# RELOCATE: A Simple Training-Free Baseline for Visual Query Localization Using Region-Based Representations

Savya Khosla Sethuraman T V Alexander Schwing Derek Hoiem
University of Illinois Urbana-Champaign
{savyak2, st34, aschwing, dhoiem}@illinois.edu

## Abstract

*We present* RELOCATE*, a simple training-free baseline designed to perform the challenging task of visual query localization in long videos. To eliminate the need for task-specific training and efficiently handle long videos,* RELOCATE *leverages a region-based representation derived from pretrained vision models. At a high level, it follows the classic object localization approach: (1) identify all objects in each video frame, (2) compare the objects with the given query and select the most similar ones, and (3) perform bidirectional tracking to get a spatio-temporal response. However, we propose some key enhancements to handle small objects, cluttered scenes, partial visibility, and varying appearances. Notably, we refine the selected objects for accurate localization and generate additional visual queries to capture visual variations. We evaluate* RELOCATE *on the challenging Ego4D Visual Query 2D Localization dataset, establishing a new baseline that outperforms prior task-specific methods by 49% (relative improvement) in spatio-temporal average precision.*

## 1. Introduction

Visual Query Localization (VQL) requires localizing the last appearance of an object of interest in a long video. The object of interest is specified via a reference image, also known as the *visual query*. Figure 1 provides illustrative examples. VQL is an important task, *e.g.*, for surveillance, legal investigations, wildlife monitoring, or simply for tracking down a misplaced item. However, the task presents several unique challenges that push the boundaries of contemporary computer vision methods. For instance, unlike classic object detection models that are trained to identify a fixed set of object categories [5, 32], VQL requires localizing an open-ended range of objects. Additionally, while typical object tracking methods are initialized with a bounding box close to the object's temporal location in the video [19, 22, 41], the VQL reference image (visual query) often originates outside the video, *i.e.*, there may be no exact or neighboring frame for reliable matching. Further, the object's appearance may vary significantly from the visual query due to changes in orientation, scale, context, lighting, motion blur, and occlusions. Compounding these issues, the object of interest usually appears briefly in a long, untrimmed video.

Classic VQL methods typically use a stage-wise pipeline [12, 39, 40]: (1) identify all objects in each video frame, (2) compare these objects with the given query to select the most similar ones, and (3) perform bidirectional tracking to obtain a spatio-temporal response. While this pipeline is well-grounded in the object localization literature, it is effective only for large, consistently visible objects that closely match the visual query in short video clips. A more recent work [15] has proposed an end-to-end framework that aims to understand the holistic relationship between a given query and the video, performing spatio-temporal localization in a single step. However, this approach requires extensive training on large, annotated datasets, which can be challenging to obtain. Additionally, since such a method learns a holistic video-query relationship, it must process the entire video for each query, even when multiple queries are to be localized in the same video.

To address these limitations, we propose Region-based representations for Localizing Anything in Episodic memory (RELOCATE), a simple *training-free* baseline for VQL. Being training-free, RELOCATE eliminates the need for extensive task-specific training and large annotated datasets. Moreover, it encodes a video independently of the query, allowing the same video encoding to be reused for multiple queries. This makes the method more suitable for episodic memory tasks, as it enables us to perform the resource-intensive video encoding just once for multiple queries.

RELOCATE follows a classic stage-wise setup for object localization while integrating techniques to efficiently encode long videos, handle small and fleeting objects, and manage varying appearances of the query object. Specifically, it begins by extracting region-based representations from the video and searching for candidates that match the visual query. The selected candidates are then refined to im-

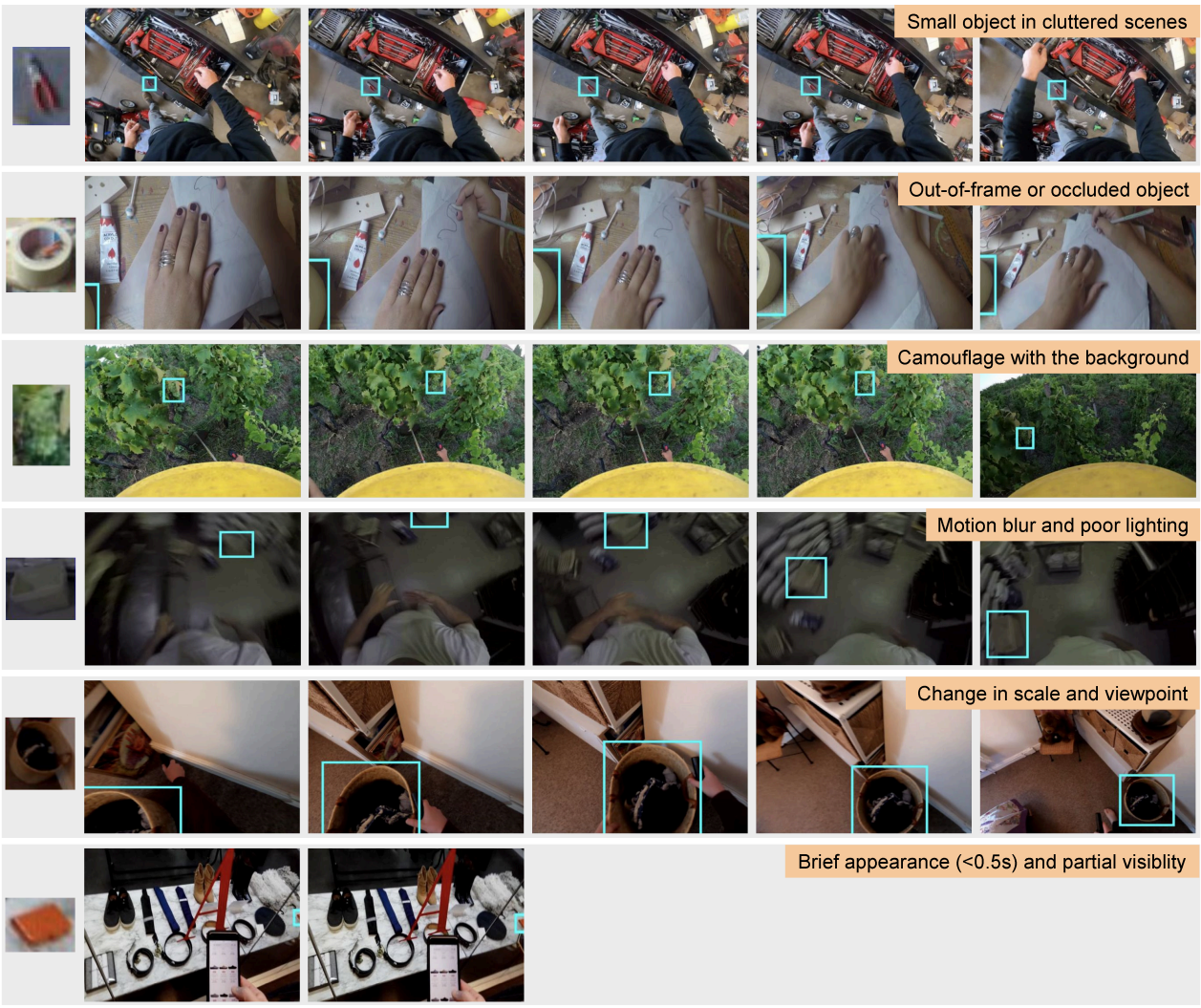



Figure 1. RELOCATE is a training-free framework designed for visual query localization. It can effectively localize target objects in long videos despite challenging conditions such as visual clutter, occlusions, background blending, motion blur, viewpoint changes, and brief object appearances. Here we show visual queries on the left and the successfully localized object appearances marked by cyan bounding boxes on the right.

