers can leverage video sequences from drones or fixed cameras to monitor animal health and detect abnormal movements. Achieving pixel-level segmentation enables the identification of individual animals within large herds and facilitates the tracking of subtle behavioral changes that may signal disease [12]. Naturally, robotics, autonomous vehicles, and mixed reality also benefit from advancements in VOS [8, 13]. Moreover, as the demand for artificial intelligence-based solutions grows, so does the need for large amounts of data. To meet this growing requirement, several specialized tools for video annotation have emerged and expanded in recent years (*e.g.*, label-studio[4] (previously LabelImg), Labelbox[5], and Dataloop[6]).

In summary, the need for efficient and flexible tools becomes increasingly important as video applications continue to expand in scale and complexity, from content creation, live streaming, robotics and large-scale surveillance.

1.2 Challenges and Motivation

A central challenge for future VOS methods is designing systems that can: (1) Robustly track arbitrary objects in unconstrained video sequences and (2) Design an efficient interaction user workload in terms of annotation and the need for constant monitoring (*i.e.*, reviewing the results), to minimize user workload We refer to video sequences as unconstrained when they are (i) not necessarily pre-recorded (*e.g.*, live-streamed), (ii) lack fixed boundaries (*i.e.*, arbitrary length), (iii) and exhibit unpredictable visual content such as rapid appearance variations, clutter, prolonged occlusions. Hence, unconstrained videos introduce difficulties related to sequence length, visual variability, and the inability to access future frames during inference.

An additional limitation found in current VOS methods is that they are tightly coupled to the data distributions seen during training, which might limit their ability in complex scenarios (out of distribution), resulting in performance breakdowns. To mitigate this, occasional human oversight would be beneficial (*i.e.*, Humain-in-the-loop (HITL) approach), as a user could provide additional cues to correct erroneous predictions (ideally on-the-fly while the method segments the video), and solve ambiguous or uncertain scenarios.

Among the diverse approaches to VOS [8] (Figure 1.1 present the main subcategories to solve VOS), semi-automatic and interactive methods stand out for their ability to balance automation with user control. By contrast, Automatic Video Object Segmentation

4 https://github.com/HumanSignal/label-studio
5 https://labelbox.com/product/annotate/video/
6 https://dataloop.ai/platform/annotation-platform-old/video-annotation/

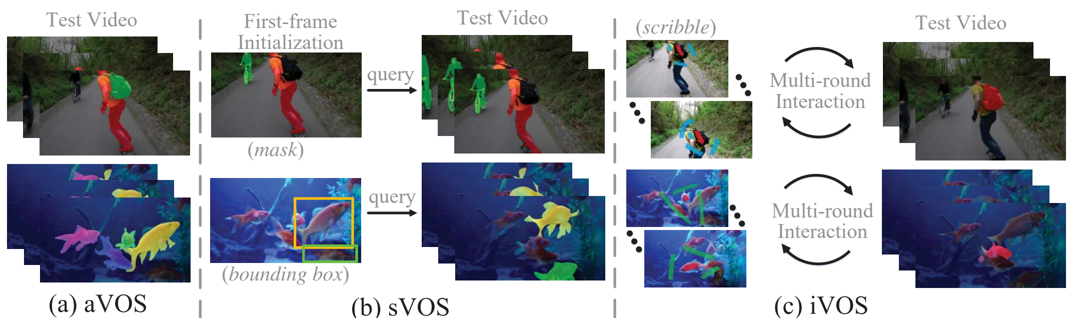


Figure 1.1: Primary sub categories to VOS, (a) Automatic Video Object Segmentation, (b) Semi-automatic Video Object Segmentation, (c) Interactive Video Object Segmentation. Figure adapted from [8].

(aVOS) methods, which require no user input, lack the flexibility to segment arbitrary objects outside the distribution seen during training. Although Semi-automatic Video Object Segmentation (sVOS) and Interactive Video Object Segmentation (iVOS), through user-cues (*e.g.*, initialization, interactions), can track arbitrary objects, they still present key limitations: **(1) Initial annotation cost**: sVOS requires significant upfront effort to create a fully annotated object mask on pixel-level, often taking more than 300 seconds per object [14] (depending on the object area). **(2) User interaction**: iVOS mitigates this effort by relying on sparse annotations (*i.e.*, typically scribbles) but instead requires multiple rounds of user intervention. While this strategy reduces the upfront burden of creating a fully annotated object mask, it shifts the workload to reviewing the video multiple times and increases the number of interactions needed. We provide in Figure 1.2 a visual representation of the user's workload distribution when segmenting a video. Note that, in contrast to iVOS, in sVOS no feedback loop from between the segmentation pipeline and the user is present. **(3) Short-term bias**: Most sVOS and iVOS methods are tailored for short, pre-recorded clips and struggle to work on longer sequences. **(4) Pre-recorded bias**: The reliance on either a full initial mask (sVOS) or iterative feedback (iVOS) renders these methods unsuitable for live video, where users cannot necessarily pause or rewind the sequence. **(5) Feedback timing**: In iVOS methods, user corrections are typically applied only after completing an entire interaction round. This delayed feedback loop is inefficient and problematic for long videos, where errors may propagate across many frames before being addressed, leading to wasted effort and reduced segmentation efficiency. Ideally, corrections should be triggered early based on model uncertainty, preventing the propagation of errors across long sequences.

Beyond these design-level constraints, there is also an evaluation-related limitation. **(6) Monitoring assumptions**: Evaluation protocols either ignore the monitoring aspect from the user's perspective (as in sVOS) or only partially consider it (as in iVOS). However, as video datasets grow in size and duration, so does the burden of continuous user oversight becomes significantly impractical. **(7) Workload metrics**: Current evaluation

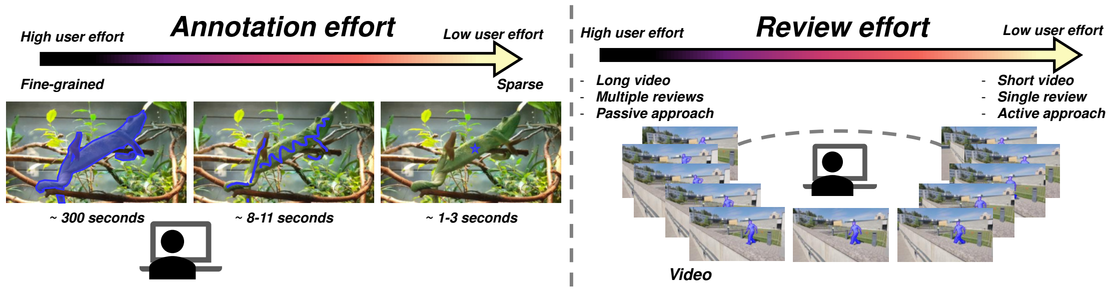


Figure 1.2: User workload dispersion for VOS tasks can be categorized into annotation effort and reviewing effort. (1) Annotation effort is directly influenced by the interaction type used to specify the object of interest. Pixel-level annotations represent the most intensive approach, while click-based inputs require the least effort. (2) Reviewing effort depends on several factors, including sequence length, the number of review rounds, and the method's ability to actively guide the user by suggesting frames or regions for correction.

metrics rarely reflect the user's workload, making it difficult to assess the true workload from a user's side.

In summary, while sVOS and iVOS methods represent a significant first step toward scalable and user-adaptive segmentation, they remain at their current state, ill-suited for unconstrained, long-form or streaming applications. To address the above challenges, we frame our central research question (**CRQ**) as follows:

How can we design a scalable, robust, and user-efficient VOS approach for unconstrained video sequences that ensures minimal user effort?

To help us answer this question, we decompose the CRQ into the following sub-questions, which are explored throughout this thesis:

- ➢ **(RQ1) User Interaction Efficiency**: How can we efficiently optimize the user's required workload (*e.g.*, initialization, monitoring, reviewing, corrections) typically required in VOS tasks while maintaining segmentation performance for unconstrained scenarios?
- ➢ **(RQ2) Scalability to unconstrained video**: How can we reliably and efficiently support unconstrained videos (not assuming offline access, fixed video duration nor object type) in a viable HITL approach for VOS methods?

1.3 Contributions and Thesis Outline

Conceptually, we propose a hybrid framework that fuses sVOS with iVOS (see Figure 1.3). Similarly to sVOS, we would like to rely only on a single segmentation pass but avoid

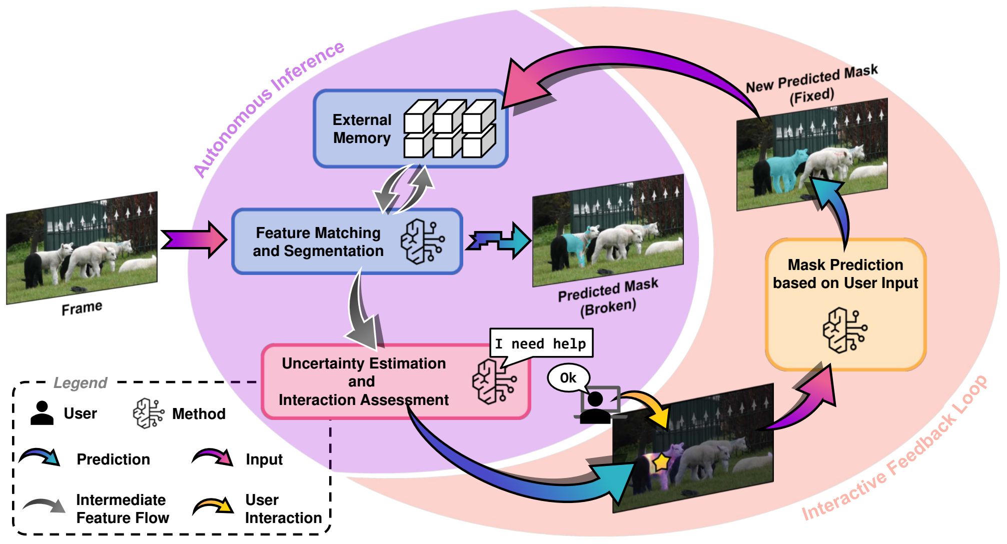

Autonomous Inference
External Memory
Feature Matching and Segmentation
Frame
Predicted Mask (Broken)
Uncertainty Estimation and Interaction Assessment
I need help
Ok
New Predicted Mask (Fixed)
Mask Prediction based on User Input
Interactive Feedback Loop
Legend
User
Method
Prediction
Input
Intermediate Feature Flow
User Interaction

Figure 1.3: A conceptual overview of our proposed baseline for a robust VOS approach suited for unconstrained videos. Note that, in this figure we assume the method to be already initialized to the object to track. Since we treat segmentation as a step-by-step process, where frames are segmented one after another, we illustrate our concept using a single frame from the video. Importantly, in our proposed solution, the segmentation is performed in a single forward pass. Our pipeline includes: (1) A segmentation module (feature matching and segmentation with an external memory) that automatically predicts object masks. (2) An uncertainty estimation module that requests user interaction when confidence is low. Finally, (3) a user feedback loop that incorporates corrective inputs into our method's memory to improve future predictions.

the cumbersome up-front initialization process. Instead of following the traditional iVOS approach, which passively waits for user input at the end of an interaction, we would like our approach to actively monitor its predictive uncertainty and request user corrections on-the-fly when necessary. Hence, have the user engage actively with the method during the segmentation process. This shift toward proactivity would allow the method to maintain segmentation accuracy over long and not necessarily pre-recorded sequences with ideally minimal human effort.

We address our CRQ through the following chapters, with our contributions summarized as follows:

➢ Chapter 4: Minimizing User Effort in Interactive Video Object Segmentation with Click based Interactions
We present a lightweight interaction scheme using clicks instead of scribbles (the standard format) to reduce annotation effort in iVOS as a proof-of-concept. We

decouple mask prediction and propagation into separate modules and introduce a complementary hardware-independent evaluation metric. Our approach achieves competitive results against scribble approach based, but at a lower interaction cost.

- Chapter 5: Extending Semi-Automatic Video Object Segmentation for Unconstrained Video Scenarios
  We propose a training-free memory management scheme to adapt contemporary sVOS methods for unconstrained videos. We address the continuous accumulation of intermediate frames typically found in recent sVOS approaches, which bloats memory and degrades prediction capabilities over time. To this end, we limit the memory growth and update the memory whenever an incoming frame (*i.e.*, embedding) improves the diversity of the stored representations. We quantify the diversity of the embeddings stored in the memory by computing the corresponding Gram matrix. This enables us to have a generalized update strategy, thus eliminating the need for a meta-parameter to control the memory update frequency. In all sVOS baselines extended with our module, we consistently improve the performance and achieve competitive results to State-of-the-Art (SOTA).

- Chapter 6: Introducing Proactive and Scalable Interactive Video Object Segmentation for Unconstrained Videos
  We introduce Lazy interactive Video Object Segmentation (ziVOS), a new sub-task that bridges sVOS and iVOS for unconstrained video segmentation with minimal user effort by interacting only on a single segmentation pass. We present a corresponding baseline, which proposes a trade-off between automation and robustness by computing its prediction uncertainty on the fly (on pixel level) and inquiring the user for help when its uncertainty is too high. Depending on how quickly the uncertainty degrades, the system can request a user correction, generate a pseudo-correction, or proceed without intervention if uncertainty remains low. Pseudo-corrections help avoid unnecessary user involvement, keeping the system efficient and reducing user fatigue. In addition, we present complementary metrics to evaluate the robustness of existing sVOS methods and introduce new metrics to assess user workload.

To summarize, this thesis is organized as follows: we begin in Chapter 2 with a review of existing approaches with a particular emphasis on sVOS and iVOS. In Chapter 3 we introduce the fundamental concepts behind sVOS methods, which is the backbone of this thesis. In Chapter 4, we explore the feasibility of click-based interactions in iVOS to reduce user workload. Chapter 5 presents our diversity-driven memory management module to extend existing SOTA sVOS approaches to unconstrained videos. Chapter 6 introduces our new sub-task ziVOS, along with a tailored baseline that estimates its confidence while segmenting to request user help or to provide self-corrections when possible. Finally, Chapter 7 summarizes our key contributions and outlines potential avenues for future research.

This thesis is based on the following peer-reviewed articles:

Publication	**Chapter**
Stéphane Vujasinović, Sebastian Bullinger, Stefan Becker, Norbert Scherer-Negenborn, Michael Arens, Rainer Stiefelhagen. "Revisiting click-based interactive video object segmentation", in *International Conference on Image Processing (ICIP)*, 2022.	4
Stéphane Vujasinović, Sebastian Bullinger, Stefan Becker, Norbert Scherer-Negenborn, Michael Arens, Rainer Stiefelhagen. "READMem: Robust Embedding Association for a Diverse Memory in Unconstrained Video Object Segmentation", in *British Machine Vision Conference (BMVC)*, 2023.	5
Stéphane Vujasinović, Stefan Becker, Sebastian Bullinger, Norbert Scherer-Negenborn, Michael Arens, Rainer Stiefelhagen. "Strike the Balance: On-the-Fly Uncertainty based User Interactions for Long-Term Video Object Segmentation", in *Asian Conference on Computer Vision (ACCV)*, 2024.	6

Table 1.1: List of related publications and their corresponding chapters.

2 RELATED WORK

In the literature [8, 15], VOS approaches are typically divided into three sub-categories (see Figure 1.1), each addressing distinct requirements and needing varying degrees of user intervention: (i) **aVOS** (or unsupervised) performs video object segmentation without user input. These methods follow a model-based approach, where the method already has prior knowledge about the set of objects to track, either through offline training or by using an object detector. (ii) **sVOS** (or semi-supervised) requires a single user interaction at the beginning of the video sequence to indicate which object to track. Typically, this involves initializing the sVOS method with a fine-grained annotated mask of the target object in the first frame of the video. Note that sVOS is closely related to Visual Object Traking (VOT) but on pixel-wise level. (iii) **iVOS** approaches rely on sparse user annotations, usually in the form of scribbles. However, object masks generated from sparse annotations are less likely to be perfect, hindering performance. To mitigate this, multiple rounds of user interaction are allowed. A typical interaction round includes (a) generating a segmentation mask based on the user's sparse input (*i.e.*, scribbles), (b) propagating the mask to the remaining frames, and (c) reviewing the results to provide additional annotations if needed. The user repeats the process until they are satisfied with the final output.

Recent advances in VOS have also introduced new sub-tasks (which we present briefly for completeness): (iv) **Referring Video Object Segmentation (RVOS)**, segments a target object based on natural language descriptions. (v) **Few-shot Video Object Segmentation (FSVOS)**, segments objects based on a support set (several images of the object of interest taken from other datasets) that contains the associated fine-grained annotations. (vi) **Promptable Video Object Segmentation (PVOS)**, recently introduced by Ravi *et al*. [16] (*i.e.*, SAM 2), extends sVOS by incorporating sparse annotations while enabling corrections in a single interaction round. Note that PVOS closely resembles our proposed sub-task ziVOS (refer to Chapter 6).

In this chapter, we provide an overview of recent SOTA approaches that tackle Semi-automatic Video Object Segmentation (sVOS) in Section 2.1, and in Section 2.2 Interactive Video Object Segmentation (iVOS) approaches. In Section 2.3, we present popular datasets found in the field to evaluate VOS approaches. Note that we focus mainly on Deep Learning (DL) based solutions, which have yielded more robust and accurate segmentation in recent years.

2.1 Semi-automatic Video Object Segmentation

We can divide most sVOS approaches into three subcategories, as outlined by Zhou *et al.* [8]: (i) Online Fine-Tuning, (ii) Propagation-Based, and (iii) Matching-Based methods (the most prominent in recent years). Note that these subcategories only serve as a loose guideline to organize the landscape of the sVOS field. In practice, some methods may exhibit characteristics that span through multiple groups.

In addition, through the introduction of the Segment Anything Model (SAM) by Kirillov *et al.* [17], we nowadays have several recent pipelines relying on it for mainly refinement purposes. Given its growing adoption and impact on the field, we include it here as a fourth subcategory.

2.1.1 *Online Fine-Tuning*

Online fine-tuning approaches, often referred to as one-shot video object segmentation, dynamically adjust network parameters in response to an initial mask that defines the object of interest [18–24]. The pioneering work in this domain, OSVOS [18] proposed by Caelles *et al.*, fine-tunes a pre-trained segmentation model using the ground-truth mask from the first frame, enabling it to predict the object masks in the subsequent frames. Building on this foundation, OnAVOS [19] enhances OSVOS by incorporating online fine-tuning, which allows the model to adapt to appearance variations that arise after the initial frame. Further advancements are seen by Maninis *et al.* [20] and by Luiten *et al.* [23], which integrate object detectors as auxiliary components. This strategic addition helps define the location and spatial extent of the object, thereby narrowing the image regions where segmentation is performed. Another notable approach, by Meinhardt *et al.* [24] extends the capabilities of Mask R-CNN [25] by fine-tuning its appearance model for the object of interest in real time. This method not only aims for faster convergence through the learning of meta-parameters but also seeks to enhance the alignment between predicted and ground-truth masks. However, these improvements come with a trade-off in flexibility.

Despite these innovations, online fine-tuning approaches are often hindered by slow inference speeds and limited generalization capabilities [8].

2.1.2 *Propagation-Based*

In contrast, propagation-based methods leverage strong priors from previous frames (*i.e.*, the adjacent frame) to effectively propagate masks and to enhance their ability to manage rapid appearance changes [26–30]. A seminal contribution in this area is by Jampani *et al.* [31], who introduced a novel propagation-based approach in 2017. This method combines a bilateral network with a spatial network, primarily a Convolutional Neural Network (CNN), to learn the properties of a bilateral filter, used to transfer object information from all previously observed frames to the current one. Expanding on this, Khoreva *et al*. [26] utilize the previously predicted mask from adjacent frames to guide the current frame's object mask prediction. Their approach also includes on-the-fly optimization, enabling the model's weights to learn a coarse representation of the target object. Xiao *et al*. [21] further enhance this design by introducing an optical flow branch that serves as a prior to refine the segmentation mask generated by the baseline branch. Continuing this trajectory, Khoreva *et al*. [32] incorporate data augmentation in the first frame to specifically adapt a set of weights in the neural network to the target object. Ci *et al*. [33] explicitly learn to encode location and appearance embeddings of the current frame, improving the refinement of the predicted mask. Chen *et al*. [34] employ advanced tracking techniques that introduce a state estimation process, leading to remarkable performance improvements.

However, these methods tend to implicitly encode target-specific information into the network weights, which can reduce flexibility. Additionally, they are susceptible to error accumulation and often lack long-term context for re-identifying objects after occlusions [8].

2.1.3 *Matching-Based*

Matching-based methods [8] learn to construct an embedding space, where the distance between two embedding vectors from different frames are near each other if they encode similar information (*e.g.*, appearance, texture) and otherwise are further apart. Here, an initial frame is encoded into an embedding space, which indicates the user's interest for an image region (this initial frame is important, as it's purpose is to guide the segmentation for the subsequent frames). Given a new frame, which the network also encodes in the same embedding space, the goal for the network is to robustly associate resembling feature vectors from both encoded images. This matching is the key process to identify the new image regions that is relevant for the segmentation. In [35–40], the embeddings of the initial frame and adjacent frames are matched with the embeddings of the query frame through global and local matching, while Zhou *et al*. [41] also combine online fine-tuning approaches.

However, leading sVOS methods rely primarily on the *external memory* design introduced by Oh *et al*. [42], which uses features from the initial frame along with previously processed frames (*i.e.*, intermediate frames) as references during feature matching. Using a non-local approach, these methods match the embeddings from the query frame with the embeddings of reference frames. Based on STM [42], Xie *et al*. [43] and Seong *et al*. [44] perform local-to-local matching to alleviate distractor-induced mismatches. Seong *et al*. [45] also introduce multi-scale memory matching. To enhance pixel discrimination, Yong *et al*. [46] explore a unique approach that leverages both frequency and spectral domains. To reduce noisy predictions caused by ever-expanding memory, Cheng *et al*. [47] introduce a top-k filtering strategy during non-local matching and propose an efficient extension [48] that decouples feature extraction of the frame and corresponding mask, while improving the matching process. Park *et al*. [49] depart from the frame-to-frame propagation scheme and instead adopt a clip-to-clip approach to accelerate the propagation process.

However, as many of these methods are primarily tailored for short-term videos, recent works have sought to address this limitation. For instance, Li *et al*. [50] propose a compact global module to summarize object segmentation information within a fixed-size memory. Similarly, Liang *et al*. [51] apply a weighted average to the extracted features to merge new features with existing ones in memory and discard obsolete information using a *least frequently used* (LFU) mechanism. Liu *et al*. [52] explore the use of a quality-aware module (QAM) (based on the seminal work of Huang *et al*. [53]) to assess the quality of the predicted mask on-the-fly before integrating it into memory. Li *et al*. [54] introduce a spatio-temporal aggregation module to maintain a fixed-size memory bank. Cheng and Schwing [55] adopt the Atkinson-Shiffrin model [56] for their sVOS pipeline (XMem), which comprises: a *sensory memory* [57] that learns an appearance model for each incoming frame, a *working memory* based on STCN [48], and a *long-term memory* that consolidates working memory embeddings into a compact representation.

2.1.4 *Segment-Anything*

Contemporary works [58–60] leverage Segment Anything Model (SAM) [17] or its variants [61–63] to refine the original mask predicted by an sVOS baseline [39, 55], typically matching-based. However, in contrast to our framework, these methods refine every n-th mask predicted by the sVOS backbone using a SAM-based approach [17, 61–63], and require continuous user monitoring to determine when interventions are necessary. Furthermore, they limit the impact of refinements on subsequent frames, as they do not update the memory with the refined masks, thereby reducing the potential to improve future predictions. Other approaches have moved away from mask-based propagation as a backbone, shifting instead to tracking dense points [64], which are then used as prompts for SAM, as proposed by Rajič *et al*. [65].

Recently, Ravi *et al*. [16] introduced SAM2[1], performing on average 10% better than contemporary works on popular datasets. In addition, the authors define a new sub-task for VOS, *i.e.*, (PVOS), where the goal is to provide click-based inputs to indicate which object to segment in a given video sequence while allowing users to also provide corrections during the segmentation process to refine the model's prediction. Since its introduction, newer methods have tackled the memory management of SAM2, which relied primarily on a FIFO (First-In First-Out) based update. Here, most notably SAM2Long [66], which extends SAM2 for long-term applications by replacing the FIFO update strategy with training-free by viewing the memory as a tree structure. The authors maintain a set of pre-defined branches that should each include diverse hypotheses and prune over the

1 Meta Blog Post regarding SAM2: https://ai.meta.com/sam2/

ones that are likely to fail and create new ones to replace the pruned ones to maintain the same potential in diversity. However, each branch contains a separate set of memory. Yang *et al*. [67] extends SAM2 to better handle the presence of distractors by targeting the memory management aspect. The authors introduce a motion-aware memory (*i.e.*, Kalman Filter), to update the memory by taking into account an affinity score and a motion score.

2.2 Interactive Video Object Segmentation

To briefly restate, the goal in iVOS is to reduce the user's workload during video annotations [14], compared to sVOS, by relying on multiple user interaction rounds. Early methods for addressing iVOS use hand-crafted features (*e.g.*, click-based interactions [68, 69]), but they require a large number of user interactions, making them impractical at a large scale [8]. With the release of the DAVIS interactive benchmark [14], a standardization for the evaluation of iVOS methods has been introduced, where a key aspect is to identify methods that can be used in real-world applications.

2.2.1 *Related Work*

The most common practice in SOTA methods is to tackle the problem following the *Interaction-Propagation* design introduced by Benard *et al*. [70], where they combine two separate networks: an interactive segmentation network based on [71] that predicts and refines segmentation masks based on the interactions of the user as well as a sVOS network [18] which propagates the predicted masks derived from the interaction module to the remaining video frames.

Oh *et al*. [72] connect the interaction and propagation module with an additional component (*i.e.*, feature aggregation), which requires a joint training of all modules. Heo *et al*. [73] use a global and local transfer mechanism during propagation for conveying segmentation results specifically to adjacent and distant frames. A more efficient approach is explored by Miao *et al*. [74] by using a single backbone network for the interaction and the propagation modules. A modular architecture on the popular *Interaction-Propagation* design is proposed by Cheng *et al*. [75] and where a third neural network (*Difference-Aware Fusion*) is added to capture and integrate masks differences between each round. An alternative to the *Interaction-Propagation* design was explored by Chen *et al*. [76], who approached the problem by describing it as a pixel-wise classification task (*i.e.*, k-nearest neighbors in an embedding space) using convolutional neural networks. Recently Chen *et al*. [77] presented an interactive annotation tool, where the user annotates objects using tracked boxes and scribbles.

A persistent bottleneck is determining which frame to annotate for the next round. Hence recent works focus on the applicability of iVOS methods for real-world scenarios by designing networks that automatically determine suitable frames for the user to annotate. Recent approaches address this by identifying a quartet of candidate frames for the user to annotate [78], estimating which frame would yield the most improvement [79], or using a weakly supervised method to indicate the frame and type of interaction [80] to the user. To determine which frame or set of frames to annotate, these methods map each frame in the sequence into an embedding space, restricting them to short videos, as it requires storing the embedding of every frame. Here, each embedding encodes the frame's representation and the quality of the corresponding predicted mask. The best candidate frame is selected by comparing each embedding w.r.t. others and against those of annotated frames, either through an agent [79] or by choosing the embedding that is furthest from any previously annotated embedding [80].

A majority of recent methods focus on scribble-based interactions as it is the default user annotation provided by the DAVIS benchmark. However, since the introduction of SAM [17] more methods have been incorporating additional types of interactions (*e.g.*, clicks, masks. bounding boxes), where previous studies [68, 69], in addition to ours [81], have focused solely on click-based interactions [75, 76].

2.3 Datasets

Several datasets are available to evaluate sVOS approaches, each addressing specific challenges. One of the most established benchmarks in the field is DAVIS [82, 83], providing high-quality and densely annotated video sequences for every single frame. In addition, it also provides the corresponding attributes per sequence (only for the 2016 version [82]), such as occlusions, motion blur, and out-of-view, to cite a few. While initially introduced in 2016 [82] and expanded in 2017 [83], it remains a widely used benchmark for VOS-related tasks, due to its structured format and strong evaluation protocol. Another popular benchmark introduced in 2018 is YouTube-VOS (Y-VOS) [84], which offers a larger set of video sequences compared to DAVIS. However, in contrast to DAVIS, it provides annotated masks only every sixth frame. Note that, unless otherwise stated, most of the datasets discussed below also offer annotations only every sixth frame.

In the last couple of years (essentially since 2023), complementary datasets have been introduced in an attempt to expand the coverage of more complex scenarios. For instance, MOSE [85], OVIS [86], which focus on sequences containing multiple occlusions and clutter situations, making them particularly relevant for evaluating sVOS method's robustness under challenging conditions. VOST [87] introduces sequences where objects undergo extreme shape and appearance variations, making it a strong benchmark for evaluating models beyond standard appearance-based segmentation. In contrast to the other datasets, PUMaVOS [88] targets a niche application, where the goal is to predict partial mask, for example, only the right-hand side of a woman's face. Meanwhile, BURST [89] compiles a large-scale dataset (*i.e.*, highest number of sequences) by aggregating existing object tracking benchmarks and converting the classical bounding-boxes into corresponding object masks.

Although some of these datasets provide a few long sequences, they are outweighed by the shorter videos, as the average duration of the above datasets ranges from 3 to 35 seconds [90]. Hence, a couple of datasets specifically tailored for the long term setting have been proposed by the community. The first dataset introduced that follows this direction is LV1 [51], providing an average video length of 1.3 minutes. However, LV1 only contains three video sequences, where only every 30 frames are annotated. More recently, Hong *et al.* introduced the LVOS datasets (version 1 [91] and version 2 [90]) in 2023. This dataset is an aggregation of existing datasets from VOS and VOT fields but where shorter sequences were removed. It provides an alternative to LV1, with more sequences (50 sequences for the validation set in the first version), where the average video length is circa 1.59 minutes. The dataset also provides a more diverse set of videos and presents mask annotations for every sixth frame. Similarly to DAVIS, the authors also provide corresponding attributes to each video (version 2).

Note that the popular VOT challenge, which focused in the past on bounding box level tracking, has pivoted to pixel level tracking, *i.e.*, sVOS, namely since the VOTS2023 challenge[2] [92]. Here, access to the ground-truth data is limited to the validation set (composed of only two videos), so that participants of the challenge avoid to fine-tune their approach to the test data.

Recently, Meta published an open-source SA-V dataset, used to trained their latest foundational model for video object segmentation, SAM2 [16], providing now the largest dataset for training VOS approaches.

2 https://www.votchallenge.net/vots2023/

3 FUNDAMENTALS

In this chapter, we provide a detailed blueprint of the popular *memory-based* paradigm [42, 47, 48], which lies at the core of this thesis. We begin in Section 3.1 by describing its architectural components and overall workflow. In Section 3.2, we outline the typical training regimes used in memory-based sVOS methods. Finally, Section 3.3 introduces standard evaluation metrics in the literature.

3.0.1 *Problem Definition*

Formally, given a video sequence $\mathcal{V} = \{\mathbf{I}_t \mid t \in \{0, 1, \ldots, T\}\}$, where $\mathbf{I}_t \in \mathbb{R}^{3 \times H_R \times W_R}$ denotes the image at index t. Here H_R and W_R denote the image resolutions in height and width. Our goal with an sVOS approach is to predict for each image a corresponding mask $\mathbf{M}_t \in \{0,1\}^{|\mathcal{O}| \times H_R \times W_R}$, which delineates the spatial extend for each object $o \in \mathcal{O}$ the user would like to track. We initialize sVOS methods (for $t = 0$) with a fine-grained annotated mask, defined as a user cue $\mathbf{U}_t \in \{0,1\}^{|\mathcal{O}| \times H_R \times W_R}$, to specify which objects to track in the given video $\mathcal{V}$. For practical evaluations, this cue is set to the ground-truth mask $\mathbf{G}_t$ ($\mathbf{U}_{t=0} = \mathbf{G}_{t=0}$). After the initialization, no additional user cue is provided, and the method segmented autonomously the remaining frame $\mathbf{I}_{t>0} \in \mathcal{V}$. Note that, while we broadly define the mask dimensions for $\mathbf{M}_t$ and $\mathbf{U}_t$, only one object mask $o \in \mathcal{O}$ can have a value of 1 at any given pixel location (h, w). This guarantees that object masks are mutually exclusive and non-overlapping.

To summarize, for a given video sequence $\mathcal{V}$ and an initial user cue $\mathbf{U}_t$, our segmentation pipeline denotes by f predicts a segmented sequence (a set of masks), such that $\mathcal{S} = \{\mathbf{M}_t \mid t \in \{1, \ldots, T\}\}$, to delineates the spatial and temporal presence of a set of objects $\mathcal{O}$ the user is interested in. Conceptually, we can visualize the segmentation process task as the following mapping function

$$f : \mathcal{V} \times \mathbf{U}_{t=0} \rightarrow \mathcal{S}. \tag{3.1}$$

Note that, we exclude $t = 0$ from the prediction in $\mathcal{S}$, since it's associated ground-truth mask $\mathbf{G}_0$ is already available and used as the initial cue to initialize the method. In Figure 3.1 we visually represent the segmentation process.

3.1 Backbone Architecture

For clarity, we assume a single-object tracking scenario throughout this chapter ($|\mathcal{O}| = 1$), which simplifies the notations and explanations. In addition, we view the segmentation

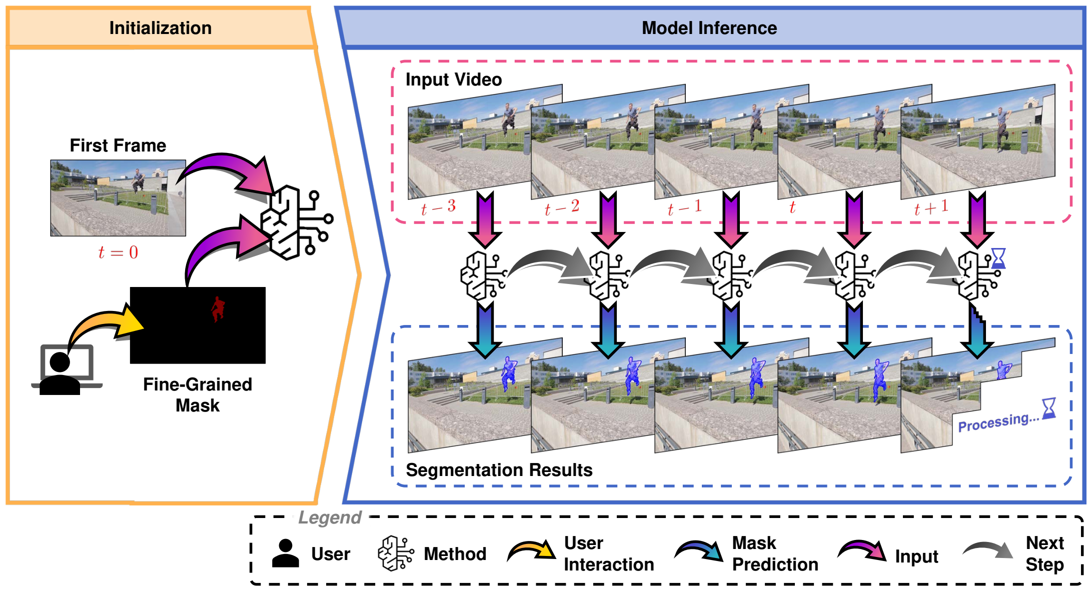


Figure 3.1: Visual representation of the sVOS process. Given a video sequence, the user first indicates the object of interest, typically through a fine-grained mask, to initialize the segmentation pipeline. Afterwards, the method automatically tracks the object by predicting a mask for each frame in a step-wise manner.

of a given video $\mathcal{V}$ as a sequential process, where each frame is processed in temporal order, *i.e.*, one after the other. More precisely, given a frame $\mathbf{I}_t \in \mathcal{V} \setminus \mathbf{I}_0$, our segmentation sVOS backbone f_θ, parameterized by θ (*i.e.*, a Deep Neural Networks (DNN)), predicts a dense probability map $\mathbf{P}_t \in [0,1]^{H_R \times W_R}$ through

$$\mathbf{P}_t = f_\theta \left(\mathbf{I}_t, \mathcal{M}^k, \mathcal{M}^v \right), \tag{3.2}$$

where $\mathcal{M}^k$ and $\mathcal{M}^v$ denote external memory banks that store intermediate representations from previously processed frames. The role of these memory components is further detailed in Section 3.1.4. In summary, at each time step $t > 0$, the segmentation of $\mathbf{I}_{t>0}$ is conditioned on the stored information of previously processed frames inside the external memories.

Each value on the probability map $\mathbf{P}_t(h,w)$ describes the Bernoulli parameter for a pixel at location (h,w), describing it's probability to belong to the foreground as

$$\mathbb{P}\left(\mathbf{M}_t(h,w) = 1 \mid \mathbf{I}_t, \mathcal{M}^k, \mathcal{M}^v; \theta \right) = \mathbf{P}_t(h,w). \tag{3.3}$$

We convert the probability map $\mathbf{P}_t$ to a binary object mask $\mathbf{M}_t \in \{0,1\}^{H_R \times W_R}$ by applying a threshold (*i.e.*, $\tau = 0.5$), as follows

$$\mathbf{M}_t(h,w) = \begin{cases} 1, & \text{if } \mathbf{P}_t(h,w) \geq \tau, \\ 0, & \text{otherwise,} \end{cases} \quad \forall h \in \{1,\ldots,H_R\},\ w \in \{1,\ldots,W_R\}. \tag{3.4}$$

3.1.1 *Overview*

Before examining each component in detail, we provide a high-level overview of the main architectural elements illustrated in Figure 3.2, along with their respective roles:

1. **Encoders** (Subsection 3.1.3): The network consists of two encoders. The first encoder (*i.e.*, key encoder enc_θ^k) processes only a given image $\mathbf{I}_t$ during tracking. The second encoder (*i.e.*, value encoder enc_θ^v) considers $\mathbf{I}_t$ together with the associated predicted object mask $\mathbf{M}_t$ to jointly encode appearance and segmentation information.

2. **External Memory** (Subsection 3.1.4): An external memory module is utilized to store intermediate feature representations. These features, generated by the key and value encoders, are stored in a key memory $\mathcal{M}^k$ and a value memory $\mathcal{M}^v$.

3. **Space-Time Matching** (Subsection 3.1.5): A cross-attention operation between the stored intermediate features and the features of the current frame to be segmented (*i.e.*, the query frame), used to identify relevant object regions.

4. **Decoder** (Subsection 3.1.6): Finally, a decoder dec_θ processes the output of the cross-attention operation to predict $\mathbf{P}_t$, and finally $\mathbf{M}_t$.

3.1.2 *Initialization*

As highlighted previously, the first step is to initialize the method.The network leverages the ground-truth annotation $\mathbf{G}_0$, which serves as the initial reference cue to help the method identify the target object in subsequent frames $t > 0$. During initialization: (1) the key encoder enc_θ^k processes $\mathbf{I}_0$ to extract key embeddings. (2) In parallel, the value encoder enc_θ^v encodes jointly $\mathbf{I}_0$ and $\mathbf{G}_0$ into value embeddings. Next, the key and value embeddings are stored respectively in the external memories *i.e.*, $\mathcal{M}^k$ and $\mathcal{M}^v$. The stored information serves as a the initial reference to identify relevant object regions for segmentation (*i.e.*, guiding the matching process). In the following, we assume that the initialization is complete and focus solely on the tracking process (*i.e.*, $t > 0$).

