OD-VOS: Object Detection for Video Object Segmentation

Omer Benharush
University of Michigan
Ann Arbor, MI
omerb@umich.edu

Sarah Chan
University of Michigan
Ann Arbor, MI
sjchan@umich.edu

Jack Kernan
University of Michigan
Ann Arbor, MI
jrkernan@umich.edu

Max Rucker
University of Michigan
Ann Arbor, MI
mruck@umich.edu

***Abstract*—Video object segmentation (VOS) often requires manual intervention for selecting object masks, which can be impractical for various applications in robotics. In our project, OD-VOS, we propose a novel approach that integrates Vision Transformer for Open-World Localization (OWL-ViT) with XMem. OWL-ViT is an object detection network that can identify objects based on text queries, and XMem is a VOS framework which efficiently stores memory inspired by the Atkinson-Shiffrin memory model. OD-VOS uses OWL-ViT to automate the mask selection process for XMem. Our method eliminates the need for manual mask selection, thus streamlining the segmentation pipeline and reducing human effort. By automating the mask selection process, our framework enhances the applicability of VOS techniques. The project page is available at: https://deeprob.org/w24/reports/group8. The code is available at: https://github.com/jrkernan/ROB498FinalProject-XMem.**

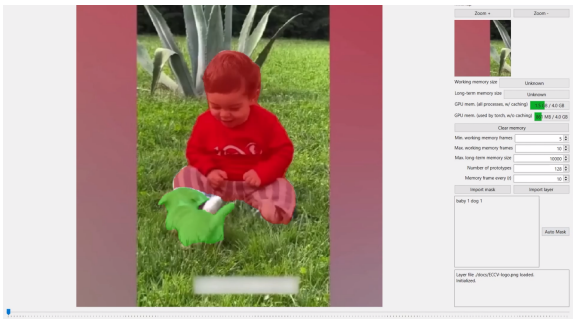

Fig. 1. An example of OD-VOS being used to mask a video with a baby and a puppy.

I. Introduction

VOS is a computer vision task that involves separating specific objects of interest from the background in a video sequence. Contrary to traditional image segmentation, which involves static images, VOS focuses on segmenting objects across multiple frames in a video.

XMem is a VOS model which addresses the complications of memory constraints in other VOS techniques. Unlike traditional VOS methods, XMem leverages a memory mechanism to store and retrieve information from previous frames. This memory mechanism allows the model to learn and remember object appearances and motion patterns over time, facilitating accurate and consistent segmentation results. By incorporating contextual information from past frames, XMem enhances the model's ability to handle long-term dependencies, ensuring robust performance across varied video lengths. Additionally, XMem's efficient memory utilization significantly reduces computational resources needed for traditional VOS models [1].

The traditional approach to VOS involves supervised mask selection where the user is required to manually select the object they wish to track. One disadvantages of this is the requirement of human intervention, which leads to an inefficient, time consuming process to run VOS models. This prevents usage of VOS in applications which require object tracking in real-time.

Google's OWL-ViT is a query-based object detection model that uses Vision Transformers (ViT). The model allows users to enter names of objects and the model will identify the objects and create a bounding box around them. [2].

To address the limitation of supervised mask selection in XMem, we created OD-VOS. By integrating the query based input from OWL-ViT into the GUI for XMem, OD-VOS presents a streamlined process for running the XMem model. The user is now only required to type the name of one or multiple objects they would like XMem to track. OD-VOS uses OWL-ViT to detect the location of the object(s) indicated by the user which feed into the XMem interactive GUI. This is beneficial because it promotes real-time processing of video streams, enabling OD-VOS to be used in robotics applications such as video surveillance and autonomous navigation, where timely analysis is crucial.

In this report, we will discuss existing work relating to VOS, our findings from reproducing the XMem model, our method for integrating OWL-ViT with XMem, and final remarks and ideas for future work.