prove precision, and the most relevant candidate is tracked across video frames to generate an initial prediction. In a subsequent iteration, this prediction is used to create new visual queries, which are then employed to relocalize the object, aiming to capture appearances that differ significantly from the original visual query.

We evaluate RELOCATE using the Ego4D Visual Query 2D (VQ2D) Localization benchmark [12] and observe significant improvements over prior methods. We achieve a 49% increase in spatio-temporal average precision and a 33% boost in temporal average precision compared to prior state-of-the-art, all without task-specific training.

In summary, the main contributions of this work are:

1. We propose RELOCATE, a *training-free* method for localizing objects in long videos. Despite no task-specific training, RELOCATE significantly improves upon prior VQL work on the Ego4D VQ2D benchmark.
2. We demonstrate the benefits of using region-based representations from pretrained vision models for object localization. Particularly, we show that these representations form a detailed yet compact encoding for long videos. Furthermore, they allow us to efficiently and robustly perform object retrieval using a simple matching function like cosine similarity.
3. We propose techniques to improve the robustness and precision of a stage-wise object localization framework. Specifically, to enhance localization accuracy, we introduce a refinement step that allows RELOCATE to closely

examine the selected candidates and improve their representations, while filtering out incorrect selections. Additionally, we introduce a technique to generate multiple visual queries from a single query, which helps capture varying appearances, changing contexts, and occlusions of objects in a dynamic video setting.

## 2. Related Work

**Object Instance Recognition.** Early approaches to object instance recognition retrieve images with similar keypoints to the visual query [18, 24, 25, 37]. This works well for objects with distinctive textures, but fails for low-texture, blurry, and highly occluded objects, which are common in the VQL task. Subsequent research shifts towards using features extracted from convolutional neural networks (CNNs) as descriptors for image retrieval [1–3, 7, 30, 31]. However, these early CNN-based methods suffer when faced with large changes in scale, rotation, or viewpoint. The advent of vision foundation models such as CLIP [28], DINO [6], and DINOv2 [26] has enabled new strategies for instance recognition and retrieval. Several works [4, 9, 34, 35] have leveraged the cross-modal capabilities of CLIP for specialized retrieval tasks. However, these approaches are primarily designed for scenarios where query objects occupy a substantial portion of the target frame, which is not the case in VQL. Recent work on region-based representations [36] shows that pooling DINOv2 features over SAM regions provides effective example-based object category retrieval. In this work, we apply a similar approach for our initial search but propose enhancements to refine, track, and expand the initial query.

**Video Object Tracking.** Conventional tracking approaches are initialized with a bounding box in an initial frame of the target video, and they track objects through gradual appearance changes. In contrast, VQL often involves queries from frames outside the target video, leading to significant variations in appearance, viewpoint, and context. Nevertheless, tracking remains crucial in VQL. Early tracking approaches relied on motion and appearance cues, using correlation filters and keypoint matching [13, 16, 33]. More recently, trackers leverage large Transformer models to effectively track one or more objects in long videos [8, 11, 19, 21, 23, 41]. Some recent works have also extended SAM [17] for tracking [10, 29, 42]. In this work, we use SAM 2 [29].

**Visual Query Localization.** The Ego4D benchmark [12] recently introduced VQ2D, a benchmark for VQL. The initial approach, proposed as a baseline in Ego4D, employs a three-stage detection and tracking framework: performing frame-level detection, identifying the most recent detection peak across time, and applying bi-directional tracking to determine the complete temporal extent of the target object [12]. Subsequent works enhance the framework's performance through various refinements like incorporating negative frame sampling to reduce false positives [39] and leveraging background objects as contextual cues [40]. In this work, we adopt a similar stage-wise framework, introducing key design decisions and refinements that significantly improve the baseline. A more recent effort trains a network to learn the query-video relationship and perform VQL in a single step. However, it relies on a large amount of annotated data, which is costly to obtain. In contrast, we present a training-free solution.

## 3. RELOCATE

We focus on VQL, the task of localizing the last occurrence of a query object in a video. Formally, given a video $\mathcal{V}$, a visual query $\mathcal{Q}$, and a query time $T$, the objective is to predict a response track $\mathcal{R} = \{b_s, b_{s+1}, \ldots, b_e\}$ that localizes and tracks the latest occurrence of the query object prior to time $T$. Here, $\mathcal{Q}$ is specified by a bounding box in a reference frame, $s$ and $e \leq T$ denote the first and the last frames in which the query object is visible, and $b_i$ represents a bounding box around the object in frame $i$.

As shown in Figure 2, RELOCATE *prepares* video representations to facilitate a swift *search* for objects that match the given visual queries. It then *refines* the search results for better precision and *tracks* the latest match across frames. After making a prediction, it can *reiterate* the localization process using the previous prediction for better results.

**Prepare (Section 3.1).** A given video is first distilled into semantically meaningful object-level representations, or *object tokens*. For this, RELOCATE uses a segmentation model to generate object-wise binary masks for each object across all video frames and a feature extractor to produce dense feature maps for each frame. The features within each object mask are then pooled to produce an object token.

**Search (Section 3.2).** After preparing object tokens from the video, RELOCATE creates a similar region-based representation for the visual query, referred to as the *query token*. It then searches the object tokens for candidates that match the query token.

**Refine (Section 3.3).** The candidates identified in the initial search are then refined to improve spatial precision and remove any spurious matches. For example, Figure 2 illustrates how refinement enhances spatial precision by capturing the base of the monitor.

**Track (Section 3.4).** The latest refined search is tracked across video frames using an off-the-shelf tracking model to produce a response track localizing the most recent occurrence of the query object in the video.

**Reiterate (Section 3.5).** To better capture the visual variations of the query object, we leverage the object's appearance in the tracked frames to create more visual queries and then re-apply RELOCATE to the video segment that follows the previously predicted track.

