UGO: Unified Architecture for General Multi-Object Tracking by Segmentation

Jer Pelhan, Alan Lukežič, Matej Kristan
Faculty of Computer and Information Science, University of Ljubljana
jer.pelhan@fri.uni-lj.si

Abstract

General multi-object tracking (GMOT) tracks all instances of a user-specified category from a single first-frame exemplar. Prior work relies on bounding boxes and surrogate training, and struggles with non-rigid objects, crowded scenes, and distractors. We introduce UGO, a unified GMOT tracker that pairs a pretrained exemplar-conditioned detection head with an instance-propagation head in a common architecture. A novel training-free, energy-minimization consolidation method converts overlapping proposals into exclusive pixel-wise masks and detections, resolving over-segmentation, duplicates, and conflicts. A hierarchical memory spanning global and instance levels improves recall and per-instance segmentation accuracy using a new memory management protocol. UGO sets a new state-of-the-art on GMOT benchmarks and video object counting, and is competitive with specialist MOT methods, establishing a strong paradigm for unified, open-category multi-object tracking. The code is available at https://github.com/jerpelhan/UGO.

1 Introduction

Given a single instance exemplar, e.g., specified by a bounding box in the first frame, the task of general multi-object tracking (GMOT) is to track *all instances of the same category* throughout the video. This extends two classical problems: single-object tracking (SOT) [16], which extracts a trajectory of a single selected instance of an arbitrary category, and multi-object tracking (MOT) [13], which detects and associates multiple instances of a predefined class using a category-specific detector. GMOT thus unifies the generalization requirement of SOT with the coverage of MOT, while operating without category-specific training data, making it a fundamentally challenging problem.

Most GMOT methods [15; 38; 41; 12] are direct adaptations of the classical online MOT paradigm, combining a detector, Kalman-based instance propagation, and Hungarian matching to associate detections with instance predictions. Alternatively, tracking and detection association is posed as an end-to-end trainable detection problem [23] in the context of query-based detectors [10]. Since sufficiently diverse GMOT training data is unavailable, these methods rely on surrogate training on detection datasets, which cannot capture the complex dynamics of instance interactions. Moreover, all current methods localize targets as bounding boxes, which may be adequate for approximating pedestrians, but poorly approximate general non-compact and deformable objects, making the association unreliable, particularly in crowded scenes with overlapping instances.

In general, all instances are better represented by segmentation masks, which enable pixel-level instance separation. This is explored in video object segmentation [42], where masks of all tracked instances are jointly inferred. However, architectures for joint instance prediction do not scale to a large number of objects and require dedicated training sets. Alternatively, recent video foundation models [33] hold the potential to independently track all instances, but these become unreliable under strong instance interactions and require external initialization at each new instance.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

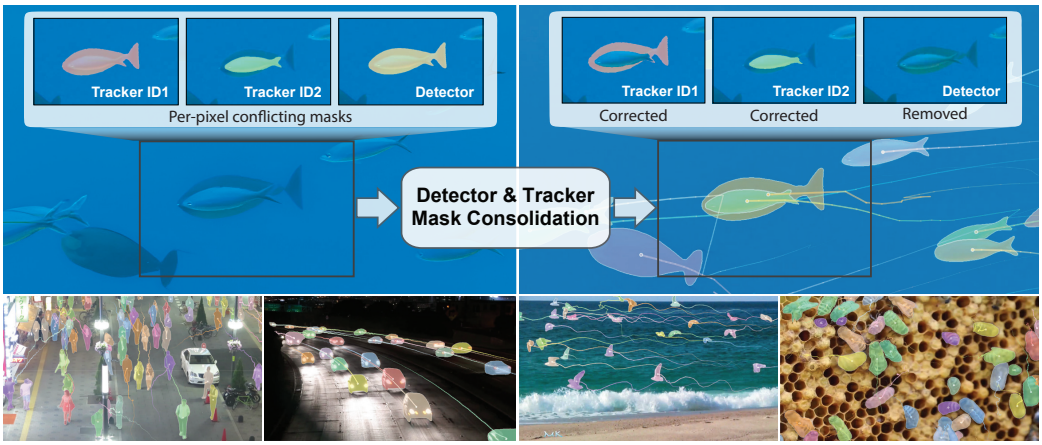

Figure 1: First row: Tracker ID=1 oversegments an occluding fish tracked by ID=2, while a single detector fires on both. UGO resolves the oversegmentations and redundant detection by a novel consolidation module. Second row: The new paradigm enables robust general multi-object tracking by segmentation in cluttered dense scenes through occlusions.