II. Related Work

Referring video object segmentation (R-VOS) is a method of supervised VOS which uses language expressions. The goal of R-VOS is to be able to use language queries to identify and segment objects in a video. Some previous R-VOS models include GLEE [3], DsHmp [4], and ReferFormer [5].

GLEE [3] provides an object-level for object detection in images and videos. Its particularly strong aspects include its ability to handle multi-modal (video, image, and query) inputs and its zero-shot capabilities. DsHmp [4] focuses on improving

VOS by separating the perception of static elements from hierarchical motion language cues. ReferFormer [5] offers a more user-friendly version of VOS which interprets natural language queries, allowing the user to provide more descriptive context for the object they would like to mask, similar to our project.

However, the existing R-VOS models were not designed to handle long videos. They struggle with scalability which hinders their ability to provide accurate and reliable segmentation results over the entire length of long videos. This limitation undermines the utility of R-VOS in real-world applications where users need to analyze and track objects across lengthy sequences. We aim to develop a model that can input language cues for VOS while also performing well across diverse video lengths.

III. Algorithmic Recreation

For the start of this project, we had to recreate and implement XMem for our video object segmentation. XMem is a VOS algorithm specifically designed for long-term videos, leveraging an architecture similar to the Atkinson-Shiffrin memory model having sensory, working, and long-term memory. Each of these three pieces of memory work together to influence the masking of the selected object while keeping memory usage minimal. The architecture of XMem can be split up into two main sections: feature extraction and memory storage. A high-level overview can be seen in Figure 2.

A. Feature Extraction

Feature extraction is how XMem identifies objects in each video frame. The architecture of this section includes the encoders, query, readout features, and decoder. The encoder is based on the Resnet-50 model that reads in the current frame and sends that to the query (q) which is another convolutional neural network to extract the features from the current frame. Once the current features are extracted, they are then combined with the features saved in the working and long-term memory. These saved features are extracted using a key-value pairing to call upon each piece of memory. Once combined, the feature combination is sent as the readout features (F) which is decoded along with the sensory memory and upscaled to create the output mask.

B. Memory Storage

After the mask is created, each memory system has to be updated accordingly. The sensory memory is the first to be updated. This is done by taking the output mask and running it through a Resnet-18 encoder. Once encoded, the data is passed through a Gated Recurrent Unit (GRU) to create the sensory memory model that is used in the decoder to create the mask. This is updated every frame as a sort of short-term memory, but is reset every r frames to avoid holding onto sensory information too long. Additionally, every r frames the mask from the current frame is saved to the working memory. This gives the model a longer memory than just sensory,

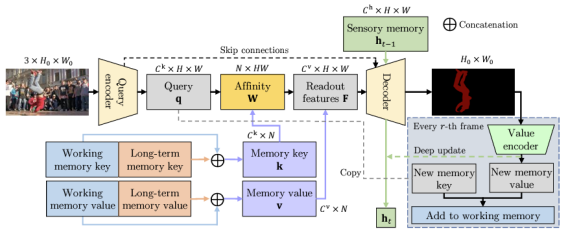

Fig. 2. XMem Architecture overview. Describes how sensory, working, and long-term memory interact to create the VOS mask.

providing the feature detector with relevant information from older frames.

Once the working memory is filled up over time and reaches it's maximum capacity, it moves frames into the long-term memory. These are chosen based on how often the memory gets called upon for the feature detection. The frames that get used the most, aka the more important memories, are removed from working memory and then inserted into the long-term memory.

Lastly, the long-term memory will continue to increase until it is full, where it will start removing the least called-upon values and deleting them entirely from the memory bank.

IV. Recreation Experiments and Results

A. Recreation Experimental Setup

To evaluate the recreation of XMem we tested our version on three datasets that XMem was initially tested on. These three datasets include hand-crafted image masks as ground truth to evaluate the masks created by VOS algorithms. Two datasets come from the densely annotated video segmentation (DAVIS) 2016 [6] and 2017 [7] datasets, along with one other dataset that is a significantly longer video [7]. The long video was used twice for testing, once with the regular video along with another version of the video that loops three times. To quantitatively evaluate the differences in the masking, the Jaccard index J, contour accuracy F, and the average of both $J\&F$ are calculated for each dataset and compared against previous VOS methods, including the results of XMem described in the original paper.

