Online Episodic Memory Visual Query Localization with Egocentric Streaming Object Memory

Supplementary Material

This document reports implementation details on the proposed ESOM methodology, along with additional results and discussion. Specifically, Section A explains the ESOM architecture by providing details of the Object Memory representation (sub-section A.1), the Object Tracking (OT) module (sub-section A.2.2), on the Object Discovery (OD) module (Section A.2.2), and on the Query Retrieval and Localization (QRL) algorithm (sub-section A.3). Section B reports details on the implementation of the baselines methods on AMEGO [21] and VideoAgent [15]. Section C reports information on the evaluation dataset and Section D reports additional and qualitative results.

A. Implementation Details of ESOM

A.1. Object Memory

Visual representation filter. To improve retrieval time and accuracy, the `.write()` operation incorporates a filtering mechanism to determine whether the visual content within $\mathbf{b}_{i,t}$, $\phi_{i,t}$, is relevant for the retrieval process. This filtering process relies on a binary classification function that assigns a label $c_{i,t}$ to each visual representation $\phi_{i,t}$, assessing its quality for effective query retrieval.
The design of $\mathcal{M}_{ego}$, which includes information about the quality of data for retrieval, was guided by experimental results. Specifically, we tested various filtering approaches to determine their effectiveness.

- MR1* (*design as reported in Section 4.1 of the main paper*): for each object $\mathbf{O}_i$, the filter assigns $c_{i,t} = 1$ at the moment of object discovery, as well as to two bounding-boxes $\mathbf{b}_{i,t}$ and their associated frames $\mathbf{F}_t$ from the most recent response track. These bounding-box and frame pairs are selected as the first and last valid patches identified by the classifier within the latest sequence of bounding boxes. For all other frames in temporal locations not mentioned, the filter set $c_{i,t} = 0$.
- MR2: for each object $\mathbf{O}_i$, the filter assigns $c_{i,t} = 1$ at the moment of object discovery, as well as to all the bounding-boxes $\mathbf{b}_{i,t}$ and their associated frames $\mathbf{F}_t$ from the most recent response track, considered useful by the classifier. This implementation was conducted to assess the impact of exploiting more object visual information for QRL.
- MR3: for each object $\mathbf{O}_i$, the filter assigns $c_{i,t} = 1$ at the moment of object discovery. For all other frames in temporal locations not mentioned, the filter set $c_{i,t} = 0$. This implementation was conducted to assess the impact of exploiting less object visual information for QRL, thus reducing the retrieval time of $\mathcal{M}_{ego}$.
- MR4: for each object $\mathbf{O}_i$, the filter assigns $c_{i,t} = 1$ to two bounding-boxes $\mathbf{b}_{i,t}$ and their associated frames $\mathbf{F}_t$ from the most recent response track. These bounding-box and frame pairs are selected as the first and last valid patches identified by the classifier within the latest sequence of bounding boxes. For all other frames in temporal locations not mentioned, the filter set $c_{i,t} = 0$. This implementation was conducted to assess the impact of considering the object at the moment of discovery in retrieval performances.

The OVQ2D performance of these memory configurations built by the Object Memory Population (OMP) algorithm and processed by the Query Retrieval and Localization (QRL) algorithm are reported in Table 6. As can be noticed, the proposed memory representation (MR1*) offers the best balance between QRL accuracy and retrieval time.

Classifier implementation details The classification model was implemented using a deep neural network consisting of a ResNet18 [23] feature extractor and a linear layer, which served as the representation filter. This model was fine-tuned on a dataset of 17,071 visual representations extracted from 250 uniformly sampled clips from the training set of the Ego4D Episodic Memory VQ2D-EgoTracks benchmark [22, 46]. Specifically, the training set included 8,534 visual representations labeled as 0 (i.e., not suitable for QRL) and 5,153 representations labeled as 1 (i.e., suitable for QRL). The validation set consisted of 2,126 visual representations labeled as 0 and 1,258 representations labeled as 1.