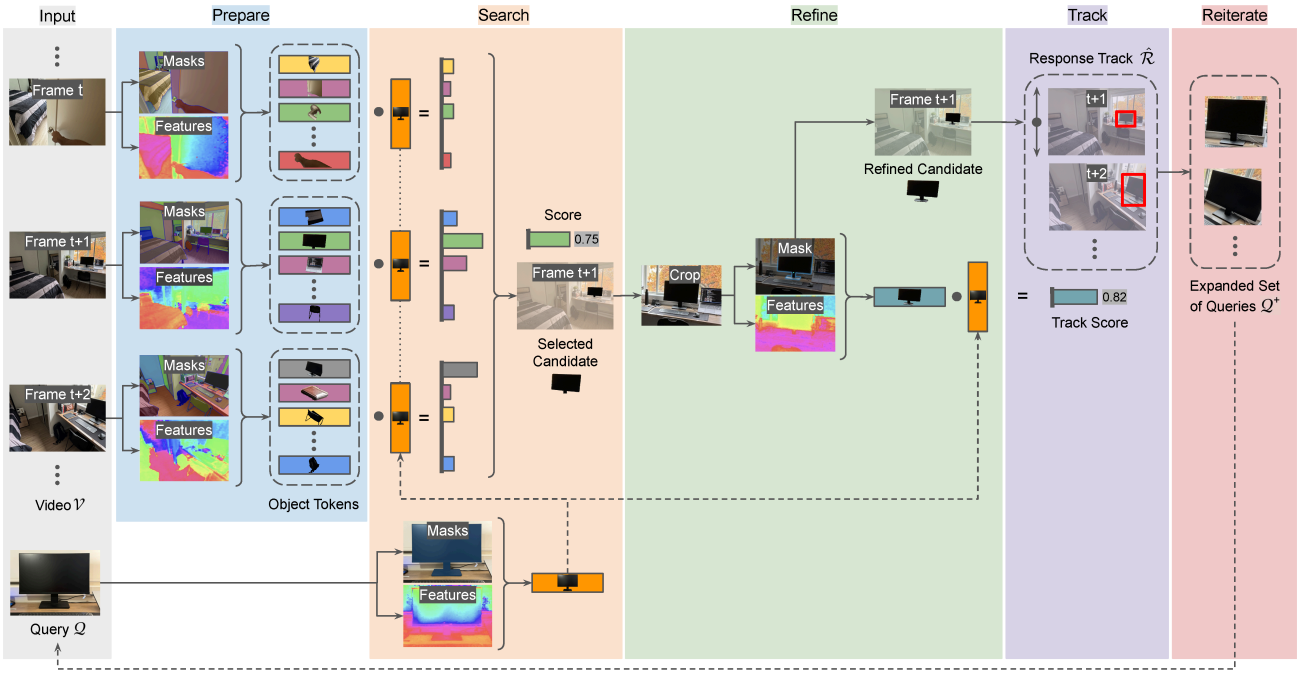



Figure 2. **RELOCATE framework:** Prepare object tokens from the given video $\mathcal{V}$ and search for candidates that match the visual query $\mathcal{Q}$. Then, refine the search results for better precision and track the latest refined candidate across video frames to get a response track prediction $\hat{\mathcal{R}}$. Finally, use the response track to create additional visual queries and reiterate the search, refinement, and tracking process.

### 3.1. Prepare

To address VQL, it is imperative to encode and represent all regions/objects in each frame of a video. However, creating patch-level features for each frame results in an excessive number of tokens for long videos, making the task computationally expensive. To address this issue, we propose using region-based representations [36] to encode each frame. As shown in Table 2, the region-based approach significantly reduces the number of tokens needed to encode a video. Moreover, this produces semantically meaningful object tokens, simplifying entity search for a given query.

Building on this insight, RELOCATE prepares object tokens from a given video as follows:

**1. Segment Objects.** We first generate a binary mask for each region/object in all frames of the video via a segmentation model. Formally, for a frame $f_t \in \mathbb{R}^{H\times W\times 3}$, we extract a set of binary masks $M_t = \{m_{t1}, m_{t2}, \dots\}$, one for each region. Here, $m_{ti} \in \mathbb{R}^{H\times W}$ is a binary mask with 1 in the area occupied by the object $i$ and 0 otherwise. In this work, we use SAM ViT-H [17] as the segmentation model, which processes each frame at a resolution of $1024 \times 1024$.

**2. Extract Features.** We then extract dense features from every frame of the video via a pretrained vision model. Formally, for every frame $f_t \in \mathbb{R}^{H\times W\times 3}$ we compute a high-dimensional feature map $h_t \in \mathbb{R}^{h\times w\times d}$. In this work, we experiment with different feature extractors and choose DINO ViT-B/8 [6] with a frame resolution of $384 \times 512$.

**3. Resize and Pool.** Subsequently, we resize the frame features $h_t \in \mathbb{R}^{h\times w\times d}$ to $\overline{h}_t \in \mathbb{R}^{H\times W\times d}$ to fit the height and width of the masks. We then pool the features within each object mask $m_{ti}$ to get an object token $o_{ti} \in \mathbb{R}^d$. Following the insights from [36], we use bilinear interpolation to resize the features and average pooling to aggregate the features within the mask.

### 3.2. Search

Videos spanning tens of minutes and processed at 5 FPS typically involve more than 150,000 object tokens, with only about 0.01% corresponding to the target response track. To find candidates similar to the visual query, RELOCATE searches over the object tokens. Concretely, given an image and a bounding box localizing the query object, we extract the query token via the process used for video preparation and proceed with the following steps:

**1. Compute Similarity.** We compute pairwise cosine similarity between all object tokens and query tokens. Initially, only one query token is generated from the given visual query, but multiple query tokens can be created for the same object. When multiple query tokens are present, we use the maximum similarity score across all tokens as the object score, ensuring that an object is selected if it matches any view of the query. Formally, for the $i^{\text{th}}$ object token in frame $t$ (*i.e.*, $o_{ti}$) and $m$ query tokens $[q_j]_{j=1}^m$, the similarity score

$s_{ti}$ is computed via

$$s_{ti} = \max_{j \in \{1,\dots,m\}} \frac{o_{ti} \cdot q_j}{\|o_{ti}\| \|q_j\|}. \quad (1)$$

We also studied test-time training (TTT) [38], but it performed worse than a simple cosine similarity search due to an insufficient number of query tokens for effective training.

**2. Perform Intra-Frame NMS.** Given that each frame can contain at most one instance of the query object, we apply non-maximum suppression (NMS) to retain only the highest-scoring object per frame, setting the scores of all other objects to zero. This process results in a single candidate object per frame, yielding a total number of candidate objects equal to the number of frames in the video.