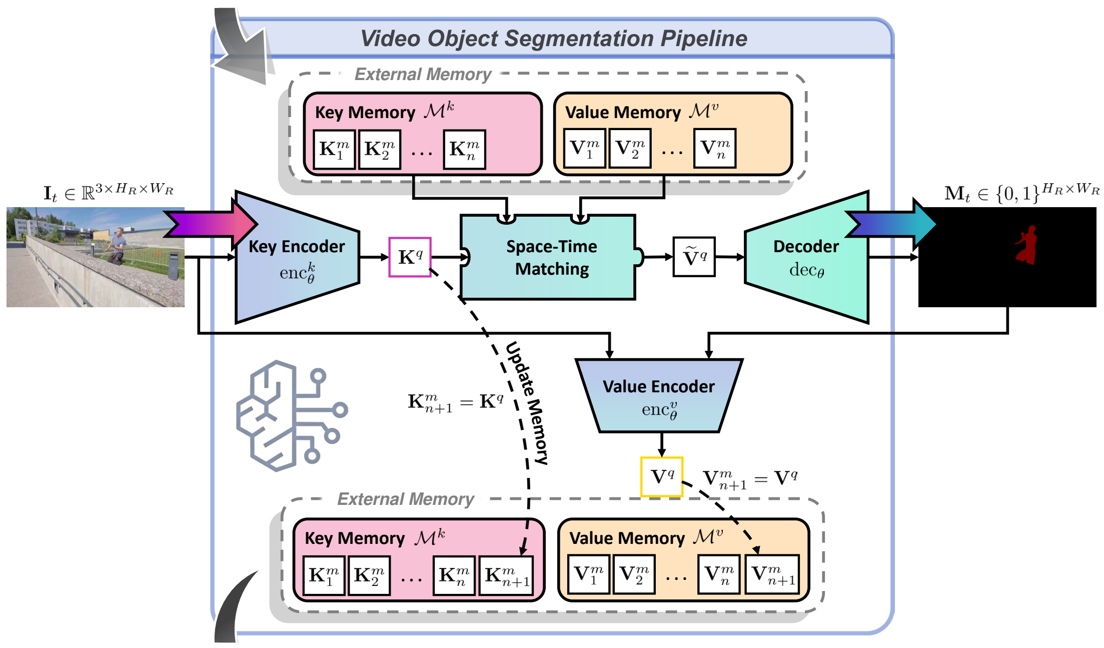

Video Object Segmentation Pipeline
External Memory
Key Memory $\mathcal{M}^k$
$\mathbf{K}_1^m$ $\mathbf{K}_2^m$ … $\mathbf{K}_n^m$
Value Memory $\mathcal{M}^v$
$\mathbf{V}_1^m$ $\mathbf{V}_2^m$ … $\mathbf{V}_n^m$
$\mathbf{I}_t \in \mathbb{R}^{3\times H_R\times W_R}$
Key Encoder enc_θ^k
$\mathbf{K}^q$
Space-Time Matching
$\widetilde{\mathbf{V}}^q$
Decoder dec_θ
$\mathbf{M}_t \in \{0,1\}^{H_R\times W_R}$
Update Memory
$\mathbf{K}_{n+1}^m = \mathbf{K}^q$
Value Encoder enc_θ^v
$\mathbf{V}^q$
$\mathbf{V}_{n+1}^m = \mathbf{V}^q$
External Memory
Key Memory $\mathcal{M}^k$
$\mathbf{K}_1^m$ $\mathbf{K}_2^m$ … $\mathbf{K}_n^m$ $\mathbf{K}_{n+1}^m$
Value Memory $\mathcal{M}^v$
$\mathbf{V}_1^m$ $\mathbf{V}_2^m$ … $\mathbf{V}_n^m$ $\mathbf{V}_{n+1}^m$

Figure 3.2: Overview of an sVOS pipeline based on the memory-based matching paradigm. We assume that initialization has already been performed. Note that, this figure provides a more detailed view of the segmentation process when performed on a single frame $\mathbf{I}t$ at step t, complementing the overview in Figure 3.1. The pipeline takes an image $\mathbf{I}_t$ as input and computes internal embeddings, namely the key $\mathbf{K}^q$ and the value $\mathbf{V}^q$ embeddings, through the query encoder enc_θ^k and value encoder enc_θ^v. These embeddings are either saved or discarded in their respective memory banks $\mathcal{M}^k$ and $\mathcal{M}^v$, depending on the memory update strategy. The method leverages the memory banks embeddings together with the predicted key embeddings $\mathbf{K}^q$ through cross-attention (*i.e.*, space-time matching) to generate pseudo-value embeddings $\widetilde{\mathbf{V}}^q$, which the decoder dec_θ uses to predict the corresponding object mask $\mathbf{M}_t$.

3.1.3 *Encoders*

The network f_θ consists of two encoders: (i) a key encoder enc_θ^k and (ii) a value encoder enc_θ^v. The key encoder $\mathrm{enc}_\theta^k : \mathbb{R}^{3\times H_R\times W_R} \to \mathbb{R}^{C_k\times HW}$ processes a query frame $\mathbf{I}_t$ and maps it to a key feature tensor $\mathbf{K}^q$ as

$$\mathbf{K}^q = \mathrm{enc}_\theta^k(\mathbf{I}_t)\,. \tag{3.5}$$

We denote the channel dimension of the key features as C_k and use the exponent q to indicate an embedding conditioned on the query frame $\mathbf{I}_t$. The original image resolution is given by $H_R \times W_R$, while $H \times W$ represents the spatial dimensions of the features after passing through the encoder. Importantly, the C_k-dimensional column vectors $\mathbf{k}_i^q = \mathbf{K}^q(\bullet, i)$ for all $i \in \{1, 2, \ldots, HW\}$ encode high-level semantic information that is robust to appearance variations. This will be particularly useful during cross-attention, as in Subsection 3.1.5. Intuitively, we select a C_k-dimensional vector at a given spatial location in the encoder's $(H \times W)$ resolution.

Similarly, the value encoder $\mathrm{enc}^v_\theta : \mathbb{R}^{3\times H_R \times W_R} \times \{0,1\}^{H_R \times W_R} \rightarrow \mathbb{R}^{C_v \times HW}$ jointly processes the input frame $\mathbf{I}^q_t$ with its associated object mask $\mathbf{M}_t$, to generate a value feature tensor $\mathbf{V}^q$ by

$$\mathbf{V}^q = \mathrm{enc}^v_\theta\left(\mathbf{I}_t, \mathbf{M}_t\right), \tag{3.6}$$

where we denote with C_v the channel dimensions of the value features. Note that, the C_v-dimensional column vectors $\mathbf{v}^q_i = \mathbf{V}^q(\bullet, i)$ for all $i \in \{1, 2, \ldots, HW\}$, capture both appearance and spatial object information (crucial for guiding future segmentation predictions) [42].

In summary, the key and value embeddings serve complementary roles: key embeddings act as "selectors," guiding the model to relevant regions, while value embeddings act as "carriers," providing information required for accurate mask prediction.

3.1.4 *External Memories*

A defining component of space-time memory-based networks, is the use of external memories, which enables the segmentation model f_θ to retain and reuse information from processed past frames seen during segmentation. This provides a richer context for segmentation compared to previous approaches (*i.e.*, Online Fine-Tuning or Propagation-Based methods).

Note that, to differentiate previously processed frames from the query frame, we will refer to them as *key-memory* and *value-memory* features, and denote them using the exponent m. We let N indicate the memory size and $n \in \{1, 2, \ldots, N\}$ the index of a memory slot. The network stores the key-value pairs of intermediate frames, *i.e.*, previously observed frames $\{\mathbf{I}_0, \mathbf{I}_1, \ldots, \mathbf{I}_{t-1}\}$, in the sets

$$\mathcal{M}^k = \{\mathbf{K}^m_n \mid n \in \{1, 2, \ldots, N\}\}, \quad \text{and} \quad \mathcal{M}^v = \{\mathbf{V}^m_n, \mid n \in \{1, 2, \ldots, N\}\}. \tag{3.7}$$

Depending on the memory management strategy, not all previously processed frames will have their key-value features stored. A common approach is to use a *modulo-based* update rule with a meta-parameter $u \in \mathbb{N}^*$, which stores the representation of a frame $\mathbf{I}_z$ only if $z \equiv 0 \pmod{u}$, for $z \in \{0, 1, \ldots, t-1\}$. Recall that the segmentation is a step-wise process; hence, the memory is updated incrementally. Consequently, (important for Chapter 5) the memory size N is proportional to the video's length T, *i.e.*, $N \propto T$. Note that, the initialization frame $\mathbf{I}_0$ and the associated ground-truth mask $\mathbf{G}_0$ embeddings are always stored in the first memory entry $n = 1$.

In addition, several methods [42, 47, 52] also include the key-value features of the preceding adjacent frame $\mathbf{I}_{t-1}$ (if not already present inside the memories $\mathcal{M}^k$ and $\mathcal{M}^v$). This leverages the idea that the most recent frame $\mathbf{I}_{t-1}$ typically undergoes fewer appearance changes relative to $\mathbf{I}_t$, and thus its representation can be particularly beneficial for accurate segmentation, especially for locating the object.

Based on the memory key and value embeddings, we construct two memory matrices $\mathbf{K}^{\mathcal{M}}$ and $\mathbf{V}^{\mathcal{M}}$ by concatenating the stored memory representations along the columns, such that

$$\mathbf{K}^{\mathcal{M}} = [\, \mathbf{K}^m_1 \;\; \mathbf{K}^m_2 \;\; \dots \;\; \mathbf{K}^m_N \,] \quad \text{and} \quad \mathbf{V}^{\mathcal{M}} = [\, \mathbf{V}^m_1 \;\; \mathbf{V}^m_2 \;\; \dots \;\; \mathbf{V}^m_N \,], \tag{3.8}$$

resulting in $\mathbf{K}^{\mathcal{M}} \in \mathbb{R}^{C_k \times NHW}$ and $\mathbf{V}^{\mathcal{M}} \in \mathbb{R}^{C_v \times NHW}$.

By retaining intermediate frame representations, the network benefits from a larger pool of reference features for cross-frame matching, which enhances the stability and robustness of matching-based methods under prolonged occlusions.

3.1.5 *Space-Time Matching*

To segment the query frame $\mathbf{I}_t$, the model retrieves relevant information from previously stored frames (inside the external memories) using a non-local cross-attention operation (no spatial or temporal constrains), referred to as space-time matching [42]. The goal of this operation is to compute a pseudo value-feature matrix $\widetilde{\mathbf{V}}^q \in \mathbb{R}^{C_v \times HW}$. In essence, we aim to determine "which" and by "how much" the memory value vectors $\mathbf{v}^m_i = \mathbf{V}^{\mathcal{M}}(\bullet, i)$, for all $i \in \{1, 2, \dots, NHW\}$, should contribute to the construction of $\widetilde{\mathbf{V}}^q$.

To this end, we first compute an affinity matrix $\mathbf{A} \in \mathbb{R}^{NHW \times HW}$ between every query key vector $\mathbf{k}^q_j = \mathbf{K}^q(\bullet, j)$, for all $j \in \{1, 2, \dots, HW\}$ and every memory key vector $\mathbf{k}^m_i = (\mathbf{K}^{\mathcal{M}})^\mathsf{T}(i, \bullet)$, for all $i \in \{1, 2, \dots, NHW\}$. Hence, the (i, j)-th entry for $\mathbf{A}$ is computed through

$$\mathbf{A}(i, j) = \operatorname{sim}\left(\mathbf{k}^m_i, \mathbf{k}^q_j\right), \tag{3.9}$$

where $\operatorname{sim} : \mathbb{R}^{C_k} \times \mathbb{R}^{C_k} \to \mathbb{R}$ denotes a similarity function (*e.g.*, dot-product, cosine similarly) that computes a similarity score between two vectors. Intuitively, $\mathbf{A}(i, j)$ quantifies how well the i-th memory key matches the j-th query key. To convert the similarity scores into attention weights, the network applies a column-wise softmax over $\mathbf{A}$, which yields an affinity matrix $\mathbf{W} \in [0, 1]^{NHW \times HW}$ by

$$\mathbf{W}(i, j) = \frac{\exp\big(\mathbf{A}(i, j)\big)}{\sum\limits_{k=1}^{NHW} \exp\big(\mathbf{A}(k, j)\big)}. \tag{3.10}$$

This normalization ensures that for each query element, the attention weights over all memory elements sums to 1. Finally, the pseudo value-feature embedding matrix $\widetilde{\mathbf{V}}^q \in \mathbb{R}^{C_v \times HW}$ for the query frame $\mathbf{I}_t$ is computed by

$$\widetilde{\mathbf{V}}^q = \mathbf{V}^{\mathcal{M}} \mathbf{W}. \tag{3.11}$$

Hence, each column vector is generated by a weighted combination (based on similarity) of the memory value columns vectors.

This is the core element of the approach, which enables the network f_θ to retrieve features from past frames and construct a first prediction towards the final segmentation.

3.1.6 *Decoder*

The final element of the network is the decoder $\text{dec}_\theta : \mathbb{R}^{C_k \times HW} \times \mathcal{F}^{\text{skip}} \to [0,1]^{H_R \times W_R}$, which transforms the aggregated pseudo value features $\widetilde{\mathbf{V}}^q$ into a dense probability map $\mathbf{P}_t$ by

$$\mathbf{P}_t = \text{dec}_\theta \left(\widetilde{\mathbf{V}}^q, \{\mathbf{F}_l\}_{l=1}^{L} \right), \tag{3.12}$$

where $\mathcal{F}^{\text{skip}} = \left\{ \mathbf{F}_l \in \mathbb{R}^{C_l \times H_l \times W_l} \mid l \in \{1, \dots, L\} \right\}$ denotes a set of intermediate feature maps extracted from the key encoder enc_θ^k. These features are passed to the decoder via skip connections to guide the upsampling process. Finally, we obtain the corresponding binary object mask $\mathbf{M}_t$ by thresholding $\mathbf{P}_t$ at $\tau = 0.5$, as described in Equation (3.4).

3.1.7 *Multi-Object Segmentation*

While the core pipeline is designed for single-object tracking, several applications often involve tracking multiple objects simultaneously. Hence, given a set of multiple objects $\mathcal{O} = \{1, 2, \dots, O\}$ to track in $\mathcal{V}$, the network stores independent groups of memory values $\{\mathcal{M}_o^v \mid o \in \mathcal{O}\}$, each dedicated to a different object indexed by o. This is necessary since value features are conditioned on object masks. In contrast, only a single key memory $\mathcal{M}^k$ is maintained, as its features depend solely on the input image, allowing the network to compute the affinity matrix $\mathbf{W}$ only once [48] (see Equation (3.10)), regardless of the number of objects $|\mathcal{O}|$.

Due to the decoder structures, a set of independent probability maps is computed $\mathcal{P}_t^{\mathcal{O}} = \{\mathbf{P}_t^o \mid o \in \mathcal{O}\}$, each representing a Bernoulli distributions over the foreground class of a specific object. However, this independence can lead to inconsistencies in overlapping regions *e.g.*, a pixel may be classified as foreground in two different probability maps. To resolve this, Oh *et al.* [93] propose to adopt a soft aggregation, where the multiple Bernoulli distributions are converted into a single categorical distribution (taking into accounts the background class). Thus, the network needs to compute a probability map that is exclusive to the background region ($o = 0$) by

$$\mathbf{P}_t^0(h, w) = \prod_{o \in \mathcal{O}} \left(1 - \mathbf{P}_t^o(h, w)\right), \quad \forall h \in \{1, \dots, H_R\},\ w \in \{1, \dots, W_R\}. \tag{3.13}$$

Next, we concatenate the multiple probability maps along the channel dimension (object indices) to perform tensor operations, which results in $\mathbf{P}_t^{\mathcal{O}} = \text{concat}\left(\mathbf{P}_t^0, \mathbf{P}_t^1, \dots, \mathbf{P}_t^O\right)$, where $\mathbf{P}_t^{\mathcal{O}} \in [0,1]^{(1+O) \times H_R \times W_R}$. Note that, since the individual probability maps in $\mathcal{P}_t^{\mathcal{O}}$

are obtained via a sigmoid activation, applying a softmax along the channel dimension of $\mathbf{P}_t^{\mathcal{O}}$ simplifies the operation to computing a ratio of probabilities. Hence, for a pixel located at (h, w), the normalized probability map along the channel axis $c \in \{0\} \cup \mathcal{O}$ is given by

$$\widetilde{\mathbf{P}}_t^{\mathcal{O}}(c, h, w) = \frac{\mathbf{Q}_t^c(h, w)}{\sum\limits_{l \in \{0,1,\ldots,O\}} \mathbf{Q}_t^l(h, w)}, \quad \text{where} \quad \mathbf{Q}_t^c = \mathbf{P}_t^c \oslash \left(\mathbf{1}_{H_R \times W_R} - \mathbf{P}_t^c\right), \tag{3.14}$$

$\oslash$ defines the Hadamard division and where the network constructs $\widetilde{\mathbf{P}}_t^{\mathcal{O}}$ by concatenating (along the channel axis) $\widetilde{\mathbf{P}}_t^{\mathcal{O}} = \text{concat}\left(\widetilde{\mathbf{P}}_t^0, \widetilde{\mathbf{P}}_t^1, \ldots, \widetilde{\mathbf{P}}_t^O\right)$, with $\widetilde{\mathbf{P}}_t^{\mathcal{O}} \in [0, 1]^{(1+|\mathcal{O}|) \times H_R \times W_R}$. Finally, the multi-object segmentation mask is computed by assigning each pixel (h, w) to the object class with the highest normalized probability through

$$\mathbf{M}_t^{\mathcal{O}}(h, w) = \underset{c \in \{0,1,\ldots,O\}}{\text{argmax}} \ \widetilde{\mathbf{P}}_t^{\mathcal{O}}(c, h, w). \tag{3.15}$$

With this strategy, the network resolves conflicting regions, converting the multiple masks predicted independently at first into a coherent single mask. This allows the final prediction to consistently assign each pixel to a single object (or background), even in regions where objects overlap.

3.2 Training Regime

At a high level, our objective is to train the segmentation network f_θ to track an object (at pixel level) across a video sequence $\mathcal{V}$, given an initial user cue. To do so, the model must learn to (i) extract meaningful visual features from individual frames, (ii) store and retrieve relevant information through memory, and (iii) produce accurate pixel-wise segmentation masks over time.

Formally, we aim to optimize the network parameters θ using Supervised Learning (SL) by minimizing the empirical risk $\mathcal{R}_{\text{emp}}(\theta)$ over a training set $\mathcal{D}_{\text{train}}$. Let $\Theta \subseteq \mathbb{R}^{|\theta|}$ be the parameter space, such that we define the optimization problem as

$$\hat{\theta} = \underset{\theta \in \Theta}{\text{argmin}} \, \mathcal{R}_{\text{emp}}(\theta), \tag{3.16}$$

where $\mathcal{R}_{\text{emp}}(\theta)$ measures the discrepancy between the predicted masks and the ground-truth annotations.

Intuitively, this training regime encourages the different modules to specialize in complementary roles:

1. The key encoder enc_θ^k learns to encode robust embeddings invariant to visual appearance changes (*e.g.*, object motion, deformation, lighting changes), while remaining temporally consistent. This consistency is essential for reliable cross-attention across frames (see Equation (3.9)).

2. In parallel, the value encoder enc_θ^v produces rich, object-centric embeddings (*e.g.*, shape, boundary, and texture cues) that are projected into a feature space that the decoder dec_θ can effectively leverage to reconstruct accurate segmentation masks.

3. In addition, the decoder dec_θ learns to exploit high-resolution spatial features from the skip connections of enc_θ^k (see Equation (3.12)), to guide the upsampling process.

3.2.1 *Simulating the Tracking Process*

A crucial aspect to account for, as highlighted in Equation (3.2), is that the network's prediction depends not only on the current input frame $\mathbf{I}_t^s \in \mathcal{V}^s$, but also on previously processed frames encoded in the memory banks $\mathcal{M}^k$ and $\mathcal{M}^v$ (see Subsection 3.1.4).

To reflect this temporal dependency during training, a tracking scenario is simulated by using synthetic video clips $\mathcal{V}^s = \{\mathbf{I}_0^s, \ldots, \mathbf{I}_T^s\}$ along with the corresponding sequence of ground-truth masks $\mathcal{G}^s = \{\mathbf{G}_0^s, \ldots, \mathbf{G}_T^s\}$, such that each $(\mathbf{I}_t^s, \mathbf{G}_t^s)$ pair is sampled from a training dataset $\mathcal{D}_{\text{train}}$. We provide further details on how to generate synthetic clips from various datasets (*e.g.*, [26, 84, 94–98]) in Subsection 3.2.4. Note that, we use the superscript s to indicate synthetic data samples generated for training purposes.

Hence, for each synthetic video clip $\mathcal{V}^s$ we train the network f_θ, by replicating the process we would have during inference. Hence, for each synthetic clip $\mathcal{V}^s$, we initialize the key and value memory $\mathbf{K}^{\mathcal{M}} = \text{enc}_\theta^k(\mathbf{I}_0^s)$ and $\mathbf{V}^{\mathcal{M}} = \text{enc}_\theta^v(\mathbf{I}_0^s, \mathbf{G}_0^s)$ as we would during inference. Note that the memory banks are reset for each new synthetic clip.

This sequential training regime implicitly encourages the key encoder enc_θ^k and value encoder enc_θ^v to encode their embeddings consistently across time.

3.2.2 *Empirical Risk Minimization*

Formally, we train the network by minimizing the standard empirical risk over the training set $\mathcal{D}_{\text{train}}$, which quantifies the average discrepancy between the predicted probability maps and the ground-truth annotations. We define the empirical risk as

$$\mathcal{R}_{\text{emp}}(\theta) = \mathbb{E}_{\mathcal{V}^s \sim \mathcal{D}_{\text{train}}} \left[\frac{1}{T} \sum_{t=1}^{T} \ell \left(f_\theta \left(\mathbf{I}_t^s, \mathcal{M}^k, \mathcal{M}^v \right), \mathbf{G}_t^s \right) \right], \tag{3.17}$$

where $\ell : [0,1]^{H_R \times W_R} \times \{0,1\}^{H_R \times W_R} \rightarrow \mathbb{R}$ denotes the loss function that quantifies the discrepancy between the network's predictions $\mathbf{P}_t$ and the ground-truth annotations $\mathbf{G}_t^s$.

In practice, we approximate the expectation over the entire training set $\mathcal{D}_{\text{train}}$ using mini-batches $\mathcal{B} \subset \mathcal{D}_{\text{train}}$, to allow for efficient computation and the use of stochastic optimization methods, *e.g.*, Stochastic Gradient Descent (SGD), Adam [99], or RMSProp [100]. Therefore, we rewrite the empirical risk in Equation (3.17) as

$$\mathcal{R}_{\text{emp}}(\theta; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(\mathcal{V}_b^s, \mathcal{G}_b^s) \in \mathcal{B}} \left(\frac{1}{T} \sum_{t=1}^{T} \ell \left(f_\theta \left(\mathbf{I}_{b,t}^s, \mathcal{M}^k, \mathcal{M}^v \right), \mathbf{G}_{b,t}^s \right) \right). \tag{3.18}$$

Each mini-batch $\mathcal{B} = \{(\mathcal{V}_b^s, \mathcal{G}_b^s) \mid b = \{1, \dots, B\}\}$ consists of synthetic video clips and their respective ground-truth. Here, $\mathbf{I}_{b,t}^s$ and $\mathbf{G}_{b,t}^s$ denote the frame and ground-truth annotations respectively of the b-th video clip for the t-th frame. Hence, for all subsequent frames belonging to the same synthetic clip $\mathbf{I}_{b,t}^s \in \mathcal{V}_b^s \setminus \{\mathbf{I}_{b,0}^s\}$ and $\mathbf{G}_{b,t}^s \in \mathcal{G}_b^s \setminus \{\mathbf{G}_{b,0}^s\}$, we predict the corresponding probability map $\mathbf{P}_{b,t}$ as in Equation (3.2). Note that we exclude the first image-mask pairs $(\mathbf{I}_{b,t=0}^s, \mathbf{G}_{b,t=0}^s)$ of each synthetic video clip $\mathcal{V}^s$ from Equations (3.17) and (3.18), as $(\mathbf{I}_{b,t=0}^s, \mathbf{G}_{b,t=0}^s)$ are only meant to initialize the memory and no prediction is performed by the network for it.

Following prior works [42, 47], to stabilize the network's training and prevent the accumulation of prediction errors when predicting on a synthetic clip $\mathcal{V}^s$, we update the value memory $\mathcal{M}^v$ using the corresponding ground-truth $\mathbf{G}_{b,t}^s \in \mathcal{G}_b^s$ rather than the predicted masks $\mathbf{M}_{b,t}^s$. Thus, we compute the value embeddings by $\mathbf{V}_{b,t}^m = \text{enc}_\theta^v(\mathbf{I}_{b,t}^s, \mathbf{G}_{b,t}^s)$ and store them in the value memory bank $\mathcal{M}^v$.

By minimizing the empirical risk over synthetic sequences and stabilizing memory updates with ground-truth masks, we train the network to effectively encode and leverage prior frame information for consistent object segmentation.

3.2.3 *Loss Function and Optimization*

In practice, Binary Cross Entropy (BCE) is used as the primary loss function ℓ to measure the discrepancy between the predicted probability map $\mathbf{P}_{t,b}$ and the corresponding ground-truth annotation $\mathbf{G}_{t,b}^s$, such that

$$\ell\left(\mathbf{P}_{t,b}, \mathbf{G}_{t,b}\right) = \frac{-1}{H_R W_R} \sum_{h=1}^{H_R} \sum_{w=1}^{W_R} g_{h,w}^s \log\left(p_{h,w}\right) + \left(1 - g_{h,w}^s\right) \log\left(1 - p_{h,w}\right), \tag{3.19}$$

where $g_{h,w}^s \in \{0, 1\}$ is the ground-truth label and $p_{h,w} \in [0, 1]$ is the predicted Bernoulli parameter at pixel location (h, w) for the b-th video of the t-th image in $\mathcal{V}_b^s$. We compute the total training loss as defined in Equation (3.18).

While BCE serves as our primary loss function, other losses can be employed to address specific challenges, for example, bootstrapped cross-entropy or Focal Loss [101], which place more emphasis on hard examples and mitigates class imbalance. Moreover, combining BCE with region-based metrics such as the Dice loss [102] allows the model to capture object-level structure and boundaries better.

After computing the empirical risk $\mathcal{R}_{\text{emp}}(\theta; \mathcal{B})$ over a mini-batch $\mathcal{B}$, we update the model parameters θ via backpropagation. We compute the gradients $\nabla_\theta \mathcal{R}_{\text{emp}}(\theta; \mathcal{B}) \in \mathbb{R}^{|\theta|}$

over the entire batch, which leads to more stable updates compared to per-video supervision. To optimize the network, we use the Adam optimizer [99], which adapts learning rates during training and supports robust convergence across diverse scenarios.

3.2.4 *Creating Synthetic Video Clips*

Following the methodologies outlined in prior works [26, 72, 93], the network is trained through a two-stage training strategy to progressively enhance its ability to segment objects in videos.

In the first stage, we leverage static datasets $\mathcal{D}^{\text{static}}_{\text{train}}$ [94–98] to construct synthetic video clips $\mathcal{V}^s$ and their corresponding mask sequences $\mathcal{G}^s$. We begin by sampling a single image and its corresponding mask $(\mathbf{I}^{\mathcal{D}}, \mathbf{G}^{\mathcal{D}}) \sim \mathcal{D}^{\text{static}}_{\text{train}}$, and apply a set of transformations to generate $T+1$ augmented pairs. We group the resulting $T+1$ image-mask pairs into a synthetic clip $\mathcal{V}^s = \{\mathbf{I}^{\mathcal{A}}_t\}^T_{t=0}$ and its corresponding mask sequence $\mathcal{G}^s = \{\mathbf{G}^{\mathcal{A}}_t\}^T_{t=0}$. We generate each pair $(\mathbf{I}^{\mathcal{A}}_t, \mathbf{G}^{\mathcal{A}}_t)$ independently for each frame and batch index by applying spatial transformations $\psi^{\text{static}}_{b,t} \sim \Psi^{\text{static}}$ to the original image-mask pair. Hence, through these augmentations, we simulate temporal variations (*e.g.*, motion blur, rotations, shearing) derived from a single sampled image-mask pair. Figure 3.3 illustrates the static images from which a synthetic video sequence $\mathcal{V}^s$ and its ground-truth counterpart $\mathcal{G}^s$ were generated.

In the second stage, we leverage real videos from DAVIS [26] and Y-VOS [84] to generate synthetic training sequences. We form each synthetic clip $\mathcal{V}^s = \{\mathbf{I}^{\mathcal{A}}_t\}^T_{t=0}$ and its corresponding mask sequence $\mathcal{G}^s = \{\mathbf{G}^{\mathcal{A}}_t\}^T_{t=0}$ by sampling $T+1$ image-mask pairs from a real video and its ground-truth annotation $(\mathcal{V}^r, \mathcal{G}^r) \sim \mathcal{D}^{\text{real}}_{\text{train}}$. To encourage sample diversity within a batch, we construct each synthetic sequence $(\mathcal{V}^s_b, \mathcal{G}^s_b)$ from a different real video. To avoid a bias toward early frames, the initial frame-mask pair $(\mathbf{I}^{\mathcal{D}}_0, \mathbf{G}^{\mathcal{D}}_0)$ (used to initialize the model during training) is sampled by selecting a random starting index uniformly from the available frames in the real video. However, this starting index must ensure that we can still draw T additional frames from the same video, that can allow for an adequate temporal spacing between each subsequent frame drawn [48]. To progressively increase the difficulty throughout the training, we gradually expand the allowed spacing between two successive samples in the latter training iterations (curriculum learning). This is presumably helpful as including more temporally distant frames in a synthetic clip, trains the network with more challenging examples. Similarly, as in the first stage, each sampled frame and its corresponding mask are augmented with spatial transformations parameterized by $\psi^{\text{real}}_{b,t} \sim \Psi^{\text{real}}$, applied independently for each frame and batch index. Figure 3.4 displays examples of a synthetic video clip $\mathcal{V}^s$ and its corresponding ground-truth sequence $\mathcal{G}^s$. These synthetic clips are generated by sampling three images from a real video and applying augmentation techniques.

By first learning from synthetic clips, the model establishes strong feature representation, enabling the network to gradually adapt to more complex scenarios encountered in real-world videos.

3.3 Evaluation Metrics

Following the standard protocol [82], we evaluate sVOS methods using two primary metrics: the region similarity $\mathcal{J}$ and the contour accuracy $\mathcal{F}$. Note that, to ease the ranking of different methods in the literature, we also report the average $\mathcal{J}\&\mathcal{F}$ score, by $\mathcal{J}\&\mathcal{F} = \frac{1}{2}(\mathcal{J} + \mathcal{F})$.

Given an evaluation dataset $\mathcal{D}$, we compute the global score $\mathcal{M}_{\mathcal{D}}$ ($\mathcal{M}_{\mathcal{D}}$ acts as a placeholder for $\mathcal{J}$ or $\mathcal{F}$) by averaging over all objects present in the $\mathcal{D}$ by

$$\mathcal{M}_{\mathcal{D}} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \frac{1}{T_o} \sum_{t_o=1}^{T_o} \mathcal{M}_t^o, \tag{3.20}$$

where $o \in \mathcal{O}$ denotes the indices of an object and t_o the frame index where the corresponding object is present is a video sequence $\mathcal{V}$. Here, $\mathcal{M}_t^o$ represents the per-frame metric of $\mathcal{J}_t^o$ or $\mathcal{F}_t^o$ (depending on whether we want $\mathcal{J}$ or $\mathcal{F}$) for the o-th object in $\mathcal{D}_{\text{eval}}$.

More specifically, for a given binary object mask prediction $\mathbf{M}_t^o \in \mathcal{S}$, for a frame t object indices o and its corresponding ground-truth annotation $\mathbf{G}_t^o \in \mathcal{G}$ we compute the per-frame metrics as follows (note that numerical stability is ensured using a small constant $\delta > 0$ in the denominator, though not shown here for brevity):

- The per-frame **Jaccard Index** $\mathcal{J}_t^o$, also known as the intersection-over-union (IoU), quantifies the region overlap between a predicted mask $\mathbf{M}_t^o$ and its corresponding ground-truth $\mathbf{G}_t^o$ by

$$\mathcal{J}_t^o = \frac{|\mathbf{G}_t^o \cap \mathbf{M}_t^o|}{|\mathbf{G}_t^o \cup \mathbf{M}_t^o|}. \tag{3.21}$$

  Hence, with $\mathcal{J}_t^o$ we measure the ratio of correctly classified pixels w.r.t. the total region covered by both masks.

- The **F-measure** $\mathcal{F}_t^o$ captures contour alignment (*i.e.*, accuracy) and is defined as the harmonic mean of contour precision c^{p} and recall c^{r} by

$$\mathcal{F}_t^o = 2\frac{c^{\mathrm{p}} \cdot c^{\mathrm{r}}}{c^{\mathrm{p}} + c^{\mathrm{r}}}. \tag{3.22}$$

  To compute c^{p} and c^{r}, we define a function $\mathrm{cont} : \{0,1\}^{H_R \times W_R} \to \mathcal{P}(\mathbb{N}^* \times \mathbb{N}^*)$ to extract a set of contour pixel coordinates (*i.e.*, (w, h)) for a given binary mask. Note that, here $\mathcal{P}$ indicates a power set, meaning it represents the set of all possible subsets of contour pixel coordinates. Hence, let $c_m \in \mathrm{cont}(\mathbf{M}_t^o)$ and $c_g \in \mathrm{cont}(\mathbf{G}_t^o)$

denote individual contour points from the predicted and ground-truth masks. We obtain c^{p} and c^{r} by

$$\begin{aligned} c^{\mathrm{p}} &= \frac{\left|\left\{ c_m \in \mathrm{cont}(\mathbf{M}_t^o) \;\middle|\; \exists\, c_g \in \mathrm{cont}(\mathbf{G}_t^o),\ \|c_m - c_g\|_2 \leq \epsilon \right\}\right|}{|\,\mathrm{cont}(\mathbf{M}_t^o)|}, \\ c^{\mathrm{r}} &= \frac{\left|\left\{ c_g \in \mathrm{cont}(\mathbf{G}_t^o) \;\middle|\; \exists\, c_m \in \mathrm{cont}(\mathbf{M}_t^o),\ \|c_g - c_m\|_2 \leq \epsilon \right\}\right|}{|\,\mathrm{cont}(\mathbf{G}_t^o)|}, \end{aligned} \tag{3.23}$$

where two contour points c_m and c_g are considered a match if their Euclidean distance satisfies $\|c_m - c_g\|_2 \leq \epsilon$. Here, ϵ is a fixed spatial tolerance defined by the benchmark [82].

To summarize, $\mathcal{J}$, $\mathcal{F}$, and $\mathcal{J}\&\mathcal{F}$ constitute the standard metrics in several sVOS benchmarks, where higher values indicate better segmentation accuracy. Notably, the metrics reflect model performance over all annotated objects rather than over the sequences for a given dataset $\mathcal{D}$.

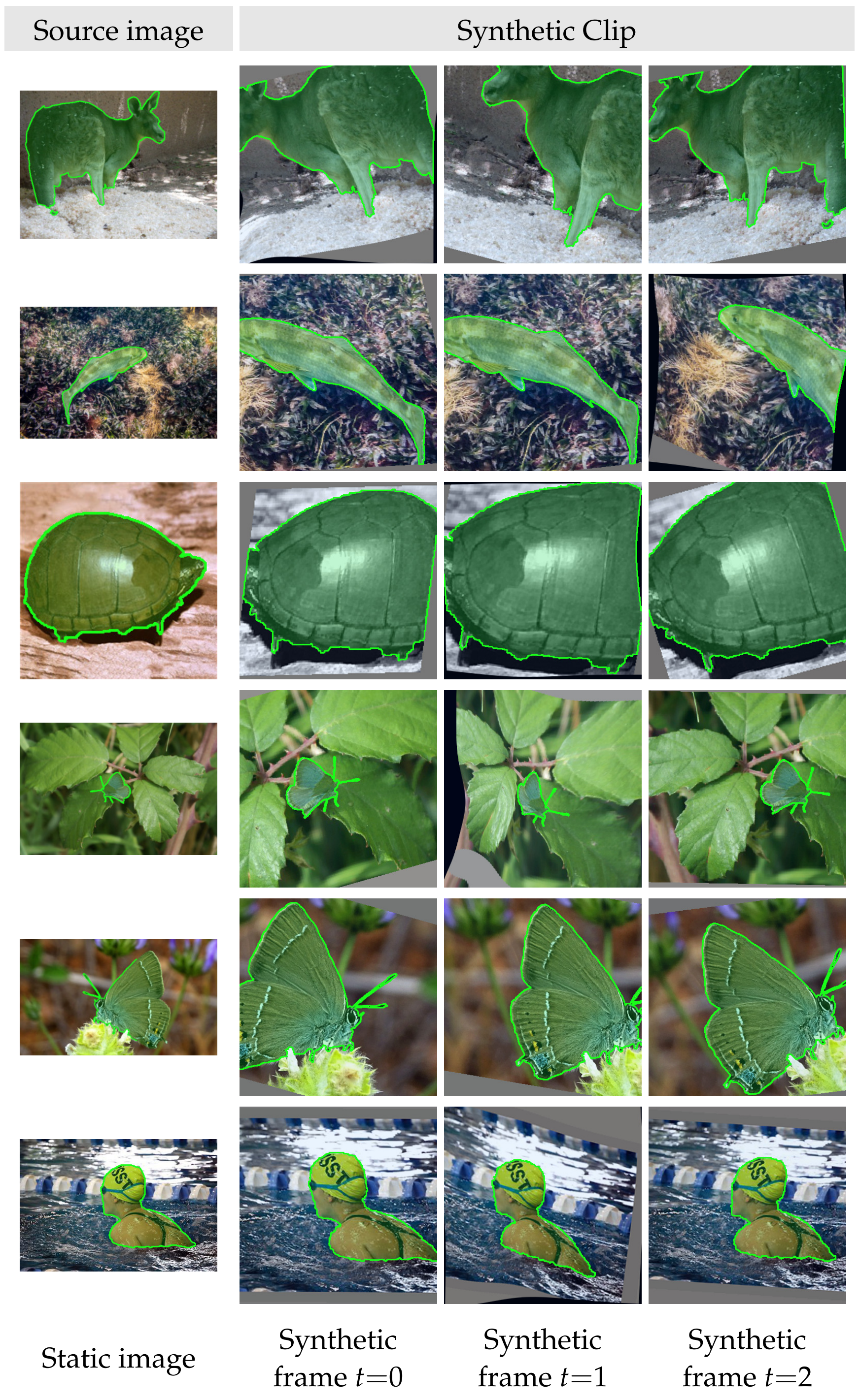


Figure 3.3: Illustration of static images [94–98] augmented into a three-frame synthetic video sequence during training. We blend and outline the contours of the corresponding ground-truth masks in green for better visualization.