The training and validation representations were labeled as 0 or 1 based on whether they were incorrectly or correctly matched by the QRL algorithm. For each frame in the 250 selected clips, the representation ϕ_i of the annotated visual query $\mathbf{Q}_i$ was extracted, and its feature representation $\Phi(\phi_i)$ was computed using Siam-RCNN [48] as $\Phi(\cdot)$. This representation was then compared to $\Phi(\phi_{i,t})$, the feature representation of the patches withing the ground-truth bounding boxes of the tracklet associated with the query $\mathbf{Q}_i$, using a classification head [48]. This comparison produced a similarity score s_t ranging from 0 to 1. Patches were assigned negative labels if $s_t \leq 0.3$ and positive labels if $s_t \geq 0.7$. After labeling, the data was randomly split into training and validation sets, with 80% allocated to the training set and 20% to the validation set.

OMP Configuration OD	OT	Memory Representation	%tAP$_{25}$ ↑	%stAP$_{25}$ ↑	%Succ ↑	Time ↓
OD$_{oracle}$	OT$_{oracle}$	MR1*	73.2	68.1	**81.92**	0.51 s
		MR2	**73.3**	**68.3**	**81.92**	4.27 s
		MR3	63.7	55.1	74.33	**0.05 s**
		MR4	70.4	66.2	80.35	0.32 s
OD$_{yolo}$	OT$_{oracle}$	MR1*	**21.0**	19.4	**40.55**	0.36 s
		MR2	**21.0**	**19.5**	**40.55**	2.71 s
		MR3	20.3	17.0	38.02	**0.05 s**
		MR4	20.7	17.2	39.05	0.21 s
OD$_{oracle}$	OT$_{egostark}$	MR1*	**0.18**	**0.05**	**31.91**	0.49 s
		MR2	**0.18**	**0.05**	**31.91**	4.98 s
		MR3	0.16	0.03	26.56	**0.05 s**
		MR4	0.16	0.04	29.02	0.29 s
OD$_{yolo}$	OT$_{egostark}$	MR1*	**0.03**	**0.02**	**4.02**	3.08 s
		MR2	**0.03**	**0.02**	**4.02**	14.90 s
		MR3	0.02	0.03	1.78	**0.05 s**
		MR4	0.02	0.05	2.90	0.69 s

Table 6. **Effect of different filtering strategies.** This table illustrates the impact of using different filtering approaches for $\mathcal{M}_{ego}$ population on QRL performance and retrieval time. MR1* refers to the representation described in the main paper. As can be noticed, MR1* constitutes the best approach at the intersection of QRL accuracy and retrieval efficiency. (Best results are highlighted in **bold**).

Tracker	Detector	HOTA ↑	OWTA ↑	AssA ↑	DetA ↑
MASA	GroundingDino	0.036	**0.211**	0.105	0.013
MASA	YOLOv10*	**0.066**	0.080	0.116	**0.038**
MASA	Ground-truth	0.501	0.501	0.252	1.000
EgoSTARK	Hand-Object Detector	0.034	0.121	0.102	0.012
EgoSTARK	GroundingDino	0.057	0.121	0.115	**0.029**
EgoSTARK	YOLOv10*	0.061	0.188	**0.151**	0.025
EgoSTARK	Ground-truth	0.217	0.390	0.347	0.137
BoT-SORT	YOLOWorld	0.014	0.073	0.042	0.005
Ground-truth	YOLOv10*	0.848	0.848	0.943	0.762
Ground-truth	GroundingDino	0.875	0.875	0.943	0.812

Table 7. **Evaluation of different tracking algorithms for the implementation of the Object Tracking (OT) module.** We evaluated the multi-object tracking (MOT) performance of our implmentations on EgoTracks's validation set. Models marked with "*" are finetuned on EgoTracks [46]. Ground-truth refers to the EgoTracks annotations used to simulate optimal object detection or tracking, and the respective results are reported in gray. We found the AssA results to be best indicators for QRL accuracy. (Best results, across real-world tracker-detector combinations, are highlighted in **bold**.)

The visual representation classifier was fine-tuned for 50 epochs using Stochastic Gradient Descent (SGD) with a learning rate of 0.01. At convergence, it achieved a test accuracy of 97%. The input patches were resized to 224×224 pixels before being fed into the model.

A.2. Object Memory Population (OMP)

A.2.1. Object Tracking (OT) module.