We address the aforementioned issues, by proposing UGO, a Unified General multiple Object tracker, that unifies recent advances in general object detection and the expressiveness of the video segmentation foundation models within a single architecture. UGO applies a video foundation backbone [33] with two lightweight heads, one for detecting instances corresponding to the user-provided exemplar, and another for frame-to-frame identity propagation. Both use the same segmentation module to predict calibrated per-pixel object presence beliefs, which may contain oversegmentation, double detections, and duplicated tracks (Figure 1). These are resolved into unique, pixel-wise exclusive tracked instance segmentations and detections of new instances by a novel training-free consolidation module, formulated as an energy-minimization problem with a global exclusivity objective. The novel design avoids the need for extensive, diverse GMOT-specific training datasets to solve complex combinatorial optimization problems efficiently. To ensure robustness to distractors and to align the detector with within-category instance appearance variation, we propose a hierarchical adaptive memory (HAM) composed of a detector category-level memory and individual per-instance memories. The memories are updated by jointly considering the tracks of all instances. UGO shows insensitivity to the hyperparameters commonly applied in MOT/GMOT settings, confirming its robust design. Our contributions are: (i) a new general multi-object tracking paradigm with independently applied single-object trackers and detectors, coupled through mask consolidation and memory feedback to achieve a global coordination; (ii) a novel training-free energy-based consolidation module that delivers per-pixel exclusive segmentation masks from per-instance tracks and detections; (iii) a hierarchical adaptive memory (HAM) with a management protocol based on explicit failure detection, ensuring robust instance detection and identity propagation in the presence of distractors. UGO unifies SOT, MOT, and GMOT in a single framework, achieving state-of-the-art performance and defining a new paradigm in category-agnostic multi-object tracking.

2 Related Work

Multi-Object Tracking (MOT). In classical MOT pipelines such as SORT [5], DeepSORT [38], Tracktor [4], and ByteTrack [44], detectors produce per-frame boxes or masks, which are then associated by motion or appearance cues. Mask-level variants [37] (MOTS) extend this pipeline to segmentation. The methods excel in closed-world setups with strong class-specific detectors available, while their stability hinges on detector recall – missed detections lead to trajectory fragmentation – thus they underperform on long-tail categories. Recent works such as TrackFormer [27] and TransTrack [35] use DETR-like queries for joint detection and identity propagation. However, as noted in [14], this creates a conflict between category semantics (detection) and instance specificity (association). In addition, most methods require per-category training and lack pixel-level exclusivity across similar instances, causing overlaps and ID switches in crowded scenes.

General single object trackers (SOT). Modern SOT enable tracking of any instance and are highly robust to appearance changes and occlusion [33; 36; 11], while recent memory mechanisms improve reliability under distractors [36; 17]. However, SOT lacks scene-level competition and cannot ensure mutual exclusivity among many look-alikes; naively running multiple SOT instances on MOT datasets is prone to identity swaps. But SOT methods cannot discover new instances appearing in the video, and optimize per-target propagation rather than frame-level joint reasoning over all instances.

General & open-category tracking. General multi-object tracking (GMOT) extends MOT to tracking any instance of an unseen category, specified by a single instance exemplar. The GMOT-40 benchmark [3] established a standardized evaluation setting, demonstrating that a straightforward combination of a one-shot detector [15] and a target association module [32; 41; 6] provides a simple, yet limited baseline. Subsequent methods, including PLGMOT [25] and S-DETR [23], introduce exemplar-conditioned detection and transformer-based association. While PLGMOT enables instance discovery, its tracking-by-detection design limits recall. S-DETR improves this via exemplar-conditioned queries and dynamic track reuse. However, both remain restricted to box-level reasoning and NMS-based conflict resolution, causing identity switches in crowded scenes. Recently, SAM3 [9] extended single-object tracking toward open-vocabulary tracking, integrating detection capabilities, but it still lacks explicit pixel-level interaction reasoning among tracked instances.

3 A Unified General Multiple Object Tracker

Given a sequence of N video frames $\{\mathbf{I}_t\}_{t=1:N}$ and an instance category exemplar bounding box $\mathbf{b}_1^E$ provided in the first frame, the task is to track all instances of the same category in the video. We propose a unified general multiple object tracker (UGO) (Figure 2). At time-step t, UGO already tracks some of the instances from the previous frame and proceeds in the current frame as follows. The input image $\mathbf{I}_t \in \mathbb{R}^{H\times W\times 3}$ is encoded by the Hiera [34] backbone into $\mathbf{F}_t \in \mathbb{R}^{h\times w\times c}$, with spatial resolution $h \times w$ and c feature channels. The features are passed to two light-weight heads. The first is the SAM2 [33] video segmentation head that propagates tracked instances from the previous time-step (Section 3.1). The second head is based on GECO2 [30] and detects all new instances and potentially also those already tracked, matching the exemplar category (Section 3.2). Both heads apply a pretrained SAM2 mask decoder, delivering compatible per-pixel logits, which enables consolidation of detector and tracker outputs into mutually exclusive masks (Section 3.3). The consolidation influences memory updating: the resolved conflicts trigger the affected tracker memory updates. The individual trackers are thus implicitly coupled through shared pixel-level decisions, forming a feedback loop that reduces future overlaps and stabilizes identity tracking (Section 3.5). This substantially simplifies trajectory management (Section 3.4).