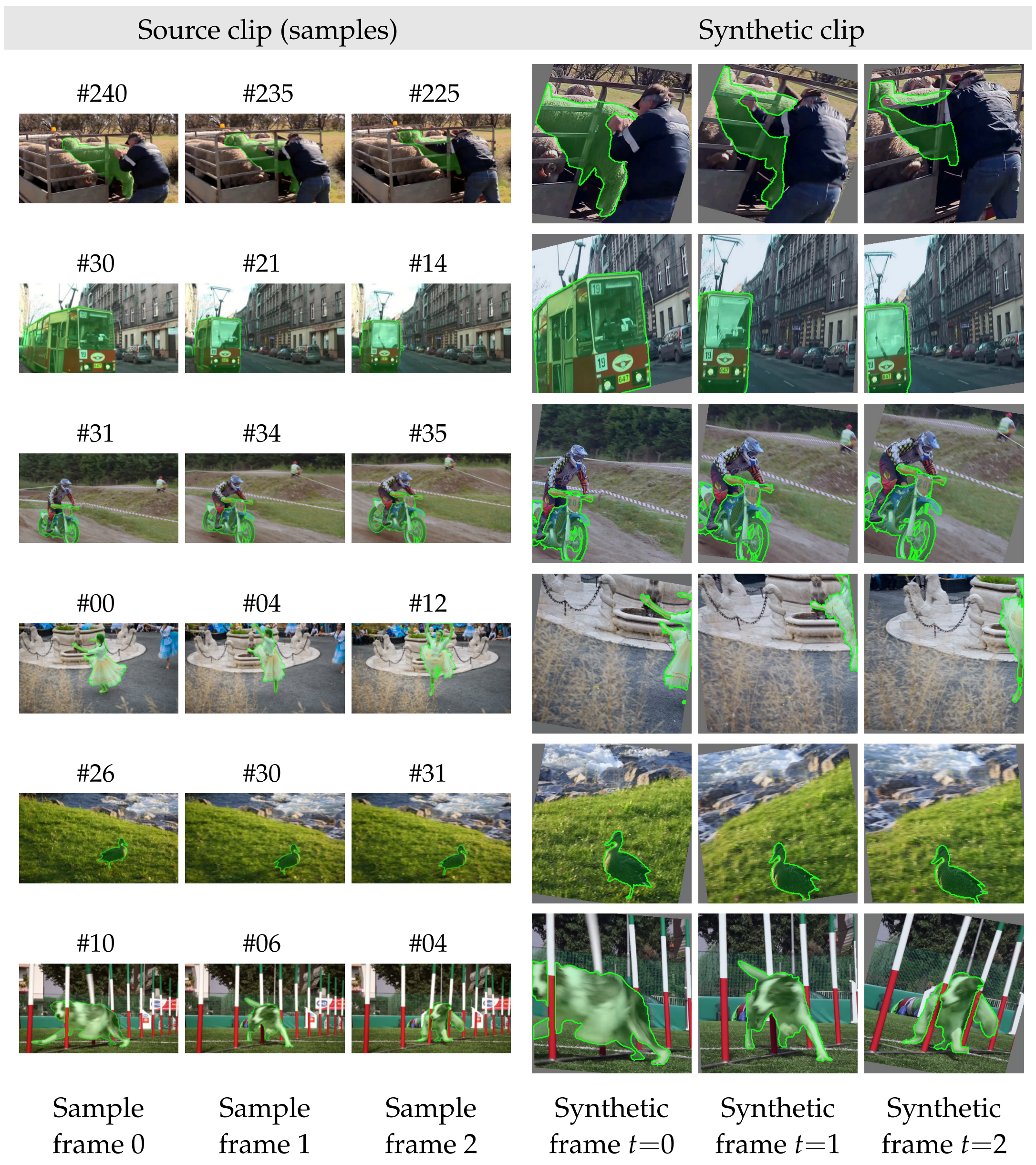


Figure 3.4: Example of source videos [26, 84] used to generate a three-frame synthetic video sequence during training. For each synthetic clip, only three frames are randomly sampled from a longer original video. We indicate the original position of these sampled frames above each image to highlight the temporal jumps between them. We blend and outline the contours of the corresponding ground-truth masks in green for better visualization.

Part II

USER-CENTRIC AND SCALABLE VIDEO OBJECT SEGMENTATION

4 CLICK-BASED INTERACTIVE VIDEO OBJECT SEGMENTATION

One of the overarching goals of this thesis is to explore more efficient user interactions (related to RQ1) when segmenting a video sequence. In this chapter, we explore the use of click-based interactions as a lightweight and intuitive medium for users to indicate which arbitrary object a VOS method should track in a given video sequence. In this context, iVOS approaches provide an initial starting point, as they leverage scribbles to define which object to segment in a video.

Our first objective is to assess whether click-based interactions are a viable option to guide and correct a VOS pipeline while preserving segmentation quality. We explore a click-based solution that aims to balance segmentation performance with interaction effort on the iVOS task. Our pipeline, **CiVOS**, comprises two decoupled modules, each responsible for a specific subtask. First, a *interaction module*, which transforms sparse user clicks into an object mask. Secondly, a *propagation module*, that propagates the mask to the remaining frames of the video. We evaluate CiVOS on the popular interactive DAVIS dataset [26], and propose three strategies for converting its default scribble-based annotations into click-based counterparts for fair comparison.

This chapter is based on our *ICIP 2022* paper [81], and is structured as follows. We outline the motivation behind CiVOS in Section 4.1. We formalize our problem Section 4.2 and review related work in Section 4.3. We describe our pipeline in Section 4.4. In Section 4.5, we detail our click generation strategies and adapt the DAVIS evaluation metric for hardware-independent comparisons. We present our experimental results in Section 4.6. Finally, we conclude the chapter with a discussion and summarize our key contributions in Section 4.7.

4.1 Motivation

At the time this research was initiated, most existing iVOS [70, 72–75] approaches relied predominantly on scribble-based interactions to fully segment a given video sequence. While faster than full mask annotations, scribbles still require significant user effort, around 11 seconds per object instance [14, 103] (depending on the level of detail). Moreover, we have to take into account the average number of times a user might need to interact (*i.e.*, interaction rounds) with the method to achieve satisfactory results or similar performance to sVOS approaches. By taking 8 as our average number of interaction rounds (as used in the DAVIS benchmark [14]), the total time required to segment a single object (excluding the review process) is actually much closer to

90 seconds. As the primary goal of iVOS approaches is to segment an arbitrary object instance with ideally minimal user effort, a natural question arises: Why not rely on click-based input to reduce the user's workload? As highlighted by Bearman *et al.* [103] and Benenson *et al.* [104], click-based annotations can reduce the interaction time from 11 seconds (scribbles) to just 1–3 seconds per object, allowing users to designate target objects more rapidly and intuitively [105].

Here, Interactive Image Object Segmentation (iIOS) methods are presumably helpful, as they have demonstrated impressive performance using solely click-based interactions to generate accurate masks for arbitrary objects in single images [71, 106–114]. To extend these benefits to video, we introduce a novel *Click-based interactive Video Object Segmentation* (CiVOS) framework, which aims to simplify the annotation process in iVOS without sacrificing segmentation quality. We adopt the popular decoupled *Interaction-Propagation* design introduced by Benard *et al.* [70]: (1) an *interactive image segmentation* pipeline [17, 111, 112] that converts sparse user clicks into an initial object mask, (2) and an sVOS-based *mask propagation* module [38, 48, 72] that extends the mask across the rest of the video.

4.2 Problem Definition

Given a video sequence $\mathcal{V} \in \{\mathbf{I}_t \mid t \in \{0, 1, \dots, T\}\}$, our goal with iVOS approaches is to predict a set of masks $\mathbf{M}_t^r \in \{0,1\}^{|\mathcal{O}| \times H_R \times W_R}$, where $\mathcal{S}_r = \{\mathbf{M}_t^r \mid t \in \{0, 1, \dots, T\}\}$ denotes the set of predicted masks across all frames at round r.

We can decompose each interaction round into three steps as shown in Figure 4.1. Thus, given a round r, the first step leverages (1) **user interaction**, where the user provides a sparse interaction map $\mathbf{U}_t^r \in \{0,1\}^{(1+|\mathcal{O}|) \times H_R \times W_R}$ (based on scribbles [47, 72–74, 78]) for the t-th image $\mathbf{I}_t \in \mathcal{V}$. Expect for the first round ($r = 1$), a user typically provides false positive and false negative annotations, each instructing the method which pixel regions belong to the background or foreground region (the set of objects $\mathcal{O}$ to track). For the first round, only false negative annotations are expected, hence $\mathbf{U}_t^{r=1} \in \{0,1\}^{|\mathcal{O}| \times H_R \times W_R}$ (2) In the second step we have the **model inference**, where the method predicts a new segmented video sequence based on the user input $\mathbf{U}_t^r$ and the current results $\mathcal{S}_r$. Conceptually, we can visualize the iVOS prediction as the following mapping

$$f : \mathcal{V} \times \mathcal{S}_r \times \mathbf{U}_t^r \to \mathcal{S}_{r+1}, \tag{4.1}$$

where $\mathcal{S}_{r+1}$ is the next's rounds refined segmented sequence and f denotes a complete iVOS pipeline. Note that in Equation (4.1), we visualize the segmentation for a given video $\mathcal{V}$ as a batch-like process under the iVOS perspective. Finally, (3) the **Review and selection** step, where the user reviews the resulting output $\mathcal{S}_{r+1}$, and selects a new frame to interact with. Steps (1), (2) and (3) are repeated over a total of R interaction rounds, *i.e.*, until the user is satisfied with the final results.

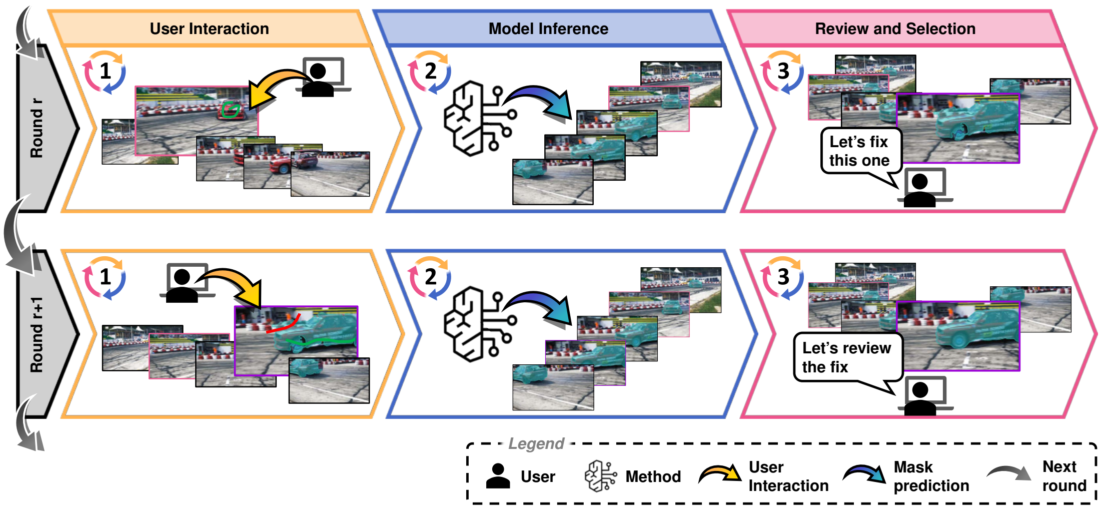


Figure 4.1: Overview of a typical interaction round, as defined by the DAVIS benchmark [14] for the iVOS task. We can decompose a round into three steps, firstly a user interaction, secondly the model's prediction and thirdly the reviewing process.

In this setup, the most popular type of interaction to generate sparse interactions maps $\mathbf{U}_t^{r,o} \in \{0,1\}^{H_R \times W_R}$, are based on scribbles $u_t^{r,o} \in \mathcal{P}(\mathbb{N}^* \times \mathbb{N}^*)$, where $\mathcal{P}$ denotes the power set describing the scribbles position as point coordinates in the image space. However, in this chapter, we address the challenge of maintaining high segmentation quality under an extreme form of user input, *i.e.* click-based interactions, such that $u_t^{r,o} \in \mathbb{N}^* \times \mathbb{N}^*$ describes only a point in the image space.

4.3 Related Work

Here, we briefly highlight prior iVOS approaches, as a more detailed review of related work in for this field is already provided in Chapter 2.2.

4.3.1 *Interactive Video Object Segmentation*

Early methods for addressing iVOS use hand-crafted features (*e.g.*, click-based interactions [68, 69]), but they require a large number of user interactions, making them impractical at a large scale [8]. With the release of the DAVIS interactive benchmark [14], a standardization for evaluating iVOS methods has been introduced, where a key aspect is to identify methods that can be used in real-world applications. A majority of recent methods focus on scribble-based interactions as it is the default user annotation provided by the DAVIS benchmark, with the exception of two studies allowing for click-based interactions [75, 76].

The current practice in state-of-the-art methods is to tackle the problem following the *Interaction-Propagation* design introduced by Benard *et al.* in [70]. The authors combine two separate networks: an iIOS network based on [71] that predicts and refines segmentation masks based on the interactions of the user as well as an sVOS network [18], which propagates the predicted masks derived from the interaction module to the remaining video frames.

As we already presented popular iVOS approaches in Chapter 2 along with the sVOS methods that compose the second component in iVOS methods, we will focus our attention on the other element composing iVOS methods, the "interaction-to-mask" part, which in our use case is based on iIOS as they convert clicks to mask.

Note, to ease up notations and explanation, we assume a single object to be segmented for a given video $\mathcal{V}$ in the following section.

4.3.2 *Interactive Image Object Segmentation*

The goal in iIOS is to predict an accurate mask of an arbitrary object instance using minimal user annotations (eg, clicks [71, 108, 110, 111, 113–115], extreme points [116–118], or bounding boxes [119–121]).

Note that clicks are the most intuitive interaction type, as pointing to an object (*i.e.*, hand, arm, fingers) is the most intuitive [103, 104] from for humans to convey information on an image and the most cost-efficient. Xu *et al.* [71] introduce deep learning in the context of iIOS to extract salient features. By relying on click-based interactions, the authors can simulate user inputs and generate a large number of training samples. Liew *et al.* [106] extend the previous work by exploiting the context of local regions around user interactions. Mahadevan *et al.* [107] introduce iterative training for interactive segmentation networks. Li *et al.* [108] consider the multi-modal nature of iIOS and propose a network to infer multiple masks in order to cover all plausible solutions as well as a second network to select a result from the set of potential object masks. Song *et al.* [109] use a *Markov Decision Process* trained with reinforcement learning to automatically generate interactions based on the initial interactions of the user. Jang *et al.* [110] propose the *Back-propagation Refinement Scheme* (BRS) to constrain their network to predict correct labels at user-specified locations by optimizing the interaction maps (*i.e.*, the input of the network) through forward and backward passes. Sofiuuk *et al.* [111] elaborate the idea further and introduce *feature*-BRS (f-BRS). Instead of optimizing the interaction maps, f-BRS optimizes auxiliary parameters in the last layers of the network. This reduces the computational time during the forward and backward passes. Lin *et al.* [112] emphasize the role of the first interaction by introducing a first click attention module, which leverages the first click as guidance for the following interactions. Kontogianni *et al.* [113] treat user corrections as training samples to update the network parameters, which also enables cross-domain adaptation. Sofiiuk *et al.* [114] expand upon the iterative training strategy proposed in[107] by introducing a new loss function and an encoding layer that allows for the integration of additional external information without affecting the pre-trained weights of the backbone encoder. In this work, they also replace the previous f-BRS backbone with an HRNet [122] combined with an OCR [123] network, which helps maintain high-quality features throughout the network and results in a more precise segmentation mask.

Since the introduction of SAM [17] by Kirillov *et al.*, a plethora of SAM-based methods have been proposed to solve the task in medical imaging [124] and natural images [125]. For instance, SAM-HQ [61] by Ke *et al.* improves upon SAM by better handling complex shapes, such as thinner structures and objects with holes. Additionally, faster approaches like FastSAM [62] and MobileFast [63] have been developed to enhance performance and efficiency.

4.4 Framework

Our proposed method (*i.e.*, CiVOS), to reduce the annotation effort on iVOS, relies on a modular design [47]. We have essentially an interaction module (see Subsection 4.4.2) and a propagation module (see Subsection 4.4.3). We provide in Figure 4.2 an overview of our pipeline.

4.4.1 *Click-based Interactive Video Object Segmentation*

As VOS can be defined as a binary classification problem that aims to distinguish pixels of a specific object from other (background) structures, incorrectly classified pixels can be regarded as either false positives or as false negatives. Hence, to rectify the misclassified pixel regions, the user would indicate false negative and false positive regions with corresponding interactions. As illustrated in Figure 4.2, we build on the modular concept introduced by MiVOS [75] for the proposed CiVOS framework, which comprises two fundamental deep learning architectures, an interaction and a propagation module. To simplify explanations and notations, we assume a single-object scenario.

Given a click-based user input $\mathbf{U}^r_{t_r}$ (*i.e.*, false positive and false negative interactions) for a frame $\mathbf{I}^r_{t_r} \in \mathbb{R}^{3\times H_R \times W_R}$, the interaction module predicts a corresponding segmentation mask $\mathbf{M}^r_{t_r} \in \{0,1\}^{H_R \times W_R}$. Here, $r \in \mathcal{R} = \{1, 2, \dots, R\}$ denotes the interaction round, and $t_r \in \{0, 1, \dots, T\}$ is the index of the frame t annotated at round r. The superscript r in both $\mathbf{I}^r_{t_r}$ and $\mathbf{M}^r_{t_r}$ explicitly indicates the interaction round.

Subsequently, the propagation module propagates the mask $\mathbf{M}^r_{t_r}$ bidirectionally (backward and forward) starting from the annotated frame $\mathbf{I}^r_{t_r}$. In each direction, the propagation is applied to all frames between t_r and the nearest previously annotated frame $t_{r^\star} \in \mathcal{T}_\mathcal{R}$ for that direction, where $r^\star < r$, and $\mathcal{T}_\mathcal{R} = \{t_1, t_2, \dots, t_r\}$ denotes the set of all annotated frames. If no such annotation exists in a given direction, the propagation proceeds to the beginning ($t = 0$) or end ($t = T$) of the video sequence, respectively.

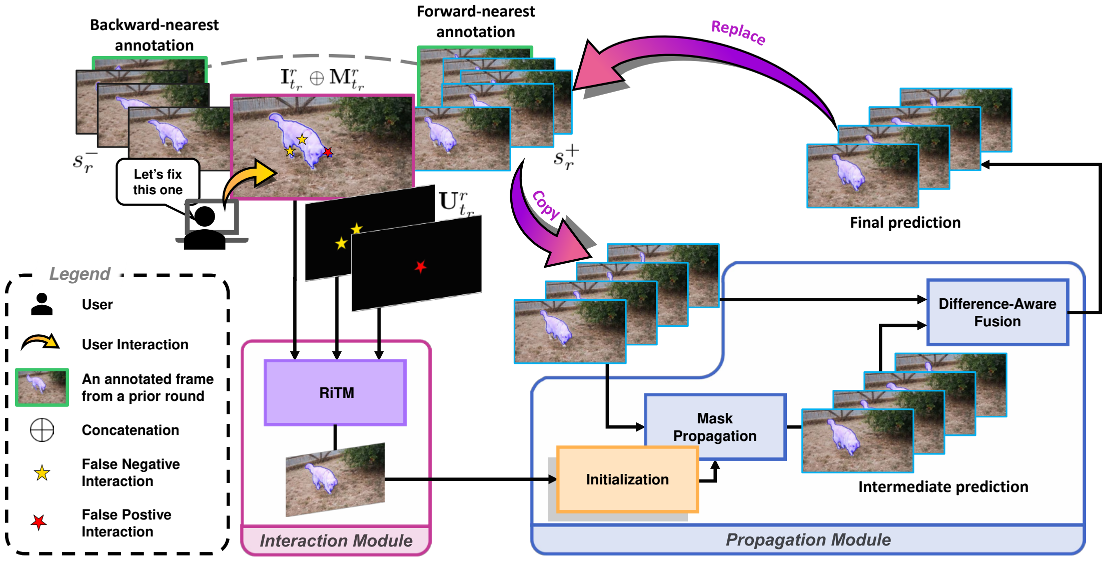


Figure 4.2: Illustration of our CiVOS pipeline, where the user corrects the results of a previous round prediction. To maintain clarity, we illustrate the process only for the forward propagation t_r^+ of round r. The user reviews the current segmentation results of round r, and decides to annotate frame $\mathbf{I}^r_{t_r}$ by corrective clicks (*i.e.*, positive and negative) denoted by a yellow star and red star respectively to indicate false positive and false negative region. The user input $\mathbf{U}^r_t$ and the mask predicted in a previous round r^*, $\mathbf{M}^*_t$ are passed through by the interaction module, which predicts a corresponding object mask (refined mask) $\mathbf{M}^r_t$. This mask initializes the Propagation Module, which processes the set of frames indexed by s_r^+ to first predict intermediate mask, where are then fused with the mask from the previous round to obtain the final results.

More formally, we define the forward propagation boundary t_r^+, *i.e.*, the last frame to which the mask is propagated in the forward direction as

$$t_r^+ = \min\left(\{t_{r^\star} - 1 \mid r^\star < r,\ t_{r^\star} > t_r\} \cup \{T\}\right), \tag{4.2}$$

and the backward propagation boundary t_r^- as

$$t_r^- = \max\left(\{t_{r^\star} + 1 \mid r^\star < r,\ t_{r^\star} < t_r\} \cup \{0\}\right). \tag{4.3}$$

Note that we subtract and add 1 respectively to t_r^+ and t_r^- as otherwise we would propagate mask information to an annotated frame. These definitions ensure that the mask is propagated from frame t_r in both directions, up to the closest previously annotated frame $t_{r^\star}$. If no such annotation exists in a given direction, propagation extends to the beginning or end of the video sequence, respectively.

4.4.2 *Interaction Module*

We consider RITM [114] as our interaction module, to generate object masks based on click-based inputs. Here, HRNet [122] and OCR [123] serve as the backbone for the segmentation network.

As illustrated in Figure 4.2, the inputs for the interaction module are: the current frame used for interactions $\mathbf{I}^r_{t_r}$ two encoded interaction maps [104, 114]; *i.e.*, $\mathbf{U}^r_{t_r} \in \{0,1\}^{2\times H_R \times W_R} \in$ that represent incorrectly segmented or missing mask areas and the latest available mask $\mathbf{M}^{r^\star}_t$, predicted in round $r^\star < r$. We use a convolutional block (*i.e.*, *Conv1S* [114]) as illustrated in Figure 4.2 to take additional external inputs into account without affecting the pre-trained weights of the backbone segmentation network (*i.e.*, HRNet [122]). Here, the mask $\mathbf{M}^{r^\star}_t$ is concatenated channel-wise with the interaction maps $\mathbf{U}^r_t$, before passing through a convolutional layer. The output of the convolutional layer (*i.e.*, *Conv1S*) is a tensor with the same dimensions as the tensor given by the first layer of HRNet, such that an element-wise summation can be applied along the channel axis. As in f-BRS [111] the predicted mask is optimized to predict the user-specified labels at the locations where the interactions occurred.

4.4.3 *Propagation Module*

We consider the propagation branch of MiVOS [75] (*i.e.*, sVOS approach), inspired by [72] as the mask propagation component, which relies on a two-encoder- and one-decoder-structure as illustrated in Figure 4.2. Note that, since MiVOS bases its design on STM [72], we already covered the conceptual fundamentals of how MiVOS works in Chapter 3. However, for completion we include a small but more technical recap with the particular fusion module for the iVOS task, which allows the network to refine the prediction by fusing the masks of a previous round $\mathbf{M}^{r^\star}_t$, with the latest prediction $\mathbf{M}^r_t$.

The backbone feature extractor for both encoders (*i.e.*, the target encoder and the memory encoder) use ResNet50 [126] up to and including the fourth stage (*i.e.*, resnet layer *conv*4_5), which computes 1024 different feature channels. Two separate convolution layers are added at the end of each encoder, as shown in Figure 4.2, to extract key and values embeddings, such that the keys $\mathbf{K} \in \mathbb{R}^{C_k \times HW}$ and the values $\mathbf{V} \in \mathbb{R}^{C_v \times HW}$.

The propagation is a step-wise process, where only one frame is segmented at a time, as illustrated. Note that for the forward direction, we have the following range in steps $s^+_r \in \{1, 2, \ldots, t^+_r - t_r\}$, and vice versa for the backward direction with $s^-_r \in \{1, 2, \ldots, t_r - t^-_r\}$. For ease of notation, we consider both indices s^+_r and s^-_r as an index denoted by s in the subsequent notation, as the process is the same regardless of the propagation direction. Hence for a given frame $\mathbf{I}^r_t$, in each step s we propagate to all remaining frame $\mathbf{I}^r_{t_r - s^-_r} < \mathbf{I}^r_{t_r} < \mathbf{I}^r_{t_r + s^+_r}$ is given to the target encoder (the addition or subtraction depends on the direction of the propagation). In contrast to the target encoder, the memory encoder encodes the images with their respective masks, referred to as memory frames and stores them inside an external memory $\mathcal{M}_s^r$. The memorization of key and value embeddings follows the same congruence condition described in Chapter 3, based on the segmentation's direction (*i.e.*, forward or background) relative to the interacted frame t_r. Note that the memory is re-initialized for each round, for each direction, and that the embeddings stored on memory slot $n = 1$ in $\mathbf{K}^{\mathcal{M}} \in \mathbb{R}^{C_k \times NHW}$ and $\mathbf{V}^{\mathcal{M}} \in \mathbb{R}^{C_v \times NHW}$ are conditioned on the interacted frame, *i.e.*, $\mathbf{I}_{t_r}^r$ and $\mathbf{M}_{t_r}^r$.

A similarity matching between the query key embeddings (current frame to segment for step s) $\mathbf{K}^q$ and the memory key embeddings $\mathbf{K}^{\mathcal{M}}$ as described in Section 3.1.5, to infer a weight matrix $\mathbf{W} \in [0,1]^{NHW \times HW}$, indicating which features from $\mathbf{K}^{\mathcal{M}}$ show the highest similarity to $\mathbf{K}^q$. The attentional weight matrix $\mathbf{W}$ allows us to weight the memory values $\mathbf{V}^{\mathcal{M}}$ only at relevant feature positions. With $\mathbf{V}^q$ encoding which features belong to the foreground or background. The weighted memory values $\mathbf{V}^{\mathcal{M}}$ are then concatenated with the target values $\mathbf{V}^q$, which encode appearance information of the frame $\mathbf{I}_{t_r+s}^r$ to segment at a given step s. The resulting concatenation is given to the decoder to predict a corresponding mask of $\mathbf{M}_{t_r+s}^r$. The decoder is based on [72] and employs several refinement modules [93] as building blocks. At each stage, a refinement module takes the up-sampled feature output from the previous refinement module along with the feature maps of the corresponding target encoder via skip connections.

Since the propagation process is independent of the masks inferred in previous rounds $\mathbf{M}_t^{r^\star}$, the information contained in those masks and, thus, the intent of the previous user interactions may be lost. To prevent this, a learnable fusion component (*Difference-Aware Fusion*) based on [75] , infers a new mask $\mathbf{M}_t^r$. A fusion between $\mathbf{M}_t^r$ and $\mathbf{M}_t^{r^\star}$.

4.5 Interactive Benchmark

4.5.1 *Interactive DAVIS*

To evaluate CiVOS, we utilize the popular interactive 2017 DAVIS benchmark [14, 82, 83, 127] containing high-quality segmentation annotations [83]. Here, a robot agent simulates scribble-based user interactions. Since the process is round-based, the robot annotates at the beginning of each round r a single frame with one or several scribbles. At the end of the round, the evaluation algorithm compares the predicted results $\mathcal{S}_r$ w.r.t. the ground-truth sequence $\mathcal{G}$ to determine the frame index t^* with the worst segmentation and provide corrective scribbles. Formally, our goal is as follows

$$t^* = \underset{t \in \{0,1,\ldots,T\}}{\operatorname{argmin}} \sum_{o \in \mathcal{O}} \operatorname{IoU}(\mathbf{M}_t^o, \mathbf{G}_t^o), \tag{4.4}$$

where $\mathbf{M}_t \in \mathcal{S}_r$ and $\mathbf{G}_t \in \mathcal{G}$. Here $\mathcal{O}$ denotes the set of objects to segment in the video, and $\operatorname{IoU}(\cdot,\cdot)$ is a function that computes the Jaccard index for two binary masks.

The benchmark uses a maximum of 8 interactions rounds while evaluating the submitted iVOS approaches with a time budget (which is set to 30 seconds per round for each object in the sequence). We also use these values in our evaluations reported in Table 4.1 and Table 4.2. Here, the standard metrics [82] are the region similarity $\mathcal{J}$ (*i.e.*, the Jaccard index) and the contour accuracy $\mathcal{F}$. To aggregate the results of several successive interactions, the DAVIS benchmark uses the area under the curve of $\mathcal{J}\&\mathcal{F}$ (denoted as AUC-$\mathcal{J}\&\mathcal{F}$). The required time to predict the masks is used as x-axis. The benchmark further introduces the $\mathcal{J}\&\mathcal{F}@60s$ metric to denote the obtained $\mathcal{J}\&\mathcal{F}$ score at 60 seconds.

For the iVOS setting, an additional metric is introduced by Caelles *et al.* [14]: R-AUC-$\mathcal{J}\&\mathcal{F}$, which considers the time taken to reach a certain level of accuracy.

In this setup, the goal is to promote VOS for real world applications in an video editing perspective. Hence to promote fast and practical methods, the authors introduce the AUC–$\mathcal{J}\&\mathcal{F}$ metrics, which integrate the $\mathcal{J}\&\mathcal{F}$ scores after each round (8 rounds in total), under the area, where the abscise is the time required to achieve these result.

4.5.2 *Metric Adaptation and Click Simulation*

The AUC-$\mathcal{J}\&\mathcal{F}$, as well as the $\mathcal{J}\&\mathcal{F}@60s$ metric, incorporate the processing time to promote fast methods suitable for real-world applications. Thus, the evaluation scores are hardware-dependent and not reproducible on different systems. The metrics favor high-end machines as they are able to achieve high $\mathcal{J}\&\mathcal{F}$ scores faster in the allowed time budget. To enable a fair and reproducible comparison across different systems, we propose R-AUC-$\mathcal{J}\&\mathcal{F}$ that adapts the AUC-$\mathcal{J}\&\mathcal{F}$ metric by incorporating the accumulated interaction rounds instead of the processing time.

As we consider a click-based approach to tackle iVOS, we propose three new click-based interaction generation strategies denoted as a_1, a_2, a_3 for the interactive DAVIS framework. We illustrate each strategy in Figure 4.3. The strategies a_1 and a_2 rely on the generated scribbles of the robot agent to extract click-based inputs. In the case of a_1, in each round, a single click is generated for each object based on all assigned scribbles to the corresponding object. In contrast, a_2 computes a click for each scribble.

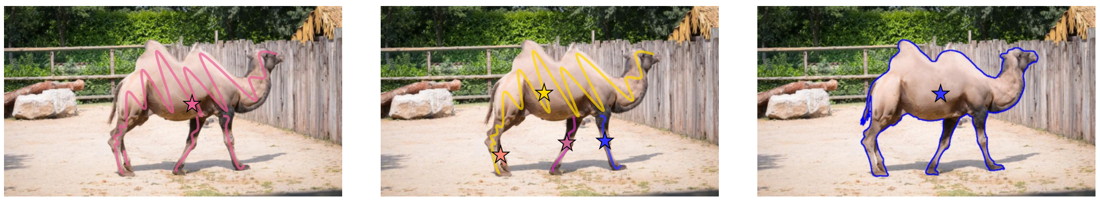

(a) Click Generation a_1 (b) Click Generation a_2 (c) Click Generation a_3

Figure 4.3: Visualization of our click generation strategies. For (a) and (b) a colored star ★ indicates the position of a generated click based on the corresponding colored scribble. For (c) we leverage the mask information directly to compute the central position.

More formally, let $u_t^r = \{(h_i, w_i)\}_{i=1}^N$ denote the set of individual coordinates belonging to provided scribbles at frame index t and round r. We define the central coordinates (h_c, w_c), constrained to lie on the scribble u_t^r, as

$$\begin{aligned} (h_c, w_c) &= \underset{(h_i, w_i) \in u_t^r}{\operatorname{argmin}} \left(\sqrt{\left(\overline{h} - h_i\right)^2 + (\overline{w} - w_i)^2} \right), \quad \text{where} \\ \overline{h} &= \frac{1}{|u_t^r|} \sum_{(h_i, \cdot) \in u_t^r} h_i \quad \text{and} \quad \overline{w} = \frac{1}{|u_t^r|} \sum_{(\cdot, w_i) \in u_t^r} w_i \,. \end{aligned} \tag{4.5}$$

We generate a point based on the a_1 strategy, by treating all individual scribbles belonging to the same object as a single set u_t^r. In contrast, when generating a point through the a_2 strategy, each scribble is treated as an independent set u_t^r, even if they correspond to the same object.

Lastly, we exploit directly the erroneous regions between prediction and ground-truth to emulate novel user interactions (a_3) at the center location of the region. For this strategy, the simulated clicks are located at the center of the corresponding regions.

4.6 Experiments

4.6.1 *Quantitative Results*

To provide a fair comparison of CiVOS against current SOTA methods, we generate click-based interactions from the scribbles provided by the DAVIS robot agent following the a_2 strategy presented in Section 4.5.2. Since the first interaction round only includes one scribble per object, we also utilize only one click per object regardless of the click generating strategy used (*i.e.*, a_1, a_2, a_3).

We compare CiVOS against SOTA methods relying on scribble-based interactions on the interactive 2017 DAVIS benchmark [14, 82, 83, 127] in Table 4.1. Specifically, we compare our approach against a MiVOS [75] variation that accepts click-based interactions. CiVOS achieves competitive results (*i.e.*, R-AUC-$\mathcal{J}\&\mathcal{F}$ = 0.76, AUC-$\mathcal{J}\&\mathcal{F}$ = 0.83) to scribble-based methods despite relying only on click-based interactions.

Table 4.2 displays the R-AUC-$\mathcal{J}\&\mathcal{F}$ results of CiVOS on the DAVIS 2017 validation set by taking into account different click-based interaction strategies (*i.e.*, a_1,a_2,a_3). The number of interactions allowed per object per round depends on the extent of the erroneous regions, *i.e.*, small incorrectly classified areas are not considered during this process. We are able to boost the performance of CiVOS (*i.e.*, R-AUC-$\mathcal{J}\&\mathcal{F}$ = 0.78) by extracting directly the central coordinates of the misclassified regions rather than arbitrarily placed points (points generated from scribbles might be located on the object edge or far from the object center) of the erroneous regions. Also, we do not observe any improvement of the R-AUC-$\mathcal{J}\&\mathcal{F}$ score after a third interaction on the same object within the same round of interaction. Thus, limiting the number of interactions up to a maximum of three interactions per round and per object provides a good trade-off between user workload and mask accuracy.

Methods	Training Interaction	Testing Interaction	R-AUC-$\mathcal{J}\&\mathcal{F}\uparrow$	AUC-$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}\&\mathcal{F}$ @60s $\uparrow$
MANet [74] (CVPR 20)	Scribble	Scribble	0.72	0.79	0.79
ATNet [73] (ECCV 20)	Scribble	Scribble	0.75	0.80	0.80
MiVOS [75] (CVPR 21)	Scribble	Scribble	**0.81**	**0.87**	**0.88**
GIS-RAmap [78] (CVPR 21)	Scribble	Scribble	0.79	0.86	0.87
MiVOS [75] (CVPR 21)	Click	Click	0.70	0.75	0.75
CiVOS (ours)	Click	Click	**0.76**	**0.83**	**0.84**

Table 4.1: Quantitative evaluation on the interactive DAVIS 2017 validation set. We evaluate current SOTA scribble-based iVOS methods with the newly presented R-AUC-$\mathcal{J}\&\mathcal{F}$ metric for reproducible comparison and test CiVOS and MiVOS on click-based interactions (using the a_2 strategy).

Maximal Number of Clicks	1	2	3	4	5	6	7
Interaction Strategy a_1	0.69	–	–	–	–	–	–
Interaction Strategy a_2	0.72	**0.76**	0.76	0.75	0.75	0.75	0.76
Interaction Strategy a_3	0.74	0.77	**0.78**	0.78	0.78	0.78	0.78

Table 4.2: R-AUC-$\mathcal{J}\&\mathcal{F}$ results on the DAVIS 2017 validation set by using different click generating strategies. CiVOS achieves the best result by incorporating at most three clicks for each object per round and by relying on the central coordinates of erroneous regions.

4.7 Discussion

In this chapter, we introduced CiVOS, a modular iVOS framework that reduces annotation effort by leveraging click-based interactions while maintaining competitive segmentation performance compared to scribble-based alternatives. We proposed a complementary hardware-independent metric, R-AUC-$\mathcal{J}\&\mathcal{F}$, to support reproducible evaluation and introduced several strategies to convert scribble annotations into equivalent click-based inputs. To foster future research in this area, we make our code publicly available[1].

These results provide an initial answer to RQ1, where we assess the feasibility of using clicks as an effective form of interaction in VOS systems. Specifically, we established that such sparse inputs can reduce user effort without significantly compromising segmentation quality (in the context of pre-recorded, and short video sequences). Despite these results, several challenges remain before we can establish a definitive answer to RQ1, as reducing user effort is not limited to minimizing interaction time or improving intuitiveness. For instance, it also involves decreasing the cognitive load associated with monitoring and reviewing the segmentation process, which we will explore in Chapter 6.

Note that the current iVOS setting, which assumes access to the full video sequence, works well for short, pre-recorded videos where users can interact at optimal frames and freely rewind. However, this paradigm is ill-suited for long-term or streaming video sequences, where delayed feedback can allow errors to propagate, increasing the cost and complexity of reviewing. While manageable in short clips, this effort becomes a significant bottleneck as sequence length increases.

To summarize, while this chapter demonstrates that click-based interactions are a viable and efficient alternative to scribbles for iVOS, the current approach remains limited to pre-recorded, short video sequences. It assumes full access to the video in advance and does not account for the reviewing effort. These constraints highlight the need to go beyond the traditional iVOS setting and explore how video object segmentation systems can operate in unconstrained environments, *i.e.*, where interaction opportunities are limited and no review is possible. These limitations motivate our Chapter 6, where we investigate how to make interaction more efficient and proactive, enabling segmentation systems to decide when and how to involve the user during ongoing video analysis.

1 https://github.com/Vujas-Eteph/CiVOS

5 DIVERSITY DRIVEN MEMORY MANAGEMENT FOR VIDEO OBJECT SEGMENTATION

This chapter addresses the scalability challenge posed in RQ2, where we aim to extend VOS methods to operate effectively on unconstrained video sequences. Unlike the previous chapter, we omit user interactions and focus solely on improving the segmentation aspect. This allows us to work within the sVOS setting, where no user input is provided after initialization.

As a proof of concept, we adapt and extend several state-of-the-art sVOS pipelines to handle unconstrained scenarios. Central to our contribution is **READMem** (Robust Embedding Association for a Diverse Memory), a training-free, plug-and-play module designed to enhance the memory update strategy for SOTA sVOS methods. Rather than following standard heuristics, READMem estimates the diversity of memory samples and updates the external memory only when a new frame adds ideally novel and non-redundant information. We demonstrate that READMem successfully extends all sVOS baselines to operate on unconstrained videos by evaluating them on LV1 [51] and VOTS2023 [92] while not hindering performance on short-term videos like DAVIS [26].