In this section, we provide details regarding the implementation of OT module. In Table 7, we report the performance of the different implementations as measured by standard multiple object tracking (MOT) metrics [32].

OT$_{egostark}$**.** OT$_{egostark}$ implements a multiple-object tracking approach using multiple instances of single-object trackers. This approach is motivated by the lack of effective multiple-object tracking solutions for egocentric vision, whereas single-object tracking has been extensively studied [11, 12, 46]. For single-object tracking, we use the STARK framework [52], fine-tuned on the EgoTracks dataset as described in [46], resulting in the EgoSTARK tracker. To track multiple objects, we instantiate multiple EgoSTARK trackers simultaneously. A new tracker instance is initialized in frame $\mathbf{F}_t$ whenever the OD module identifies a valid object, which is then stored in memory. Each bounding box predicted by the OD module serves as a template to initialize a corresponding tracker. Once initialized, a tracker runs across subsequent frames to predict the bounding boxes of the object it was assigned to, thereby updating the spatio-temporal representation of the respective object. Due to the challenges of object tracking in egocentric vision [11, 12, 46], we found different tracker instances to often end in tracking the same object. In order to prevent duplicate tracker instances, we calculate the IoU between all tracked objects at frame t. If two tracking predictions overlap with an IoU greater than 0.5, the trackers are considered to be duplicate and only the tracker instance and their corresponding prediction with the lower object instance ID (i.e. the older object) is retained.

Furthermore, to prevent incorrect tracking predictions from causing a tracker to deviate from a searching area that contains the target object and wander across the frame, the box used to compute the search region for the subsequent frame is not updated if the confidence score of the EgoSTARK instance falls below 0.4. This strategy is based on the observation that, in egocentric vision, objects often leave the frame during rapid head movements but typically reappear in a similar location when the head returns to its original position. Before writing the OT module predictions to $\mathcal{M}_{ego}$, the bounding boxes $\mathbf{b}_{i,t}$ with confidence score $s_{i,t} > \lambda_{ot} = 0.5$ are retained.

OT$_{masa}$**.** As an alternative approach, we implemented the state-of-the-art association method MASA [27] as a multi-object tracker, utilizing object detections provided by the OD module. We set a score threshold of 0.1 and limited object comparisons to the last ten frames using the MASA adapter. To evaluate the impact of considering more frames, we tested with 150 previous frames, but this led to a drop in the OWTA metric from 0.2111 to 0.1496. This result highlights the challenges the MASA adapter faces with long-term tracking.

OT$_{yoloworld}$**.** Finally, we investigated the tracking algorithm BoT-SORT [1] included in the YOLOWorld [8] implementation.

OT$_{oracle}$ We implement the oracular object tracker as follows: whenever a detected object intersects a GT bounding box from EgoTracks, we put the rest of the track (from that box onward) in memory. EgoTracks extends the Episodic Memory VQ2D benchmark introducing an object track for each query, hence GT tracks from EgoTracks are valid GT VQ2D tracks. When we combine OT$_{oracle}$ with OD$_{oracle}$, we simply insert in memory all object tracks from EgoTracks.

A.2.2. Object Discovery (OD) module

Clustering-based object taxonomy on EgoTracks. As detecting objects from egocentric videos is not straightforward [57], we tested different object detectors including the off-the-shelf detectors GroundingDino [30] and the hand-object detector of [43] (here referred to as Hand-Object Detector) which do not require training on the considered settings, as well as Faster R-CNN R101-FPN [40] and YOLOv10 [49] instances trained on different data sources, including EgoObjects [57] and curated versions of EgoTracks [46]. Targeting visual object tracking, EgoTracks contains spatially sparse bounding box annotations across contiguous frames (i.e., only few object instances are annotated in each frame). Each object instance is associated to a textual description rather than an object class (e.g., "a blue bottle"). To enable the training of object detectors, we first clustered object descriptions with K-Means to obtain an initial set of $1,169$ classes, followed by manual refinement in order to adjust them, leading to a diverse, granular taxonomy comprising 295 classes.