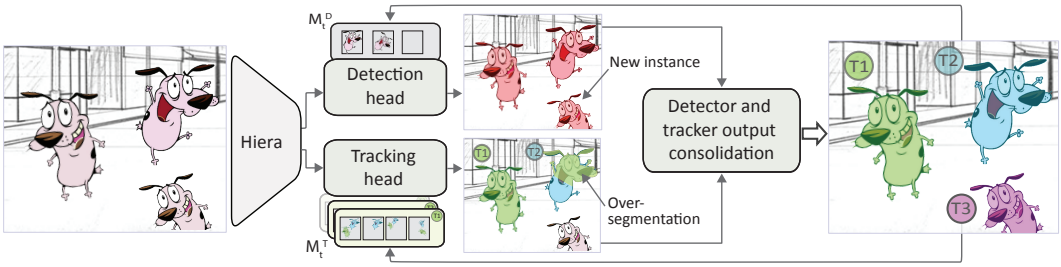

Figure 2: UGO unifies detection and frame-to-frame association in a single network. Detector and instance tracker heads produce compatible per-pixel outputs, which enables optimization-based, training-free consolidation and robust tracking.

3.1 Frame-to-frame instance propagation

A visual model $\mathcal{M}_{t-1,k}^T$ is maintained for each k-th tracked instance in form of a memory (Section 3.5). For each instance k, SAM2 head attends the backbone features to the corresponding memory and produces a logit map $\mathbf{T}_{t,k} \in \mathbb{R}^{H\times W}$, which reflects a belief that each pixel contains the instance, conditioned on its visual model. The output of the frame-to-frame propagation module is thus a set of per-instance tracker logits $\mathcal{T}_t = \{\mathbf{T}_{t,k}\}_{k=1:N_\mathcal{T}}$.

3.2 General instance localization

Since only a single user-provided exemplar of the tracked instance category is available in the first frame, all other instances (in the first frame and also entering in later frames) have to be detected to initialize new per-instance trackers. We utilize the general-object detector GECO2 [30], a light-weight detection head, operating on the SAM2 backbone. The object category is specified by exemplar memory $\mathcal{M}_t^D = \{\mathbf{b}_i^E\}_{i=1:N_B}$, which is initialized by the first-frame exemplar $\mathbf{b}_1^E$ and is updated during tracking (Section 3.5). The detection head localizes the instances and applies the SAM2 segmentation head to produce detection logits $\mathcal{D}_t = \{\mathbf{D}_{t,i}\}_{i=1:N_\mathcal{D}}$, where $\mathbf{D}_{t,i} \in \mathbb{R}^{H\times W}$ reflects the belief of each pixel belonging to the instance, conditioned on the memory.

3.3 Detector and tracker output consolidation

The use of the same pre-trained SAM2 segmentation head by the detector and tracker ensures calibrated and compatible logits in $\mathcal{T}_t$ and $\mathcal{D}_t$, which could be naïvely converted to masks by thresholding at zero. However, these masks indicate all pixels that *may* belong to the detector/tracker instance, not accounting for other instances. Thus, the masks will not be mutually exclusive, with some objects covered by both detectors and trackers, multiple trackers including distractor objects, and masks of different objects potentially overlapping due to over-segmentation (Figure 3). Because of these nontrivial interactions, the classical MOT bipartite [18; 7] or quadratic [39] optimization, even if redesigned to operate at the per-mask level, and disregarding the significant compute complexity, cannot be applied for conflict resolving.

To address these issues, we propose a new formulation that ensures mutually exclusive masks. We start by defining the logits tensor $\mathbf{L} \in \mathbb{R}^{H\times W\times S}$ obtained by concatenating the $N_\mathcal{D}$ detector and $N_\mathcal{T}$ tracker logit maps with a small constant background logit map, i.e., $\mathbf{L} = \mathrm{cat}(\mathcal{D}_t, \mathcal{T}_t, \mathbf{1}\lambda_{\mathrm{BG}})$. For notation compactness, let $\mathbf{L}(\mathbf{x}, s)$ retrieve the logit value of map s at pixel position $\mathbf{x}$. Next, let $\mathbf{Y} \in [1, \ldots, S]^{W\times H}$ denote the mutually-exclusive pixel labeling with $\mathbf{Y}(\mathbf{x})$ retrieving the logit map identity assigned to pixel $\mathbf{x}$.