**3. Perform Inter-Frame NMS.** Since we perform bidirectional tracking at a later stage, selecting one candidate from a potential track is sufficient. So, we perform NMS across consecutive frames to select the highest-scoring match for a query object, suppressing lower-scoring instances in neighboring frames. More precisely, we iteratively select the object with the maximum score and nullify candidate scores in preceding and subsequent frames until the score drops below 80% of this peak score. As a result, we end up with sparsely selected high-scoring objects across video frames.

**4. Select Candidates.** Finally, we select up to $k = 10$ candidate objects that exceed a $t_{\text{sim}} = 0.7$ similarity threshold, yielding an average of 7.5 candidates per query.

### 3.3. Refine

The objects of interest are often small and situated in cluttered scenes, making it challenging for the segmentation model and feature extractor to produce high-quality object tokens. To address this, we propose refining the selections made after the initial search. We find that this refinement significantly enhances the spatial precision of RELOCATE and helps filter out any spurious selections made after the initial search (see Section 4.3 for an ablation).

We refine the candidates selected from the search in conjunction with the query to generate a more precise set of candidates. This is done as follows:

**1. Get Object-Centric Crop.** We crop the video frames containing the selected candidates and visual queries such that these objects occupy a larger area at the center of the cropped view. These crops are then resized to the original frame dimensions. To prevent pixelation in the case of extremely small objects, we ensure that no frame is cropped with a zoom factor exceeding 2.5 times.

**2. Generate Refined Token.** We process the object-centric crops of candidates and queries using the segmentation model and feature extractor to generate refined object and query tokens. This process yields improved tokens because the segmentation model and feature extractor can produce more accurate masks and detailed features for the objects from the object-centric crop.

**3. Recompute Similarity and Filter.** We recompute the cosine similarity scores for the selected candidates using the refined object tokens and query tokens, and filter out candidates that have a score below threshold $t_{\text{sim}} = 0.7$.

### 3.4. Track

RELOCATE uses an off-the-shelf tracker to bidirectionally track the refined candidate that shows up last in the video, yielding the response track $\hat{\mathcal{R}} = \{b_s, b_{s+1}, \dots, b_e\}$. Further, it assigns the similarity score of the candidate as the track score. In this work, we use SAM 2 [29] for tracking.

### 3.5. Reiterate

The query object often appears multiple times in a given video, and RELOCATE is highly successful at localizing at least one of these appearances from a single visual query (see Section 4.1). However, our goal is to identify the *latest* appearance, which may be heavily occluded or seen from a significantly different viewpoint compared to the visual query. To achieve this, we generate additional visual queries using the response track predicted with the original query and reiterate the process of search, refinement, and tracking. This expanded pool of queries offers diverse views of the object, enhancing the likelihood of detection even when it is obscured or appears in a radically altered form. In practice, we perform this query expansion and reiteration step only once. This is carried out as follows:

**1. Generate Query Tokens.** Given the response track $\hat{\mathcal{R}} = \{b_s, b_{s+1}, \dots, b_e\}$, we apply the segmentation model and feature extractor to produce region tokens for the objects within the bounding boxes across the frames comprising the response track.

**2. Filter Queries.** We filter out low-quality query tokens generated in the previous step, specifically targeting three types: (1) query tokens with very low cosine similarity (less than 0.5) to the original query token, (2) queries associated with extremely small bounding boxes (occupying less than 0.07% of the frame area), and (3) queries derived from blurry frames (indicated by a Laplacian operator variance below 100).

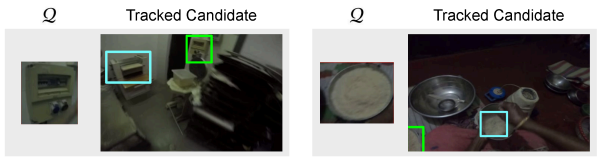
**3. Search, Refine, and Track.** After expanding the query pool, we search the video segment following the last frame of the previously predicted response track. The search results are then refined, and the latest refined candidate is tracked across frames to generate a new prediction. If the score of the new prediction exceeds a threshold relative to the previous track score, we update the previous prediction.

## 4. Experiments

In this section, we evaluate RELOCATE (Section 4.1), discuss key design decisions (Section 4.2), and present ablation studies (Section 4.3).

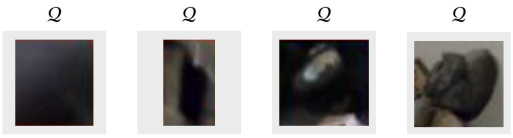
| Method | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| *Validation Set* | | | | |
| SiamRCNN [12] | 0.15 | 0.22 | 43.2 | 32.9 |
| NFM [39] | 0.19 | 0.26 | 47.9 | 37.9 |
| CocoFormer [40] | 0.19 | 0.26 | 47.7 | 37.7 |
| VQLoC [40] | 0.22 | 0.31 | 55.9 | 47.1 |
| **RELOCATE** | **0.33** | **0.41** | **58.0** | **50.5** |
| *Test Set* | | | | |
| SiamRCNN [12] | 0.13 | 0.21 | 41.6 | 34.0 |
| CocoFormer [40] | 0.18 | 0.26 | 48.1 | 43.2 |
| VQLoC [40] | 0.24 | 0.32 | 55.9 | 45.1 |
| **RELOCATE** | **0.35** | **0.43** | **60.1** | **50.6** |

Table 1. **Results on Ego4D VQ2D benchmark.** The validation results are taken from [40], and the test results are obtained from the challenge leaderboard.




(a) An object visually similar to the query (blue boxes) is mistakenly tracked instead of the correct query object (green boxes).


(b) The visual query shows the target object from a distinctly different viewpoint compared to its single appearance in the video (green boxes).




(c) These visual queries lack discriminative features due to motion blur, low contrast with background, or poor visibility.

Figure 3. **Examples where a wrong object is localized.** Among 100 randomly sampled instances from the VQ2D validation set, RELOCATE localized an instance of the query object in 93 cases, and failed to localize the correct object only under extreme conditions.

## 4.1. Evaluation

**Dataset.** We evaluate RELOCATE on the Ego4D VQ2D dataset [12], a large collection of egocentric videos annotated for VQL within episodic memory. On average, the videos in this dataset are 140 seconds long, and the target response tracks last for roughly 3 seconds. The dataset comprises 13,600 training, 4,500 validation, and 4,400 test queries, annotated across 262, 87, and 84 hours of video, respectively. We use the validation set for our development and ablations. To our knowledge, VQ2D is the only publicly available dataset for VQL.

**Metrics.** Following the official metrics outlined by the benchmark, we report spatio-temporal average precision ($\text{stAP}_{25}$), temporal average precision ($\text{tAP}_{25}$), success, and recovery. $\text{stAP}_{25}$ and $\text{tAP}_{25}$ evaluate the accuracy of the predicted response tracks' spatio-temporal and temporal extents using an Intersection over Union (IoU) threshold of 0.25. Success measures whether the IoU between predictions and ground truth exceeds 0.05, and recovery measures the proportion of predicted frames where the bounding box achieves an IoU of at least 0.5 with the ground truth.