This chapter is based on our *BMVC 2023* paper [128], and is structured as follows: We outline the motivation in Section 5.1, define our problem in Section 5.2 and provide a small overview of related work in Section 5.3. We describe our READMem module in Section 5.4, present our quantitative and qualitative results in Section 5.5 and 5.6. Finally, we discuss the limitations of our approach in Section 5.7 and provide concluding remarks in Section 5.8.

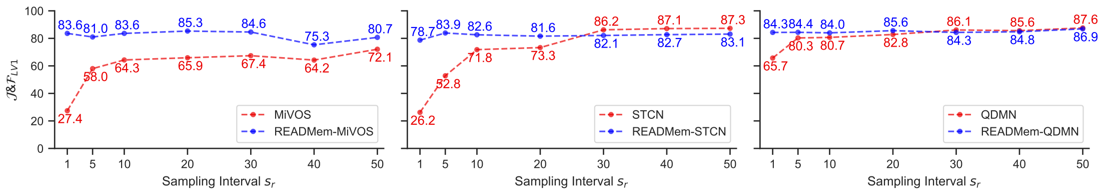
5.1 Motivation

When this research was initiated, most sVOS approaches were designed for short, prerecorded video clips, with limited attention given to their applicability in long-term or streaming scenarios. Current sVOS methods [46–48, 52, 54, 55, 129] predominantly utilized the space-time memory network design [42], which stores deep representations of previous frames and their estimated masks in an external memory for the segmentation of a query image. Although some attention has recently been given to the memory update strategy [52, 55, 129], most methods rely on a simple frame aggregation approach, which inserts every t-th frame into an ever-expanding memory. This approach works well for short sequences but fails on longer videos due to the saturation of the GPU memory and increasing computational demands.

As an example, let us consider a practical application, where we leverage the popular sVOS framework MiVOS [47] to track on pixel level an object of interest in a given video (with a resolution of 480 by 960 pixels) using an Nvidia GeForce GTX 1080 Ti (which we use in our experiments). The GPU offers about 11 GiB in VRAM, and MiVOS needs around 0.9 GiB of VRAM without taking into account the external memory. When saving the combined memory key and value in the external memory of MiVOS, the network adds approximately 18 MiB to the GPU's memory footprint. This memory footprint grows linearly under the following space complexity $\mathrm{O}(N \cdot H \cdot W \cdot (C_k + C_v))$, where N denotes the memory size, H and W the strided resolution of the image after passing through an encoder, and C_k with C_v denotes the channel dimension of the key and value embeddings. Hence, we can hypothetically store around 560 frames representations into the GPU in addition to the base network MiVOS. Note that, in our thought experiment we only considered the allocated GPU memory (effectively used by the network) to estimate the number of frames we can save before reaching an *out-of-memory* (OOM) error in the GPU. However, in practice, the actual bottleneck often stems from the reserved GPU memory. This is influenced by factors such as the framework's internal pre-allocation strategies for efficiency, memory fragmentation, concurrent workloads from other processes or networks, and the specific CUDA runtime version. This makes it difficult to accurately estimate its growth compared to the allocated memory. Hence, our original estimates of 560 frames presents mainly a theoretical bound given the GeForce GTX 1080 Ti. Moreover in our experimental evaluations we could only store up to 140 frames. Now considering that the long-term VOS datasets [90] for the long-term setting present an average sequence length of 2470 frames for LV1 [51], 2073 for VOTS 2023 [92], and 574 for LVOS [91], the theoretical bound is barely sufficient to allow us to evaluate MiVOS on LVOS if we would like to aggregate every frame representation in the external memory.

Moreover, the specificity of the sVOS approach, *i.e.*, space-time operation (see Subsection 3.1.5), induces a time complexity that scales based on the memory size N, leading to an overall complexity[1] of $\mathrm{O}(N \cdot H^2 \cdot W^2 \cdot (C_k + C_v))$, which stems from the cross-attention in Equation (3.9) (*i.e.*, $\mathrm{O}(N \cdot H^2 \cdot W^2 \cdot C_k)$) and the value aggregation in Equation (3.11) (*i.e.*, $\mathrm{O}(N \cdot H^2 \cdot W^2 \cdot C_v)$). We do not include the base network in this analysis, as its time complexity is uncorrelated to the memory size N. As the framework continuously accumulates memory embeddings, N grows linearly. Through the continuous memory aggregation scheme, this dependence on N causes the method to slow down proportionally to the duration it tracks an object (refer to Table 5.8 in Subsection 5.7). Beyond the continuous increase in memory footprint and the resulting slowdown with growing memory, Cheng *et al.* [47] show that, past a certain point, the memory contains redundant information. This redundancy adds noise and dilutes the significance of more pertinent information.

1 We exclude the column-wise softmax operation from the final Big O notation because its complexity is asymptotically negligible compared to the dominant terms involving the key C_k and value C_v dimensions.



(a) MiVOS *vs.* READMem-MiVOS

(b) STCN *vs.* READMem-STCN

(c) QDMN *vs.* READMem-QDMN

Figure 5.1: Performance comparison of sVOS baselines (MiVOS [47], STCN [48], QDMN [52]) with and without the READMem extension on the LV1 [51] dataset (composed of long videos) when varying the sampling interval s_r.

Hence, this continuous frame aggregation strategy neither considers redundancies nor novel information while adding new embeddings to the memory. A naive approach to better deal with long videos is to increase the sampling interval s_r, which dictates how frequently the memory's state is updated by saving only every t-th frame representation. As shown in Figure 5.1, increasing the sampling interval SOTA methods [42, 47, 48] leads to improved performance. Due to a higher sampling interval, the memory stores embeddings of frames that are further apart in time from each other. These embeddings are more likely to reflect appearance variations resulting in a more diverse set of embeddings in the memory, contributing to better predictions. However, a high sampling interval may lead to the omission of potentially useful frames [51, 55], which is particularly detrimental for short sequences. Moreover, as noted by [55], a higher sampling interval leads to unstable performance.

To address the aforementioned points, we propose a simple yet effective extension in the form of a module named *READMem* for updating the memory. Drawing inspiration from [130] in the visual object tracking field, our approach focuses on increasing the diversity of the embeddings stored in the memory rather than simply aggregating every t-th frame. This selective approach allows us to save only key embeddings in the memory, thus alleviating the memory saturation faced by previous methods on long videos and even unconstrained video lengths.

5.2 Problem Definition

To briefly restate, sVOS methods rely on an initial user cue $\mathbf{U}_{t=0}$ to specify which object to track throughout a video $\mathcal{V} = \{\mathbf{I}_t \mid t \in \{0, 1, \ldots, T\}\}$, where $\mathbf{I}_t$ denotes an RGB image. This cue is typically provided as a fully annotated mask $\mathbf{M}_{t=0} \in \{0,1\}^{H_R \times W_R}$ (the ground-truth $\mathbf{G}_{t=0}$) of the first frame $\mathbf{I}_{t=0} \in \mathbb{R}^{3 \times H_R \times W_R}$. The method then segments the subsequent frames in $\mathcal{V}$ in a step-wise process, resulting in a segmented sequence, denotes as $\mathcal{S} = \{\mathbf{M}_t \mid t \in \{1, 2, \ldots, T\}\}$. A more detailed description of the general sVOS framework is provided in Chapter 3.

In our use case, we target the ever-growing memory issue in STM-based approaches by considering their application to unconstrained video sequences $\mathcal{V}$ (*i.e.*, unbounded length, unpredictable object variations), where $T \to \infty$. Indeed, as $N \propto T$ this leads to a memory size $N \to \infty$, which is impractical in unconstrained scenarios (*e.g.*, streaming scenarios, unbounded sequence length). Hence, in this chapter, we focus on limiting the memory growth in SOTA methods $\mathcal{M}_t^e$, where e acts as a placeholder for key k or value v embedding, while simultaneously preserving the memory's representation capabilities. To approximate unconstrained sequences in our evaluations, we will rely on long videos. To this end, we fix the size of the memory to N slots, such that $|\mathcal{M}_t^e| = N$ and introduce a diversity criterion to retain only embeddings e that positively enhance the memory's diversity. We can formulate the memory selection process as the following optimization problem

$$\mathcal{M}_t^e = \underset{\substack{\mathcal{M}_t^e \subseteq \mathcal{M}_{t-1}^e \cup \{e_t\}, \\ |\mathcal{M}_t^e| = N}}{\operatorname{argmax}} \operatorname{div}(\mathcal{M}_t^e), \tag{5.1}$$

where $\operatorname{div}(\cdot)$ denotes a diversity measure over the selected embeddings, which we will further elaborate in Subsection 5.4.2. Conceptually, the memory from the previous step $\mathcal{M}_{t-1}^e$, which has a fixed size of N, is combined with the new embedding e_t to form a temporary candidate pool of $N + 1$ embeddings. The optimization problem then selects the top N embeddings from the $N + 1$ candidate pool to the ones that yield the highest diversity score.

However, solving the global maximization problem in Equation (5.1) is computationally infeasible, as it would require storing and evaluating all past embeddings up to time t, which contradicts our goal of maintaining a bounded memory. Therefore, we adopt a greedy strategy, where at each time step t, upon receiving a new embedding e_t, we decide whether to add or replace an existing embedding to optimize Equation (5.1), based on its contribution to the overall diversity.

5.3 Related Work

Online fine-tuning approaches [18–22, 24] adapt the network parameters on-the-fly based on an initial mask indicating the object of interest, but suffer from slow inference and poor generalization [8].

In contrast, *propagation-based methods* [26–30] utilize strong priors offered by previous (adjacent) frames to propagate the mask, allowing them to better deal with fast appearance changes. However, they are prone to error accumulation and lack long-term context for identifying objects after occlusion [8].

Matching-based methods [35–40, 42, 47, 48, 55] encode all frame representations in the same feature space to facilitate the matching process during cross-attention. Most SOTA

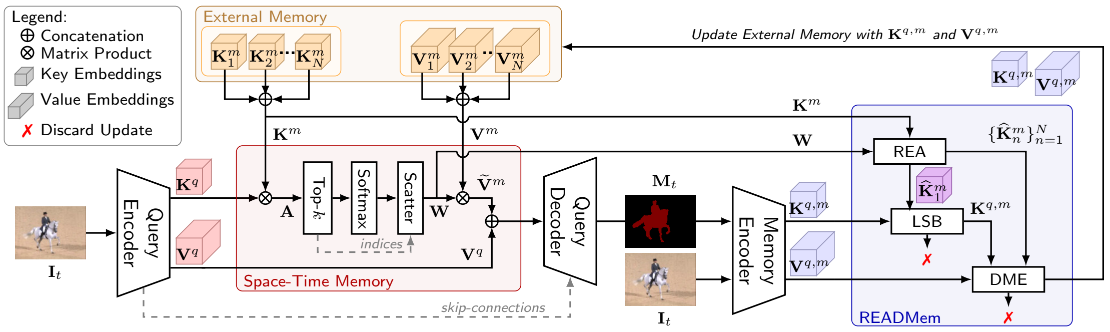


Figure 5.2: Overview of READMem-MiVOS (MiVOS [47] using READMem) after initialization. We omit the temporary memory for simplicity.

approaches follow the design proposed by Oh *et al.* [42] (Space-Time-Memory (STM)), where they encode features of the initial frame along with previously processed frames (intermediate frames). However, while STM-based methods [42, 47–49, 55] methods display incredible performance on short-term benchmarks [83, 84], they are hampered in real-world applications by their ever-expanding memory.

To address memory limitations, Li *et al.* [50] proposes a compact global module to summarize object segmentation information within a fixed memory size. Similarly, Liang *et al.* [51] apply a weighted average on the extracted features to merge new features with existing ones into the memory and discard obsolete features through a *least frequently used* (LFU) approach. Liu *et al.* [52] explore the use of a quality-aware-module (QAM) [53] to assess the quality of the predicted mask on-the-fly before integrating it in the memory. Li *et al.* [54] use a spatio-temporal aggregation module to update a fixed-sized memory bank. Cheng and Schwing [55] follow the Atkinson-Shiffrin [56] model for their sVOS pipeline (XMem), which comprises: a *sensory memory* [57] that learns an appearance model for every incoming frame, a *working memory* based on STCN [48], and a *long-term memory* that consolidates the working memory embeddings in a compact representation.

5.4 Framework

This section presents our READMem module – an extension for space-time memory-based sVOS pipelines [42, 47, 48]. Figure 5.2 illustrates our READMem module embedded in MiVOS [47].

5.4.1 *Backbone*

Since READMem is built upon the popular space-time memory network paradigm [42, 47, 48, 52], we provide a brief description of the corresponding sVOS backbone elements, more specifically for MiVOS [47]. We provide a more detailed overview of the fundamental concepts in Chapter 3, but for completeness we present a brief description of MiVOS's [47] key components.

(1) A query encoder (an extension of ResNet50 [126]) that encodes a query frame $\mathbf{I}_t \in \mathbb{R}^{3\times H_R\times W_R}$ seen at time step t into a query key feature $\mathbf{K}^q \in \mathbb{R}^{C_k\times HW}$ and a query value feature $\mathbf{V}^q \in \mathbb{R}^{C_v\times HW}$. With C_k and C_v we denote the number of feature channels, while H and W indicate the spatial image dimensions using a stride of 16, while H_R and W_R denote the initial image resolution. Here, the exponent q denotes the embeddings conditioned on the query image $\mathbf{I}_t$, which we are currently processing through the query encoder.

(2) A memory encoder that extracts latent feature representations from a query frame $\mathbf{I}_t$ and its corresponding segmentation mask $\mathbf{M}_t \in \{0,1\}^{H_R\times W_R}$ into a pair of a memory key $\mathbf{K}^m \in \mathbb{R}^{C_k\times HW}$ and a memory value $\mathbf{V}^m \in \mathbb{R}^{C_v\times HW}$. Here, the exponent m denotes the embeddings conditioned on the query image $\mathbf{I}_t$ and $\mathbf{M}_t$, we are processing through the memory encoder.

(3) An external memory that maintains a set of memory keys $\mathcal{M}^k = \left\{\mathbf{K}^m_n\right\}_{n=1}^N$ and a set of memory values $\mathcal{M}^v = \left\{\mathbf{V}^m_n\right\}_{n=1}^N$. The variable N denotes the total number of memory slots, while n represents the index of a slot. In addition, some methods [42, 47, 52] use a temporary memory that only contains the latent representation (*i.e.*, memory key and value features) of the previous adjacent frame. Note that our READMem extension targets this specific aspect in each STM-based architecture.

(4) The *Space-Time Memory* (STM) operation (*i.e.*, cross attention) to determine relevant memory information. It matches the channels of the query key $\mathbf{K}^q$ with all channels of every memory key $\mathbf{K}^{\mathcal{M}} \in \mathbb{R}^{C_k\times NHW}$, which is the concatenated version of $\mathcal{M}^k$, to infer an affinity matrix $\mathbf{A} \in \mathbb{R}^{NHW\times HW}$. Concretely, the channel level similarities of the query $\mathbf{K}^q$ and every memory key in $\mathbf{K}^{\mathcal{M}}$ are computed through

$$\mathbf{A} = (\mathbf{K}^{\mathcal{M}})^{\mathsf{T}}\mathbf{K}^q. \tag{5.2}$$

Note that MiVOS [47] introduced a noise reduction, which filter's out low affinities in the affinity matrix $\mathbf{A}$ through a top-k filter applied along the columns (memory dimension). This produces a sparse matrix (filtered) $\mathbf{F} \in \mathbb{R}^{NHW\times HW}$ by

$$\mathbf{F}(i,j) = \begin{cases} \mathbf{A}(i,j) & \text{, if } \mathbf{A}(i,j) \in \underset{\mathcal{A}_j\subset\{a|a\in\mathbf{A}(\bullet,j)\},|\mathcal{A}_j|=k}{\operatorname{argmax}} \left(\sum_{a\in\mathcal{A}_j} a\right), \\ 0 & \text{, otherwise} \end{cases} \tag{5.3}$$

where i and j denote the row and column indices respectively, and $\bullet$ refers to either the complete row or column of a matrix. Based on the sparse affinity matrix $\mathbf{F}$, the method computes soft-weights $\mathbf{W} \in \mathbb{R}^{NHW\times HW}$ through a softmax applied along the memory dimension (*i.e.*, column axis in this case) as in Equation (3.10).

All the channels of every memory value feature $\mathbf{V}_n^m \in \mathcal{M}^v$ are weighted with $\mathbf{W}$ to compute a pseudo memory feature representation $\widetilde{\mathbf{V}}^m \in \mathbb{R}^{C_v \times HW}$ through $\widetilde{\mathbf{V}}^m = \mathbf{V}^{\mathcal{M}}\mathbf{W}$, where $\mathbf{V}^{\mathcal{M}} \in \mathbb{R}^{C_k \times NHW}$ is the concatenated form of the memory values $\mathbf{V}_n^m$ along the columns.

(5) Lastly, a decoder inspired by [42], that predicts a segmentation mask $\mathbf{M}_t$ for the query frame $\mathbf{I}_t$ using refinement modules [93], skip-connections and bilinear upsampling.

5.4.2 *READMem*

Diversification of Memory Embeddings (DME): As mentioned, contemporary methods insert the embeddings of every t-th frame into the memory, along with the corresponding mask. This leads to an inevitable memory overload when accumulating embeddings of unconstrained sequences. In contrast, READMem proposes an alternative frame insertion scheme, maximizing the diversity of latent representations stored in the memory by only adding frames that enhance information. This allows us to limit the number of memory slots without degrading the performance and avoid GPU memory saturation when processing long videos.

In pursuit of enhancing the diversity of the memory embeddings $\mathcal{M}^k = \{\mathbf{K}_n^m\}_{n=1}^N$, we require a mean to quantify this diversity. If we conceptually consider the embeddings to form a parallelotope, where the key embeddings $\{\mathbf{K}_n^m\}_{n=1}^N$ act as edges, we can quantify the diversity by calculating the volume of this geometric object. A conventional approach to estimate the volume involves concatenating the keys $\text{vec}(\mathbf{K}_n^m) \in \mathbb{R}^{C_k HW \times 1}$ (we take a flattened representation, where $\text{vec}(\cdot)$ flattens $\mathbf{K}_n^m \in \mathbb{R}^{C_k \times HW}$ following a column-major ordering) into a matrix $\mathbf{X} \in \mathbb{R}^{C_k HW \times N}$ and inferring the determinant of $\mathbf{X}$, where $\mathbf{X} = [\text{vec}(\mathbf{K}_1^m) \;\; \text{vec}(\mathbf{K}_2^m) \;\; \cdots \;\; \text{vec}(\mathbf{K}_n^m)]$. However, since $N \ll C_k HW$, $\mathbf{X}$ is a non-square matrix, which prevents the computation of the determinant. To circumvent this issue, we leverage the Gram matrix $\mathbf{\Gamma} \in \mathbb{R}^{N \times N}$, computed through

$$\mathbf{\Gamma} = \mathbf{X}^\mathsf{T}\mathbf{X}. \tag{5.4}$$

As demonstrated by Ernest Vinberg [131] (pages 195-196) the determinant[2] of the Gram matrix (*i.e.*, Gramian) allows us to compute the square volume (vol) of the parallelotope, such that $(\text{vol } P(\{\mathbf{K}_n^m\}_{n=1}^N))^2 = \det(\mathbf{\Gamma})$. Therefore, we can quantify the diversity of our memory with $\det(\mathbf{\Gamma})$. We use the set of memory keys $\mathcal{M}^k = \{\mathbf{K}_n^m\}_{n=1}^N$ to compute the

2 In the paper, we compute the determinant of $\mathbf{\Gamma}$ by performing an LU decomposition [132] and then multiplying the diagonal elements of the upper triangular factor $\mathbf{U}$. However, since the $\mathbf{\Gamma}$ is positive-definite, we can leverage the Cholesky decomposition [133] instead, known to be more efficient for such matrices.

diversity of the memory, as they encode visual semantics that are robust to appearance variation [42]. Hence, we construct $\mathbf{\Gamma}$ by

$$\mathbf{\Gamma}(\mathcal{M}^k) = \begin{pmatrix} \langle \mathrm{vec}(\mathbf{K}^m_{n=1}), \mathrm{vec}(\mathbf{K}^m_{n=1}) \rangle & \cdots & \langle \mathrm{vec}(\mathbf{K}^m_{n=1}), \mathrm{vec}(\mathbf{K}^m_{n=N}) \rangle \\ \vdots & \ddots & \vdots \\ \langle \mathrm{vec}(\mathbf{K}^m_{n=N}), \mathrm{vec}(\mathbf{K}^m_{n=1}) \rangle & \cdots & \langle \mathrm{vec}(\mathbf{K}^m_{n=N}), \mathrm{vec}(\mathbf{K}^m_{n=N}) \rangle \end{pmatrix}. \tag{5.5}$$

The size of the memory N is bounded by the dimension of the feature space since otherwise $\det(\mathbf{\Gamma}) = 0$ [134]. However, this is not a relevant limitation since in practice, N is several magnitudes smaller than the dimension of the feature space $C_k \times HW$ (*i.e.*, $N \ll C_k \times HW$).

To increase the diversity of the embeddings stored in the memory, we want to maximize the absolute Gramian $|\det(\mathbf{\Gamma}(\mathcal{M}^k))|$. As the annotated frame $\mathbf{I}_{t=0}$ provides accurate and controlled segmentation information [42, 52], the corresponding memory key $\mathbf{K}^m_{n=1}$ and value $\mathbf{V}^m_{n=1}$ are exempt from the following update strategy:

(1) For each memory slot $n \in \{2, \ldots, N\}$, we substitute the respective memory key $\mathbf{K}^m_n$ with the query-memory key $\mathbf{K}^{q,m}$ (memory key of the current query frame $\mathbf{I}^q$ and mask $\mathbf{M}^q$) and construct a temporary Gram matrix $\mathbf{\Gamma}^t_n$. Using the temporary matrices $\mathbf{\Gamma}^t_n$, we build a set $\mathcal{E}^t$ containing the absolute values of the determinants, such that we have $\mathcal{E}^t = \{|\det(\mathbf{\Gamma}^t_n)|\}^N_{n=2}$.

(2) The memory is updated based on whether the highest value in $\mathcal{E}^t$ surpasses the absolute value of the current Gramian $|\det(\mathbf{\Gamma})|$. If this condition is met, then the corresponding memory slots n (position of the highest value in $\mathcal{E}^t$) is replaced with the query-memory key $\mathbf{K}^{q,m}$ and value $\mathbf{V}^{q,m}$ embeddings.

Robust Embeddings Association (**REA**): Given that we compute an inner product between two embeddings (enforced by the Gram matrix), we capture the global similarity of two frames in a single score. However, as two frames, $\mathbf{I}_t$ and $\mathbf{I}_{t+\Delta t}$ are further apart in time, it becomes more likely for the object of interest to experience in-between: motion, ego-motion, or size variations. Consequently, a channel-wise embedding may encode different information (specifically foreground *vs.* background) for the same image region in frame $\mathbf{I}_t$ and frame $\mathbf{I}_{t+\Delta t}$ due to the spatial disparity in the object's location. As a result, the resulting similarity between the two frames is inherently low when comparing their embeddings. To address this issue and dampen positional variation without relying on a motion model or fixed-sized search region (which would introduce additional hyper-parameters), our proposal utilizes transition matrices $\{\mathbf{T}_n\}^N_{n=1}$. Wherein the core idea is to leverage the cross-attention already at hand in the memory-based networks (*i.e.*, $\mathbf{A}$ and $\mathbf{W}$) to identify the best channel-wise mapping between two embeddings. Specifically, through the transition matrix $\mathbf{T}_n \in \mathbb{R}^{HW \times HW}$, we project the n-th memory embedding $\mathbf{K}^m_n \in \mathbb{R}^{C_k \times HW}$ from its original Frame of Reference (FOR) to the query's FOR (see Figure 5.3) by

$$\widehat{\mathbf{K}}^m_n = \mathbf{K}^m_n \mathbf{T}_n, \tag{5.6}$$

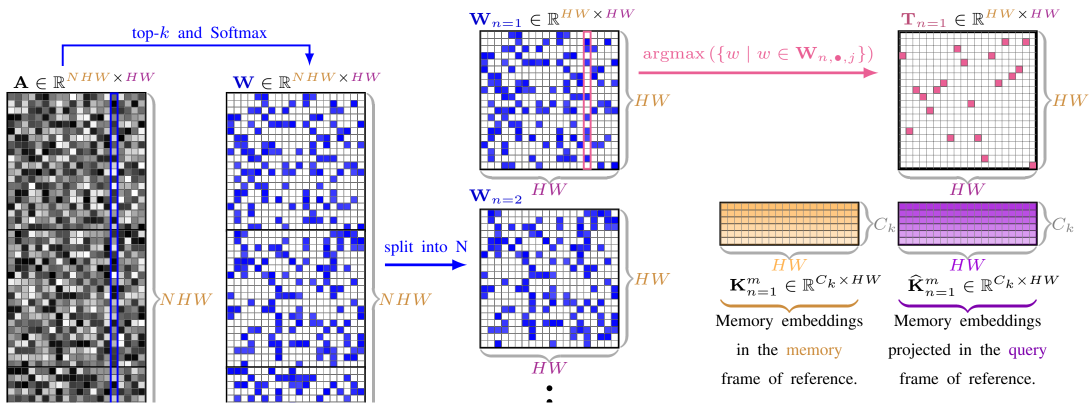

top-k and Softmax
$\mathbf{A} \in \mathbb{R}^{NHW \times HW}$
$\mathbf{W} \in \mathbb{R}^{NHW \times HW}$
NHW
NHW
split into N
$\mathbf{W}_{n=1} \in \mathbb{R}^{HW \times HW}$
HW
$\mathbf{W}_{n=2}$
HW
HW
argmax $(\{w \mid w \in \mathbf{W}_{n,\bullet,j}\})$
$\mathbf{T}_{n=1} \in \mathbb{R}^{HW \times HW}$
HW
HW
C_k
C_k
$\mathbf{K}^m_{n=1} \in \mathbb{R}^{C_k \times HW}$
Memory embeddings in the memory frame of reference.
$\widehat{\mathbf{K}}^m_{n=1} \in \mathbb{R}^{C_k \times HW}$
Memory embeddings projected in the query frame of reference.

Figure 5.3: Inference of a transition matrix $\mathbf{T}_n$ and the corresponding pseudo-memory embeddings $\widehat{\mathbf{K}}^m_n$ for robust embeddings association (REA).

where $\widehat{\mathbf{K}}^m_n \in \mathbb{R}^{C_k \times HW}$ denotes the pseudo-embedding of the embedding $\mathbf{K}^m_n$, but expressed in the query's *frame of reference*. This approach effectively compensates for the target's spatial disparity between two distant frames (refer to Table 5.3 and 5.4).

Thus, we compute the similarity w.r.t. $\mathbf{K}^{q,m}$ not with the memory keys $\{\mathbf{K}^m_n\}^N_{n=1}$ but with the pseudo memory keys $\{\widehat{\mathbf{K}}^m_n\}^N_{n=1}$. Figure 5.3 depicts the operation that maps key embeddings from the memory's FOR to the query's FOR through the transition matrix $\mathbf{T}_n$. We use $\mathbf{W}$, the filtered and adapted version of $\mathbf{A}$ (see Equation (3.10)), to only consider strong affinities between the point-wise embeddings (channels) of the query and memory key which are more likely to encode similar information. This ensures that the corresponding mapping matrix $\mathbf{T}_n$ is constructed based on relevant channels (refer to Table 5.4). Although $\mathbf{W}$ entails strong similarities, using $\mathbf{W}_n$ as transition matrix $\mathbf{T}_n$ potentially leads to the aggregation of multiple memory channels onto a single pseudo memory channel – which is certain to degrade the similarity. Hence, to avoid the summation, we need to map at most one element from the memory FOR to the query FOR, *i.e.*, filtering $\mathbf{W}_n$. Generating a bijective mapping between the memory key $\mathbf{K}^m_n$ and the corresponding pseudo query key $\widehat{\mathbf{K}}^m_n$ would constraint the foreground-background ratio of $\widehat{\mathbf{K}}^m_n$ based on $\mathbf{K}^m_n$. However, it is unlikely that the area taken by the object in the image is unchanged from one frame to the other. To avoid this foreground-background restriction, we should allow point-wise embeddings from the memory key $\mathbf{K}^m_n$ to be re-used for the creation of the pseudo memory key $\widehat{\mathbf{K}}^m_n$, as long as they are sufficiently similar. Thus, an appropriate function that validates the aforementioned criteria is argmax applied along the columns of $\mathbf{W}$ to maximize the re-usability of point-wise embeddings when producing the pseudo memory key $\widehat{\mathbf{K}}^m_n$ on the query FOR. We obtain $\mathbf{T}_n \in \mathbb{R}^{HW \times HW}$ by dividing $\mathbf{W} \in \mathbb{R}^{NHW \times HW}$ into N-square matrices, such that $\mathbf{W}_n \in \mathbb{R}^{HW \times HW}$, and by applying argmax along the columns by

$$\mathbf{T}_n(i,j) = \begin{cases} 1 & \text{, if } i \in \underset{i \in \{1,\dots,HW\}}{\operatorname{argmax}} \left(\{w \mid w \in \mathbf{W}_n(i,j)\}\right) \\ 0 & \text{, otherwise} \end{cases} . \tag{5.7}$$

Lower Similarity Bound of Memory Embeddings (LSB): To ensure the presence of the foreground object in a candidate frame, we set a lower bound on similarity $l_{bs} = 0.5$. We validate object presence in the query image $\mathbf{I}_t$ by computing a similarity score between its key embeddings $\mathbf{K}^q$ and the annotated frame's key embeddings $\mathbf{K}^m_1$, such that we consider storing $\mathbf{K}^q$ only if $\langle \mathbf{K}^q, \widehat{\mathbf{K}}^m_{n=1} \rangle > l_{bs}$. Otherwise, the memory would simply integrate the most diverse set of frames seen in the segmentation process, however, if a subset includes multiple views of the object of interest, the remaining subset of the memory would likely be filled with embeddings where the object is not present.

Initialization: We integrate every t-th frame (that satisfies the lower similarity bound) into the memory – until the memory is completely filled. Afterward, the method only integrates embeddings of frames that enhance the diversity of the memory.

5.5 Quantitative Experiments

In our experiments, we use the Long-time Video [51] (LV1) and VOTS2023 [92] datasets to demonstrate the scalability and versatility of our approach for long-term sequences. We include the D17 dataset [83] in our evaluations to encompass scenarios with shorter sequences. Importantly, we want to clarify that our method is originally designed for managing the memory of sVOS task and, as such, is not modifying the underlying architecture of the sVOS baselines [47, 48, 52], which are not tailored towards handling specific challenges found only in VOT datasets (*e.g.*, small object-to-image ratio, presence of numerous distractors). We perform all experiments on an Nvidia GeForce GTX 1080 Ti.

5.5.1 *Quantitative Results*

Table 5.1 presents the quantitative results of recent sVOS methods (MiVOS [47], STCN [48] and QDMN [52]) with our READMem extension along with (at the time) the state-of-the-art sVOS method XMem [55] on the LV1 [51] and the D17 [83] datasets. We use the $\mathcal{J}$ (intersection over union) and $\mathcal{F}$ (contour accuracy) metrics (higher is better) introduced by [83], with both metrics are averaged into the $\mathcal{J}\&\mathcal{F}$ to quantify the performance on the LV1 and D17 benchmarks.

To ensure a fair comparison, we keep the default settings for XMem [55] and use the same sampling interval (*i.e.*, $s_r = 10$) for all other methods (in contrast to previous works [51, 55] that use dataset-specific sampling intervals). In our experiments, we follow the evaluation of [52] and limit the number of memory slots of the baselines and

Method	LV1 [51] (Long Sequences)			D17 [83] (Short Sequences)		
	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Sampling Interval $s_r = 10$						
XMem† [55] (ECCV 22)	**89.8**	**88.0**	**91.6**	85.5	82.3	88.6
XMem [55] (ECCV 22)	83.6	81.8	85.4	**86.7**	83.0	**90.1**
MiVOS [47] (CVPR 21)	64.3	63.6	65.0	84.3	81.4	87.1
READMem-MiVOS (ours)	$83.6^{\uparrow 19.3}$	$82.1^{\uparrow 18.5}$	$85.1^{\uparrow 20.1}$	84.3	81.4	87.1
READMem-MiVOS* (ours)	$86.0^{\uparrow 21.7}$	$84.6^{\uparrow 21.0}$	$87.4^{\uparrow 22.4}$	$84.6^{\uparrow 0.3}$	$81.8^{\uparrow 0.4}$	$87.3^{\uparrow 0.2}$
STCN† [48] (NeurIPS 21)	71.8	69.4	74.3	83.7	80.7	86.6
READMem-STCN† (ours)	$82.6^{\uparrow 10.8}$	$81.5^{\uparrow 12.1}$	$83.7^{\uparrow 9.4}$	83.7	80.7	86.6
READMem-STCN†* (ours)	$81.8^{\uparrow 10.0}$	$80.8^{\uparrow 11.4}$	$82.8^{\uparrow 8.5}$	83.7	80.7	86.6
QDMN* [52] (ECCV 22)	80.7	77.8	83.6	86.0	83.2	88.8
READMem-QDMN* (ours)	$84.0^{\uparrow 3.3}$	$81.3^{\uparrow 3.5}$	$86.7^{\uparrow 3.1}$	$86.1^{\uparrow 0.1}$	**83.3**$^{\uparrow 0.1}$	$88.9^{\uparrow 0.1}$
Sampling Interval $s_r = 1$						
XMem† [55] (ECCV 22)	61.5	60.4	68.4	84.1	81.1	**90.8**
XMem [55] (ECCV 22)	79.3	77.3	84.2	83.9	80.6	87.1
MiVOS [47] (CVPR 21)	27.4	27.4	27.4	84.3	81.3	87.3
READMem-MiVOS (ours)	$83.6^{\uparrow 56.2}$	**82.3**$^{\uparrow 54.9}$	$85.0^{\uparrow 57.6}$	84.3	$81.4^{\uparrow 0.1}$	$87.1^{\downarrow 0.2}$
STCN [48]† (NeurIPS 21)	26.2	22.9	29.4	83.2	79.9	86.5
READMem-STCN† (ours)	$80.8^{\uparrow 54.6}$	$78.4^{\uparrow 55.5}$	$83.2^{\uparrow 53.8}$	$83.8^{\uparrow 0.6}$	$80.4^{\uparrow 0.5}$	$87.2^{\uparrow 0.7}$
QDMN [52] (ECCV 22)	65.7	63.5	67.9	85.1	82.2	88.0
READMem-QDMN (ours)	**84.3**$^{\uparrow 18.6}$	$81.9^{\uparrow 18.4}$	**86.7**$^{\uparrow 18.8}$	**85.3**$^{\uparrow 0.2}$	**82.4**$^{\uparrow 0.2}$	$88.1^{\uparrow 0.1}$

Table 5.1: Quantitative evaluation of sVOS methods [47, 48, 52, 55] with and without READMem on the LV1 [51] and D17 [83] datasets. The symbol † denotes no pre-training on BL30K [47], while * indicates the use of a flexible sampling interval as in QDMN [52].

the corresponding READMem counterparts to 20. Note that this value differs from the memory slot limit (*i.e.*, 50) used by [51, 55]. In contrast to [51, 55], we do not adapt the sampling interval to the sequence length when dealing with long videos as we argue that the sampling interval should not be correlated to the video's length and is not a standard practice when dealing with short videos (D17 [83] and Y-VOS [84]). Instead, we opt for a first-in-first-out (*i.e.*, FIFO) replacement strategy when updating the memory [16] of the mentioned baselines.

Our results in Table 5.1 demonstrate that READMem improves long-term performance while preserving good short-term capabilities. Moreover, by setting $s_r = 1$ we minimize the likelihood of omitting key frames and enhance the stability of the method [55]. As a result, we attain competitive results against the state-of-the-art method (*i.e.*, XMem [55]). We do not employ READMem on XMem [55], as a long-term memory is already present, consolidating the embeddings of the working memory [48] and updated by a *least-frequently-used* (*LFU*) approach [51, 55].

To enhance the soundness of our READMem extension, we conduct additional experiments on the VOTS2023 dataset [92]. We tabulate in Table 5.2 the results of sVOS baselines [47, 48, 52] with and without READMem on the VOTS2023 tracking benchmark. In contrast to previous VOT challenges [135–137], VOTS2023 introduced new evaluation metrics split into: (i) a primary performance metric: The Tracking Quality (**Q**) and (ii) secondary metrics: the Accuracy (**ACC**), Robustness (**ROB**), Not-Reported Error (**NRE**), Drift-Rate Error (**DRE**) and Absence-Detection Quality (**ADQ**). We refer the reader to the VOTS2023 [92] paper for more details about the metrics. For the evaluation, we use the same settings as described above and rely on the official VOT evaluation toolkit[3]. We observe from Table 5.2 that the READMem variants consistently outperform their baseline counterpart.

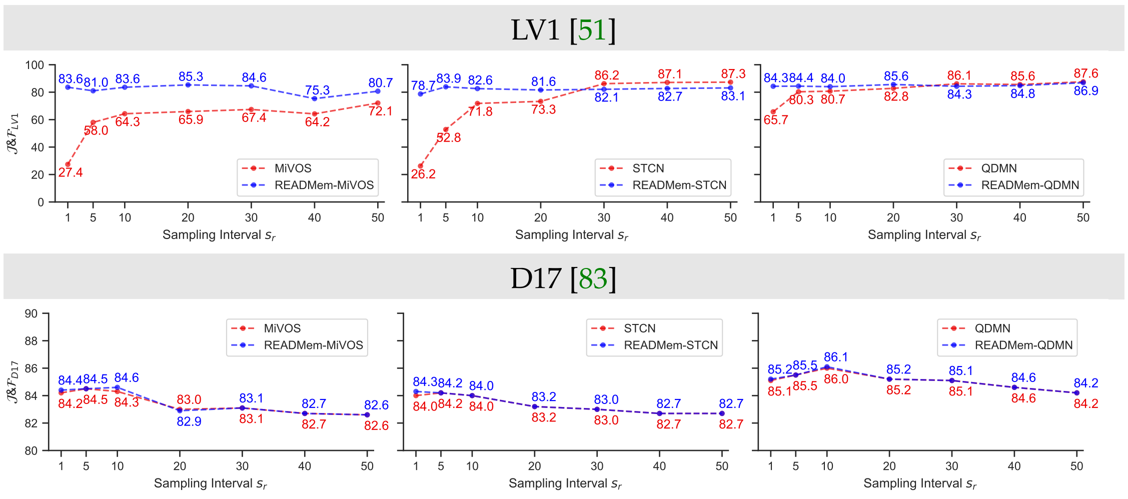
	(Higher is better)				(Lower is better)	
Method	**Q**	**ACC**	**ROB**	**ADQ**	**NRE**	**DRE**
MiVOS [47] (CVPR 21)	0.38	0.55	0.54	0.75	0.41	0.06
MiVOS [47] ($s_r = 50$) (CVPR 21)	$0.39^{\uparrow 0.01}$	0.55	$0.58^{\uparrow 0.04}$	$0.67^{\downarrow 0.08}$	$0.35^{\downarrow 0.06}$	$0.07^{\uparrow 0.01}$
READMem-MiVOS (ours)	$0.43^{\uparrow 0.05}$	$0.57^{\uparrow 0.02}$	$0.60^{\uparrow 0.06}$	$0.67^{\downarrow 0.08}$	$0.33^{\downarrow 0.08}$	0.06
STCN [48] (NeurIPS 21)	0.40	0.55	0.62	0.67	0.29	0.08
STCN [48] ($s_r = 50$) (NeurIPS 21)	0.40	0.55	$0.61^{\downarrow 0.01}$	$0.61^{\downarrow 0.06}$	0.29	$0.10^{\uparrow 0.02}$
READMem-STCN (ours)	$0.42^{\uparrow 0.02}$	$0.56^{\uparrow 0.01}$	$0.66^{\uparrow 0.04}$	$0.57^{\downarrow 0.10}$	$0.25^{\downarrow 0.04}$	$0.09^{\uparrow 0.01}$
QDMN [52] (ECCV 22)	0.44	0.59	0.62	0.69	0.28	0.10
QDMN [52] ($s_r = 50$) (ECCV 22)	$0.42^{\downarrow 0.02}$	0.59	$0.60^{\downarrow 0.02}$	$0.63^{\downarrow 0.06}$	$0.30^{\uparrow 0.02}$	$0.11^{\uparrow 0.01}$
READMem-QDMN (ours)	$0.45^{\uparrow 0.01}$	0.59	$0.63^{\uparrow 0.01}$	$0.67^{\downarrow 0.02}$	$0.27^{\downarrow 0.01}$	$0.09^{\downarrow 0.01}$

Table 5.2: Quantitative evaluation of sVOS methods [47, 48, 52] with and without READMem on the VOTS2023 [92] datasets. We use the same settings as described in Section 5.5.1 and the official VOT evaluation toolkit. Note that these results were obtained on rescaled videos, where the height of each video in the datasets was standardized to 480 pixels (as in DAVIS [83]) and the width correspondingly to retain the same aspect ratio.