The Jaccard index J is defined as the intersection over the union between the predicted and the ground truth mask. This essentially describes quantitatively the amount of mislabeled pixels by the predicted masks. The equation is defined as:

$$J = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{1}$$

The contour accuracy F is a function that describes the F-accuracy and is comprised of the precision P_c and recall R_c of the predicted masks against the ground truth masks. This describes the precision of the predicted masks. The equation is defined as:

$$F = \frac{2P_cR_c}{P_c + R_c} \tag{2}$$

For the recreation, we ran our recreation on an NVIDIA GTX 4090 and used the vos-benchmark [8] script provided in

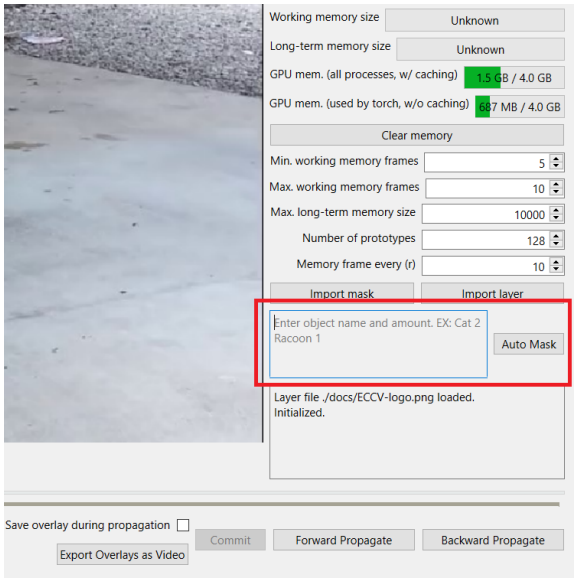

Fig. 3. This is a figure showing a section of the GUI. The red box highlights the additions we made for the algorithmic extension.

the paper to calculate J, F, and $J\&F$. This script automatically calculates each term using the output masks from each VOS algorithm and the hand-crafted masks.

B. Results

In table I, we can see the results of our recreation as well as previous VOS methods. From these results, we can see that our recreation of XMem is true to the original results provided in the XMem paper. XMem consistently performs better across all datasets when compared to other algorithms, even when tested on shorter videos. We can see that our method has results within $\pm 0.1\%$ of the results from the original paper.

V. Algorithmic Extension

As previously mentioned, one limitation of VOS is the fact that on the first frame of the video, the user has to manually select the objects to be tracked. Our algorithmic extension to the XMem paper seeks to improve this aspect of VOS by allowing users to simply type the name of the objects looking to be tracked. This simplifies the user experience, automatically masking the objects that are entered into the text box.

Google's OWL-ViT provided the exact functionality we were looking for. To integrate OWL-ViT with XMem we started with the XMem code-base which was provided in the paper. In order to demonstrate their model, the authors of the XMem paper, Ho Kei Cheng and Alexander G. Schwing, used a VOS GUI developed by Cheng and a couple other reseachers for a previous paper on Modular interactive VOS (MiVOS) [9]. This GUI retains the "interaction-to-mask" module for object detection given a user input and replaces the propagation module with XMem, allowing users to demo XMem for any video of their choosing.

The GUI code leverages PySide6 for development of the application's interface, utilizing widgets and tools provided by the Qt framework for Python. This means that all GUI modifications, front-end or back-end, take place in a single gui.py file. The first step we took to implement the extension was modifying the GUI. As seen in Fig. 3, we added a text box for user input and a button to automatically mask the objects that the user provided in the text box. This was done by modifying gui.py following the PySide6 syntax. We created objects for the text box and the button and then made a layout object that attached the two. Finally, we inserted the layout into the minimap area of the GUI, as that's where it makes the most sense visually. While PySide6 automatically handles formatting, we also increased the size of the text box to ease the user experience.