**Results.** Table 1 shows results on the validation and test sets. Despite no task-specific training, RELOCATE outperforms the next-best baseline by 49% $\text{stAP}_{25}$, 33% $\text{tAP}_{25}$, 8% Success, and 12% Recovery on the test set. Note, these are relative improvements. Figure 1 shows some qualitative examples of RELOCATE in action.

Additionally, a manual analysis of 100 randomly sampled examples from the VQ2D validation set shows that RELOCATE successfully localizes the last occurrence of the object in 61 cases, localizes an earlier instance in 32 cases, and identifies the wrong object in 7 cases. Figure 3 categorizes these failure modes for the cases where RELOCATE localizes the wrong object, highlighting that the framework typically fails to localize the correct object only in extreme situations, such as when the visual query is ambiguous or there exists another object that closely resembles the query.

**Timing Analysis.** On average, preparing a 1000-frame video (*i.e.*, extracting object tokens) takes 1422.5 seconds with our current setup. Subsequent steps are faster: search takes 0.8 seconds, refinement 12.6 seconds, tracking 26.3 seconds, and generating additional queries from a response track is instantaneous. Note that our current implementation is not optimized for speed. For applications requiring faster localization, several simple optimizations can be made with minimal impact on the metrics, such as using HNSW [20] for search, increasing the batch size for feature extraction, using faster SAM variants [27] for region generation, and employing faster trackers. We leave these enhancements for future work.

## 4.2. Design Decisions

**Region vs. Patch Representations.** Most contemporary approaches to object localization and tracking encode video frames using patch-based representations, where each frame is divided into non-overlapping patches and encoded with a vision transformer. Region-based representations derived from pretrained vision models [36] have recently been proposed as an efficient alternative. These representations offer both semantic richness and compact encoding—qualities that align with the requirements of VQL.

Notably, region token count depends on the scene content (detected objects or regions) rather than spatial param-

| Encoding Method | Number of tokens | |
|---|---|---|
| | 384 × 512 Frames | 1024 × 1024 Frames |
| Patches | 3,072,000 | 16,384,000 |
| **Regions** | **130,507** | **130,507** |

Table 2. **Token count for patch-based and region-based encodings.** Region-based representations typically use fewer tokens than the patch-based method, and their token count does not scale with resolution. Token counts are reported for a 1000-frame video, using a patch size of 8 and SAM ViT-H for region-based encoding.

eters (image dimensions and patch size). As a result, in most practical cases, a region-based approach yields a much lower token count than the patch-based approach. As shown in Table 2, a region-based encoding of a 200-second VQ2D video produces 24× fewer tokens compared to a patch-based encoding at an image resolution of $384 \times 512$ with a patch size of 8. Moreover, this difference drastically increases if we increase the resolution; *e.g.* if the image resolution is increased to $1024 \times 1024$, a region-based encoding produces 126× fewer tokens than a patch-based encoding. Thus, the decoupling of token count from spatial parameters allows the region-based approach to leverage the benefits of smaller patches and high-resolution images without incurring the computational overhead typically associated with increased token counts.

Beyond their compactness, region-based representations excel at capturing semantic information, enabling spatio-temporal object localization through simple token similarity matching. Figure 4 illustrates this through cosine similarity heatmaps, where region-based representations show precise areas of high similarity for the query objects, while patch-based representations result in less focused similarity distributions. Our comparative analysis reveals that while patch encodings achieve better frame-level retrieval (65% vs. 60%), they perform significantly worse at spatial localization (42% vs. 58%) compared to region-based representations. For specific applications, one could combine patch-based frame retrieval with region-based object localization. For episodic memory tasks, however, we believe that a streamlined approach using only region-based representations is advantageous as it avoids the computational overhead of storing and searching through substantially more patch tokens and eliminates the need to generate regions during retrieval.

**Feature Extraction.** To search and retrieve objects that match a given query, VQL requires capturing fine-grained details from each video frame, and the DINO models are particularly well-suited for this task [14, 36]. We compare DINO [6] and DINOv2 [26] using various ViT backbones, with results shown in Table 3. Our findings indicate that DINO ViT-B/8 outperforms ViT-B/16, likely due to its smaller patch size enabling finer feature extraction. Additionally, DINOv2 ViT-L/14 shows stronger frame-level retrieval, and DINO ViT-B/8 is better at spatial localization of objects. We choose DINO ViT-B/8 as our feature extractor due to its higher spatio-temporal localization success rate, smaller model size for faster inference, and reduced feature dimensionality requiring less memory.

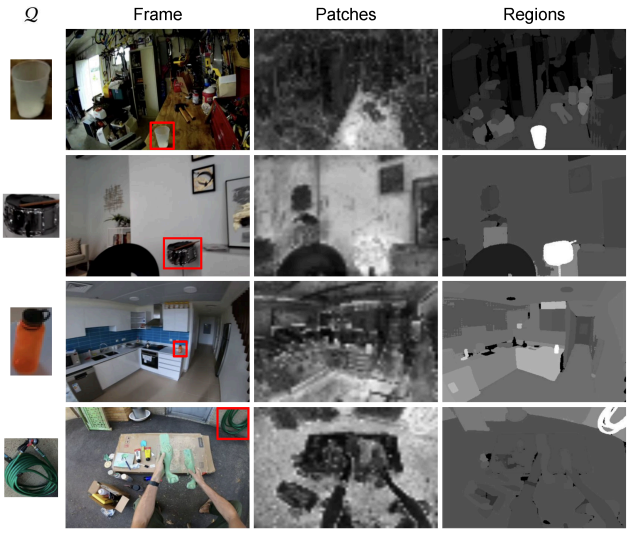



Figure 4. **Comparing cosine similarity heatmaps for patch-based and region-based representations.** Region-based representations yield distinct, high-similarity matches for query objects (brighter regions), while patch-based representations produce more diffuse similarity patterns.

| Extractor | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| DINO ViT-B/16 | 0.254 | 0.301 | 50.0 | 47.7 |
| DINOv2 ViT-L/14 | **0.331** | **0.452** | 55.8 | 45.3 |
| **DINO ViT-B/8** | **0.333** | 0.409 | **58.0** | **50.5** |