Data cleaning for EgoTracks. Since EgoTracks contains annotations in contiguous frames, we uniformly sampled one-third of all frames. EgoTracks contains annotations for very small objects to assess the ability of visual object trackers to continuously follow objects subject to scale changes. Since these objects would be unfeasible to detect from single frames and may be source of ambiguity due to occlusion or motion blur, we discarded bounding box annotations with areas falling in the first quartile of all bounding box areas in the training set. Figure 7 shows a few examples of annotation bounding boxes that fall in the first quartile and therefore discarded from the dataset. This leads to a reduced training set of $474,761$ annotated images, while we consider a validation set of $2,466$ images.

Faster R-CNN - based detector. A first version of the object discovery module is obtained by training a Faster R-CNN instance on the above described dataset with object class taxonomies obtained through clustering of object descriptions. The performance of this module is limited as shown later in the quantitative evaluation, so this implementation of the OD module is not considered in the main paper.

Figure 7. **Visualization of discarded bounding-boxes.** Small-area bounding boxes similar to those shown in this figure have been discarded for the training and evaluation of the module. This is because we found that they lacked sufficient visual information, leading to an increased number of false positive detections. These detections provided no useful data for tracker initialization within the OT module or for effective QRL.

YOLO - based detector. A second version of the object discovery module was based on the YOLOv10 [49] object detector, given its capacity for real-time inference. We trained the YOLOv10 detector on the same set of refined EgoTracks annotations. Given the sparsity of EgoTracks annotations, both the Faster R-CNN and YOLO detectors tend to recognize a few object instances per image, which may limit the ability of the object discovery module to identify all possible objects of interest. To address this issue, we trained the YOLOv10 object detector on the EgoObjects [57] dataset, which contains dense object annotations for egocentric frames with a fixed taxonomy of 638 classes. To make the EgoObjects and EgoTracks sets compatible, we mapped each EgoTracks object description to its corresponding class from the EgoObjects taxonomy. This is done by first describing the object description and class name with a *Sentence Transformer*, and then selecting the class maximizing the cosine similarity between the embeddings. To reduce noise, we discarded all annotations with a similarity lower than 0.7.

Examples of class mapping. Table 8 shows some examples of object descriptions retrieved from the EgoTracks annotations and mapped to EgoObjects objects. Note that some objects such as "air conditioner", "headphones" or "yoga mat" have a reasonable match, while "washing machine" contains "flat washer", which has a different meaning despite a close embedding. We did not manually adjust

EgoObjects Classes	accordion	air conditioner	headphones	washing machine	...	yoga mat
EgoTracks Mapped Classes	-	air conditioner, air conditioner control, air conditioner., air conditioning machine	head phones, headphone, headphones, headset, earphone, earphones	flat washer, washing machine	...	gym mat, mat, workout mat, yoga mat, exercise mat, fitness mat

Table 8. **Examples of class mappings from EgoObject classes to EgoTracks object descriptions.** The first row reports EgoObject classes, while the second row contains descriptions of EgoTracks matched with embedding similarity.

these inconsistencies to keep the process fully automated. The table contains one class with no matches (accordion), due to the absence of EgoTracks objects with similar descriptions. Out of the total number of class descriptions (3491), we mapped 1,241 captions, while the remaining 2,250 classes were excluded from the dataset. This resulted in a high-quality subset of the data suitable for training and evaluation.