We display the performance of MiVOS [47], STCN [48] and QDMN [52] with and without the READMem extension when varying the sampling interval s_r on the D17 [83] dataset, using the same configuration as in Section 5.5.1.

In Figure 5.4, we observe that increasing the sampling interval generally improves the performance of all methods on long videos, regardless of the baseline employed.

3 https://github.com/votchallenge/toolkit – version 0.6.4



(a) MiVOS *vs.* READMem-MiVOS (b) STCN *vs.* READMem-STCN (c) QDMN *vs.* READMem-QDMN

Figure 5.4: Performance comparison of sVOS baselines (MiVOS [47], STCN [48], QDMN [52]) with and without the READMem extension on the LV1 [51] and D17 [83] datasets when varying the sampling interval s_r.

However, this trend does not hold when working with short video sequences. It is essential to utilize a sampling interval that does not negatively impact the performance on both long and short video sequences. This is where our READMem extension becomes valuable, as it enables the sVOS pipeline to use a small sampling interval (typically $s_r \in [1-10]$) that achieves and maintains high performance for the long and short video settings.

5.5.2 *Ablation Study*

For clarity and to isolate the benefits of each design decision, we present our ablation results solely for READMem-MiVOS in Table 5.3. We observe that integrating embeddings of frames that diversify the memory (DME) and enforcing a lower bound on similarity (LSB) results in a significant performance improvement for long videos. Moreover, using the robust association of memory embeddings (REA) also leads to a substantial performance increase. Although including the adjacent frame leads to a slight improvement for long videos, it is particularly important when dealing with short sequences as the previous adjacent frame provides recent information (not guaranteed by our module).

In Table 5.4, we compare different approaches for inferring the transition matrix $\mathbf{T}_n$. Our results indicate, as expected that using the weight matrix $\mathbf{W}$ to generate the transition matrices $\mathbf{T}_n$ leads to better performance compared to the affinity matrix $\mathbf{A}$. Furthermore,

Configuration	$\mathcal{J}\&\mathcal{F}_{\text{LV1}}$	$\mathcal{J}\&\mathcal{F}_{\text{D17}}$
MiVOS [47] + adj. frame (baseline)	64.3	84.3
MiVOS [47] + DME (ours)	$69.5^{\uparrow 5.2}$	$81.3^{\downarrow 3.0}$
MiVOS [47] + DME + LSB (ours)	$75.0^{\uparrow 10.7}$	$81.3^{\downarrow 3.0}$
MiVOS [47] + DME + LSB + adj. frame (ours)	$77.4^{\uparrow 13.1}$	$84.3^{\uparrow 0.0}$
MiVOS [47] + DME + LSB + adj. frame + REA (ours)	$\mathbf{86.0}^{\uparrow 21.7}$	$\mathbf{84.6}^{\uparrow 0.3}$

Table 5.3: Demonstrating the benefits of our memory extension, *i.e.*, diversity of the embeddings (DME), robust embedding association (REA) and lower similarity bound (LSB).

Computation of the transition matrix $\mathbf{T}_n$	$\mathcal{J}\&\mathcal{F}_{\text{LV1}}$ using $\mathbf{W}_n$	$\mathcal{J}\&\mathcal{F}_{\text{LV1}}$ using $\mathbf{A}_n$
Hungarian Method [138]	76.8	75.1
argmax along the columns	**86.0**	73.4
argmax along the rows	79.8	**77.1**

Table 5.4: Performance comparison on the LV1 [51] dataset when computing the transition matrix $\mathbf{T}_n$ through different methods. As a reminder, the baseline (*i.e.*, MiVOS [47]) achieves a $\mathcal{J}\&\mathcal{F}$ score of 64.3 on LV1 [51] using the same configuration.

we note that using the argmax function along the columns axis (memory) leads to the best performance in comparison to applying argmax along the rows axis (query) or when using a bijective mapping (Hungarian Method [138]).

Table 5.5 tabulates the $\mathcal{J}\&\mathcal{F}$ score and Gramian of READMem-MiVOS on the three sequences of LV1 [51] for three different sampling interval s_r. In Figure 5.5, we record the Gramian (diversity of the memory) for MiVOS with and without our READMem extension on *blueboy*, *dressage* and *rat* sequences from LV1 [51] with different sampling intervals (*i.e.*, 1, 5 to 10). We note that with READMem, the diversity is effectively enhanced throughout the segmentation process as the gramian continuously grows with the percentage of sequence processed. Note that this leads to an overall better segmentation as indicated in Table 5.6, where a high Gramian generally correlates to higher performance. Interestingly, the memory reached full capacity later than expected for the *rat* sequence with sampling intervals of 5 or 10. This behavior likely results from the lower similarity bound, as in many frames, the rat is only partially visible, *i.e.*, either occluded or out of view.

	blueboy		*dressage*		*rat*	
$\mathbf{s_r}$	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$
1	**89.5**	$\mathbf{14.6 \times 10^{-7}}$	83.7	3.36×10^{-8}	77.7	4.22×10^{-10}
5	87.0	6.35×10^{-7}	83.7	9.38×10^{-8}	74.9	1.39×10^{-10}
10	88.1	4.31×10^{-7}	**84.0**	$\mathbf{18.0 \times 10^{-8}}$	**86.0**	$\mathbf{82.3 \times 10^{-10}}$

Table 5.5: Performance and Gramian of READMEm-MiVOS for different sampling intervals s_r on LV1 [51]. Note that high $\mathcal{J}\&\mathcal{F}$ scores correlate with high Gramian values.

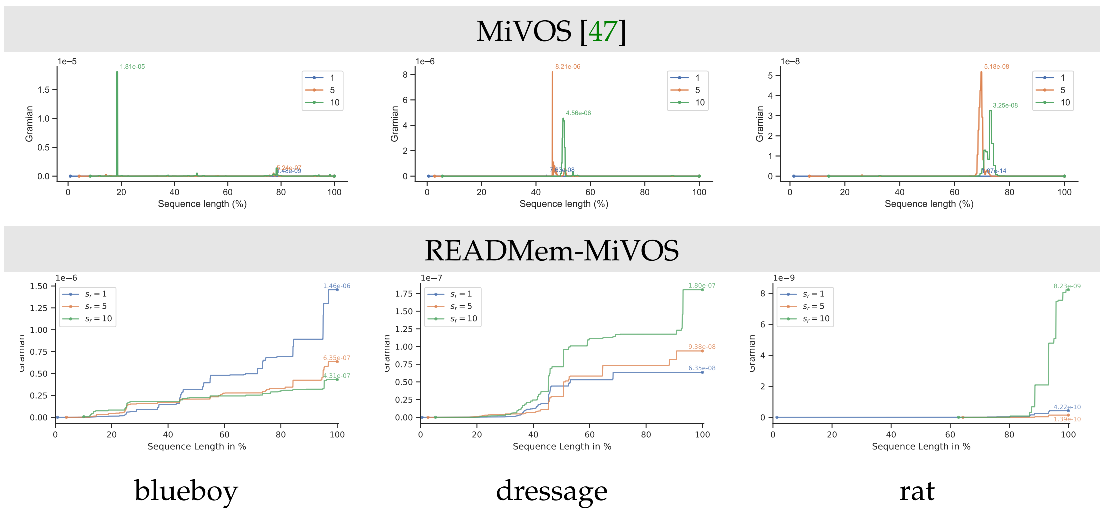
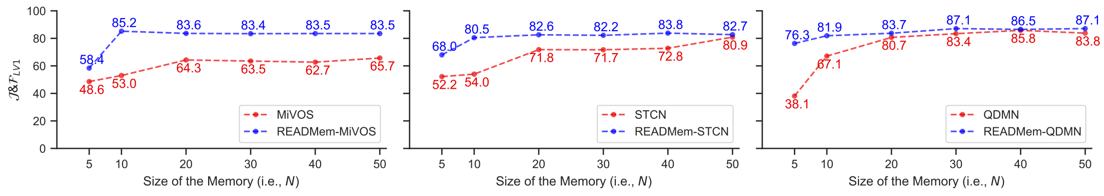


Figure 5.5: Evolution of $\lvert\boldsymbol{\Gamma}\rvert$ for different sampling interval on the LV1 [51] dataset for MiVOS without and with our READMem extension. Note that we start recording the memory's diversity only when we reached full capacity, as before this, the only criteria to store embeddings in the memory is based on the lower similarity bound.

	0.0% – 26.20%		26.20% – 57.64%		57.64% – 83.63%	
s_r	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$	$\mathcal{J}\&\mathcal{F}$	$\lvert\boldsymbol{\Gamma}\rvert$
1	**94.3**	0.61×10^{-7}	**72.5**	$\mathbf{4.80 \times 10^{-7}}$	**95.6**	$\mathbf{6.93 \times 10^{-7}}$
5	93.3	1.46×10^{-7}	59.3	2.34×10^{-7}	92.2	3.45×10^{-7}
10	90.4	$\mathbf{1.69 \times 10^{-7}}$	66.1	2.44×10^{-7}	95.4	3.09×10^{-7}

Table 5.6: Relative performance and Gramian (*i.e.*, $\lvert\boldsymbol{\Gamma}\rvert$) evolution for three sections on the *blueboy* sequence [51]. The final Gramian and $\mathcal{J}\&\mathcal{F}$ score is reported in Table 5.5.



(a) MiVOS *vs.* READMem-MiVOS

(b) STCN *vs.* READMem-STCN

(c) QDMN *vs.* READMem-QDMN

Figure 5.6: We compare the performance of sVOS baselines (MiVOS [47], STCN [48], QDMN [52]) with and without the READMem extension on the LV1 [51] dataset while varying the size of the memory (*i.e.*, N). A general tendency is that increasing the memory size, leads to better performance.

5.5.3 *Performance as a Function of Memory Size*

We explore the impact of the size of the memory on the performance of MiVOS [47], STCN [48] and QDMN [52] with and without our READMem extension on the LV1 [51] dataset. We follow the same experimental setup as in Section 5.5.1 (with $s_r = 10$), except for the varying memory size N, which ranges from 5 to 50.

From Figure 5.6, we observe that the performance of the baselines improves as the memory size increases. Similarly, although to a lesser extent, the READMem variants also demonstrate improved performance with larger memory sizes. However, the READMem variations consistently outperform their respective baselines, especially when using a smaller memory size. This is desired as a smaller memory requires less GPU resources.

Comparing Figure 5.6 with Figure 5.4, we notice that increasing the sampling interval (*i.e.*, s_r) of the baselines leads to a significant boost in performance compared to increasing the memory size (*i.e.*, N). Hence, storing a diverse set of embeddings in the memory is more beneficial than including additional ones.

5.5.4 *Initialization of the Memory*

We investigate the performance variation when employing two different initialization for READMem in Table 5.7. The strategies are as follows: (1) Integrates every t-th frame into the memory until full, while (2) Fills the memory slots with the embeddings of the annotated frame and includes a new frame to the memory if the conditions on the lower bound on similarity and the Gramian are met (follows a greedy approach). The second strategy yields worse results on the short scenarios and is slightly below the performance of strategy (1) on LV1 [51]. We argue that with longer sequences the memory has more opportunities to integrate decisive frame representations in the memory to use as a reference. Hence, initialization plays a crucial role in short videos, but as the method

Initialization	READMem-MiVOS $\mathcal{J}\&\mathcal{F}_{\text{LV1}}$	READMem-MiVOS $\mathcal{J}\&\mathcal{F}_{\text{D17}}$	READMem-STCN $\mathcal{J}\&\mathcal{F}_{\text{LV1}}$	READMem-STCN $\mathcal{J}\&\mathcal{F}_{\text{D17}}$	READMem-QDMN $\mathcal{J}\&\mathcal{F}_{\text{LV1}}$	READMem-QDMN $\mathcal{J}\&\mathcal{F}_{\text{D17}}$
(1)	83.6	84.6	82.6	84.0	84.0	86.1
(2)	82.7	73.7	85.3	73.6	72.5	73.3

Table 5.7: Performance variation when leveraging two different initialization strategies for READMem-MiVOS. Besides the initialization strategy, the remaining parameters are consistent to Section 5.5.1 (we set $s_r = 10$).

observes longer videos and has access to a larger pool of frames to select from, the importance diminishes.

5.6 Qualitative Results

We display qualitative results for the READMem variations of MiVOS [47], STCN [48] and QDMN [52] along with their baseline on the LV1 [51] dataset. We also provide the results for XMem [55], which represents the state-of-the-art.

In Figure 5.7, Figure 5.8 and Figure 5.9 we display the results for the *blueboy*, *dressage* and *rat* sequences in LV1 [51] respectively when setting the sampling interval to $s_r = 1$. Moreover we provide in Figures 5.10, 5.11 and 5.12 the results for a sampling interval $s_r = 10$. The estimated segmentation mask of the baselines (MiVOS [47], STCN [48], and QDMN [52]) are visualized in red, while the results of the READMem-based variations (READMem with a baseline) are highlighted in blue. The intersection between the prediction of a baseline and its corresponding READMem variation is depicted in turquoise. The ground-truth contours are highlighted in yellow. We depict XMem [55] results in purple.

5.6.1 *Qualitative Results on LV1 [51] with $s_r = 1$*

In this subsection we present the results of all baselines without and with our extension (along the ground-truth) when setting the sampling interval to $s_r = 1$. This effectively eliminates the need for a meta-parameter that controls the update frequency of the memory and solely relies on a diversity criterion to update the memory. For completeness we also provide the results of XMem [55], *i.e.* the state-of-the-art approach at the time this research was initiated.

5.6.2 *Qualitative Results on LV1 [51] with* $s_r = 10$

Similarly to Subsection 5.6.1 we present the results of all baselines without and with our extension (along the ground-truth) when setting the sampling interval to $s_r = 10$, while also providing the results for XMem [55].

5.7 Limitations

We are aware of the limitations imposed by the hand-crafted threshold for the lower similarity bound l_{sb}, although to avoid any fine-tuning, we set the threshold value to 0.5. A more thoughtful approach would incorporate a learnable parameter. This approach could potentially lead to improved performance, albeit at the expense of the plug-and-play nature of our extension. Another point for improvement is to reduce the participation of the background when computing the similarity between two embeddings. A possible enhancement is to integrate either the segmentation mask estimated by the sVOS pipeline or use the memory values to estimate a filter that can be applied to the memory keys before computing a similarity score.

An additional factor to consider is the overhead READMem introduces to an sVOS backbone, specifically regarding its GPU memory footprint and inference time relative to performance gains as N grows (not considering the encoder and decoder complexity as they are not influenced by the memory size). Here, our DME component (refer to Subsection 5.4.2), designed to enhance the diversity of the external memory $\mathcal{M}$, adds a total time complexity of $\mathrm{O}(N^4 + N^2 \cdot H \cdot W \cdot C_k)$ to the baseline sVOS approach. This complexity is dominated by two factors: (1) the computation of the similarities in Equation (5.5), which costs $\mathrm{O}(N^2 \cdot H \cdot W \cdot C_k)$ as we recompute the similarities for every new query frame through the REA module; (2) the repeated use of the LU decomposition [132] (which has a time complexity of $\mathrm{O}(N^3)$) for all N Gram matrices $\mathbf{\Gamma}$ variants, to identify which slot n to replace with the incoming embeddings, adding a further $\mathrm{O}(N^4)$. Meanwhile, our REA module introduces a time complexity of $\mathrm{O}(N \cdot H \cdot W \cdot (C_k + H \cdot W))$, which arises from the two-stage process of building and then using a transition matrix $\mathbf{T}$ to remap features for the reference change operation in Equation (5.6). Hence, the total complexity for REA is the sum of two parts: (1) building N transition matrices $\mathbf{T}$ such that the number of non-zero elements are $\mathrm{nnz}(\mathbf{T}) = H \cdot W$ via a column-wise argmax operation, which entails a time-complexity of $\mathrm{O}(N \cdot H^2 \cdot W^2)$; (2) using the transition matrix $\mathbf{T}$ (*i.e.*, acting as a look-up table) to permute key features, inducing a complexity of $\mathrm{O}(N \cdot C_k \cdot \mathrm{nnz}(\mathbf{T})) = \mathrm{O}(N \cdot C_k \cdot H \cdot W)$.Note that, while other operations like flattening the key matrices (*i.e.*, $\mathrm{vec}(\cdot)$) are performed, their $(O)(N \cdot H \cdot W \cdot C_k)$ complexity is negligible and thus absorbed by the higher-order terms. To summarize, when comparing to the initial backbone's time complexity $\mathrm{O}(N \cdot H^2 \cdot W^2 \cdot (C_k + C_v))$, to the DME's complexity of $\mathrm{O}(N^4 + N^2 \cdot H \cdot W \cdot C_k)$, and to the REA's $\mathrm{O}(N \cdot H \cdot W \cdot (C_k + H \cdot W))$. In a purely asymptotic analysis, the dominant term is $\mathrm{O}(N^4)$. However, given that in practi-

Method	Memory Size (N)	$\mathcal{J}\&\mathcal{F}_{\text{LV1}}$	FPS	VRAM (MiB)
MiVOS (baseline)	10	53.0	14.5	1092
MiVOS-READMem (ours)	10	$85.2^{\uparrow 32.2}$	$9.6^{\downarrow 4.9}$	1148
MiVOS (baseline)	20	64.3	9.3	1631
MiVOS-READMem (ours)	20	$83.6^{\uparrow 19.3}$	$6.0^{\downarrow 3.3}$	1738
MiVOS (baseline)	30	63.5	6.6	2173
MiVOS-READMem (ours)	30	$83.4^{\uparrow 19.9}$	$4.7^{\downarrow 1.9}$	2312

Table 5.8: Quantitative results of MiVOS [47] and our variant of MiVOS-READMem on the LV1 [51] datasets regarding the network's footprint (allocated memory) on the GPU and inference time w.r.t. to memory size and performance. We use a sampling interval set to $s_r = 10$. READMem leads to a drop of about 30% in FPS, but allows us to gain better performance for the same memory size.

cal scenarios with typical parameters values of $H = 30$, $W = 60$, $C_k = 128$ and $C_v = 512$ (for an image resolution of 480 by 960 pixels), and where the user set's $N \in 10$ to 30, the N^4 component only becomes the predominant bottleneck (overshadowing all other terms) when N exceeds approximately 1300. In practice, the selection of N represents a trade-off between the desired inference time, overall performance gains, and the GPU's memory footprint. We provide a quantitative evaluation of READMem's overhead w.r.t. the original MiVOS [47] baseline in Table 5.8, showcasing its behavior when varying the memory size N between 10 and 30.

We can optimize our READMem module, by altering the frame of reference used by our REA module for the features permutation. Currently we re-project all N memory keys frames into the frame of reference of the query frame. A more efficient design is to leverage a fixed frame of reference (*e.g.*, the initialization frame at $t = 0$). Thus, the memory features remain in a stable projection, and only the new query key's needs to be projected in the reference frame of the initialization frame. This would reduce the computational overhead of our READMem module such that: (1) our REA module's cost would decrease from $\mathrm{O}(N \cdot H \cdot W \cdot (C_k + H \cdot W))$ to $\mathrm{O}(H \cdot W \cdot (C_k + H \cdot W))$, not dependent on the memory size; (2) and the second part of our DME component would decrease from $\mathrm{O}(N^2 \cdot H \cdot W \cdot C_k)$ to $\mathrm{O}(N \cdot H \cdot W \cdot C_k)$.

5.8 Discussion

We presented READMem, a modular framework for matching-based sVOS methods, to improve the diversity of the embeddings stored in the memory for unconstrained videos and address the expanding memory demand. Our evaluation displays that sVOS methods [47, 48, 52] using our extension consistently improve their performance compared to their baseline the long video sequences (*i.e.*, LV1 [51]) without compromising their efficiency on shorter sequences (*i.e.*, D17 [83]). Moreover, we achieve competitive results with SOTA and provide a comprehensive overview of each design decision supported by an extensive ablation study.

In this chapter, we addressed the scalability challenge posed by **RQ2**, focusing on extending sVOS methods to operate effectively on unconstrained video sequences. Through READMem, we propose a modular, plug-and-play solution designed to enhance an arbitrary matching based VOS solutions that relies on an external memory. By selectively storing frame embeddings based on a diversity criterion, our module enables existing methods to handle longer sequences in a training-free manner. Our approach effectively improves the diversity of the embeddings stored in the memory and performs a robust association with query embeddings during the update process. Moreover, READMem removes the reliance on a manually tuned meta-parameter to regulate memory update speed, a common practice in existing approaches, where this parameter is often adapted for each benchmark to achieve optimal performance. Our experiments demonstrate that integrating READMem into several SOTA sVOS baselines (*i.e.*, MiVOS [47], STCN [48], and QDMN [52]) consistently improves performance on long video sequences (*e.g.*, LV1 [51]), while preserving accuracy on shorter ones like DAVIS 2017 [83]. Overall, we attain competitive results against the SOTA. These improvements are further supported by an extensive ablation study analyzing the effect of each component in our design. To support reproducibility and future research, we make our code publicly available[4].

In the mean time, other studies have tackled the memory management aspect of VOS methods. Notably RMem [139] by Zhou *et al*., which consider the update of the memory (of limited size) as a multi-armed bandit problem. They goal is to maximize a given reward function (Upper confidence bound (UCB)) to determine which embedding to replace in the memory, by considering two parameters: (i) *relevance*, which denotes the contribution of each saved embedding during the space-time operation and (ii) *freshness* which measures how temporally distant a given memory frame embedding is from the currently process frame. The least relevant and fresh frame is then replaced from the memory. While the proposed method ensures regular update through *freshness*, its reliance on *relevance* risks a convergence to a local optimum, as the memory might favor retaining visually similar frames, potentially leading to a redundant memory bank. Inspired by SAM2 [16], SAM2Long [66] by Ding *et al*. manages the memory of SAM2 through a restrained tree design. Their approach maintains a predefined number of pathways (alternative memories) for each object tracked. Each memory is update separately based on the confidence and occlusion score provided by SAM2, while trying to ensure a diverse set of memory embeddings. A key trade-off of this method is its substantial memory requirement, as k unique pathways are stored per tracked object. Recently, DAM4SAM [140] by Videnovic *et al*. present a memory update strategy that

4 https://github.com/Vujas-Eteph/READMem

focuses on improving sVOS approaches against distractors. The authors propose two distinct memory: (i) RAM (Recent Appearance Memory), which limits the memory size to a pre-defined size and updates it's embeddings in a FIFO fashion (guarantying that the object is present with SAM2); (ii) DRM (Distractor Resolving Memory), similarly to RAM but with two additional constraints, updating the memory every 5-th frame and concatenating anchor frames (frames that are very likely to contain distractors). This distractors detection takes advantage of the three object masks prediction strategy by SAM2, as SAM2 predicts preemptively (through a low IoU score) the presence of a distractors before complete confusion, *i.e.*, where SAM2 completely switches to the distractors in later frame.

In summary, this chapter provides a partial response to RQ2 (*i.e.*, the scalability issue) by demonstrating that sVOS methods can be extended to unconstrained video sequences in an offline setting, without compromising segmentation quality or introducing significant computational overhead. However, a key aspect of unconstrained video processing remains outside the scope of this work: segmentation is initialized using fine-grained masks, and user interaction is deliberately omitted to focus solely on method scalability. As a result, the scenario considered, while unconstrained in terms of sequence length and visual complexity, does not completely reflect the practical constraints of online applications, where detailed initialization is impractical and prolonged errors may go unnoticed without timely correction.

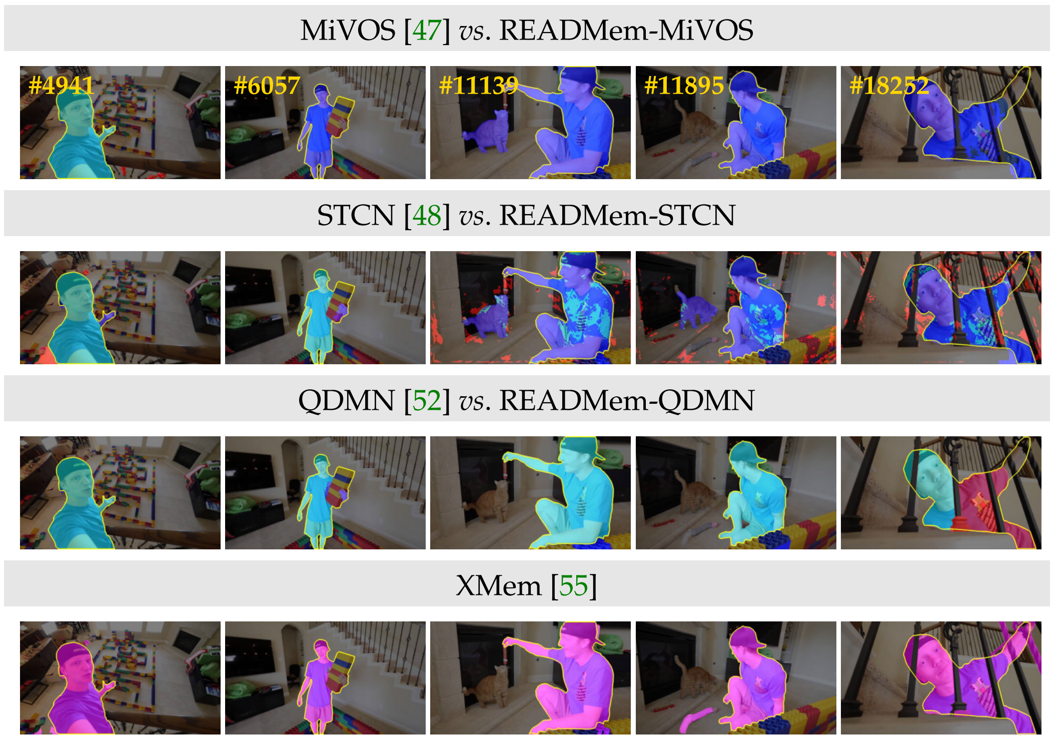

MiVOS [47] *vs.* READMem-MiVOS
#4941
#6057
#11139
#11895
#18252
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.7: Results on the *blueboy* sequence of LV1 [51] with $s_r = 1$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

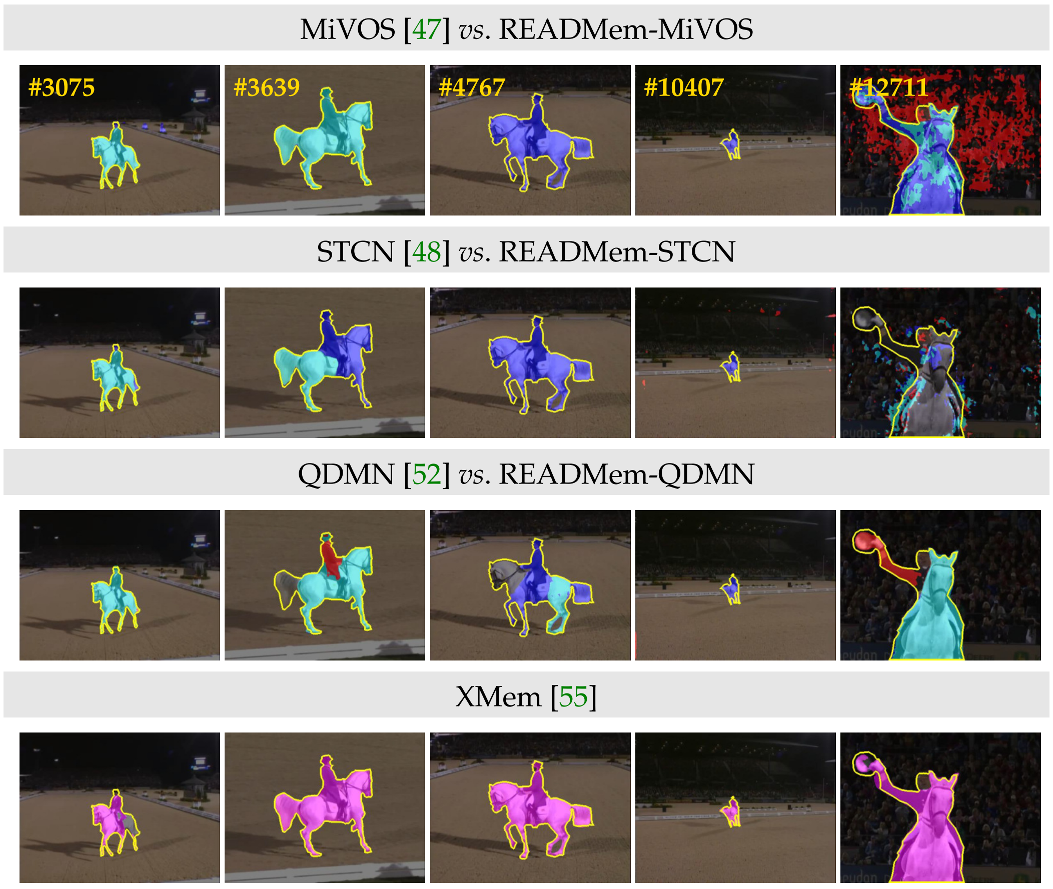

MiVOS [47] *vs.* READMem-MiVOS
#3075
#3639
#4767
#10407
#12711
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.8: Results on the *dressage* sequence of LV1 [51] with $s_r = 1$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

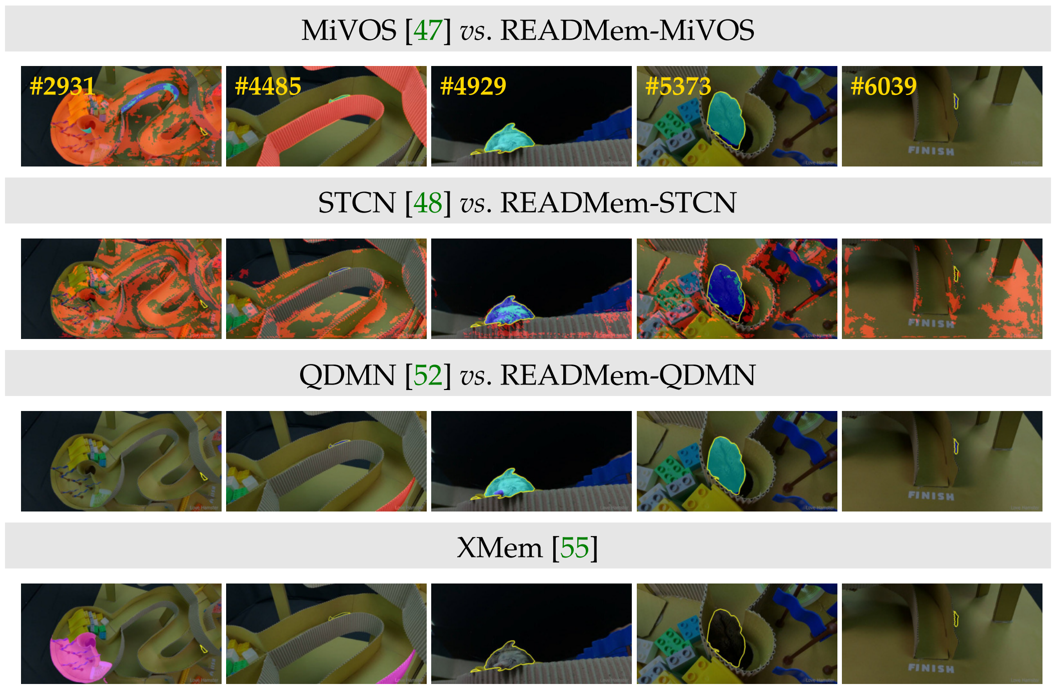

MiVOS [47] *vs.* READMem-MiVOS
#2931
#4485
#4929
#5373
#6039
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.9: Results on the *rat* sequence of LV1 [51] with $s_r = 1$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

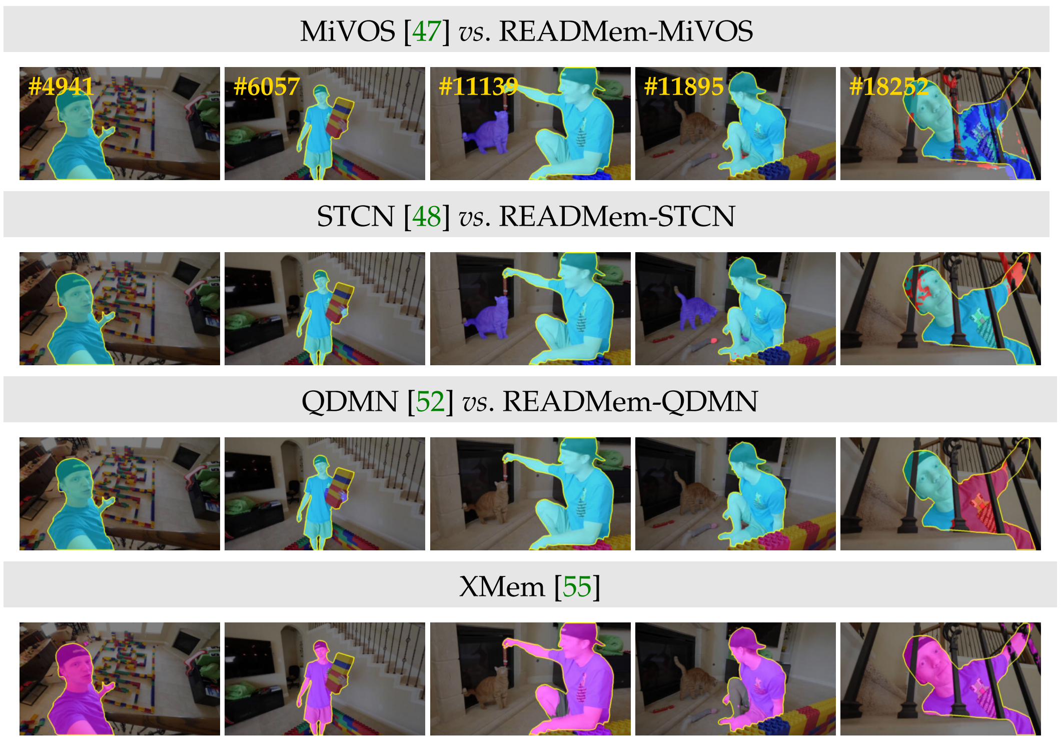

MiVOS [47] *vs.* READMem-MiVOS
#4941
#6057
#11139
#11895
#18252
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.10: Results on the *blueboy* sequence of LV1 [51] with $s_r = 10$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

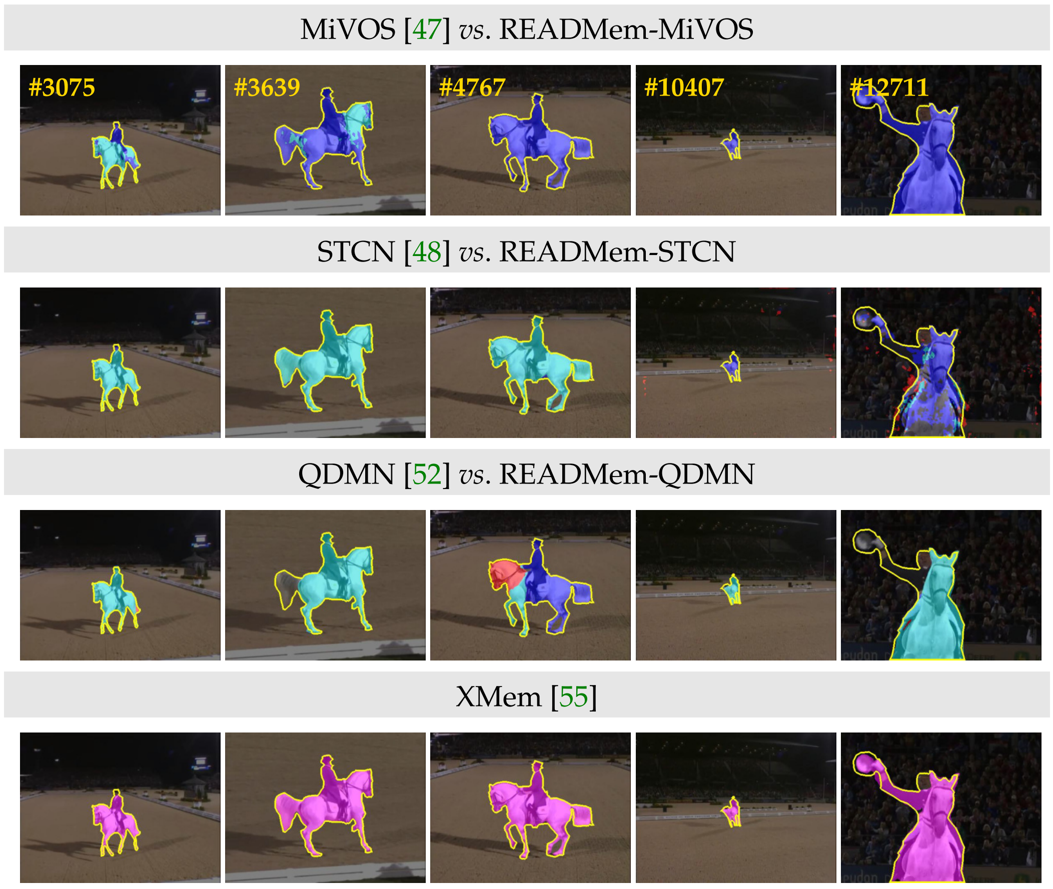

MiVOS [47] *vs.* READMem-MiVOS
#3075
#3639
#4767
#10407
#12711
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.11: Results on the *dressage* sequence of LV1 [51] with $s_r = 10$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

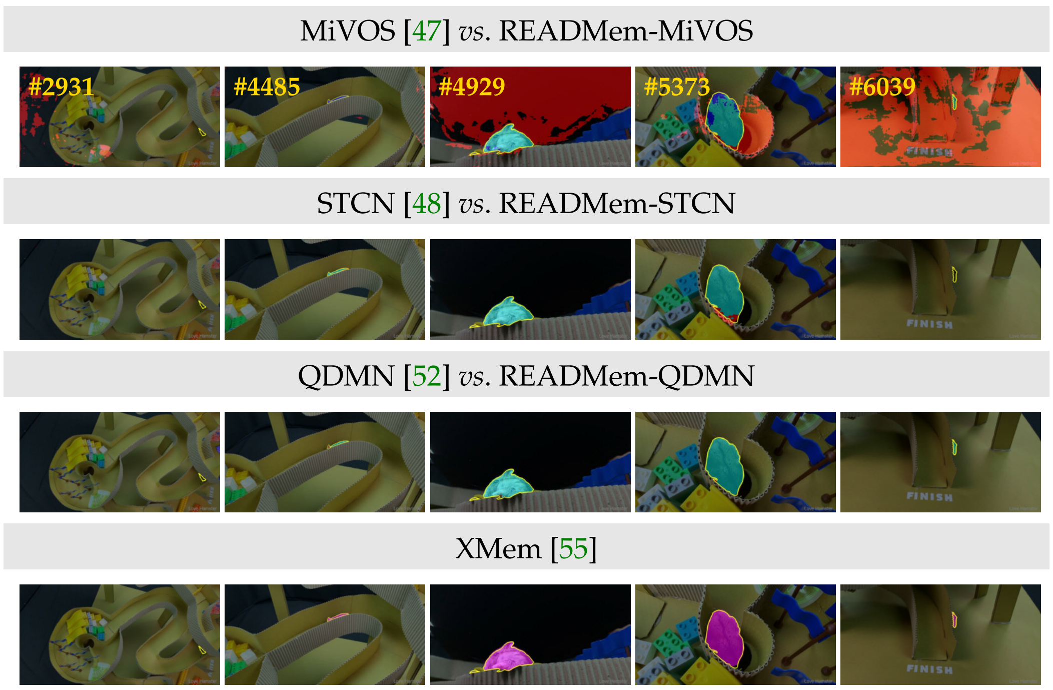

MiVOS [47] *vs.* READMem-MiVOS
#2931
#4485
#4929
#5373
#6039
STCN [48] *vs.* READMem-STCN
QDMN [52] *vs.* READMem-QDMN
XMem [55]

Figure 5.12: Results on the *rat* sequence of LV1 [51] with $s_r = 10$. We depict the results of: the baselines in red, the READMem variations in blue, the intersection between both in turquoise, the ground-truth contours in yellow and XMem [55] results in purple.