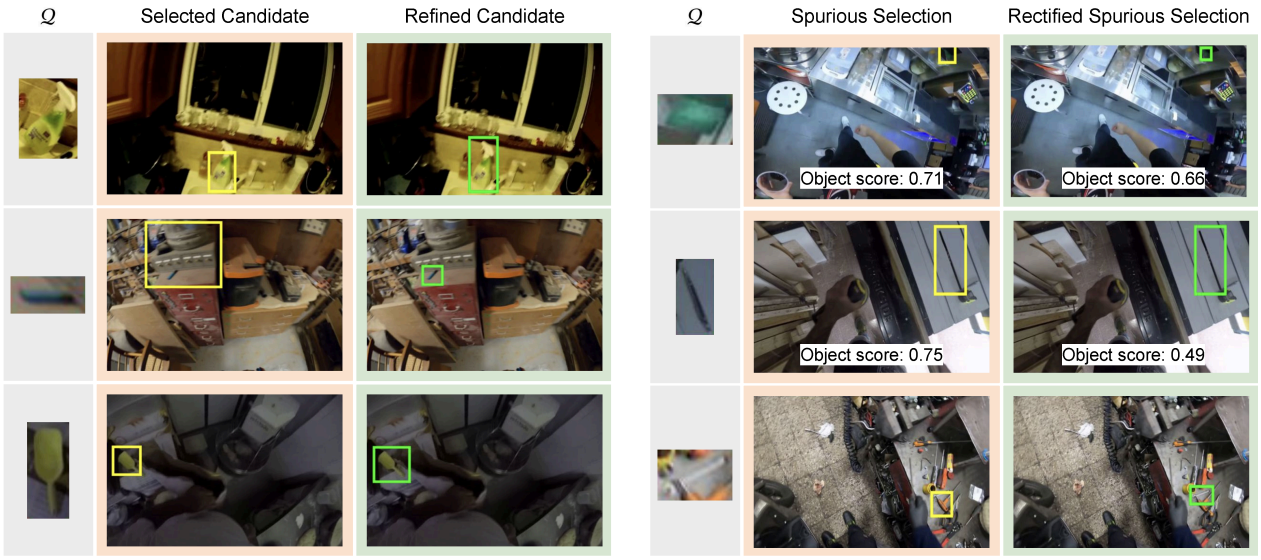
Table 3. **Performance comparison of feature extractors.** DINOv2 ViT-L/14 shows superior frame-level retrieval while DINO ViT-B/8 is better at spatial localization.

| Method | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| RELOCATE-NoRefine | 0.246 | 0.364 | 49.1 | 39.8 |
| RELOCATE-NoReiter | 0.247 | 0.293 | 50.9 | 48.2 |
| **RELOCATE** | **0.333** | **0.409** | **58.0** | **50.5** |

Table 4. **Impact of refining search results and reiterating after query expansion.** Ablating either refinement (RELOCATE-NoRefine) or reiteration (RELOCATE-NoReiter) significantly reduces the performance of RELOCATE.

### 4.3. Ablation Study

**Refining Search Results.** Results in Table 4 show that removing search refinement (RELOCATE-NoRefine) from the framework significantly degrades RELOCATE's performance. As illustrated in Figure 5a, refinement enhances spatial precision for localizing small objects and fine parts




(a) Cropping around objects during refinement helps capture small objects or object parts that full-frame processing might miss.

(b) Refinement fixes incorrect selections due to feature bleeding (row 1) and removes false selections by lowering their scores (rows 2 and 3).

Figure 5. **Impact of refining search results.** Initial candidates are shown in yellow boxes, and refined candidates are shown in green.

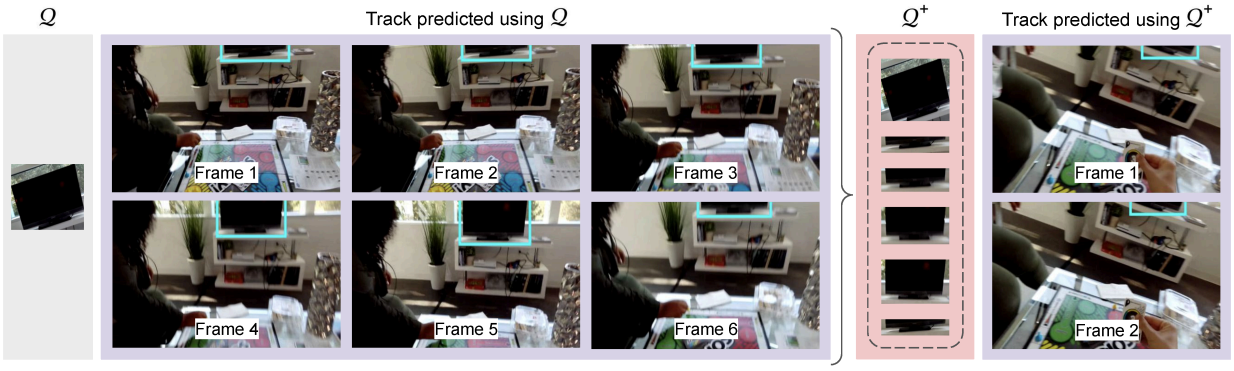



Figure 6. **Impact of query expansion and reiteration.** Using only the initial query $\mathcal{Q}$, RELOCATE misses the object's final appearance due to its partial view and low feature similarity. However, the expanded query set $\mathcal{Q}^+$ captures multiple object perspectives, enabling successful detection of the final occurrence.

by generating regions from an object-centric crop that captures more of the object of interest. Additionally, refinement removes incorrect selections from the initial search by rescoring candidates using region tokens generated from an object-centric crop. Furthermore, for small or thin objects, feature bleeding can result in the selection of an object adjacent to the target object. Object-centered cropping can also help avoid this by providing a clearer view of the target. This is demonstrated in Figure 5b.

**Query Expansion and Reiteration.** Results in Table 4 show that query expansion and reiteration significantly improve RELOCATE's performance compared to the variant without this step (RELOCATE-NoReiter). Figure 6 shows an example where query expansion using an initial prediction helps capture a wider range of views for the visual query. This, in turn, helps in successfully localizing the final occurrence of the object that was missed when only the initial visual query was used. Additional examples where reiteration after query expansion leads to successful localization are illustrated in rows 2, 5, and 6 of Figure 1.

## 5. Conclusion

We present RELOCATE, a framework utilizing region-based representations from pretrained vision models to address VQL. Despite a simple design and no task-specific training, RELOCATE can localize target objects in long videos, even under challenging conditions such as clutter, occlusion, blur, and viewpoint changes. On the VQ2D benchmark, it significantly outperforms existing methods that are specifically trained on this dataset.