Quantitative comparison. Despite the domain difference between EgoObjects and EgoTracks, we noticed improved qualitative and quantitative results. To further adapt the model to the EgoTracks domain, we fine-tuned the YOLOv10 object detector, pre-trained on EgoObjects, on the EgoTracks dataset with labels mapped to the EgoObjects taxonomy. Table 9 compares the performance of the considered approaches with three off-the-shelf object detectors: GroundingDino [30], the Hand-Object-Detector of [43], and YoloWorld [8]. We do not train these detectors on the EgoTracks data and use the official checkpoints and implementations provided by the authors. Since Visual Query Localization does not require to correctly identify object classes, all methods are evaluated with AP and Recall, merging all ground truth and predicted classes in a single "object" class. It is worth noting that AP and Recall measure different aspects of the detector. Indeed, AP measures the average ratio between correct predictions and all predictions made by the detector, whereas Recall measures the fraction of objects which have been correctly found by the detector, regardless of the total number of objects. In our settings, it is desirable to have a large recall (future queries have more chance to end up in the memory), but, since it is not possible to track all objects and a selection must be made based on the confidence score, in practice, a very low value of AP is not desirable. The harmonic mean between AP and Recall is hence adopted as a way to identify the most balanced detector. Results in Table 9, obtained with a confidence score threshold $\lambda_{od} = 0.01$, show that Faster R-CNN trained on labels clustered from EgoTracks achieves the second-highest recall (0.5224), but also yields a very low AP value (0.0219). On the contrary, the YOLOv10 instance achieves the highest AP (0.1983) but also a low recall (0.1634). We attribute these skewed results to the sparse nature of annotations in EgoTracks (only a few object instances annotated per frame) and the different ways the two methods handle negative examples. The harmonic mean results of the first one are rather low (0.0420), while the second result is higher but not optimal at 0.1791, highlighting the unbalanced nature of the two approaches. Training YOLOv10 on EgoObjects, which contains much denser annotations increases Recall to 0.3780 at the expense of AP (0.0247), which leads to a low harmonic mean of 0.0463. Mapping EgoTracks annotations to the EgoObjects taxonomy and training YOLOv10 using both datasets provides the most balanced detector, with a harmonic mean of 0.1882, an AP of 0.1358 and a Recall of 0.3067.

YOLOWorld achieves the lowest precision value among all the methods (0.0207), but the highest recall, equal to 0.5969. Due to the low precision value, the harmonic mean has a low score equal to 0.0401. GroundingDino achieves a low AP and a reasonable recall value (0.0450 and 0.2572), which is likely due to the large domain shift between data used for training GroundingDino and the challenging egocentric observations of EgoTracks. Using a Hand-Object-Detector as an object detector leads to lower results, which is because the Hand-Object-Detector only focuses on interacted objects, hence neglecting many non-interacted objects which may nevertheless be queried in the future. Lastly, RT-DETR (used in VideoAgent [15]) achieves results comparable to GroundingDino, due to similar reasons.

Qualitative Examples. Figure 8 reports qualitative examples, reporting the predictions of the discussed models on a frame from the validation set of EgoTracks. Results are consistent with the evaluation metrics in Table 9, Faster R-CNN (A) is the model with the second-highest recall, indeed it is the model with the highest number of bounding boxes containing both correctly localized objects and errors. YOLOv10 trained on EgoTracks (B) has fewer predictions than Faster R-CNN, but they are more accurate. Training YOLOv10 on EgoTracks (C) shows similar results to Faster R-CNN, with the second-densest predictions, translated into the second-highest recall model. YOLOv10 trained on EgoObjects fine-tuned in EgoTracks (D) has the most balanced results; predictions are well localized, reflecting the best-performing model in the HM metric. YOLOWorld (E)

Detector	Training Dataset	AP ↑	Rec ↑	HM ↑
Faster R-CNN	EgoTracks*	0.0219	0.5224	0.0420
YOLOv10	EgoTracks*	**0.1983**	0.1634	0.1791
YOLOv10	EgoObjects	0.0247	0.3780	0.0463
YOLOv10	EgoObjects + EgoTracks**	0.1358	0.3067	**0.1882**
YOLOWorld	-	0.0207	**0.5969**	0.0401
GroundingDino	-	0.0450	0.2572	0.0765
Hand-Object Detector	-	0.0247	0.0381	0.0299
RT-DETR	-	0.0418	0.3374	0.0743

Table 9. **Comparison of the different object detectors.** * denotes that the object taxonomy has been obtained through clustering. ** denotes that classes have been obtained by mapping EgoTracks descriptions to the EgoObject taxonomy. HM: Harmonic Mean between AP and Recall. (Best results are highlighted in **bold**.)

achieves results comparable to Faster R-CNN (A), demonstrating the highest recall with dense object predictions and minimal errors, which can be further reduced by increasing the threshold. iGroundingDino (F) contains several correctly regressed objects, but as shown in the frame, no ground truth annotations were found. Hand-Object Detector (G) has the worst performance, containing only two predictions, one of which is related to the interacted object. The discovery mechanism of this detector leads to the lowest values for both precision and recall. The predictions of VideoAgent (H) are partially correct, despite the presence of a few wrong bounding boxes. Finally the predictions marked with I are the ground truth ones.