6 UNCERTAINTY-GUIDED INTERACTIVE VIDEO OBJECT SEGMENTATION

In this final chapter, we address the joint challenge posed by RQ1 and RQ2 by exploring how VOS systems can operate effectively in an online unconstrained video context (*i.e.*, hypothetical streaming scenario), where user effort must remain minimal, and model decisions must be made on-the-fly without access to future frames. In earlier chapters, we explored each research question in isolation. In Chapter 4, we reduced annotation effort through click-based interactions, partially answering RQ1. However, interactions occurred in multiple rounds and were evaluated on short sequences, where user monitoring and review were still feasible. In Chapter 5, we tackled RQ2 by improving the scalability of sVOS methods for long, unconstrained videos, but in an offline setting that assumed detailed initial masks and omitted user feedback entirely.

Building on these contributions, this chapter introduces the task of Lazy interactive Video Object Segmentation (ziVOS), which unifies the goals of scalability (RQ2) and efficient interaction (RQ1) for unconstrained video segmentation, particularly in challenging scenarios such as long-term or streaming video. Unlike traditional settings, ziVOS assumes a single-pass segmentation process, where future frames are inaccessible, and rewinding is not possible. These constraints render conventional iVOS approaches ineffective, as they rely on multiple rounds of interaction and retrospective correction, while sVOS methods lack the ability to recover from drift without supervision.

To address this, we propose **Lazy-XMem**, a baseline framework that proactively calls for user help (*i.e.*, interactions) while segmenting a video sequence. Starting from a single-click initialization (as in Chapter 4), Lazy-XMem estimates segmentation uncertainty on-the-fly to determine whether to request user input, apply pseudo-corrections, or continue autonomously. This approach enables the system to maintain segmentation quality while minimizing unnecessary supervision, making it suitable for unconstrained applications where user attention is limited. We evaluate our approach on the LVOS dataset [91] and introduce complementary metrics to measure segmentation robustness and user workload under streaming constraints.

This chapter is based on our *ACCV 2024* paper [128], and is structured as follows: We outline the motivation in Section 6.1, formulate our problem in Section 6.2, and discuss related work in Section 6.3. In Section 6.4, we describe Lazy-XMem and our complementary metric for the ziVOS task in Section 6.5. We present our quantitative results in Section 6.6, ablation studies in Section 6.7 and qualitative results in Section 6.8. We conclude this chapter by discussing the limitations of our pipeline in Section 6.9 and providing a summary in Section 6.10.

6.1 Motivation

When this research was initiated, most state-of-the-art sVOS approaches were designed for short, pre-recorded video sequences, with limited attention given to their applicability in long-term or streaming scenarios. Similarly, most interactive segmentation methods (iVOS) have been constrained to short-form video, where the full sequence is available for analysis, and embeddings from all frames can be stored and queried to guide user interaction. These assumptions are incompatible with real-world settings such as live streams or surveillance, where frames arrive sequentially and future content is unknown.

In this chapter, we introduce a video object segmentation (VOS) variant that bridges interactive and semi-automatic approaches termed Lazy Video Object Segmentation (ziVOS). In contrast to both tasks, which handle video object segmentation in an offline manner (*i.e.*, pre-recorded sequences), we propose through ziVOS to target online recorded sequences. Here, we strive to strike a balance between performance and robustness for long-term scenarios by soliciting user feedback's on-the-fly during the segmentation process. By trading-off between automation and reliability, we aim to maximize the tracking duration of an object of interest while requiring minimal user corrections to maintain tracking over an extended period. We propose Lazy-XMem as a competitive baseline that estimates the uncertainty of the tracking state to determine whether a user interaction is necessary to refine the model's prediction. We introduce complementary metrics alongside those already established in the field to quantitatively assess the performance of our method and the user's workload. We evaluate our approach using the recently introduced LVOS dataset, which offers numerous long-term videos.

VOS is a fundamental challenge involving various tasks, including sVOS and iVOS [8]. In sVOS, given an initial segmentation mask for the first video frame, methods classify each pixel in the subsequent video frames as a part of the object of interest (*i.e.*, foreground) or the background. Here, a user only interacts at the start of the sequence by providing the corresponding annotation mask to indicate which object to segment in the video. In contrast, iVOS methods incorporate user interactions in a multi-round scheme, where the user interacts with the method before each round to improve the segmentation quality on the subsequent rounds. Both applications are suited for pre-recorded sequences, *i.e.*, offline segmentation, as sVOS methods assume that the user has unlimited time to annotate the initial frame with utmost accuracy, whereas iVOS approaches expect the user to inspect the segmentation quality of the previous round, and interact for multiple rounds until the desired segmentation quality is achieved. However, while sVOS methods demonstrate impressive performances on short-term datasets [83, 84], their applicability to long-term sequences remains under-explored [39, 51, 52, 55, 88, 91, 128, 141] and yet to be addressed by iVOS methods. This underscores a gap in methodologies suited for prolonged sequences, where maintaining error-free segmentation under challenging conditions becomes increasingly difficult.

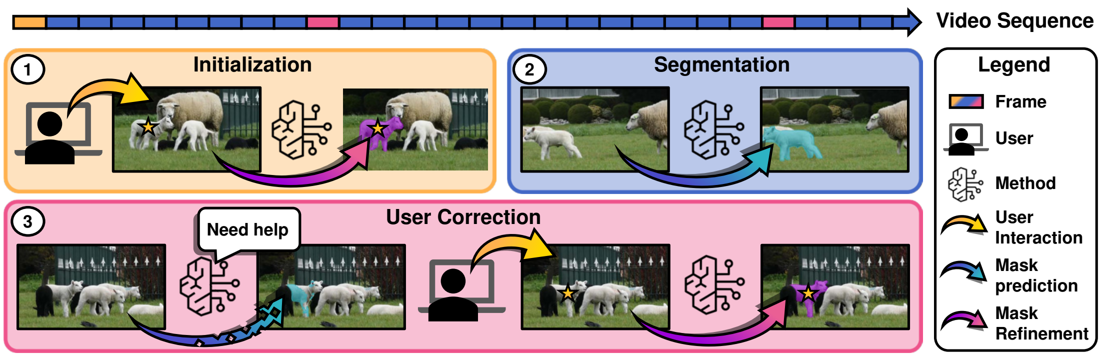


Figure 6.1: Visual representation of the ziVOS framework. (1) The user initiates the segmentation by clicking to identify the object of interest in the video, (2) thus indicating which object to segment. Only when requested by the method (3) does the user provides corrective clicks on-the-fly.

In this paper, we explore a hybrid framework, named ziVOS (depicted in Figure 6.1), that bridges the methodologies of sVOS and iVOS, focusing on maintaining robust object tracking with minimal user interactions. Unlike iVOS, we discard the round based scheme and integrate user corrections on-the-fly, refining the model's prediction as needed, while the method segments the video. Moreover, distinct from sVOS, in ziVOS the object of interest is indicated with a user interaction (*i.e.*, click). To achieve this, we only allow one interaction per frame and object, and solely rely on click-based interactions, as pointing an object is the quickest, most intuitive and predictable interaction type for humans [105, 142]. Hence, we propose ziVOS to emulate a human-in-the-loop process when segmenting a video in an online fashion, which is better suited for dynamic applications, when user engagement is feasible and where maintaining consistent object tracking in challenging conditions is more critical than achieving segmentation accuracy. Concretely, our objective shifts from segmenting an object with high accuracy to maximizing the number of frames in which the object is segmented above a minimal alignment ratio (*i.e.*, Intersection over Union (IoU)), denoted as τ_{iou}, by integrating user corrections on-the-fly (only at critical events), while simultaneously reducing the user's workload.

We propose Lazy-XMem, as a baseline for future works addressing ziVOS. Lazy-XMem assess the uncertainty of a predicted object mask on-the-fly and refines it accordingly (through SAM-HQ [61]), if the uncertainty is too high, through either pseudo-corrections or user-corrections. In our approach, the Shannon entropy [143] serves as a proxy to estimate the performance of the tracking state – alignment (*i.e.*, IoU) ratio between the predicted mask and a hypothetical ground-truth. Similarly, recent studies in iVOS [79, 80] evaluate which frame to interact with at the end of a round. They compare the embeddings of each frame in the video sequence against all other frame embeddings to determine which frame to suggest to the user for new interactions, limiting this strategy to only pre-recorded videos. In contrast, our criterion is solely defined w.r.t. the tracker's state, how accurate the prediction is for the current observed frame, allowing us to work with non-prerecorded sequences. To our knowledge, only QDMN by Liu *et al*. [52] also estimates the tracking state in an online fashion by predicting a quality score through a second head (following a similar design to [53]). However, unlike prior works [52, 79, 80], we estimate the tracker's state on pixel level in a *post-hoc* fashion, removing the need for training an additional auxiliary network. An additional benefit to computing the uncertainty on the pixel level is that we can visually indicate ambiguous regions to the user where an interaction might be the most helpful. Moreover, depending on the confidence of the predicted mask, Lazy-XMem decides whether the currently predicted mask will be stored in the memory. Hence, by selectively refining masks based on entropy-driven uncertainty estimation, we aim to maintain a balance between robustness and user-workload in ziVOS, specifically in long-term scenarios. Hence, while on-the-fly interactions may suggest constant user supervision, our approach minimizes this need by prompting the user (ideally) only at critical events – when Lazy-XMem is uncertain about its prediction.

Crucially, we did not conduct a user study in this work to estimate cognitive load, thus not relying on subjective metrics like the NASA Task Load Index (NASA-TLX) [144] or System Usability Scale (SUS) [145]. Our goal is to explore a user-efficient interactive design, adapted for long-term user involvement in VOS for continuous and robust object tracking, where the user would only intervene when necessary (asked by the method), allowing them to focus on other tasks simultaneously.

6.2 Problem Formulation

Similarly to Chapter 5, for a given video sequence $\mathcal{V} = \{\mathbf{I}_t \mid t \in \{0, 1, \ldots, T\}\}$ where $\mathbf{I}_t \in \mathbb{R}^{3 \times H_R \times W_R}$ we assume $T \to \infty$. Our goal is still to predict a corresponding segmented sequence $\mathcal{S} = \{\mathbf{M}_t \mid t \in \{0, 1, \ldots, T\}\}$, where $\mathbf{M}_t \in \{0, 1\}^{|\mathcal{O}| \times H_R \times W_R}$ corresponds to a predicted object masks computed by a segmentation pipeline.

Analogously to iVOS, ziVOS utilizes sparse user inputs, *i.e.*, interaction maps (sparse matrices) $\mathbf{U}_t \in \{0, 1\}^{|O| \times H \times W}$, to indicate which objects to track in a given video sequence $\mathcal{V}$ by providing user-clicks for the t-th frame. Similarly to sVOS, we perform only a single segmentation pass and segment the sequence in a step-wise manner. Hence, the method can only leverage information from previously seen frames when segmenting a video. However, unlike sVOS, we do not restrict the number of user input to solely the beginning of a video. Instead, we allow user interactions to be distributed throughout the sequence $\mathcal{V}$ to allow the user to provide corrections on-the-fly. It is important to note that only a single click per object is allowed for any given frame. In addition, unlike sVOS the initial user cue is not provided through a full ground-truth mask $\mathcal{G}_t$, but rather through sparse user inputs $\mathbf{U}_t^o$ like in iVOS. Importantly, in ziVOS, the method actively calls the user for interaction in an attempt to lower the monitoring effort.

Conceptually, we can summarize the overall segmentation process through an ziVOS segmentation pipeline, denoted by f as

$$f : \mathcal{V} \times \{\mathbf{U}_t^o \mid t \in \mathcal{T}_U\} \rightarrow \mathcal{S} \times \mathcal{C}, \tag{6.1}$$

where $\mathcal{T}_U \subseteq \{0, 1, \ldots, T\}$ denotes the (dynamically determined) set of frame indices at which the user is called to interact with the method, which can happen at arbitrary time steps. Note that, due to the streaming and interactive nature of the problem, user interactions are not predefined. Instead, the method decides on-the-fly when to request user assistance by computing an associated internal confidence state. Here, $\mathcal{C}$ denotes the set of per-frame confidence states, such that $\mathcal{C} = \{\mathbf{C}_t \mid t \in \{0, 1, \ldots, T\}\}$. Note that we denote through $\mathcal{C}$ a general confidence space that accommodates scalar values, spatial maps, or higher-dimensional tensors, depending on the underlying method design. Finally, depending on the internal confidence state for the t-th frame, $\mathbf{C}_t$, the method applies a decision function $g : \mathbf{C}_t \rightarrow \{0, 1\}$, which calls the user for assistance if $g(\mathbf{C}_t) = 1$, such that $\mathcal{T}_U = \{t \in \{0, 1, \ldots, T\} \mid g(\mathbf{C}_t) = 1\}$.

6.3 Related Work

6.3.1 *Semi-Automatic Video Object Segmentation*

Early deep learning methods in sVOS follow an *online fine-tuning approache* [18–22, 24], which adapts the network's parameters on-the-fly while segmenting the objects of interest in the video sequence. This results, in slow inference times and poor generalization capabilities [8]. Concurrently, *propagation-based methods* [26–30] propagate the masks from the previous adjacent frame to the current one for segmentation, but they are prone to error accumulation and often fail during occlusions [8]. *Matching-based methods* [35–41] leverage features from the initial and previous adjacent frame to segment the current frame. The leading methods in the field further integrate features from in-between frames (previously processed) into an external memory [38, 39, 42–49], using cross-attention to link features from previous frames to the current frame to segment. However, these methods are limited in real-world applications due to their expanding memory requirements, making long-term segmentation on consumer-grade GPUs challenging.

Recent works address this bottleneck by selectively integrating frame representations into the external memory [52, 54, 128] or by generating compact representations to summarize similar features together [50, 51, 55, 88, 141]. These methods effectively manage the memory footprint, enabling more efficient sVOS on long-term videos. Newer methods [141] also explore improved ways to differentiate similar objects (distractors) from each other. Additionally, new datasets have recently been introduced [85, 89, 91] to provide an alternative to the classical DAVIS [83] and YouTube-VOS [84] datasets, with some targeted specifically for long-term video segmentation [51, 91] and tracking [92].

Contemporary works [58–60] leverage SAM [17] or a variant [61–63] to refine the original mask predicted by an sVOS baseline [39, 55]. However, in contrast to our framework, they refine every n-th mask predicted by the sVOS backbone with a SAM based approach [17, 61–63], and require continuous user monitoring to identify when interventions are needed. Furthermore, they diminish the influence to enhance the predictive accuracy for subsequent frame as they do not update the memory with the refined mask.

6.3.2 *Interactive Image Segmentation*

In iIOS, methods predict a mask for an object of interest based on user interactions for a single image. These approaches aim to reduce the user's workload by replacing densely annotated mask for sparse annotations (*e.g.*, clicks [71, 108, 110, 111, 113–115], extreme points [116–118], or bounding boxes [119–121]). Most notable approach is f-BRS [111] which optimizes internal auxiliary features of the segmentation network to align its prediction's at the clicked position with the user annotated label. A follow up work by Sofiiuk *et al*. [114], replaces the previous f-BRS backbone with an HRNet [122] + OCR [123] network, to maintain high quality features through out the network to obtain a preciser segmentation mask. Since the introduction of SAM [17], a plethora of SAM-based methods have been proposed to solve the task in medical imaging [124] and natural images [125]. For instance, SAM-HQ [61] improves upon SAM by better handling complex shapes, such as thinner structures and objects with holes. Additionally, faster approaches like FastSAM [62] and MobileFast [63] have been developed to enhance performance and efficiency.

6.3.3 *Interactive Video Object Segmentation*

Originally intended to reduce the user's workload during video annotations [14], iVOS methods integrate user interactions in a round-based process. Most approaches follow the design introduced by Benard *et al*. [70], which combines sVOS and iIOS pipelines. The blueprint process for the iVOS task is as follow: (1) Firstly, the method predicts a segmentation mask for each frame in the video (through a sVOS baseline), based on an initial mask provided for a frame. (2) Next, a user scrolls through the resulting masks and selects a frame to interact with (*e.g.*, through clicks [68, 69, 81], scribbles [47, 72–74, 78]). Based on the provided interaction, a new mask is predicted for the annotated frame, serving as a new starting point when repeating step (1). Steps (1) and (2) are repeated one after the other, until the user is satisfied with the final results.

A persistent bottleneck is determining which frame to annotate for the next round. Recent approaches address this by identifying a quartet of candidate frames for the user to annotate [78], estimating which frame would yield the most improvement [79], or using a weakly supervised method to indicate the frame and type of interaction [80] to the user. To determine which frame or set of frames to annotate, these methods map each frame in the sequence into an embedding space, restricting them to short videos, as it requires storing the embedding of every frame. Here, each embedding encode the frame's representation and the quality of the corresponding predicted mask. The best candidate frame is selected by comparing each embedding w.r.t. others and against those of annotated frames, either through an agent [79] or by choosing the embedding that is furthest from any annotated embedding [80]. In contrast, our approach introduces corrections on-the-fly by directly assessing the tracking state during segmentation, thereby proposing an online methodology that is also not restricted to short sequences.

6.3.4 *Uncertainty Estimation in Video Object Segmentation*

Uncertainty estimations is essential to improve the reliability and explainability of a model, however estimating the uncertainty of DNN, remains a challenging topic. To our knowledge, only the work by Liu *et al.* [52] incorporates a confidence score to asses the tracking state on-the-fly for the sVOS task by leveraging an auxiliary head (*i.e.*, QAM module), predicting a confidence score on how likely the predicted mask would align with a ground-truth annotation. Similar to our approach, QDMN manages its memory updates based on a threshold value that determines whether a predicted mask, given its confidence level, is reliable enough to be stored in the external memory. However, as the QAM module only predicts a single score per object, it is unable to guide the user during an interaction, as to where a correction might be the most valuable. In contrast, we explore uncertainty estimation through information theory [143] (*i.e.*, Shannon entropy) and update the memory with the refined mask.

6.4 Framework

We present Lazy-XMem, depicted in Figure 6.2, as a baseline for future works targeting ziVOS. Lazy-XMem comprises the following key components: (1) An sVOS baseline, to predict object masks; (2) An uncertainty assessment component; (3) A mask refiner, to refine the original prediction from the sVOS baseline; (4) An interaction-issuer, to issue either pseudo- or user-corrections and (5) a memory update mechanism.

6.4.1 *Backbone*

We rely on XMem [55] as our sVOS baseline. Initialized with an object mask at the beginning, the network predicts masks for subsequent frames. For simplicity, we assume the network segments a single object. The key components are:

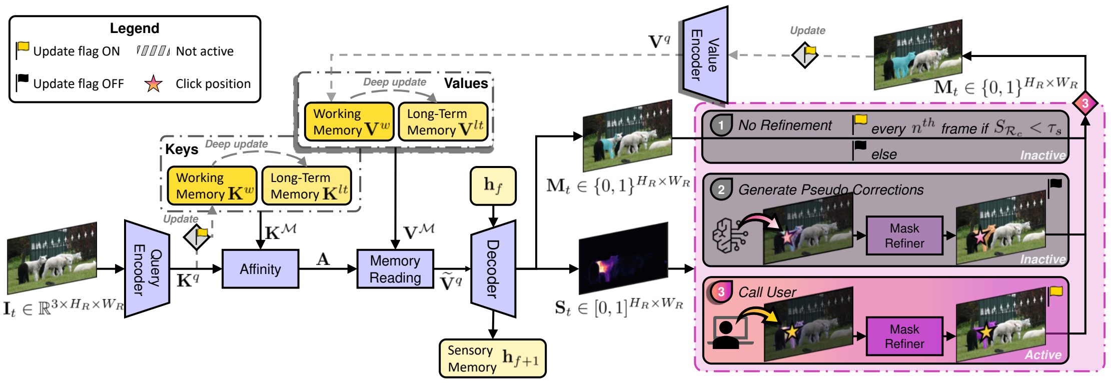

Legend
Update flag ON
Not active
Update flag OFF
Click position
Keys
Deep update
Working Memory $\mathbf{K}^w$
Long-Term Memory $\mathbf{K}^{lt}$
Values
Working Memory $\mathbf{V}^w$
Long-Term Memory $\mathbf{V}^{lt}$
Update
Query Encoder
Affinity
Memory Reading
Decoder
Value Encoder
Sensory Memory $\mathbf{h}_{f+1}$
$\mathbf{h}_f$
$\mathbf{I}_t \in \mathbb{R}^{3\times H_R\times W_R}$
$\mathbf{K}^q$
$\mathbf{K}^{\mathcal{M}}$
$\mathbf{A}$
$\mathbf{V}^{\mathcal{M}}$
$\widetilde{\mathbf{V}}^q$
$\mathbf{V}^q$
$\mathbf{M}_t \in \{0,1\}^{H_R\times W_R}$
$\mathbf{S}_t \in [0,1]^{H_R\times W_R}$
1 No Refinement
every n^{th} frame if $S_{\mathcal{R}_c} < \tau_s$
else
Inactive
2 Generate Pseudo Corrections
Mask Refiner
Inactive
3 Call User
Mask Refiner
Active

Figure 6.2: Overview of Lazy-XMem for Lazy Video Object Segmentation. Our method is relies on an sVOS baseline (*i.e.*, XMem [55]). We leverage the entropy to estimate on-the-fly the tracking state. Based on the tracking state's, the method either uses the original mask of the sVOS baseline, or refine the original mask by generating pseudo-interactions, or requesting user interaction.

Convolutional Blocks: (1) Firstly, a *query encoder*, that extracts query key features $\mathbf{K}^q \in \mathbb{R}^{C_k \times HW}$ from the current image to segment. (2) A *decoder*, which predicts an object mask $\mathbf{M}_t \in \{0,1\}^{H_R \times W_R}$ for a query frame $\mathbf{I}_t \in \mathbb{R}^{3\times H_R \times W_R}$. (3) Lastly, a *value encoder*, that extracts value features $\mathbf{V}^m \in \mathbb{R}^{C_v \times HW}$ based on the current image $\mathbf{I}_t$ and the predicted mask $\mathbf{M}_t$.

Memories: Unlike previous works [42, 47, 48], XMem [55] employs three distinct memories: a *working memory*, a *long-term memory*, and a *sensory memory*.

(1) The *working memory*, is updated every n-th frame with query and value representations $\mathbf{K}^w \in \mathbb{R}^{C_k \times NHW}$ and $\mathbf{V}^w \in \mathbb{R}^{C_v \times NHW}$, until it reaches full capacity (*i.e.*, we have N query and value representations).

(2) When the working memory is full, it is distilled into $l \in \{1, 2, \dots, L\}$ prototype features $\mathbf{k}_l^p \in \mathbb{R}^{C_k}$ and $\mathbf{v}_l^p \in \mathbb{R}^{C_v}$, based on usage frequency during memory reads. These prototypes are added to the long-term memory $\mathbf{K}^{lt} \in \mathbb{R}^{C_k \times L}$ and $\mathbf{V}^{lt} \in \mathbb{R}^{C_v \times L}$, with least-frequent-usage (LFU) filtering to remove obsolete features. Where, we denote through $\mathbf{K}^{lt} = [\,\mathbf{k}_{l=1}^p\ \mathbf{k}_{l=2}^p\ \dots\ \mathbf{k}_{l=L}^p\,]$ a concatenated form composed of $\mathbf{k}_l^p$ vectors, and similarly $\mathbf{V}^{lt}$, build by concatenating the corresponding $\mathbf{v}_l^p$ vectors.

(3) The *sensory memory*, uses GRU cells [146] to update a hidden representation $\mathbf{h}_f$ every frame, where $\mathbf{h}_f \in \mathbb{R}^{C_h \times H \times W}$, encodes prior information, *i.e.*, like position [55].

Memory Reading: During the memory read operation, feature representations from both working and long-term memories are used, totaling $Z = NHW + L$ elements. The model computes the similarity between memory keys $\mathbf{K}^{\mathcal{M}} \in \mathbb{R}^{C_k \times Z}$ and query key $\mathbf{K}^q$ using an anisotropic ℓ_2 [55] as a similarity function $\mathrm{sim}(\cdot,\cdot)$, that results in an affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times HW}$ following Equation (3.9). Note that, we have $\mathbf{K}^{\mathcal{M}} = \mathbf{K}^w \oplus \mathbf{K}^{lt}$. Next, the model applies a softmax along the rows yields the weighted matrix $\mathbf{W}$ as in Equation (3.10). A new value $\widetilde{\mathbf{V}}^q \in \mathbb{R}^{C_v \times HW}$ is then generated through Equation (3.11), where $\mathbf{V}^{\mathcal{M}} = \mathbf{V}^w \oplus \mathbf{V}^{lt}$ is the concatenated working and long-term memories values, such that $\mathbf{V}^{\mathcal{M}} \in \mathbb{R}^{C_v \times Z}$. The keys provide robust semantic information for matching, while the values encode boundary and texture cues needed for decoding [42].

6.4.2 *Uncertainty Estimation via Entropy*

To estimate the uncertainty of the tracking state (*i.e.*, the predicted segmentation mask), we leverage the Shannon entropy [143], denoted as $\mathbf{S}$. We consider pixels as discrete random variables whose classes c belong to a set $\mathcal{O}$, contains every object observed in a given video, including the background (*i.e.*, $o = 0$). We use the output values of the softmax layer as an approximation of a probability mass functions $p_{\mathcal{O}}(o \mid x_{h,w})$ for each pixel x located at (h, w). Here, $o \in \mathcal{O}$ denotes the object class of the pixel, and (h, w) specifies the pixel's location in terms of height h and width w within a mask $\mathbf{M} \in \{0, \ldots, |\mathcal{O}|\}^{H_R \times W_R}$. For notation clarity, we exclude the temporal index t in this section, even though it remains implicitly present in our computations. As the number of classes $|\mathcal{C}|$ can vary over time (*i.e.*, from one video to another, or one frame to another in the same video), we normalize the entropy for consistency and comparability. Formally, we express the entropy of a pixel $x_{h,w}$ by

$$\mathbf{S}_{h,w} = -\frac{\sum_{o \in \mathcal{O}} p_{\mathcal{O}}(o \mid x_{h,w}) \log \left(p_{\mathcal{O}}(o \mid x_{h,w}) \right)}{\log(|\mathcal{O}|)}, \tag{6.2}$$

where $\mathbf{S} \in [0,1]^{H \times W}$ denotes the corresponding entropy map (*i.e.*, uncertainty map) of the current frame to segment.

To isolate the uncertainty for a specific object o, we use a dilated mask $\mathbf{M}_o^d$ based on the original object mask $\mathbf{M}_o$ predicted by the network. Here, the dilated mask allows us to exclude the background noise, while still considering the uncertainty around the object's edges[1]. We display on the left-hand side of Figure 6.3 the masked entropy results w.r.t. to the entropy map $\mathbf{S}$. Formally, we compute the total uncertainty for an object via the joint entropy $S^{\mathcal{R}_o}$ of a considered object region $\mathcal{R}_o = \{(h, w) \mid \mathbf{M}_o^d(h, w) = 1\}$, through

$$S^{\mathcal{R}_o} = \sum_{r \in \mathcal{R}_o} \mathbf{S}_r \left(x_r \mid x_{r-1}, \ldots, x_1 \right) \approx \sum_{r \in \mathcal{R}_o} \mathbf{S}_r, \tag{6.3}$$

where we essentially sum the conditional entropies of each random variable within the considered region. This approach captures the inter-dependencies among all variables, reflecting their collective impact on $\mathcal{R}_o$. However, computing the joint entropy is impractical as the network does not provide any joint or conditional distributions for a formal evaluation. Additionally, the computational cost would grow exponentially with respect to the number of classes $\mathcal{O}$ and the size of the region $\mathcal{R}_o$ (*i.e.*, $\mathrm{O}(|\mathcal{O}|^{|\mathcal{R}_o|})$ time

1 Subsequent experiments show that the use of the dilated mask yields similar results to the unmodified mask version. However, we retained it for consistency.

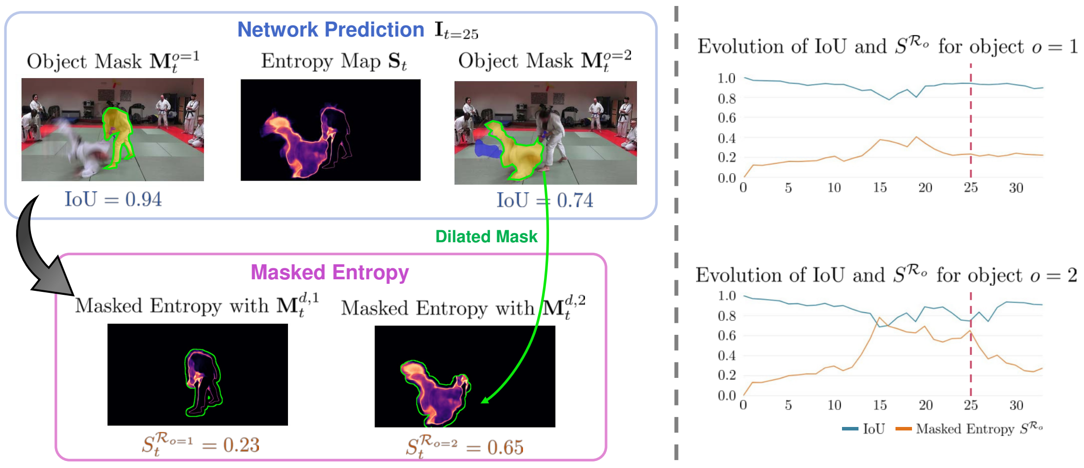

Network Prediction $\mathbf{I}_{t=25}$
Object Mask $\mathbf{M}_t^{o=1}$
Entropy Map $\mathbf{S}_t$
Object Mask $\mathbf{M}_t^{o=2}$
IoU = 0.94
IoU = 0.74
Dilated Mask
Masked Entropy
Masked Entropy with $\mathbf{M}_t^{d,1}$
Masked Entropy with $\mathbf{M}_t^{d,2}$
$S_t^{\mathcal{R}_{o=1}} = 0.23$
$S_t^{\mathcal{R}_{o=2}} = 0.65$
Evolution of IoU and $S^{\mathcal{R}_o}$ for object $o = 1$
Evolution of IoU and $S^{\mathcal{R}_o}$ for object $o = 2$
IoU
Masked Entropy $S^{\mathcal{R}_o}$

Figure 6.3: **Left**: Predicted object masks for frame $\mathbf{I}_{t=25}$ of the *judo* sequence [26]. Also shown is the associated entropy map $\mathbf{S}_t$ and its conversion to masked entropy maps for each object o present, to isolate the object-specific uncertainty. For mask predictions, the original boundary is shown in green, the predicted mask in yellow, and false negative regions in blue. The actual IoU and associated $S_t^{\mathcal{R}_o}$ values are provided for frame $t = 25$. **Right**: Evolution of the IoU per object w.r.t. $S_t^{\mathcal{R}_o}$.

complexity). To reduce the computational complexity, we assume zero mutual information between the predicted probability distributions of pixels in the region $\mathcal{R}_o$. This allows us to sum the entropy of each pixel $\mathbf{S}_{h,w}$ belonging to the region of interest $\mathbf{M}_o^d$ (refer to Figure 6.3), allowing us to significantly reduce the computational cost (*i.e.*, to $O(|\mathcal{O}| \times |\mathcal{R}_o|)$ complexity). Additionally, considering that the object size may vary from one image to another, we divide $S^{\mathcal{R}_o}$ by the size of the corresponding region $|\mathcal{R}_o|$. This dampens the fluctuation of $S^{\mathcal{R}_o}$ due to object size variations. As an example we display on the right-hand side of Figure 6.3 the evolution of the associated entropy of two objects w.r.t. to the actual IoU for a small sequence.

6.4.3 *Mask Refinement*

For the mask-refinement component, we rely on SAM-HQ [61], which extends SAM [17] to segment intricate object structures in more details, while preserving its zero-shot capabilities and flexibility. SAM-HQ [61] introduces two additional components on top of SAM [17]: (1) An *HQ-output token* to correct the original SAM's mask. (2) A *global-local features fusion*, which fuses early features with later ones (*i.e.*, after the first and last global attention block respectively) to enrich the features used by the mask decoder. For more details about SAM [17] and SAM-HQ [61] we refer the reader to the original sources.

6.4.4 *Issuing Corrections*

For a given object c, we record the corresponding masked entropy $S^{\mathcal{R}_o}$ at each frame, such that $\mathbf{s}_{\mathcal{R}_o} = \left[S_1^{\mathcal{R}_o}, \ldots, S_t^{\mathcal{R}_o}\right]$, where t denotes the latest frame processed by the method. Let $\mathbf{s}'_{\mathcal{R}_o}$ denote the derivative of $\mathbf{s}_{\mathcal{R}_o}$, such that $\mathbf{s}'_{\mathcal{R}_o} = \left[\Delta S_2^{\mathcal{R}_o}, \ldots, \Delta S_t^{\mathcal{R}_o}\right]$, and where $\Delta S_t^{\mathcal{R}_o} = S_t^{\mathcal{R}_o} - S_{t-1}^{\mathcal{R}_o}$. Depending on $\Delta S_t^{\mathcal{R}_o}$ we either generate a pseudo- or request a user-correction. Importantly, we intend through the use of pseudo-corrections and user-corrections to allow the method to detect when it is about to fail, preventing further degradation if left unaddressed. This assumes, that the method's predicted mask is still partially aligned with the object of interest.

User-Correction (U-C): We prompt a user correction whenever $\Delta S_t^{\mathcal{R}_o} \geq \tau_u$, where τ_u denotes the user threshold beyond which our approach requests a user interaction. The user indicates a foreground region of the object to track via a positive click, which is then processed by the mask refiner to generate a new mask. Note that the original mask is not used during refinement due to its high uncertainty. This provides the added benefit of simplifying the user's task during the initial correction phase, as they only need to consider a positive interaction rather than also managing negative ones. The user is then free to continue interacting in subsequent frames to improve the mask further if deemed necessary, through positive or negative interactions. Note that we follow the definition in iIOS [26], which denotes a positive or negative click interaction type as a point-wise indication of a falsely classified region as either foreground or background. To further ease the user's workload, we overlay the entropy map directly onto the processed image. This visual guide directs the user's attention towards the most uncertain regions in the model's predictions, facilitating efficient interaction.

Pseudo-Correction (P-C): Through pseudo-corrections, our goal is to allow for the method to self correct itself, without directly calling the user for help. By recognizing whenever the uncertainty is high enough, while also not exceeding τ_u, such that we call a pseudo-correction whenever $\tau_u > \Delta S_t^{\mathcal{R}_o} \geq \tau_p$, where τ_p denotes a threshold above which a pseudo-correction is generated. We argue that this intermediate state strikes a balance, where we enable the method to preemptively address growing uncertainty in its prediction and correct it's state before significant drift. As a result, this allows us to further reduce the number of user-corrections needed, as tabulated in Table 6.3, thereby improving the methods interaction efficiency. We generate a pseudo-correction p_t^o for object o given a frame t through

$$p_t^o = \underset{(h,w)}{\operatorname{argmax}}\big(\mathbf{M}_t^o \odot \mathbf{E}_t^o \odot (\,\mathbf{1}_{H_R \times W_R} - \mathbf{S}_t\,)\big), \quad \text{where} \tag{6.4}$$

$$\mathbf{E}_t^o(h,w) = \min_{(h_{\text{bound}}, w_{\text{bound}}) \in \Omega_t} \sqrt{(h - h_{\text{bound}})^2 + (w - w_{\text{bound}})^2}, \tag{6.5}$$

where Ω_t denotes the set of pixel that belong to the boundaries of the object mask $\mathbf{M}_t^o$ (dilated version), $\mathbf{E}_t^o \in [0,1]^{H_R \times W_R}$ a distance field, and $\odot$ represents the Hadamard

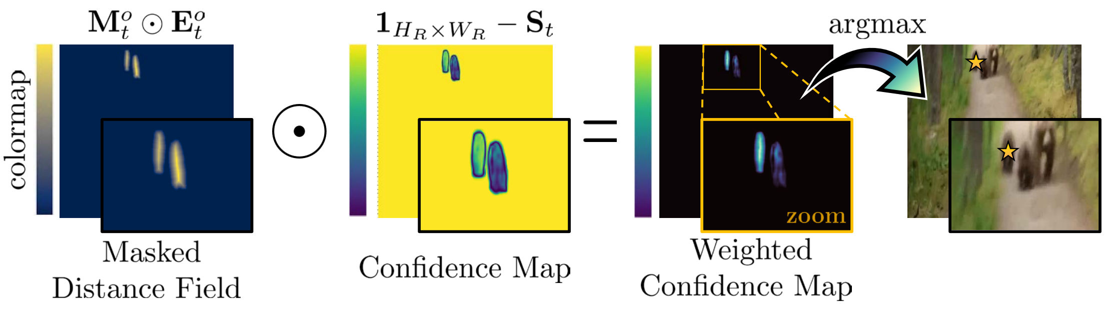

$\mathbf{M}_t^o \odot \mathbf{E}_t^o$
$\mathbf{1}_{H_R \times W_R} - \mathbf{S}_t$
argmax
colormap
zoom
Masked Distance Field
Confidence Map
Weighted Confidence Map

Figure 6.4: We depict the generation of a pseudo-correction, as defined in Equation (6.4). For improved visualization, we provide a zoomed-in view in the bottom-right corner of each heat-map and the image. Though the yellow star ★ we indicate the locations of the generated pseudo-correction.

product. We depict the intuition behind Equation (6.4) in Figure 6.4. The coordinates of the pseudo-correction are then given to the mask-refiner to produce a new object mask $\mathbf{M}_t^o$. We limit the generation of pseudo-corrections to one per target object. Also, note that these pseudo-corrections are inherently positive, as they are always generated within the boundaries of the previously predicted mask.