## References


[1] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *CVPRW*, 2014. 3

[2] Artem Babenko and Victor S. Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[3] Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 3

[4] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and A. Bimbo. Effective conditioned and composed image retrieval combining clip-based features. In *CVPR*, 2022. 3

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ArXiv*, 2020. 1

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 3, 4, 7

[7] Vijay Chandrasekhar, Jie Lin, Olivier Morère, Hanlin Goh, and Antoine Veillard. A practical guide to cnns and fisher vectors for image instance retrieval. *Signal Processing*, 2016. 3

[8] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 3

[9] Yanbei Chen and Loris Bazzani. Learning joint visual semantic matching embeddings for language-guided retrieval. In *ECCV*, 2020. 3

[10] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *ArXiv*, 2023. 3

[11] Yutao Cui, Jiang Cheng, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *CVPR*, 2022. 3

[12] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh K. Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Z. Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen, Abrham Kahsay Gebreselasie, Cristina González, James M. Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jáchym Kolár, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran K. Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbeláez, David J. Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard A. Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2021. 1, 2, 3, 6

[13] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 2014. 3

[14] Dongsheng Jiang, Yuchen Liu, Songlin Liu, Jin'e Zhao, Hao Zhang, Zhen Gao, Xiaopeng Zhang, Jin Li, and Hongkai Xiong. From clip to dino: Visual encoders shout in multimodal large language models. *ArXiv*, 2023. 7

[15] Hanwen Jiang, Santhosh K. Ramakrishnan, and Kristen Grauman. Single-stage visual query localization in egocentric videos. *ArXiv*, 2023. 1, 2

[16] Zdenek Kalal. Tracking-learning-detection. *PAMI*, 2012. 3

[17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *ICCV*, 2023. 3, 4

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 3

[19] Fan Ma, Mike Zheng Shou, Linchao Zhu, Haoqi Fan, Yilei Xu, Yi Yang, and Zhicheng Yan. Unified transformer tracker for object tracking. In *CVPR*, 2022. 1, 3

[20] Yury Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *PAMI*, 2016. 6

[21] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *CVPR*, 2022. 3

[22] Christoph Mayer, Martin Danelljan, Ming-Hsuan Yang, Vittorio Ferrari, Luc Van Gool, and Alina Kuznetsova. Beyond sot: Tracking multiple generic objects at once. In *WACV*, 2022. 1

[23] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*, 2021. 3

[24] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006. 3

[25] Stepán Obdrzálek and Jiri Matas. Sub-linear indexing for large scale object recognition. In *BMVC*, 2005. 3

[26] Maxime Oquab, Timoth'ee Darcet, Théo Moutakanni, Huy Q. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Huijiao Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *ArXiv*, 2023. 3, 7

[27] PyTorch.org. Accelerating generative ai with pytorch: Segment anything, fast. https://pytorch.org/blog/accelerating-generative-ai/, 2023. Accessed: 2024-11-11. 6

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3

[29] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya K. Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross B. Girshick, Piotr Doll'ar, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *ArXiv*, 2024. 3, 5, 2

[30] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *CVPRW*, 2014. 3

[31] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. A baseline for visual instance retrieval with deep convolutional networks. In *ICLR*, 2014. 3

[32] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *PAMI*, 2015. 1

[33] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *IJCV*, 2008. 3

[34] Aneeshan Sain, Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Subhadeep Koley, Tao Xiang, and Yi-Zhe Song. Clip for all things zero-shot sketch-based image retrieval, fine-grained or not. In *CVPR*, 2023. 3

[35] Kuniaki Saito, Kihyuk Sohn, Xiang Zhang, Chun-Liang Li, Chen-Yu Lee, Kate Saenko, and Tomas Pfister. Pic2word: Mapping pictures to words for zero-shot composed image retrieval. In *CVPR*, 2023. 3

[36] Michal Shlapentokh-Rothman, Ansel Blume, Yao Xiao, Yuqun Wu, TV Sethuraman, Heyi Tao, Jae Yong Lee, Wilfredo Torres, Yu-Xiong Wang, and Derek Hoiem. Region-based representations revisited. In *CVPR*, 2024. 3, 4, 6, 7

[37] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. In *ICCV*, 2003. 3

[38] Yu Sun, X. Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2019. 5

[39] Mengmeng Xu, Cheng-Yang Fu, Yanghao Li, Bernard Ghanem, Juan-Manuel Pérez-Rúa, and Tao Xiang. Negative frames matter in egocentric visual query 2d localization. *ArXiv*, 2022. 1, 3, 6

[40] Mengmeng Xu, Yanghao Li, Cheng-Yang Fu, Bernard Ghanem, Tao Xiang, and Juan-Manuel Pérez-Rúa. Where is my wallet? modeling object proposal sets for egocentric visual query localization. In *CVPR*, 2022. 1, 3, 6

[41] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021. 1, 3, 2

[42] Jinyu Yang, Mingqi Gao, Zhe Li, Shanghua Gao, Fang Wang, and Fengcai Zheng. Track anything: Segment anything meets videos. *ArXiv*, 2023. 3

# RELOCATE: A Simple Training-Free Baseline for Visual Query Localization Using Region-Based Representations

## Supplementary Material

| k | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| 5 | 0.302 | 0.371 | 56.5 | 49.9 |
| 10 | 0.333 | 0.409 | 58.0 | 50.5 |
| 25 | 0.329 | 0.404 | 58.2 | 50.6 |
| 50 | 0.330 | 0.409 | 58.5 | 50.8 |

Table 5. **Effect of initially selected candidates on model performance.** Our final evaluations use $k = 10$.

| $t_{\text{sim}}$ | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| 0.6 | 0.348 | 0.446 | 58.4 | 47.8 |
| 0.7 | 0.333 | 0.409 | 58.0 | 50.5 |
| 0.8 | 0.258 | 0.316 | 52.9 | 48.0 |

Table 6. **Effect of candidate selection threshold on model performance.** Our final evaluations use $t_{\text{sim}} = 0.7$.

| $t_{\text{nms}}$ | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| 0.6 | 0.308 | 0.379 | 57.1 | 50.9 |
| 0.7 | 0.320 | 0.393 | 57.8 | 51.0 |
| 0.8 | 0.333 | 0.409 | 58.0 | 50.5 |
| 0.9 | 0.324 | 0.404 | 58.3 | 50.8 |

Table 7. **Effect of inter-frame NMS threshold on model performance.** Our final evaluations use $t_{\text{nms}} = 0.8$.

| $t_{\text{q}}$ | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| 0.4 | 0.320 | 0.402 | 58.2 | 50.2 |
| 0.5 | 0.333 | 0.409 | 58.0 | 50.5 |
| 0.6 | 0.320 | 0.396 | 58.0 | 50.4 |

Table 8. **Effect of query selection threshold on model performance.** Our final evaluations use $t_{\text{q}} = 0.5$.

| Method | $\text{stAP}_{25}$ | $\text{tAP}_{25}$ | Success | Recovery |
|---|---|---|---|---|
| SAM 2 [29] | 0.290 | 0.329 | 55.0 | 42.7 |
| **RELOCATE** | **0.378** | **0.458** | **63.0** | **49.1** |

Table 9. **Evaluating SAM 2 on VQ2D.** Here, we evaluate on 100 randomly sampled examples from the VQ2D validation set.

| Category | SAM 2 | RELOCATE |
|---|---|---|
| Last occurrence localized | 54 | 61 |
| Prior occurrence localized | 24 | 32 |
| Wrong object localized | 18 | 7 |
| No track returned | 4 | 0 |

Table 10. **Response track prediction analysis of SAM 2 and RELOCATE.** We compare the predictions of SAM 2 and RELOCATE on 100 sampled examples from the VQ2D validation set. Predictions are categorized into four types, and the count for each category is reported.