OD$_{yolo}$ and OD$_{gdino}$. Given these results, in the following experiments, we consider the YOLOv10 detector trained on EgoObjects and EgoTracks as a representative of trained methods and both GroundingDino and Hand-Object-Detector as representative of off-the-shelf methods. These are referred to as OD$_{yolo}$ and OD$_{gdino}$ respectively in the main paper.

OD$_{oracle}$. The oracular object detector is obtained using the ground truth object annotations in EgoTracks as valid detections. As previously mentioned, EgoTracks is aligned to the Episodic Memory benchmark of Ego4D, so EgoTracks detections include valid queries.

A.3. Query Retrieval and Localization (QRL)

We implemented two distinct approaches for localizing a visual query $\mathbf{Q}$ within a memory $\mathcal{M}_{ego}$: an implementation based on SiamRCNN with a classification head [48](QRL$_{siamrcnn}$); and an approach based on DINOv2 with cosine similarity [21, 34] (QRL$_{dino}$).

In QRL$_{siamrcnn}$, a Feature Pyramid Network (FPN) [28] serves as the backbone $\Phi(\cdot)$ for features extraction, generating features for the visual representation of the object $\Phi(\phi_{i,t}), \phi_{i,t} \neq \emptyset$ stored in $\mathcal{M}_{ego}$ as well as the features

Figure 8. **Visualization of predictions from the object detection models.** Here we qualitatively show the performance of the different object detectors over the same sampled frame. A: Faster R-CNN on EgoTracks, B: YOLOv10 on EgoTracks, C: YOLOv10 on EgoObjects, D: YOLOv10 on EgoObjects, fine-tuned in EgoTracks, E: YOLOWorld, F: GroundingDino, G: is the Hand-Object Detector H: RT-DETR, I: Ground-truth annotations.

for the visual query $\Phi(\mathbf{Q})$. To determine whether $\mathbf{Q}$ exists within $\mathcal{M}_{ego}$, each $\Phi(\phi_{i,t})$ is compared with $\Phi(\mathbf{Q})$ using a siamese network head, $\Psi(\Phi(\phi_{i,t}), \Phi(\mathbf{Q}))$, which implements a bilinear operation [48] that predicts instance similarity score belonging to the interval [0, 1].

In QRL$_{dino}$, DINOv2 [34] is employed as the backbone $\Phi(\cdot)$ for extracting feature representations $\Phi(\phi_{i,t})$ and $\Phi(\mathbf{Q})$. In this approach, the similarity operation $\Psi(\Phi(\phi_{i,t}), \Phi(\mathbf{Q}))$ is implemented using cosine similarity [21], which provides scores belonging to the interval [0, 1].

In both cases, for each object i, the similarity scores are averaged into a single score r_i. If the maximum score r_i exceeds the threshold λ_{ref} = 0.5, the object $\mathbf{O}_i$ is considered a match for query $\mathbf{Q}$. Once a match is identified, the most recent sequence of contiguous bounding boxes from $\mathbf{O}_i$'s spatio-temporal history is retrieved and returned as the response track $\mathbf{r}$.

A.4. ESOM Memory Overhead

The GPU memory overhead is 423MB for OD_{yolo} and 2,350MB for $\text{OT}_{egostark}$ when tracking 10 objects concurrently, 2,320MB for OT_{masa}, and 2,474MB for $\text{QRL}_{siamrcnn}$.

B. Online Object Memory Baselines

Here we provide details of the implementations of the baselines used for comparison.

B.1. AMEGO

The AMEGO framework [21], was employed to construct an episodic memory model, serving as a performance benchmark for our proposed approach. We utilized *ce683bd* version of the AMEGO codebase, available at , with minor modifications to accommodate the EGO4D dataset structure. Our implementation focused specifically on the Hand-Object Interaction (HOI) module of AMEGO. The Location Segmentation module was excluded from our analysis as it fell outside the scope of our research objectives. To ensure a fair and consistent comparison, we maintained the original hyperparameters and thresholds to align with the baseline model published on the official GitHub repository. Since their experiment has been made using a different dataset, our modifications were limited to data input/output functions to handle the EGO4D data format, without altering the core algorithmic components.