Essentially, we operate under the assumption that tracking will eventually fail. Our objective is to delay this failure for as long as possible before requiring user intervention. Hence, we leverage this intermediate state of generating pseudo-corrections (between no interaction and user interaction) to proactively mitigate escalating uncertainty.

6.4.5 *Interaction and Uncertainty Driven Memory Updates*

At each user-correction, we update the working memory of our sVOS baseline with the newly refined mask. This update strategy, termed *Interaction-Driven Update* (IDU), improves the method's robustness as the refined mask can influence the segmentation of the subsequent frames. An additional update mechanism, named *Uncertainty-Driven Update* (UDU), prevents updating the working memory with the original representation when the corresponding uncertainty $S^{\mathcal{R}_o}$, exceeds τ_p (similarly to QDMN [52]).

6.5 Lazy interactive Video Object Segmentation Metrics

Since we introduce ziVOS, we propose complementary metrics to the standard $\mathcal{J}\&\mathcal{F}$ presented by Perazzi *et al*. [82] to quantify the user's workload in providing on-the-fly corrections and to evaluate the robustness of a given method.

6.5.1 *Robustness Metric*

We take inspiration from Kristan *et al.* [92] and propose R@τ_{IoU} (higher is better) to measure the robustness of a method. Given a threshold value τ_{IoU}, we compute the ratio of frames, in which the predicted object mask attains an IoU above or equal to τ_{IoU} for all objects in a given dataset. More formally, let $\mathcal{O}$ be the set of objects in the dataset, where $\mathcal{T}_o$ is the set of index frames in which object o is present, we define $R@\tau_{\text{IoU}}$, such that

$$R@\tau_{\text{IoU}} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \frac{1}{|\mathcal{T}_o|} \sum_{t \in \mathcal{T}_o} \mathbb{1}_{[\text{IoU}(\mathbf{M}_t^o, \mathbf{G}_t^o) \geq \tau_{IoU}]}, \tag{6.6}$$

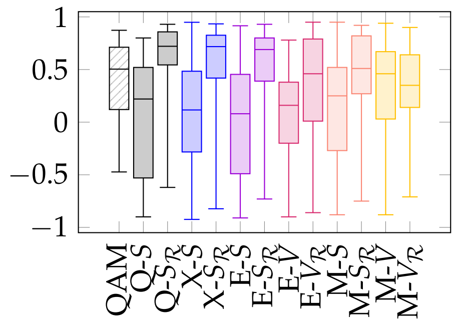
where $\mathbf{M}_t^o$ and $\mathbf{G}_t^o$ denote respectively the predicted mask and ground-truth annotation for object o at frame t. Like in [92], whenever the method correctly predicts the absence of an object we set $\mathbb{1}_{\text{IoU}(\mathbf{M}_t^o, \mathbf{G}_t^o) \geq \tau_{IoU}}$ to 1, otherwise to 0.

6.5.2 *User-Workload Metrics*

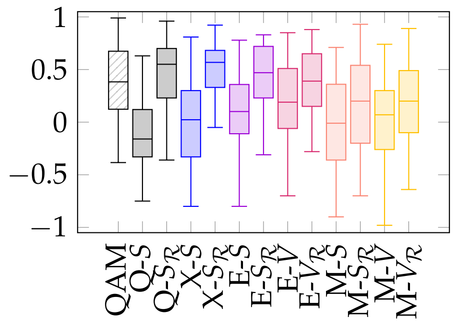
To quantitatively evaluate the workload for the user we introduce the following metrics: (1) Number of Correction (NoC) to denote the total number of user-corrections issued by the model to refine its current prediction. (2) Interaction Density Index (IDI) (higher is better), is introduced as an intuitive metric that reports the average time between two user-corrections reported in seconds. Note that some sequences might have no user interactions; however, to include every sequence in the evaluation, we consider the initialization and the end of a sequence as user-interactions. (3) As IDI does not reflect the underlying distribution of the interactions, we provide through Average Correction Interval (ACI) (which encapsulates both NoC and IDI) a score to indicate this distribution. In essence, we compute the cumulative count over user interactions and their respective distance to each other. Consequently, a low ACI score indicates more spread out user interactions, while a higher score indicates consecutive interactions more closer to each other within a short period. More formally, let $\mathcal{N}_o = \{t_{p=0}, \cdots, t_{p=P_o} \mid t_p \in \mathcal{T}_o\}$ denote the set containing the frame indexes t_p where a user prompt p is issued for object o. Hence, we define ACI, such that

$$\text{ACI} = \sum_{o \in \mathcal{O}} \frac{1}{|\mathcal{T}_o|} \sum_{i=1}^{|\mathcal{T}_o|} \sum_{j=1}^{i} n_j, \tag{6.7}$$

where $n_j = \sum_{t_p=1}^{\mathcal{N}_o} \mathbb{1}_{[j = t_p - t_{p-1}]}$ denotes the number of occurrences a user provided corrections at a distance of j frames from one prompt to the next.



(a) On DAVIS 2017 [83]



(b) On LVOS [91]

Figure 6.5: Comparison of correlation coefficients across the DAVIS 2017 [83] and LVOS [91] datasets: i) the QAM module [52] and entropy based QDMN [52] (as Q-S and Q-$S_{\mathcal{R}}$), ii) entropy results for a single baseline (X-S, X-$S_{\mathcal{R}}$), an ensemble (E-S, E-$S_{\mathcal{R}}$), and Monte-Carlo methods (M-S, M-$S_{\mathcal{R}}$), and iii) epistemic uncertainty variants for ensemble and Monte-Carlo (E-V, E-$V_{\mathcal{R}}$, M-V, M-$V_{\mathcal{R}}$).

6.6 Experiments

In Section 6.6.1, we assess the effectiveness of the proposed masked entropy $S^{\mathcal{R}_c}$ to estimate the tracking's state on-the-fly. We present the evaluation protocol for the ziVOS benchmark in Table 6.6.2, and present our results on the LVOS dataset [91] in Table 6.6.3. Note that all experiments are conducted on an NVIDIA GeForce GTX 1080 Ti.

6.6.1 *Entropy as a Proxy*

To evaluate the effectiveness of our entropy based solution to estimate the tracking's state, we compare against the following approaches: (1) Using the Quality-Aware Module (QAM) from QDMN [52], which predicts a confidence score through an auxiliary network. (2) Computing the entropy S and its masked version $S_{\mathcal{R}}$ for various models: single models denoted as Q and X respectively for the QDMN[52] and XMem [55] networks, an ensemble model denoted as E, and a Monte-Carlo dropout model denoted as M. (3) We also consider for the ensemble and Monte-Carlo dropout variants the epistemic uncertainty, denoted as V (with the masked version $V_{\mathcal{R}}$, similarly to $S_{\mathcal{R}}$). We refer the reader to Subsection 6.8.3 for the implementation details of the Ensemble and Monte-Carlo Dropout approaches.

For each method, we compute the Spearman coefficient [147] to measure the correlation between each variant's output to estimate its tracking state w.r.t. the actual IoU. We conduct our evaluations on the DAVIS 2017 [83] and LVOS [91] validation sets, featuring short and long videos respectively. Figure 6.5 presents the distribution (*i.e.*, box-plots) of the correlation coefficients when computing the coefficient for every object present in a dataset. Aside from the QAM based methods, we expect an inverse correlation, however, to facilitate the comparison, we invert the correlation results for all methods except for the QAM version. Consequently, values closer to 1 indicate a higher correlation, suggesting a more accurate estimate of the tracking state. Across both the DAVIS 2017 [83] and LVOS [91] datasets, variants employing masked entropy (*i.e.*, $S_{\mathcal{R}}$) demonstrate notably stronger correlations. This highlights the effectiveness of isolating uncertainty at the object level using a mask. Among the different model variants – single (Q and X), ensemble (E), and Dropout (M) – the single models (Q and X) leveraging $S_{\mathcal{R}}$ outperform even the advanced learning-based QAM module [52].

Hence, by examining Figure 6.5, the most effective method for estimating the tracking state on-the-fly appears to be the masked entropy approach, particularly the X-$S_{\mathcal{R}}$ variant, as its median value is closer to 1 and the distribution is notably narrower. This underscores the efficacy of masked entropy as a straightforward yet robust approach to estimate the tracking's state on-the-fly.

Throughout this research, we have developed a tool to assess the correlation between uncertainty estimation and accuracy of sVOS method. To foster further research into this direction we make our tool publicly available[2].

6.6.2 *Evaluation Protocol for ziVOS*

As our goal is to improve the robustness of video object segmentation methods by incorporating user corrections on-the-fly, while minimizing the user's workload, we only allow one interaction per object per frame. Moreover, we limit the types of interaction to only clicks, as pointing an object is the quickest and most intuitive interaction type for humans [105, 142]. As we incorporate interactions on the fly, the user has a limited time to interact with the frame. Moreover, the corrections mask cannot be improved iteratively on the same frame, hence if another correction is necessary, the user has to interact on a subsequent frame to improve the mask.

For practical reasons, we do not rely on real users during the ziVOS evaluation but rather on a *simulated agent*[3] to automatically simulate user input, similarly to the DAVIS 2017 [26] interactive robot[4]. To evaluate ziVOS methods, the agent simulates a positive user interaction u_t^o at frame t at the center of object o, whenever the ziVOS method requests a user intervention. This single interaction is then processed by the mask-refiner to generates a new mask. We adopt this simple interaction model for simulated corrections as this design prevents additional complexity in the evaluation protocol, and leverages the capabilities of SAM [17] models to accurately estimate the object of interest from a single interaction. In addition, this evaluation design ensures

2 https://github.com/Vujas-Eteph/ziVOS_Analyis
3 https://github.com/Vujas-Eteph/iVOTS_robot
4 https://github.com/albertomontesg/davis-interactive/tree/master/davisinteractive/robot

a clear and controlled setting, allowing us to assess our core contribution of improved robustness through proactive user interaction requests.

6.6.3 *Quantitative Results (Lazy interactive Video Object Segmentation)*

In Table 6.1, we present quantitative results of Lazy-XMem compared to SOTA methods on the LVOS validation set [91] by following the ziVOS evaluation process outlined in Figure 6.6.2. Hence, unlike sVOS, which relies on curated masks, we rely on click to indicate which object to track in the video. As a results, we employ imperfect masks generated by the mask refiner (*i.e.*, SAM-HQ [61]), which more closely resembles real-world scenarios. We provide additional results on LVOS [91] when following the protocol in sVOS in our supplementary material.

To allow for a better comparability, we also evaluate a modified version of QDMN [52], that adopts the same design as LazyXMem for integrating user and pseudo-corrections, with the notable exception that the tracking state estimation is based on the QAM module [52]. Moreover, we evaluate an alternative approach that simply requests user corrections at random intervals throughout the sequence (denoted as Rand-Lazy-XMem). Finally, we introduce Lazy-XMem†, a variant of Lazy-XMem, which operates *without* user corrections. Consequently, Lazy-XMem† constitutes an sVOS approach relying only on the (i) pseudo-interactions, (ii) UDU, (iii) IDU and (iv) mask refinement components. This facilitates direct comparison w.r.t. sVOS methods and allows us to verify our design choices for sVOS applications. We report the popular $\mathcal{J}\&\mathcal{F}$ metric, alongside our complementary metrics (see Section 6.5).

As shown in Table 6.1, our proposed Lazy-XMem† achieves competitive results w.r.t. to the SOTA sVOS methods. However the robustness is still close to the original XMem [55] version, despite of the increase in accuracy. By incorporating user corrections (*i.e.*, Lazy-XMem), we manage to improve the robustness by 13 points on average over all robustness metrics, while requesting in total 325 interactions from the user for the entire datasets, averaging one interaction every 18.4 seconds. This corresponds to approximately 1.05% of the total number of frames in the LVOS validation set, which contains $30,876$ frames [91].

While Lazy-XMem incorporates user corrections on-the-fly to enhance its robustness, it requires a continuous participation of the user throughout the segmentation process. Therefore, we present Lazy-XMem as an alternative to sVOS and iVOS methods to segment offline and online videos, in scenarios where user engagement is feasible and where segmenting over an extended period with high reliability is the priority.

6.6.4 *Quantitative Results (Perfect Mask Initialization)*

Table 6.2 reports the evaluation of sVOS and ziVOS methods on the LVOS validation set [91], using ground-truth annotations to indicate which object to segment in the

		Robustness			User-Workload		
Method	$\mathcal{J}\&\mathcal{F}$	R@0.1	R@0.25	R@0.5	ACI	NoC	IDI
		sVOS Methods					
QDMN[52] (ECCV 22)	44.2	47.8	45.5	36.2	-	-	-
XMem [55] (ECCV 22)	52.8	57.0	55.0	49.0	-	-	-
DEVA [148] (ICCV 23)	55.1	63.6	59.3	52.4	-	-	-
Cutie-base [141] (CVPR 24)	57.0	59.2	57.8	52.4	-	-	-
Cutie-small [141] (CVPR 24)	57.6	58.6	57.0	52.5	-	-	-
Lazy-XMem† (ours)	56.4	58.8	56.8	50.6	-	-	-
		ziVOS Methods					
Rand-Lazy-XMem (ours)	61.3	67.9	65.8	59.3	5.17	335	17.9
Lazy-QDMN (ours)	52.7	58.2	52.0	42.9	5.64	360	16.7
Lazy-XMem (ours)	**64.3**	**70.2**	**67.8**	**62.3**	5.02	**325**	18.4

Table 6.1: Quantitative evaluation of ziVOS and sVOS methods on the LVOS validation set [91] following the ziVOS framework. Here, we initialize each methods with an imperfect mask, in contrast to sVOS, to indicate which object to segment in the sequence.

sequence (as in sVOS). We re-evaluated each method, and compute the robustness metric $R@\tau_{IoU}$, expect for DDMemory [91] as the code is unavailable at the time of writing. Similarly to Table 6.1, Lazy-XMem† with only pseudo-interaction achieves competitive results to SOTA sVOS methods. However, by including user interactions on-the-fly to aid Lazy-XMem, we manage to improve the results robustness for the cost of 315 interactions (about 1.02% of the total number of frames in LVOS).

6.7 Ablations

6.7.1 *Design-Level Ablations*

To provide more insights into our pipeline, we detail the influence of each design choice in Section 6.3. Using the Uncertainty Driven Update (UDU), we achieve improvements over the baseline by selectively integrating memory predictions that present sufficiently low uncertainty. By soliciting user interactions to refine the initial mask predicted by the sVOS baseline (*i.e.*, XMem [55]), we achieve slight improvements at the cost of 507 interactions across the dataset. While, storing the refined masks as references for future segmentation after a user correction through the Interaction Driven Update (IDU), we attain substantial improvements in both robustness and user workload. However, using

		Robustness			User-Workload		
Method	$\mathcal{J}\&\mathcal{F}$	R@0.1	R@0.25	R@0.5	ACI	NoC	IDI
sVOS							
QDMN[52] (ECCV 22)	48.2	54.0	50.1	41.5	-	-	-
XMem [55] (ECCV 22)	53.7	54.6	51.7	41.3	-	-	-
DDMemory [91] (ICCV 23)	60.7	-	-	-	-	-	-
DEVA [148] (ICCV 23)	58.2	65.3	62.7	56.8	-	-	-
Cutie-base [141] (CVPR 24)	60.3	62.9	62.0	58.3	-	-	-
Cutie-small [141] (CVPR 24)	59.0	61.3	59.0	56.5	-	-	-
Lazy-XMem† (ours)	57.2	60.3	58.5	49.6	-	-	-
ziVOS							
Rand-Lazy-XMem (ours)	60.3	66.3	64.3	58.8	5.05	320	18.2
Lazy-XMem (ours)	63.5	70.0	68.3	63.1	4.86	315	18.9

Table 6.2: Quantitative evaluation of sVOS and ziVOS methods on the LVOS validation set [91], when initialized with the ground-truth annotations (curated masks as in sVOS).

the original mask from XMem [55], associated with a high uncertainty, as an additional prompt to the user's interaction for the mask refiner leads to a decrease in performance.

By generating pseudo-interactions following the strategy outlined in Section 6.4.4 to refine XMem's initial mask, we enhance the robustness even further while slightly reducing the user's workload. However, saving the resulting refined mask from a pseudo-interaction (pseudo-IDU), affect only marginally the robustness, but increases the user workload considerably. When discarding the user interactions and only relying on pseudo-corrections, *i.e.*, Lazy-XMem†, we obtain a similar setup to sVOS methods and manage to improve the results of the XMem [55] baseline, even attain competitive results against the current SOTA sVOS methods as shown in Section 6.1. Thus, our extension improves the baseline by: (1) discarding non-confident predictions from being added to the memory; (2) issuing pseudo-corrections to prompt SAM-HQ [61], thereby refining the baseline's initial prediction when the method's uncertainty increases sharply (Section 6.4.4); (3) and requests user-corrections on-the-fly as needed to improve the robustness.

6.7.2 *Decision Policies for Mask Refinement*

Table 6.4 tabulates the results when relying directly on the masked entropy $S^{\mathcal{R}_c}$ and its respective derivative $\Delta S^{\mathcal{R}_c}$ as a condition to request user help. To isolate the influence

Configuration											
	Pseudo		*User*			**Robustness**			**User-Workload**		
UDU	Corr.	IDU	Corr.	IDU	$\mathcal{J}\&\mathcal{F}$	R@0.1	R@0.25	R@0.5	ACI	NoC	IDI
-	-	-	-	-	52.8	57.0	55.0	49.0	-	-	-
✓	-	-	-	-	54.7	56.3	54.5	50.0	-	-	-
✓	✓	-	-	-	56.4	58.8	56.8	50.6	-	-	-
✓	✓	✓	-	-	53.1	57.0	55.1	49.6	-	-	-
✓	-	-	✓	-	55.6	58.2	56.4	51.8	7.80	507	12.6
✓	-	-	✓	✓	62.9	67.8	66.2	60.9	5.05	327	18.3
✓	✓	-	✓	✓	**64.3**	**70.2**	67.8	**62.3**	5.02	**325**	18.4
✓	✓	✓	✓	✓	**64.3**	70.1	**68.2**	62.1	5.91	352	17.3

Table 6.3: Ablation study for Lazy-XMem on the ziVOS framework. We initialize each method with an imperfect mask, to indicate which object to segment in the sequence.

each strategy for calling the user's help, we discard the mask refiner and the pseudo interaction. We only consider user interactions and rely directly on the ground-truth annotations to correct the model's predictions, instead of the mask refiner. We can see in Table 6.4, that both strategies enhance the robustness and the accuracy, especially when updating the memory of the sVOS baseline (XMem [55]) with the refined masks through the Interaction Driven Update (IDU). However, by issuing an interaction based on the derivative $S^{\mathcal{R}_c}$, we manage to significantly reduce the number of user calls from 787 to 327.

6.8 Qualitative Results

In this section we provide qualitative results that highlight both success and failure cases whenever Lazy-XMem issues either pseudo- or user-corrections to generate a refined mask. Figures 6.6 and 6.7 displays success and failure cases, respectively, for generating a refined mask through pseudo-corrections. While figures 6.8 and 6.9 show results when a refined mask is generated via a simulated user-correction, as described in Table 6.6.2.

We indicate a ground-truth mask in yellow, the original prediction in turquoise, the refined mask in orange or purple after a pseudo-and user-correction respectively. We mark the location of a pseudo- or user-corrections through a yellow star ☆.

For small objects, we provide a cropped version to better visualize the different predictions. In these cases, a small image of the original image is shown on the first column, surrounded by a red border. Note that in Figure 6.9, we do not display refined

		Robustness			User-Workload		
Configuration	$\mathcal{J}\&\mathcal{F}$	R@0.1	R@0.25	R@0.5	ACI	NoC	IDI
XMem [55] (baseline)	53.7	54.6	51.7	41.3	-	-	-
Call user corrections based on $S^{\mathcal{R}_c}$							
XMem + UDU	54.7	56.3	54.5	50.0	56.1	3647	1.9
XMem + UDU + IDU	63.5	67.6	66.1	61.7	12.1	787	8.5
Call user corrections based on $\Delta S^{\mathcal{R}_c}$							
XMem + UDU	55.6	58.2	56.4	51.8	7.80	507	12.6
XMem + UDU + IDU	62.9	67.8	66.2	60.9	5.05	327	18.3

Table 6.4: Results for Lazy-XMem when requesting user corrections through $S^{\mathcal{R}_c}$ or $\Delta S^{\mathcal{R}_c}$ (note that for this table we discard the pseudo-interaction). We initialize each method with perfect masks. UDU denotes Uncertainty Driven Update.

masks for the third, fourth and fifth rows, as Lazy-XMem missed for those instances the generation of either a user- or pseudo-corrections.

6.8.1 *Pseudo-Corrections*

Through the pixel wise uncertainty estimation, we are able to identify confusing and confident regions, helpful for the generation of pseudo-corrections, allowing us to correct the segmentation whenever a distractors is present and anticipate when the method is likely to fail as shown in Figure 6.6. We can observe that our proposed pseudo-correction generation strategy successfully recovers the original object of interest in the presence of distractors (*e.g.*, rows two, three, and four). Additionally, objects that are about to be lost are also recovered (*e.g.*, rows one, three, and five).

Note that for small objects (refer to Figure 6.7), the mask refinement incorrectly generates masks, although the pseudo-correction location's lies on the target, as seen in rows two, three, and five. In the first row, the small gorilla (target) is lost in favor to the adult gorilla, since the uncertainty is lower the method fails to issue correct pseudo-corrections or request a user-corrections. Ideally, the method should detect the transition from the small gorilla to the adult gorilla, while the pixel level uncertainty for both objects is still high, to indicate confusion. In row 6, we note that the pixel uncertainty for the foot region and the ball (target) are very similar, consequently the method is unable to find a correct location to generate a pseudo-correction as both object are as likely considered to be the actual object to track by the sVOS baseline, here the method failed to actually issue a user-correction.

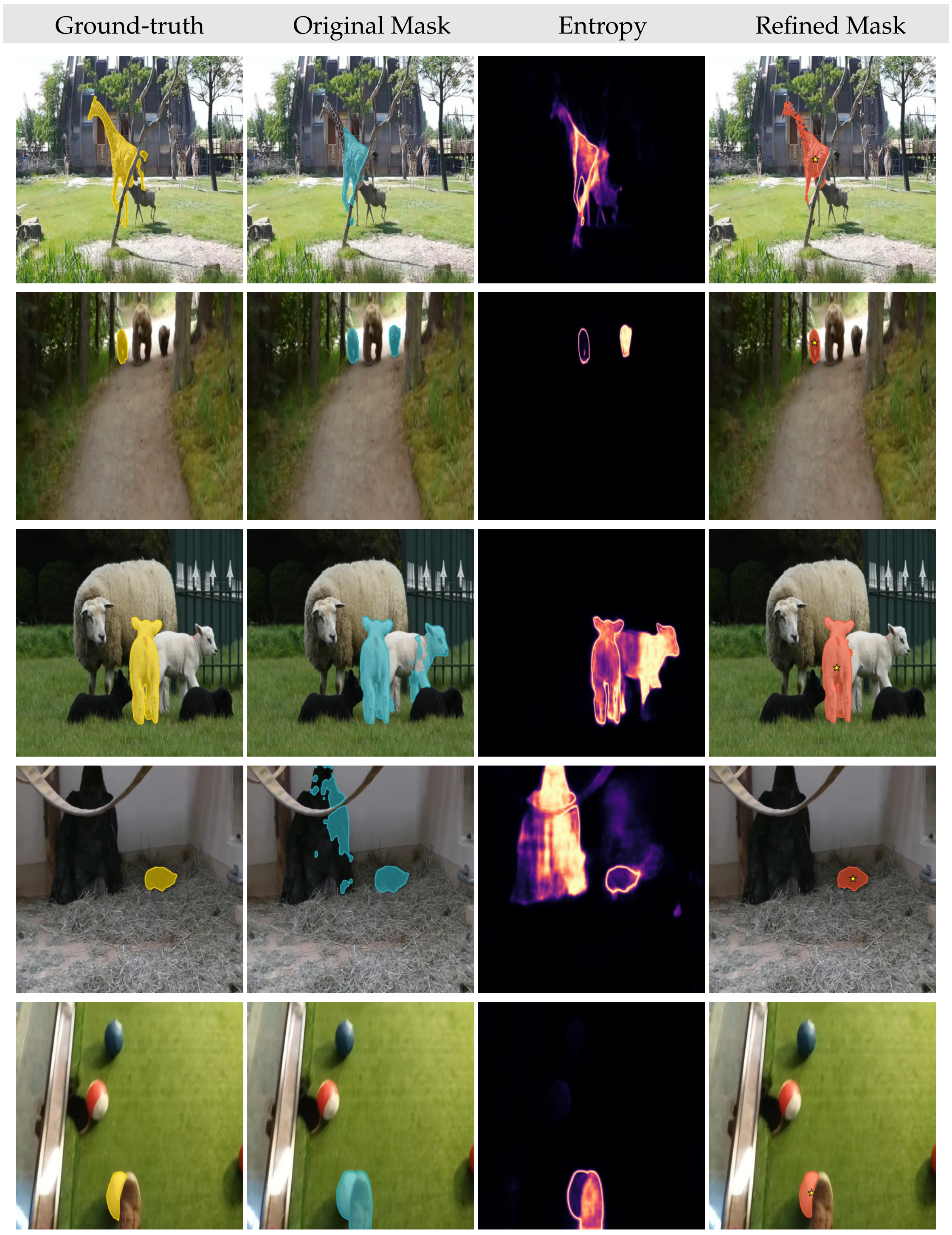


Figure 6.6: Qualitative results on the validation set of LVOS [91] when refining the mask through pseudo-corrections (Success cases).

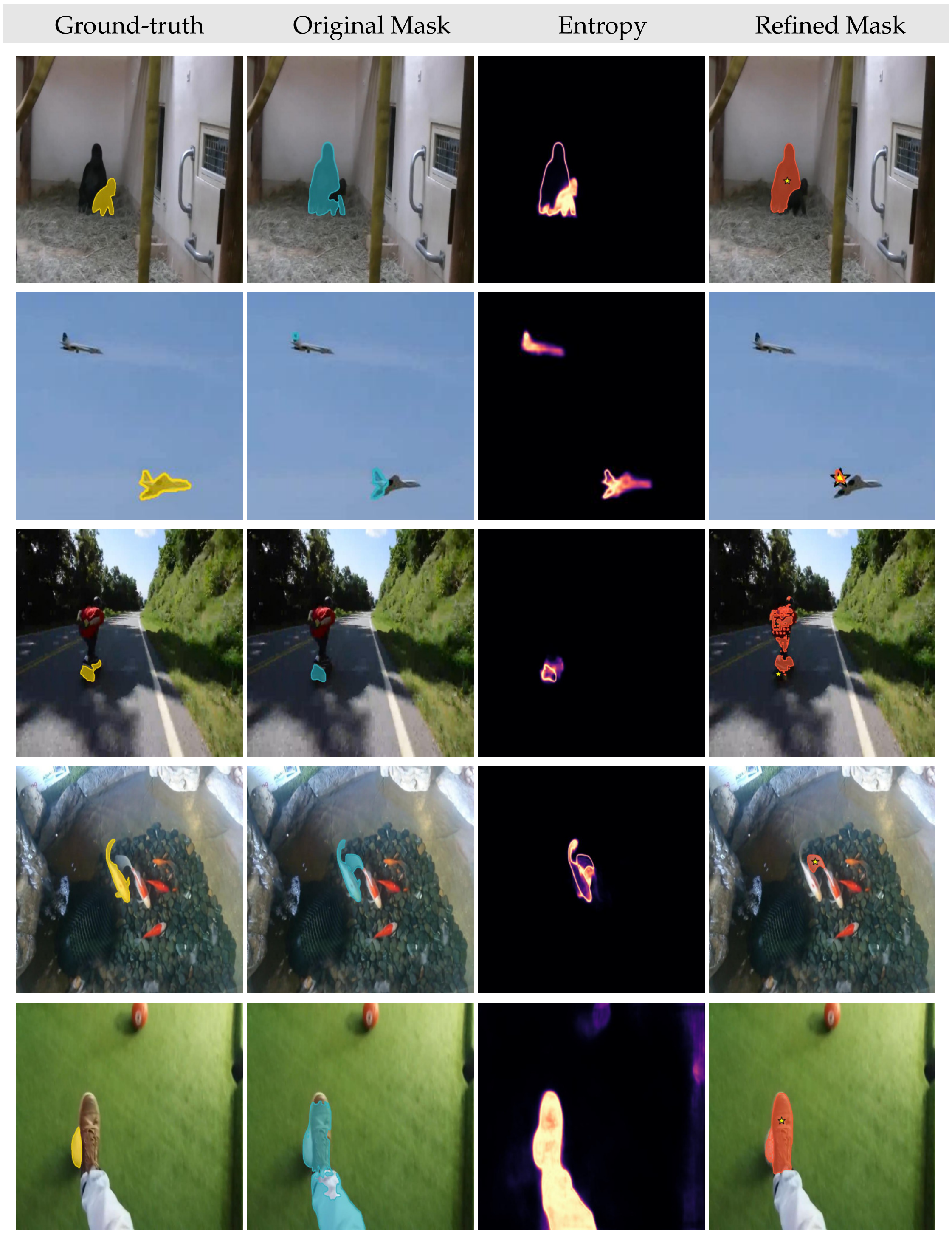


Figure 6.7: Qualitative results on the validation set of LVOS [91] when refining the mask through pseudo-corrections (Failure cases).

As seen in figure Figure 6.6, the actual object of interest, is the little gorilla, however, during the training process, the method identifies the big gorilla as the target of interests, as seen in the entropy map.

6.8.2 *User-Corrections*

In the first and last rows of Figure 6.8, we note that the method correctly issues a user interactions, as only the ear of the sheep and the back of the zebra are still segmented, preventing the loss of the target. Similarly, in the second and third rows, the method manages to issue and interaction to the user while losing the target in favor to a distractors. Note that in the third row, the method correctly issues a user-correction instead of a pseudo-correction, as otherwise the pseudo-correction would be generated on the wrong sheep.

In 6.9, we observe that the method sometimes unnecessarily calls for user interaction even when a good portion of the object is correctly predicted (*i.e.*, first and second row), and where a pseudo-correction would be more appropriate (first row).

Additionally, there are instances where a user (or pseudo) correction is missed, as seen in rows three, four and five. In the fourth row, the tracker confidently segments a distractor after the disappearance of the object of interest, while indicating the actual object with some uncertainty. Lastly, when the SVOS backbone loses track of the object of interest, it is unable to recover it, as shown in the fifth row.

6.8.3 *Implementation Details*

For our sVOS baseline, we rely on the original weights provided by the authors of XMem [55], which is trained on the static and DAVIS 2017 training set [83]:

Deep Ensemble variant: We experiment with an ensemble approach that combines three independently trained XMem models (each initialized with a different seed). The first model is trained on the static [48] and DAVIS 2017 training set [83]. The second model (which we use as a baseline in Lazy-XMem) is trained similarly to the first model but also includes the synthetic dataset BL30K [47]. The third model is trained like the first model but with the addition of the MOSE [85] dataset. Note that for computational reason, the ensemble size as limited to three models.

Monte Carlo (MC) variant: We rely on spatial pooling [149] applied to the key-projection of XMem [55], with a dropout ratio of 0.2 for our Monte Carlo Dropout variant during training, which is maintained during inference. Hence, during inference we generate 20 stochastic forward passes per frame. The resulting probability maps are averaged to compute the final segmentation result.

We seek to isolate the epistemic uncertainty, as it encapsulates the divergence in the predicted probabilities across the different models forming the ensemble, as it provides a measure of model disagreement. To quantify the epistemic (model) uncertainty for

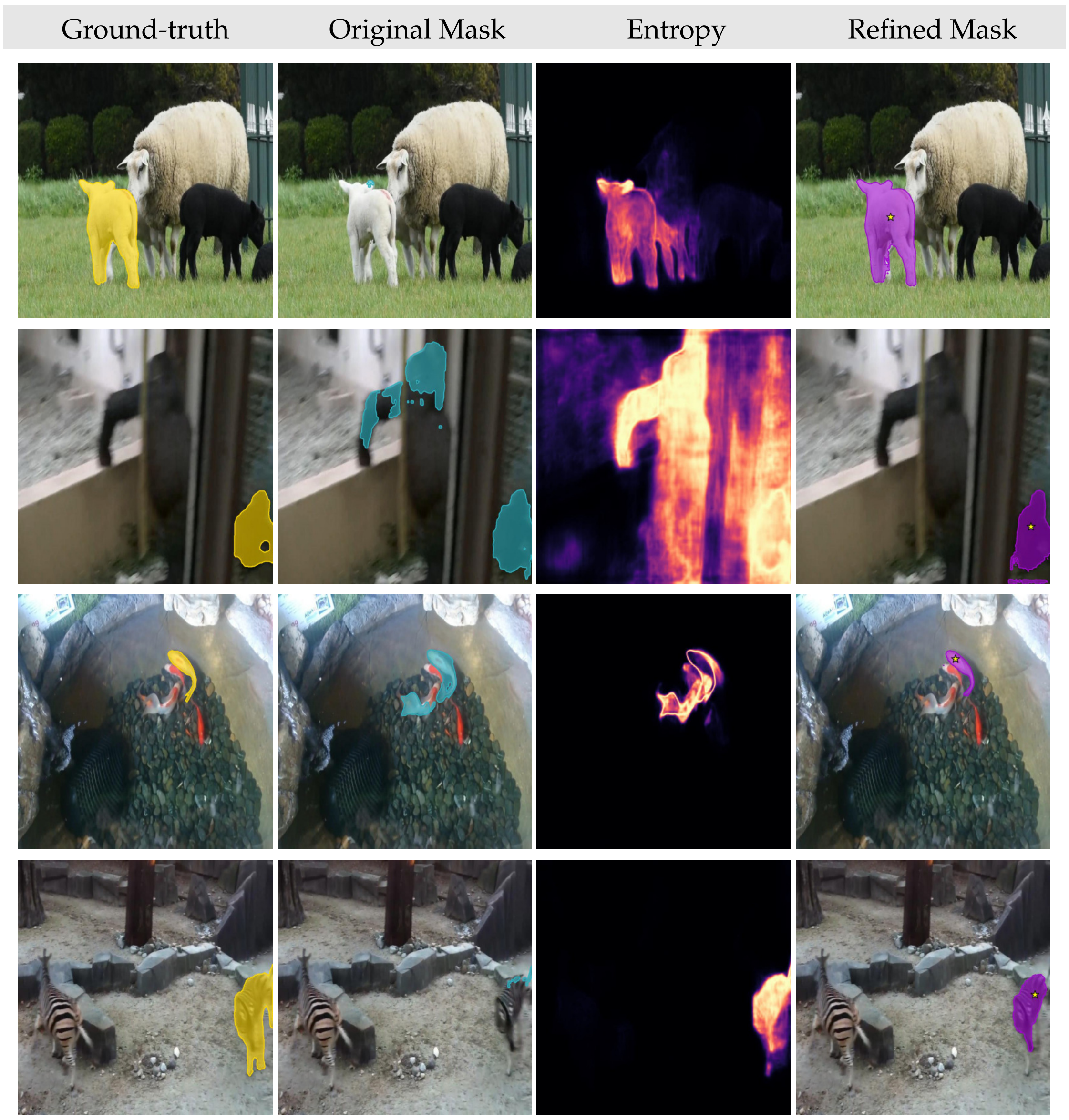


Figure 6.8: Qualitative results on the validation set of LVOS [91] when refining the mask through user-corrections (Success cases).

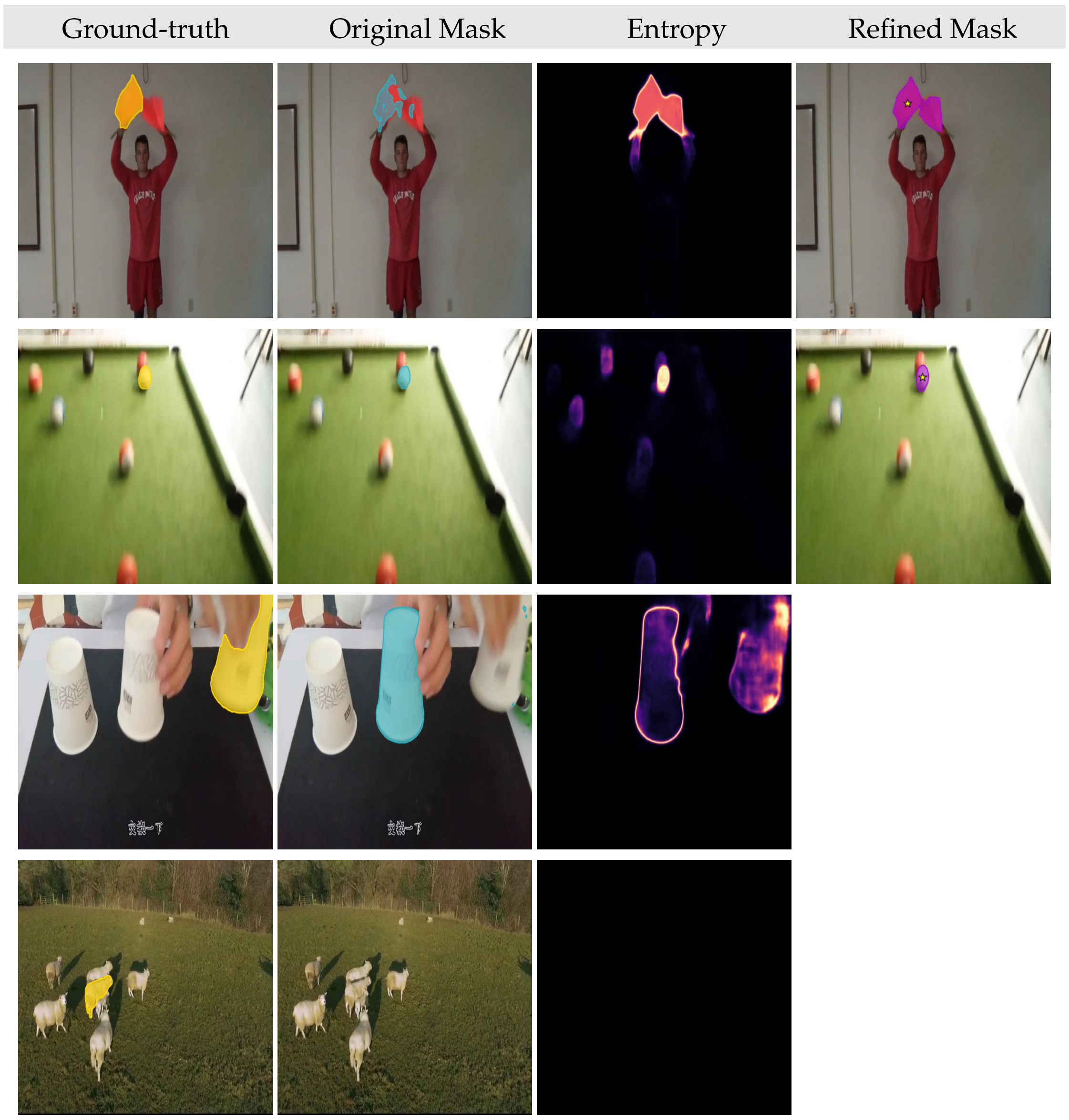


Figure 6.9: Qualitative results on the validation set of LVOS [91] when refining the mask through user-corrections (Failure and miss cases). Here we considered a missed opportunity to generate a pseudo- or user-correction whenever the IoU between the original prediction and the ground-truth annotation is below 0.1. These missed opportunities explain the absence of entries in the Refined Mask columns for the last two rows.

our Monte Carlo Dropout (MCD) and Deep Ensemble variant, we decompose the total predictive uncertainty $\mathbf{S}_t$ into its aleatoric (data) and epistemic components. Intuitively, to extract the epistemic uncertainty, we consider the total uncertainty of the Ensemble or MCD model (encapsulating aleatoric and epistemic uncertainty), and subtract the aleatoric uncertainty to it [150]. The idea is that my having multiple models providing a prediction for the same input, the aleatoric uncertainty, which is only related to the data, should be unchanged from one model to another. In contrast, since the epistemic uncertainty is model-specific it will vary between each model. By averaging the uncertainty of each individual model, these model-specific variations are down-weighted, leaving behind the aleatoric uncertainty. Hence, by subtracting the average uncertainty of each model from the total uncertainty of the whole ensemble, we isolate the epistemic component. Formally, we compute the epistemic uncertainty $\mathbf{V}_t$ for frame t through $\mathbf{V}_t = \mathbf{S}_t - \mathbb{E}[\mathbf{S}_t^m]$ where $\mathbf{S}_t$ is the total uncertainty, computed as the entropy of the ensemble's average probability maps $\mathbf{P}_t$, such that $\mathbf{P}_t = \mathbb{E}[\mathbf{P}_t^m]$. And $\mathbb{E}[\mathbf{S}_t^m]$ denotes the aleatoric uncertainty, where $\mathbf{S}_t^m$ designates the entropy of each model individually such that $\mathbf{S}_t^m$ is only computed from $\mathbf{P}_t^m$. Here the exponent m denotes the model index.