This supplementary material is structured as follows. In Appendix A we analyze the sensitivity of RELOCATE to its hyperparameters. In Appendix B we study the performance of SAM 2 on the VQL task.

## A. Hyperparameter Sensitivity Analysis

We analyze RELOCATE's sensitivity to four key hyperparameters: (1) the maximum number of initially retrieved candidates $k$, (2) the candidate selection threshold $t_{\text{sim}}$, (3) the inter-frame NMS threshold $t_{\text{nms}}$, and (4) the query selection threshold $t_{\text{q}}$. Tables 5-8 and Figure 7 present model's performance across different hyperparameter configurations.

For the initial retrieval count $k$, we observe stable performance across values from 10 to 50, with only a slight degradation at $k = 5$. The candidate selection threshold $t_{\text{sim}}$ leads to a noticeable decline in performance when set above 0.7. The inter-frame NMS threshold $t_{\text{nms}}$ demonstrates consistent performance across the range 0.7-0.9, suggesting robustness to this parameter. Similarly, the query selection threshold $t_q$ shows minimal variation in performance between 0.4 and 0.6.

Overall, these results indicate that our model maintains stable performance across a wide range of hyperparameter values, with selected values of $k = 10$, $t_{\text{sim}} = 0.7$, $t_{\text{nms}} = 0.8$, and $t_{\text{q}} = 0.5$ providing a robust operating point.

## B. Evaluating SAM 2 on VQ2D

Jiang et al. [15] demonstrated significant limitations in VQL capabilities among contemporary tracking systems. Specifically, they showed that STARK [41], a state-of-the-art visual tracker at the time, achieves only a 0.04 $\text{stAP}_{25}$ score on the VQ2D validation set. Since then, tracking systems have advanced considerably. To evaluate the capabilities of current tracking systems, we test SAM 2 [29] on the VQL task.

To adapt SAM 2 for VQ2D, we prepend the query frame to the target video and use the query bounding box from the annotations as the prompt for mask generation. SAM 2 then propagates the generated mask across all subsequent frames, tracking multiple occurrences of the query object. We select the last contiguous track as the response track prediction.

We evaluate SAM 2 on 100 randomly sampled examples

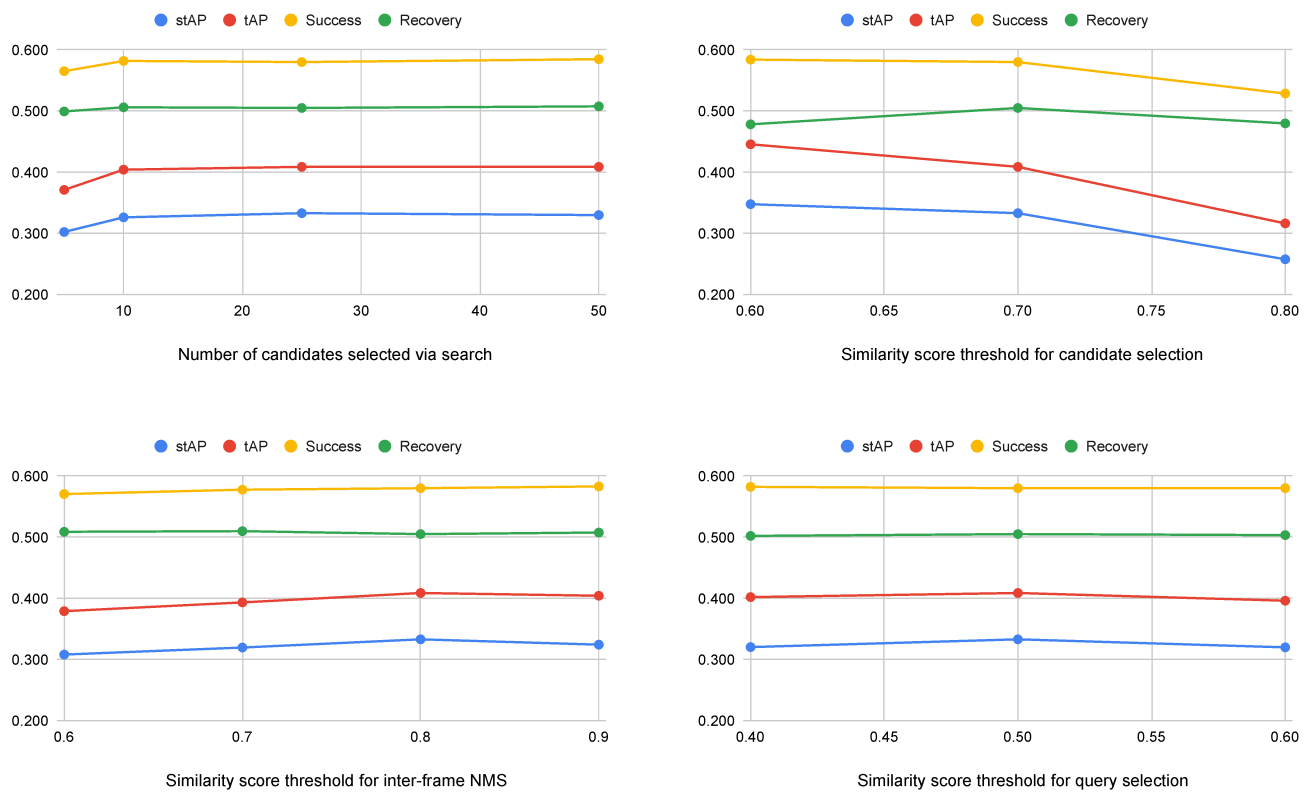



Figure 7. **Hyperparameter sensitivity analysis of** RELOCATE**.** Empirical evaluation demonstrates RELOCATE's robustness across different hyperparameter configurations.

previously used for the manual analysis of RELOCATE reported in Section 4.1, and the results are shown in Tables 9 and 10. While SAM 2 shows competitive performance on VQ2D (Table 9), it underperforms compared to RELOCATE. Our qualitative analysis (Table 10) reveals that SAM 2 has a higher tendency to localize incorrect objects or produce no tracks compared to RELOCATE. On an NVIDIA A40, with our implementation, SAM 2 takes an average of 110.7 seconds to locate a query object in a 1000-frame video. In comparison, RELOCATE incurs a one-time cost of 1422.5 seconds to prepare a 1000-frame video, followed by 73.6 seconds to process each query. However, the processing time of RELOCATE can be significantly reduced by using batch processing and faster SAM variants.