In the GUI, the user enters the objects to mask and the number of each object to mask. Once the "Auto Mask" button is pressed, the user input is sent through the autonomous masking pipeline. First, the objects and amounts from the input are separated and the image is retrieved as the current frame from the video. Then, for every object the user enters, a query is generated in the form of "a photo of a object" and run through the custom owl_vit_to_bbx function that will be described below. That function outputs n bounding boxes, where n is the provided amount of that particular object. Then, the interaction-to-mask module, developed in MiVOS [9] and provided by the GUI, is ran with simulated interactions on the center of the bounding boxes creating the initial masks for the objects.

In normal operation of the GUI, clicking anywhere on an object creates a mask for that object. We simulate this interaction by first resetting the visualization. Then a ClickInteraction object is created using the current frame. After, the push_point function is run on that object with the location of the click to simulate being passed as a parameter. Finally, a mask is predicted off that click, and the visualization is updated.

Lastly, the owl_vit_to_bbx function. The function takes in three parameters, the path to the image, the object name, and the amount of bounding boxes to return for that particular object. Using the transformers library, an OWL-ViT model is loaded. That model is given the image and the object name formatted as a text query. The model then outputs a list of possible bounding boxes, each with a varying degree of confidence, given the text query. Those possible bounding boxes are sorted by degree of confidence and then returns the top specified number of bounding boxes for the specified object object.

VI. Extension Experiments and Results

A. Experimental Setup

To test OD-VOS we follow the same framework to run XMem normally. Using the GUI we are able to upload any video of our choosing and run the model. Instead of clicking on the objects of interest, we can use our added functionality to type in the object name and the quantity of that object and automatically mask the first frame. Once the masks are set,

TABLE I
VOS RESULTS FROM XMEM AND OTHER VOS ALGORITHMS

Algorithm	DAVIS 2017			DAVIS 2016			Long Video			Long Video (x3)		
	J	F	$J\&F$	J	F	$J\&F$	J	F	$J\&F$	J	F	$J\&F$
MiVOS	81.7	87.4	84.5	89.6	92.4	91.0	80.2	82.0	81.1	78.0	79.0	78.5
STCN	82.2	88.6	85.4	90.8	92.5	91.6	82.9	89.2	87.3	83.3	85.9	84.6
JOINT	82.0	88.6	83.5	-	-	-	64.5	69.6	67.1	55.7	59.7	57.7
AOT	82.3	87.5	84.9	90.1	92.1	91.1	83.2	85.4	84.3	79.6	82.8	81.2
XMem	82.9	89.5	86.2	90.4	92.7	91.5	88.0	91.6	89.8	88.2	91.8	90.0
XMem (Our Recreation)	82.9	89.6	86.3	90.4	92.7	91.6	88.0	91.5	89.7	88.3	91.7	90.0

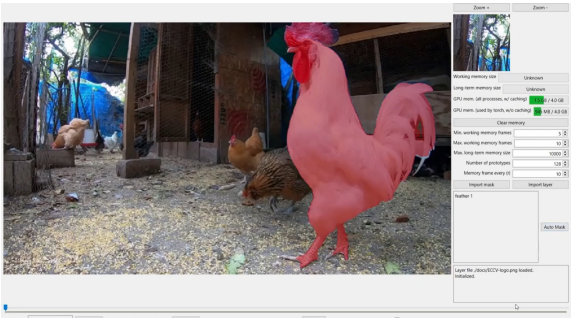

Fig. 4. This figure shows our extension in action. One object is masked using the text input and the 'Auto Mask' button.

we can press the forward propagate button as normal to run XMem across every frame in the video. Once the propagation finishes we can playback the video at the normal frame rate with the mask tracking the objects on every frame.