B.2. VideoAgent

The VideoAgent framework [15], has been utilized as a comparative baseline for our episodic memory model. We employed the *21e4eb7* version of the VideoAgent repository. Our implementation focused on the Object Memory module, and we opted to omit the captioning module since it was not pertinent to our research scope. To maintain experimental integrity, we preserved the original configuration parameters.

B.3. Memory Representation

As with the other OMP configurations discussed in this paper, the object tracklets produced by VideoAgent and Amego OMP are stored in memory using the MR1* representation, as it provides the best balance between accuracy and memory efficiency (see Table 6). Specifically, for each $\mathbf{O}_i$ found by either AMEGO or VideoAgent object detection and tracking algorithms, $\mathcal{M}_{ego}$ retains the RGB patch ($\phi_{i,t}$) and the $\mathbf{b}_{i,t}$ at the time of object discovery, along with two RGB patches from the most recent response track. These patches correspond to the first and last valid patches identified by the classifier within the most recent sequence of bounding boxes. For all other temporal locations not covered above, $\mathcal{M}_{ego}$ stores only the spatio-temporal information provided by $\mathbf{b}_{i,t}$.

https://github.com/gabrielegoletto/AMEGO.git
https://github.com/YueFan1014/VideoAgent.git

OMP Configuration OD	OT	%tAP$_{25}$ ↑	%stAP$_{25}$ ↑	%Succ ↑	Size ↓
OD_{oracle}	$\text{OT}_{egostark}$	**18.3**	**5.1**	**31.91**	**591.4 MB**
OD_{yolo}	$\text{OT}_{egostark}$	0.03	0.2	4.02	1.7 GB MB
$\text{OD}_{oracle+yolo}$	$\text{OT}_{egostark}$	10.6	2.3	23.67	2.4 GB

Table 10. **More objects in the memory can confound QRL.** In this table, by combining the capabilities of the OD_{oracle} and OD_{yolo}, we tested wether the QRL method is robust to a memory $\mathcal{M}_{ego}$ which contains the true visual queries as well as other object found while processing $\mathcal{V}_{ego}$. (Best results are in **bold**).

B.4. Query Retrieval and Localization (QRL)

To localize the visual queries $\mathbf{Q}$ within the memory $\mathcal{M}_{ego}$ built over AMEGO and VideoAgent, we used the implementation based on SiamRCNN with a classification head ($\text{QRL}_{siamrcnn}$). Once the similarity scores are computed for the different patches, they are averaged into a single score r_i. If the maximum score exceeds the λ_{ret} = 0.5, the object $\mathbf{O}_i$ is considered a match for query $\mathbf{Q}$.

C. Evaluation Dataset

Due to the high storage requirements for the validation split of the Ego4D VQ2D benchmark [22] and the significant time needed for experiments on the full dataset, we opted to use a subset of the validation clips for our evaluations. Specifically, we randomly selected 115 video clips uniformly, containing 450 visual queries along with their corresponding ground-truth response tracks and tracklets. This subset consisted of a total of 208,365 frames. To maintain consistency with the original frame rate of the EGO4D dataset [22], we ran the OMP algorithm at 5 frames per second.

D. Additional Results

Combination of OD modules. The performance of ESOM has been evaluated under various configurations of the OD and OT modules (Table 5). However, OMP configurations using OD_{oracle} construct a memory that includes only objects that will be queried. To incorporate addi-

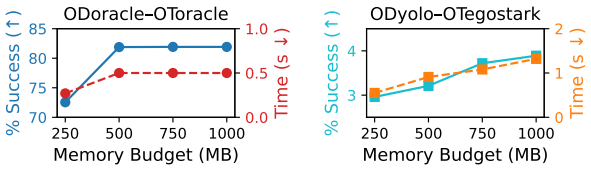

Figure 9. **Impact of memory-budget on ESOM performances.** Success rate and query time for two ESOM variants under varying memory budgets (250–1000 MB). Results show that performance remains stable as older or low-confidence objects are pruned to meet memory constraints, indicating ESOM's robustness under limited memory conditions.

tional legitimate objects into $\mathcal{M}_{ego}$, we combined the oracle detections, together with the OD_{yolo}detections (see $\text{OD}_{oracle+yolo}$ in Table 10). Adding more objects to memory reduces ESOM's performance compared to the OMP configuration with OD_{oracle}. This suggests that an increased number of objects in memory hinders the QRL algorithm's ability to accurately discriminate the visual query, resulting in confusion. However, its performance remains higher than the OMP configuration using only OD_{yolo}, further highlighting the challenges detectors face in identifying objects that will appear as queries in the future.