Thresholds: We using the training set of the LVOS dataset [91] to identify the values for $\tau_u = 0.5$, $\tau_p = 0.2$ and $\tau_m = 0.8$. Note that none of the networks used in this work is trained on the LVOS training set.

6.9 Limitations

Currently, Lazy-XMem generates only click-based pseudo-corrections, which are then given to the mask-refiner without including the predicted mask, as the empirical results were not convincing. This is due to some masks being too inaccurate to serve as a viable prompt for the refinement module. However, this also disregards the impact that a slightly inaccurate mask could provide to boost the refinement. Another aspect could also be linked to the inherent bottleneck that SAM-based models do not work well with mask-based prompts in practice. Instead, an alternative approach, explored by Delatolas *et al.* [80], involves iteratively prompting the mask-refiner with pseudo-prompts generated from the initial mask until a certain level of alignment is achieved between the SAM-predicted mask and the original sVOS initial mask. For instance, an IoU reaching 0.8 between the SAM predicted mask and the original mask. However, this method assumes that the initial mask (from the sVOS pipeline) is accurate enough to serve as a reliable base for further prompting the mask-refiner with uncertainty-based prompts. An aspect to take into account in future evaluations is the simulation of user corrections. Currently we generate these by essentially computing the central coordinates of the ground-truth mask. However, as recently shown by Antonov *et al.* [157], real user interactions are rarely centrally located and tend to be more dispersed. This discrepancy introduces a performance drop, highlighting the need for a more diverse training strategy for user correction generation in refinement models.

Our pseudo-correction (P-C) mechanism currently generates corrections and requests user interaction exclusively within the dilated object mask. This serves as a pre-emptive measure, effective before the object of interest is entirely lost. If the object is already lost, the method cannot generate pseudo-corrections or prompt the user for help. However, it can still recover independently because the stored memory embeddings retain a representation of when the object was present. As indicated in Subsection 6.4.4, our current P-C mechanism is restricted to positive pseudo-corrections within the current object mask, a strategy that has shown improvement over the baseline (refer Table 6.1 and 6.2). To incorporate negative pseudo-corrections in order to cover regions outside the predicted mask, future work should also consider the uncertainty outside the predicted masks. This could enable the method to identify and act upon confusing objects (*i.e.*, distractors) and potentially update it's memory representation in accordance, to prevent drift. In addition, extending the P-C mechanism to include negative pseudo-interactions would also diversify its ability to resolve ambiguous situations, such as over-segmentation where background pixels are incorrectly included in the object's mask.

An important consideration for future research is to complement our proposed objective user-workload metrics (see Subsection 6.5) by integrating a subjective assessment from the user's perspective. Here, the NASA Task Load Index (NASA-TLX) [144] offers a well-established methodology to evaluate mental, physical, temporal, and other workloads on the user. In addition, we can also leverage the System Usability Scale (SUS) [145] to provide a complementary measure of perceived usability.

6.10 Discussion

In this chapter, we introduced the *ziVOS* subtask, a hybrid combination of sVOS and iVOS specifically designed for online, unconstrained scenarios where on-the-fly corrections by a human user are feasible. This setting reflects real-world applications in which video frames are processed sequentially, and continuous monitoring is impractical, but maintaining robust long-term object tracks remains essential.

To address this challenge, we presented Lazy-XMem, a baseline that extends the capabilities of XMem [55] by dynamically integrating both pseudo and user corrections whenever the system detects high uncertainty. This mechanism provides an effective balance between segmentation performance and user workload since corrections are only requested during critical events, such as occlusions or distractors. Leveraging an entropy-based measure of uncertainty on pixel-level, our method estimates the tracking state online and triggers interactions only when they are needed to avert substantial errors. The pixel-wise uncertainty estimation serves a dual purpose: it facilitates efficient user guidance to problematic regions (easier to visualize problematic regions), thus streamlining interactions, and it is also employed to generate pseudo-corrections.

In addition, this entropy based regulation, allows us also to extend on if we would update the memory or not of our approach, allowing us to be more robust and allow a better scalability (RQ2) Building on the standard the $\mathcal{J}\&\mathcal{F}$ metrics [82], we proposed complementary metrics to evaluate the robustness of the segmentation (*i.e.*, the proportion of frames meeting a minimum IoU threshold) and the user workload associated with provided corrections on-the-fly. Experiments on the long-term LVOS dataset [91] confirmed that Lazy-XMem improves the robustness of the baseline, albeit at the cost of extra interactions. Overall, Lazy-XMem serves as a reference point for practical, online VOS solutions where maintaining persistent object tracks is often more critical than achieving the highest accuracy in a single pass.

In summary, this chapter jointly advances **RQ1** (by reducing the user's monitoring effort through proactive uncertainty estimation and pseudo corrections) and **RQ2** (by demonstrating scalable, long-term tracking in an online setting). The code is publicly available[5] to promote reproducibility and further developments.

5 https://github.com/Vujas-Eteph/LazyXMem

Part III

CONCLUSION

7 SUMMARY AND OUTLOOK

7.1 Summary

We address throughout this thesis the central research question *How can we design a scalable, robust, and user-efficient VOS approach for unconstrained video sequences that ensures minimal user effort?* Our goal is to enhance the robustness of VOS methods by designing a Human-in-the-loop approach that permits users to interact on-the-fly with the segmentation in a single pass while simultaneously minimizing the required effort.

To identify potential avenues, we decomposed our research question into two sub-questions: **RQ1**: How can we design an efficient interactive VOS approaches that reduce the user's effort? and **RQ2**: How can we scale VOS methods to robustly handle unconstrained video sequences (*i.e.*, long, diverse, or streaming videos) in a Human-in-the-loop environment?

Our answer builds upon two complementary paradigms: semi-supervised VOS (sVOS) and interactive VOS (iVOS). While sVOS methods propagate (user-provided) reference masks with no further intervention after initialization, iVOS methods rely on an iterative user feedback loop to progressively refine segmentation output. By fusing insights from both paradigms, we aim to achieve a low annotation effort on the user's side, alongside an efficient scalable approach that allows us to robustly track on pixel-level an arbitrary object in unconstrained videos.

In Chapter 2, we extensively reviewed state-of-the-art methods in sVOS and iVOS, summarizing prevalent approaches and standard benchmarks (*i.e.*, datasets). Moreover, we also highlighted critical shortcomings w.r.t. our central research question. In Chapter 3, we presented fundamental concepts underpinning matching-based sVOS architectures, as they form the backbone of this thesis. We detail key architectural elements (*e.g.*, encoders, decoders, and space-time memory matching), describe the training process (*i.e.*, leveraging synthetic video clips), and review essential evaluation metrics for assessing the performances of VOS methods.

We partially addressed **RQ1** in Chapter 4 by replacing scribble interactions with sparse, click-based inputs within the iVOS context to effectively reduce annotations effort from the user's side while still achieving reasonable performance compared to scribbles-based approaches. However, this proof-of-concept (**CiVOS**) was limited to short, pre-recorded videos, where multiple reviews and iterative interactions are feasible.

In Chapter 5 we specifically target the scalability concerns of **RQ2**. We tackled a prevalent limitation in existing sVOS methods, which were tailored for short video sequences due to their ever-growing memories. We proposed **READMem**, a novel plug-

and-play module (*i.e.*, training-free) for efficiently managing the memory aspect of SOTA matching-based sVOS approaches. Concretely, we selectively stored frame embeddings based on a diversity criterion. We demonstrate performance boost across significantly longer sequences while maintaining high accuracy in short videos.

Finally, in Chapter 6, we jointly addressed **RQ1** and **RQ2** in an online (*i.e.*, hypothetical streaming) scenario where video rewinding is impossible, sequences are long, and continuous user monitoring is impractical. To formalize this setting, we introduced the ziVOS task, aiming for a Human-in-the-loop setup that operates within a single segmentation pass. We proposed a baseline, **Lazy-XMem**, which assesses its pixel-level uncertainty on-the-fly during the segmentation process. This allows us to strategically request user interactions precisely when the uncertainty is high. Furthermore, to reduce user effort, Lazy-XMem generates pseudo-corrections when the uncertainty is moderately high, enabling the method to self-correct. To evaluate the robustness, we introduced complementary metrics that specifically capture the segmentation robustness and the user workload. Our evaluations confirm Lazy-XMem's effectiveness in minimizing user interventions while maintaining robust segmentation performance under our new sub-task.

7.2 Outlook

Let us now discuss interesting directions that could be addressed in future work. With the introduction of SAM [17] and more recently SAM2 [16], which demonstrated substantial approximately 10% performance gains over previous methods on popular benchmarks, the VOS field is currently experiencing a significant shift towards leveraging foundational models. Mirroring the impact of architectures like ResNet [126] and VGG [151], SAM2, in less than a year, serves as the primary backbone for most VOS applications, ranging from the medical field, robotics, to track. In addition to SAM2, Rave *et al.* [16] also introduced a new subtask: Promptable Video Object Segmentation (PVOS), which is closely related to our proposed subtask in ziVOS, allowing users to correct the segmentation process in a single pass.

However, areas for improvement remain, such as memory management, explicit handling of distractors, and pixel-level uncertainty estimation. In particular, the memory module, which currently relies on a FIFO approach, could be enhanced. Recent studies have explored better memory management strategies for matching-based approaches, which SAM2 is based on. Notably, QDMN [52], READMem [128], and RMem [139] (winner of the VOT2024 challenge) offer promising directions. Works, like SAM2Long [66], have already started to address this aspect specifically. Similarly, DAM4SAM [140] are exploring memory management solutions to better deal with distractors.

As noted by [52, 66, 152], the quality of stored embeddings can be proxied by estimating the uncertainty of a predicted mask, giving us an indication of the prediction's reliability to serve as a reference for future predictions. While some approaches estimate uncertainty at the frame level [16, 17, 52], a finer-grained (pixel-level) uncertainty estimation would allow us to identify which regions within a frame should be avoided when saving to memory. Hence, instead of predicting a scalar confidence value, auxiliary networks or modules could be designed to output pixel-wise confidence heatmaps to allow also the generation of pseudo-corrections (as in Chapter 6) to improve a method's performance. More specifically, one promising direction would be to train an auxiliary network to predict false positive and false negative maps as proposed by Qin *et al.* [153] for the iIOS task but extended for sVOS. These maps could then be directly used to generate pseudo-positive and pseudo-negative user corrections, potentially leading to more effective self-correction during segmentation. An additional alternative to Monte Carlo, Ensemble or Laplace Approximations would also be to rely on evidential deep learning [154] perspective. Here, instead of directly predicting Bernoulli parameters for each pixel (in the single-object tracking scenario), the network could predict the parameters of its conjugate prior, the Beta distribution (or the Dirichlet distribution for multi-object tracking) [155]. This would allow modeling the probability distribution of a pixel belonging to the foreground more explicitly, offering a richer uncertainty representation, particularly for out-of-distribution scenarios [156].

This explicit uncertainty modeling could be further integrated with active and lifelong learning strategies. When the method detects high uncertainty in its predictions, it could actively solicit user feedback for refinement and update its internal state accordingly. For instance, future work could explore explicitly fine-tuning the SAM2 backbone, adapting its weights or auxiliary weights based on user feedback, similar to the online adaptation strategies employed in BRS [110, 111].

Importantly, future work should focus on comprehensive user studies to asses from a user's perspective the practicability of VOS approaches for the long-term and streaming scenarios. Especially since the VOS approaches work in tandem with the user. This includes conducting comprehensive user studies through NASA-TLX [144] or SUS [145] as well as testing multiple types (*i.e.*, natural language descriptions, clicks) of interactions separately or in a combination, to identify the most adapted interaction form for particular events or use-cases. The objective would be to evaluate which modalities are perceived as: (i) *Easiest to use*: Assessing learnability and efficiency for various user groups; (ii) *Most reliable*: Achieving consistently the desired result; (iii) *Least cognitively demanding*: Estimating the sustained attention demand, decision-making frequency; and (iv) *Requiring minimal physical effort*: Evaluating the ergonomic aspects of interaction.

Finally, a major limitation of interactive approaches lies in the absence of realistic user interaction models for evaluation and potentially for training. While standard interactive segmentation benchmarks often rely on simplified inputs like center clicks, these do not fully capture the nuances of real human behavior. To address this, Antonov *et al.* [157] recently proposed R-Clicks, a network that predicts more realistic user interactions. The authors show that in realistic scenarios, networks tend to learn a bias toward central interactions and display a drop in performance when exposed to real user interactions.

Integrating such insights into future research is crucial for developing and evaluating VOS methods that better align with real-world user behavior. This could significantly influence the design of both the interaction mechanisms and the learning strategies of future systems.

BIBLIOGRAPHY

[1] *Fortune Business Insight Video On Demand Market 2024*. Available at: https://www.fortunebusinessinsights.com/industry-reports/video-on-demand-market-100140. Last accessed: 10 March 2025.

[2] *ESPN Subscriber Statistics Q1 2025*. Available at: https://www.statista.com/statistics/1054451/espn-plus-subscriber-us/. Last accessed: 09 March 2025.

[3] Statista. *Hours of video uploaded to YouTube every minute as of 2023*. https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/. Accessed: October 10, 2023. 2023.

[4] *Video Editing Software Market Statistics 2023*. Available at: https://sendshort.ai/statistics/video-editing-software/. Last accessed: 09 March 2025.

[5] *Adobe Premiere Pro Statistics 2024*. Available at: https://sendshort.ai/statistics/premiere-pro/. Last accessed: 09 March 2025.

[6] *DaVinci Resolve Statistics 2023*. Available at: https://sendshort.ai/statistics/davinci-resolve/. Last accessed: 09 March 2025.

[7] *Apple iMovie: Revenue and Growth Statistics*. Available at: https://sendshort.ai/statistics/imovie/. Last accessed: 21 April 2025.

[8] Tianfei Zhou, Fatih Porikli, David J Crandall, Luc Van Gool, and Wenguan Wang. "A Survey on Deep Learning Technique for Video Segmentation." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023).

[9] John Smith and Jane Doe. "Advancements in Video Object Segmentation for Surveillance Systems." In: *International Journal of Computer Vision (IJCV)* (2021).

[10] *fortunebusinessinsights: Video Surveillance Market Size*. Available at: https://www.fortunebusinessinsights.com/video-surveillance-market-102673. Last accessed: 13 March 2025.

[11] Emily Johnson and Li Wang. "Video Object Segmentation in Livestock Monitoring Systems: A Review." In: *Computers and Electronics in Agriculture* (2022).

[12] Oded Berger-Tal, Tal Polak, Aya Oron, Yael Lubin, Burt P Kotler, and David Saltz. "Integrating Animal Behavior and Conservation Biology: A Conceptual Framework." In: *Behavioral Ecology* (2011).

[13] Si Liu, Tianrui Hui, Shaofei Huang, Yunchao Wei, Bo Li, and Guanbin Li. "Cross-modal progressive comprehension for referring segmentation." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2021).

[14] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. “The 2018 DAVIS Challenge on Video Object Segmentation.” In: *Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)* (2018).

[15] Rui Yao, Guosheng Lin, Shixiong Xia, Jiaqi Zhao, and Yong Zhou. “Video object segmentation and tracking: A survey.” In: *ACM Transactions on Intelligent Systems and Technology (TIST)* (2020).

[16] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. “SAM 2: Segment Anything in Images and Videos.” In: *International Conference on Learning Representations (ICLR)* (2025).

[17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. “Segment Anything.” In: *International Conference on Computer Vision (ICCV)*. 2023.

[18] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. “One-shot video object segmentation.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[19] Paul Voigtlaender and Bastian Leibe. “Online adaptation of convolutional neural networks for video object segmentation.” In: *British Machine Vision Conference (BMVC)* (2017).

[20] K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. “Video object segmentation without temporal information.” In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2018).

[21] Huaxin Xiao, Jiashi Feng, Guosheng Lin, Yu Liu, and Maojun Zhang. “Monet: Deep motion exploitation for video object segmentation.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[22] Huaxin Xiao, Bingyi Kang, Yu Liu, Maojun Zhang, and Jiashi Feng. “Online meta adaptation for fast video object segmentation.” In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).

[23] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. “PReMVOS: Proposal-generation, Refinement and Merging for Video Object Segmentation.” In: *Asian Conference on Computer Vision (ACCV)*. 2018.

[24] Tim Meinhardt and Laura Leal-Taixé. “Make one-shot video object segmentation efficient again.” In: *Neural Information Processing Systems (NeurIPS)* (2020).

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In: *International Conference on Computer Vision (ICCV)*. 2017.

[26] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. "Learning video object segmentation from static images." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[27] Won-Dong Jang and Chang-Su Kim. "Online video object segmentation via convolutional trident network." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[28] Ping Hu, Gang Wang, Xiangfei Kong, Jason Kuen, and Yap-Peng Tan. "Motion-guided cascaded refinement network for video object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[29] Lu Zhang, Zhe Lin, Jianming Zhang, Huchuan Lu, and You He. "Fast video object segmentation via dynamic targeting network." In: *International Conference on Computer Vision (ICCV)*. 2019.

[30] Joakim Johnander, Martin Danelljan, Emil Brissman, Fahad Shahbaz Khan, and Michael Felsberg. "A generative appearance model for end-to-end video object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[31] Varun Jampani, Raghudeep Gadde, and Peter V. Gehler. "Video Propagation Networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[32] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. "Lucid data dreaming for video object segmentation." In: *International Journal of Computer Vision (IJCV)* (2019).

[33] Hai Ci, Chunyu Wang, and Yizhou Wang. "Video Object Segmentation by Learning Location-Sensitive Embeddings." In: *European Conference on Computer Vision (ECCV)*. 2018.

[34] Xi Chen, Zuoxin Li, Ye Yuan, Gang Yu, Jianxin Shen, and Donglian Qi. "State-Aware Tracker for Real-Time Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[35] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. "Feelvos: Fast end-to-end embedding learning for video object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[36] Zongxin Yang, Yunchao Wei, and Yi Yang. "Collaborative video object segmentation by foreground-background integration." In: *European Conference on Computer Vision (ECCV)*. 2020.

[37] Zongxin Yang, Yunchao Wei, and Yi Yang. "Collaborative Video Object Segmentation by Multi-Scale Foreground-Background Integration." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2021).

[38] Zongxin Yang, Yunchao Wei, and Yi Yang. "Associating Objects with Transformers for Video Object Segmentation." In: *Neural Information Processing Systems (NeurIPS)*. 2021.

[39] Zongxin Yang and Yi Yang. "Decoupling Features in Hierarchical Propagation for Video Object Segmentation." In: *Neural Information Processing Systems (NeurIPS)*. 2022.

[40] Suhwan Cho, Heansung Lee, Minhyeok Lee, Chaewon Park, Sungjun Jang, Minjung Kim, and Sangyoun Lee. "Tackling background distraction in video object segmentation." In: *European Conference on Computer Vision (ECCV)*. 2022.

[41] Mao Yunyao, Wang Ning, Zhou Wengang, and Li Houqiang. "Joint Inductive and Transductive Learning for Video Object Segmentation." In: *International Conference on Computer Vision (ICCV)*. 2021.

[42] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. "Video object segmentation using space-time memory networks." In: *International Conference on Computer Vision (ICCV)*. 2019.

[43] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, and Wenxiu Sun. "Efficient regional memory network for video object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[44] Hongje Seong, Junhyuk Hyun, and Euntai Kim. "Kernelized memory network for video object segmentation." In: *European Conference on Computer Vision (ECCV)*. 2020.

[45] Hongje Seong, Seoung Wug Oh, Joon-Young Lee, Seongwon Lee, Suhyeon Lee, and Euntai Kim. "Hierarchical Memory Matching Network for Video Object Segmentation." In: *International Conference on Computer Vision (ICCV)*. 2021.

[46] Yong Liu, Ran Yu, Jiahao Wang, Xinyuan Zhao, Yitong Wang, Yansong Tang, and Yujiu Yang. "Global spectral filter memory network for video object segmentation." In: *European Conference on Computer Vision (ECCV)*. 2022.

[47] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. "Modular Interactive Video Object Segmentation: Interaction-to-Mask, Propagation and Difference-Aware Fusion." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[48] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. "Rethinking Space-Time Networks with Improved Memory Coverage for Efficient Video Object Segmentation." In: *Neural Information Processing Systems (NeurIPS)*. 2021.

[49] Kwanyong Park, Sanghyun Woo, Seoung Wug Oh, In So Kweon, and Joon-Young Lee. "Per-Clip Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[50] Yu Li, Zhuoran Shen, and Ying Shan. "Fast video object segmentation using the global context module." In: *European Conference on Computer Vision (ECCV)*. 2020.

[51] Yongqing Liang, Xin Li, Navid Jafari, and Jim Chen. "Video Object Segmentation with Adaptive Feature Bank and Uncertain-Region Refinement." In: *Neural Information Processing Systems (NeurIPS)*. 2020.

[52] Yong Liu, Ran Yu, Fei Yin, Xinyuan Zhao, Wei Zhao, Weihao Xia, and Yujiu Yang. "Learning quality-aware dynamic memory for video object segmentation." In: *European Conference on Computer Vision (ECCV)*. 2022.

[53] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. "Mask Scoring R-CNN." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[54] Mingxing Li, Li Hu, Zhiwei Xiong, Bang Zhang, Pan Pan, and Dong Liu. "Recurrent dynamic embedding for video object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[55] Ho Kei Cheng and Alexander G. Schwing. "XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model." In: *European Conference on Computer Vision (ECCV)*. 2022.

[56] Richard C Atkinson and Richard M Shiffrin. "Human memory: A proposed system and its control processes." In: *Psychology of learning and motivation*. 1968.

[57] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches." In: *arXiv preprint* (2014).

[58] Yang Jinyu, Gao Mingqi, Li Zhe, Gao Shang, Wang Fangjing, and Zheng Fengm. "Track Anything: Segment Anything Meets Videos." In: *arXiv preprint* (2023).

[59] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. "Segment and Track Anything." In: *arXiv preprint* (2023).

[60] Jiawen Zhu, Zhenyu Chen, Zeqi Hao, Shijie Chang, Lu Zhang, Dong Wang, Huchuan Lu, Bin Luo, Jun-Yan He, Jin-Peng Lan, Hanyuan Chen, and Chenyang Li. "Tracking Anything in High Quality." In: *arXiv preprint* (2023).

[61] Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan Liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. "Segment Anything in High Quality." In: *Neural Information Processing Systems (NeurIPS)*. 2023.

[62] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. "Fast Segment Anything." In: *arXiv preprint* (2023).

[63] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. "Faster Segment Anything: Towards Lightweight SAM for Mobile Applications." In: *arXiv preprint* (2023).

[64] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. "Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories." In: *European Conference on Computer Vision (ECCV)*. 2022.

[65] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. "Segment Anything Meets Point Tracking." In: *Winter Conference on Applications of Computer Vision (WACV)* (2025).

[66] Shuangrui Ding, Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Yuwei Guo, Dahua Lin, and Jiaqi Wang. "SAM2Long: Enhancing SAM 2 for Long Video Segmentation with a Training-Free Memory Tree." In: *arXiv preprint arXiv:2410.16268* (2024).

[67] Cheng-Yen Yang, Hsiang-Wei Huang, Wenhao Chai, Zhongyu Jiang, and Jenq-Neng Hwang. *SAMURAI: Adapting Segment Anything Model for Zero-Shot Visual Tracking with Motion-Aware Memory*. 2024.

[68] Tinghuai Wang, Bo Han, and John Collomosse. "TouchCut: Fast image and video segmentation using single-touch interaction." In: *Conference on Computer Vision and Image Understanding (CVIU)* (2014).

[69] Suyog Jain and Kristen Grauman. "Click carving: Segmenting objects in video with point clicks." In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2016.

[70] Arnaud Benard and Michael Gygli. "Interactive video object segmentation in the wild." In: *arXiv preprint* (2017).

[71] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. "Deep interactive object selection." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[72] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. "Fast User-Guided Video Object Segmentation by Interaction-And-Propagation Networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[73] Yuk Heo, Yeong Jun Koh, and Chang-Su Kim. "Interactive Video Object Segmentation Using Global and Local Transfer Modules." In: *European Conference on Computer Vision (ECCV)*. 2020.

[74] Jiaxu Miao, Yunchao Wei, and Yi Yang. "Memory Aggregation Networks for Efficient Interactive Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[75] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. “Modular Interactive Video Object Segmentation: Interaction-to-Mask, Propagation and Difference-Aware Fusion.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[76] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. “Blazingly fast video object segmentation with pixel-wise metric learning.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[77] Bowen Chen, Huan Ling, Xiaohui Zeng, Jun Gao, Ziyue Xu, and Sanja Fidler. “Scribblebox: Interactive annotation framework for video object segmentation.” In: *European Conference on Computer Vision (ECCV)*. 2020.

[78] Yuk Heo, Yeong Jun Koh, and Chang-Su Kim. “Guided Interactive Video Object Segmentation Using Reliability-Based Attention Maps.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[79] Zhaoyuan Yin, Jia Zheng, Weixin Luo, Shenhan Qian, Hanling Zhang, and Shenghua Gao. “Learning to Recommend Frame for Interactive Video Object Segmentation in the Wild.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[80] Thanos Delatolas, Vicky Kalogeiton, and Dim P Papadopoulos. “Learning the What and How of Annotation in Video Object Segmentation.” In: *Winter Conference on Applications of Computer Vision (WACV)*. 2024.

[81] Stéphane Vujasinović, Sebastian Bullinger, Stefan Becker, Norbert Scherer Negenborn, Michael Arens, and Rainer Stiefelhagen. “Revisiting Click-Based Interactive Video Object Segmentation.” In: *International Conference on Image Processing (ICIP)*. 2022.

[82] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. “A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[83] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. “The 2017 DAVIS Challenge on Video Object Segmentation.” In: *Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)* (2017).

[84] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. “Youtube-vos: A large-scale video object segmentation benchmark.” In: *arXiv preprint* (2018).

[85] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. “MOSE: A New Dataset for Video Object Segmentation in Complex Scenes.” In: *International Conference on Computer Vision (ICCV)*. 2023.

[86] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip Torr, and Song Bai. "Occluded Video Instance Segmentation: A Benchmark." In: *International Journal of Computer Vision (IJCV)* (2022).

[87] Pavel Tokmakov, Jie Li, and Adrien Gaidon. "Breaking the "Object" in Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[88] Maksym Bekuzarov, Ariana Bermudez, Joon-Young Lee, and Hao Li. "XMem++: Production-level Video Segmentation From Few Annotated Frames." In: *International Conference on Computer Vision (ICCV)*. 2023.

[89] Ali Athar, Jonathon Luiten, Paul Voigtlaender, Tarasha Khurana, Achal Dave, Bastian Leibe, and Deva Ramanan. "BURST: A Benchmark for Unifying Object Recognition, Segmentation and Tracking in Video." In: *Winter Conference on Applications of Computer Vision (WACV)*. 2023.

[90] Lingyi Hong, Zhongying Liu, Wenchao Chen, Chenzhi Tan, Yuang Feng, Xinyu Zhou, Pinxue Guo, Jinglun Li, Zhaoyu Chen, Shuyong Gao, Wei Zhang, and Wenqiang Zhang. "LVOS v2: A Benchmark for Large-scale Long-term Video Object Segmentation." In: *arXiv preprint* (2024).

[91] Lingyi Hong, Wenchao Chen, Zhongying Liu, Wei Zhang, Pinxue Guo, Zhaoyu Chen, and Wenqiang Zhang. "LVOS: A Benchmark for Long-term Video Object Segmentation." In: *International Conference on Computer Vision (ICCV)*. 2023.

[92] Matej Kristan, Jiří Matas, Martin Danelljan, Michael Felsberg, Hyung Jin Chang, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, Zhongqun Zhang, Khanh-Tung Tran, Xuan-Son Vu, Johanna Björklund, Christoph Mayer, Yushan Zhang, Lei Ke, Jie Zhao, Gustavo Fernández, et al. "The First Visual Object Tracking Segmentation VOTS2023 Challenge Results." In: *International Conference on Computer Vision Workshop (ICCV-W)*. 2023.

[93] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. "Fast video object segmentation by reference-guided mask propagation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[94] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. "Hierarchical Image Saliency Detection on Extended CSSD." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2016).

[95] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. "Learning to Detect Salient Objects with Image-level Supervision." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[96] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. "FSS-1000: A 1000-Class Dataset for Few-Shot Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[97] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. “Towards High-Resolution Salient Object Detection.” In: *International Conference on Computer Vision (ICCV)*. 2019.

[98] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. “CascadePSP: Toward Class-Agnostic and Very High-Resolution Segmentation via Global and Local Refinement.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[99] Diederik P Kingma and Jimmy Lei Ba. “Adam: A method for stochastic gradient descent.” In: *International Conference on Learning Representations (ICLR)*. 2015.

[100] Tijmen Tieleman and Geoffrey Hinton. *Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude*. Course lecture, *Neural Networks for Machine Learning*. Available online at http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. 2012.

[101] Reza Azad, Moein Heidary, Kadir Yilmaz, Michael Hüttemann, Sanaz Karimijafarbigloo, Yuli Wu, Anke Schmeink, and Dorit Merhof. “Loss Functions in the Era of Semantic Segmentation: A Survey and Outlook.” In: *arXiv preprint* (2023).

[102] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations.” In: *International Conference on Medical Image Computing and Computer Assisted Intervention Workshop (MICCAI-W)*. 2017.

[103] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. “What’s the point: Semantic segmentation with point supervision.” In: *European Conference on Computer Vision (ECCV)*. 2016.

[104] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. “Large-scale interactive object segmentation with human annotators.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[105] Herbert H Clark. “Coordinating with each other in a material world.” In: *Discourse studies* (2005).

[106] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. “Regional interactive image segmentation networks.” In: *International Conference on Computer Vision (ICCV)*. 2017.

[107] Sabarinath Mahadevan, Paul Voigtlaender, and Bastian Leibe. “Iteratively Trained Interactive Segmentation.” In: *British Machine Vision Conference (BMVC)*. 2018.

[108] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. “Interactive image segmentation with latent diversity.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[109] Gwangmo Song, Heesoo Myeong, and Kyoung Mu Lee. "Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[110] Won-Dong Jang and Chang-Su Kim. "Interactive Image Segmentation via Backpropagating Refinement Scheme." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[111] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. "f-BRS: Rethinking backpropagating refinement for interactive segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[112] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. "Interactive Image Segmentation With First Click Attention." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[113] Theodora Kontogianni, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. "Continuous adaptation for interactive object segmentation by learning from corrections." In: *European Conference on Computer Vision (ECCV)*. 2020.

[114] Konstantin Sofiiuk, Ilya A Petrov, and Anton Konushin. "Reviving iterative training with mask guidance for interactive segmentation." In: *International Conference on Image Processing (ICIP)*. 2022.

[115] Marco Forte, Brian Price, Scott Cohen, Ning Xu, and François Pitié. "Getting to 99% accuracy in interactive segmentation." In: *arXiv preprint* (2020).

[116] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. "Deep extreme cut: From extreme points to object segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[117] Camille Dupont, Yanis Ouakrim, and Quoc Cuong Pham. "UCP-Net: Unstructured Contour Points for Instance Segmentation." In: *International Conference on Systems, Man, and Cybernetics (SMC)* (2021).

[118] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. "Interactive object segmentation with inside-outside guidance." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[119] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "" GrabCut" interactive foreground extraction using iterated graph cuts." In: *ACM transactions on graphics (TOG)* (2004).

[120] Jiajun Wu, Yibiao Zhao, Jun-Yan Zhu, Siwei Luo, and Zhuowen Tu. "Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[121] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. "Deep interactive object selection." In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[122] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. "Deep high-resolution representation learning for visual recognition." In: 2020.

[123] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. "Segmentation transformer: Object-contextual representations for semantic segmentation." In: *European Conference on Computer Vision (ECCV)*. 2021.

[124] Zdravko Marinov, Paul F Jäger, Jan Egger, Jens Kleesiek, and Rainer Stiefelhagen. "Deep Interactive Segmentation of Medical Images: A Systematic Review and Taxonomy." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2024).

[125] Chunhui Zhang, Li Liu, Yawen Cui, Guanjie Huang, Weilin Lin, Yiqian Yang, and Yuehong Hu. "A Comprehensive Survey on Segment Anything Model for Vision and Beyond." In: *arXiv preprint* (2023).

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[127] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. "The 2019 DAVIS Challenge on VOS: Unsupervised Multi-Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition Workshop (CVPR-W)* (2019).

[128] Stephane Vujasinovic, Sebastian Bullinger, Stefan Becker, Norbert Scherer Negenborn, Michael Arens, and Rainer Stiefelhagen. "READMem: Robust Embedding Association for a Diverse Memory in Unconstrained Video Object Segmentation." In: *British Machine Vision Conference (BMVC)*. 2023.

[129] Zhihui Lin, Tianyu Yang, Maomao Li, Ziyu Wang, Chun Yuan, Wenhao Jiang, and Wei Liu. "SWEM: Towards Real-Time Video Object Segmentation with Sequential Weighted Expectation-Maximization." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[130] Axel Sauer, Elie Aljalbout, and Sami Haddadin. "Tracking Holistic Object Representations." In: *British Machine Vision Conference (BMVC)*. 2019.

[131] Ėrnest Borisovich Vinberg. *A course in algebra*. American Mathematical Soc., 2003.

[132] Tadeusz Banachiewicz. "Méthode de résolution numérique des équations linéaires, du calcul des déterminants et des inverses, et de réduction des formes quadratiques." In: *Bull. Intern. Acad. Polon. Sci. A* (1938).

[133] Commandant Benoit. "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky)." In: *Bulletin géodésique* (1924).

[134] Nils Barth. "The gramian and k-volume in n-space: some classical results in linear algebra." In: *Journal of Young Investigators* (1999).

[135] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. "The eighth visual object tracking VOT2020 challenge results." In: *European Conference on Computer Vision Workshop (ECCV-W)*. 2020.

[136] Matej Kristan, Jiří Matas, Aleš Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Cehovin, Alan Lukežič, Ondrej Drbohlav, Jani Käpylä, Gustav Häger, Song Yan, Jinyu Yang, Zhongqun Zhang, Gustavo Fernández, et al. "The ninth visual object tracking VOT2021 challenge results." In: *International Conference on Computer Vision Workshop (ICCV-W)*. 2021.

[137] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, et al. "The tenth visual object tracking VOT2022 challenge results." In: *European Conference on Computer Vision Workshop (ECCV-W)*. 2022.

[138] Harold W Kuhn. "The Hungarian method for the assignment problem." In: *Naval research logistics quarterly* (1955).

[139] Junbao Zhou, Ziqi Pang, and Yu-Xiong Wang. "RMem: Restricted Memory Banks Improve Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[140] Jovana Videnovic, Alan Lukezic, and Matej Kristan. "A Distractor-Aware Memory for Visual Object Tracking with SAM2." In: *CVPR*. 2025.

[141] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. "Putting the Object Back into Video Object Segmentation." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[142] Chaz Firestone and Brian J Scholl. ""Please tap the shape, anywhere you like" shape skeletons in human vision revealed by an exceedingly simple measure." In: *Psychological science* (2014).

[143] Claude Elwood Shannon. "A mathematical theory of communication." In: *The Bell system technical journal* (1948).

[144] Sandra G Hart and Lowell E Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research." In: *Human mental workload* (1988).

[145] John Brooke et al. “SUS-A quick and dirty usability scale.” In: *Usability evaluation in industry* (1996).

[146] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning phrase representations using RNN encoder-decoder for statistical machine translation.” In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014).

[147] Charles Spearman. “The Proof and Measurement of Association between Two Things.” In: *American Journal of Psychology* (1904).

[148] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. “Tracking Anything with Decoupled Video Segmentation.” In: *International Conference on Computer Vision (ICCV)*. 2023.

[149] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. “Efficient object localization using convolutional networks.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[150] Andrey Malinin. “Uncertainty estimation in deep learning with application to spoken language assessment.” PhD thesis. 2019.

[151] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” In: *International Conference on Learning Representations (ICLR)*. 2015.

[152] Stéphane Vujasinović, Stefan Becker, Sebastian Bullinger, Norbert Scherer Negenborn, Michael Arens, and Rainer Stiefelhagen. “Strike the Balance: On-the-Fly Uncertainty based User Interactions for Long-Term Video Object Segmentation.” In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2024.

[153] Qin Liu, Meng Zheng, Benjamin Planche, Srikrishna Karanam, Terrence Chen, Marc Niethammer, and Ziyan Wu. “PseudoClick: Interactive image segmentation with click imitation.” In: *European Conference on Computer Vision (ECCV)*. 2022.

[154] Murat Sensoy, Lance Kaplan, and Melih Kandemir. “Evidential Deep Learning to Quantify Classification Uncertainty.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[155] Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-HUng Vu, Mathieu Cord, and Patrick Pérez. “Confidence Estimation via Auxiliary Models.” In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2021.

[156] Charles Corbière, Marc Lafon, Nicolas Thome, Matthieu Cord, and Patrick Pérez. “Beyond First-Order Uncertainty Estimation with Evidential Models for Open-World Recognition.” In: *ICML Workshop on Uncertainty and Robustness in Deep Learning*. 2021.

[157] Anton Antonov, Andrey Moskalenko, Denis Shepelev, Alexander Krapukhin, Konstantin Soshin, Anton Konushin, and Vlad Shakhuro. “RClicks: Realistic Click Simulation for Benchmarking Interactive Segmentation.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.