B. Qualitative Results

The videos where we tested OD-VOS can be found on the GitHub page. We have included Fig. 4 as an example of the masks produced from running it. We ran many qualitative tests and we are going to outline the results from those tests below.

OD-VOS appears to be most successful in videos with a smaller number of objects. As seen on the GitHub page, our most successful attempts came from security footage of a bear walking around a house and a video of a baby and a puppy playing together. In those instances, OWL-ViT had no trouble distinguishing the objects of interest from the background and everything ran smoothly. We also tested our model on the default video that comes with XMem (a video showing a raccoon and three cats in a garage) and it ran without issues. Videos with a small number of easily identifiable objects seemed to work the best when testing our model.

C. Limitations

OD-VOS is not without limitations however, and a couple of videos on the GitHub show this. OWL-ViT suffers when trying to identify five wolves in the video of a wolf pack walking around. It correctly masks four of the five, but misses one in the background. It also does not give the user a choice on which object to track. Given a video with multiple instances of the same object, like the chickens in the chicken coop video, the user does not have the choice to select which object they want to track. When "chicken 1" is entered into the text box for the chicken coop video, it selects a specific chicken in the back without the user having any say. These limitations lead to areas of improvement for future work.

VII. CONCLUSIONS & FURTHER WORK

XMem is a framework designed for long-term video object segmentation (VOS). Every VOS model is supervised, meaning the object mask needs to be provided on the first frame for it to run. Our algorithmic extension of XMem, OD-VOS, simplifies the user experience, allowing users to simply type the name and quantity of objects of interest to initialize the masks on the first frame. OD-VOS works extremely well on videos with a small number of unique objects, but has a couple of limitations. Videos with a large number of objects suffer from inaccuracy with identification. Additionally, videos with many instances of the same object fall short in cases where the user isn't tracking all of the objects. There is no way for the user to distinguish which specific objects they want to track, leading to unpredictability.

With this, there are many avenues for further work to be done. One idea would be to expand upon our language model to take more inputs than the names of different objects. For example, a user could describe the location or other finer details of the object that needs to be segmented for further precision of the initial mask.

REFERENCES

[1] H. K. Cheng and A. G. Schwing, "Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model," no. arXiv:2207.07115, Jul. 2022, arXiv:2207.07115 [cs]. [Online]. Available: http://arxiv.org/abs/2207.07115

[2] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby, "Simple open-vocabulary object detection with vision transformers," no. arXiv:2205.06230, Jul. 2022, arXiv:2205.06230 [cs]. [Online]. Available: http://arxiv.org/abs/2205.06230

[3] J. Wu, Y. Jiang, Q. Liu, Z. Yuan, X. Bai, and S. Bai, "General object foundation model for images and videos at scale," no. arXiv:2312.09158, Dec. 2023, arXiv:2312.09158 [cs]. [Online]. Available: http://arxiv.org/abs/2312.09158

[4] S. He and H. Ding, "Decoupling static and hierarchical motion perception for referring video segmentation," no. arXiv:2404.03645, Apr. 2024, arXiv:2404.03645 [cs]. [Online]. Available: http://arxiv.org/abs/2404.03645

[5] J. Wu, Y. Jiang, P. Sun, Z. Yuan, and P. Luo, "Language as queries for referring video object segmentation," no. arXiv:2201.00487, Mar. 2022, arXiv:2201.00487 [cs]. [Online]. Available: http://arxiv.org/abs/2201.00487

[6] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Computer Vision and Pattern Recognition*, 2016.
[7] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1704.00675
[8] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee, "Tracking anything with decoupled video segmentation," in *ICCV*, 2023.
[9] H. K. Cheng, Y.-W. Tai, and C.-K. Tang, "Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion," no. arXiv:2103.07941, Mar. 2021, arXiv:2103.07941 [cs]. [Online]. Available: http://arxiv.org/abs/2103.07941