Random performance The performance of ESOM has been evaluated by implementing a QRL method that randomly (w. uniform dist.) selects a response track from $\mathcal{M}_{ego}$. This achieves a Succ. score of 27.29 (OD_{oracle}+OT_{oracle}) and of 0.44 (OD_{yolo}+$\text{OT}_{egostark}$), while ESOM achieves 81.92 and 4.02 respectively. ESOM performance, even in suboptimal OMP configuration (OD_{yolo}+$\text{OT}_{egostark}$) are still better than random selection. This indicates that the method effectively leverages available information to enhance performance.

Constant memory budget We explored a "constant-budget" version of ESOM, in which older and low-confidence objects are pruned as soon as a predefined memory limit is reached. Results (Figure 9) demonstrate that ESOM isn't particularly affected by memory constraints.

Qualitative examples. Figure 10 shows additional qualitative examples of the behavior of QRL in retrieving and localizing different visual queries within two memory representations build by two OMP configurations.

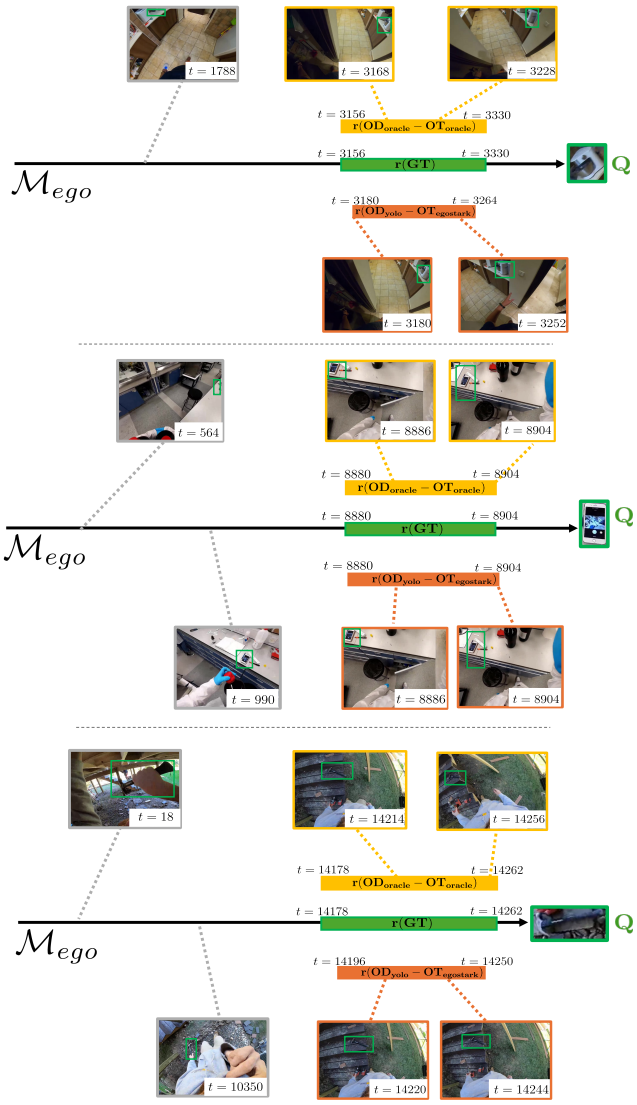

Figure 10. **Additional qualitative examples.** Successful retrieval of thre different queries Q from $\mathcal{M}_{ego}$ using the oracle OMP algorithm (OD_{oracle} - OT_{oracle}) and the real-world OMP (OD_{yolo} - $\text{OT}_{egostark}$). Sample frames from each track are shown with corresponding bounding boxes whose patches $\phi_{i,t}$ were matched during retrieval. The frames in gray correspond to the moment of the object's discovery. The green bar visualizes the temporal extent of the ground-truth track, with timestamps indicating its duration.