The solution for $\mathbf{Y}$ should ensure (i) a unique label on each pixel, (ii) among the masks competing for the same pixels, the trackers should be preferred over detectors, and (iii) among the competing tracker masks, the trackers with masks supported by detections should be preferred. Such labeling can be obtained by minimizing the objective

$$\hat{\mathcal{L}}(\mathbf{Y}) = -\sum_{\mathbf{x}\in\Omega; s=\mathbf{Y}(\mathbf{x})} \log\Big(\mathbf{L}(\mathbf{x}, s)\cdot\Theta_1(s)\cdot\Theta_2(s)\Big), \tag{1}$$

where $\Omega = \{1, \ldots, H\}\times\{1, \ldots, W\}$ denotes the set of pixel positions, and $\Theta_1(\cdot)$ and $\Theta_2(\cdot)$ are unitary potentials enforcing the required properties (i-iii) of the optimal labeling. The first potential $\Theta_1(\cdot)$ is defined as

$$\Theta_1(s) = \begin{cases} \frac{1}{2}\theta_0 & ; s \le N_\mathcal{D} \\ \theta_0 + \max\limits_{d\in\mathcal{D}_t} \mathrm{IoU}\big(\mathbf{Y}\equiv s,\, d\big) & ; N_\mathcal{D} < s \le N_{\mathcal{DT}} \\ 1 & ; s \equiv S \end{cases} \tag{2}$$

where $N_{\mathcal{DT}} = N_\mathcal{D} + N_\mathcal{T}$, θ_0 is a constant enforcing preference of trackers over detections (θ_0 vs $\frac{1}{2}\theta_0$), and the IoU term prefers trackers better supported by the detections, assuming a labeling $\mathbf{Y}$. The second potential $\Theta_2(\cdot)$ reduces the scores of instances whose masks induced by labeling $\mathbf{Y}$ significantly deviate from their initially computed masks, indicating the other instances have absorbed their pixels,

$$\Theta_2(s) = \begin{cases} \mathrm{IoU}(\mathbf{Y}\equiv s, L(:, s) > 0) & ; s \le N_{\mathcal{DT}} \\ 1 & ; s \equiv S \end{cases}. \tag{3}$$

Since (1) is not separable in $\mathbf{Y}(\mathbf{x})$, we introduce auxiliary variables $\mathbf{A}_1 \in \mathbb{R}^{1\times S}$ and $\mathbf{A}_2 \in \mathbb{R}^{1\times S}$, leading to a surrogate objective

$$\mathcal{L}(\mathbf{Y}) = -\sum_{\mathbf{x}\in\Omega,\ s=\mathbf{Y}(\mathbf{x})} \log(\mathbf{L}(\mathbf{x}, s)\mathbf{A}_1(s)\mathbf{A}_2(s)) + \sum_{s=1}^{S} N_s\left[\log^2\frac{\mathbf{A}_1(s)}{\Theta_1(s)} + \log^2\frac{\mathbf{A}_2(s)}{\Theta_2(s)}\right], \tag{4}$$

with N_s the number of pixels assigned to label s, which can be optimized by MM-based [19] iterations of the following steps (see supplementary material for full derivation).

Assuming a constant $\mathbf{Y}^{(k-1)}$, estimated at previous step, taking derivatives of (4) w.r.t. $\log \mathbf{A}_1$ and $\log \mathbf{A}_2$, and equating to zero, leads to the update,

$$\mathbf{A}_1^{(k)}(s) \propto \Theta_{1|\mathbf{Y}^{(k-1)}}(s) \; ; \; \mathbf{A}_2^{(k)}(s) \propto \Theta_{2|\mathbf{Y}^{(k-1)}}(s). \tag{5}$$

Keeping $\mathbf{A}_1$ and $\mathbf{A}_2$ constant, the cost is separable across pixels, leading to maximization over $\mathbf{Y}$ as

$$\mathbf{Y}^{(k)}(\mathbf{x}) = \underset{s \in \{1:S\}}{\operatorname{argmax}} \big(\mathbf{L}(\mathbf{x}, s) \cdot \mathbf{A}_1^{(k)}(s) \cdot \mathbf{A}_2^{(k)}(s) \big). \tag{6}$$

The optimization thus gradually re-assigns pixels claimed by several detector/tracker instances to either of them, and if a particular instance is poorly supported by the remaining pixels, it is automatically turned off. The labels $\mathbf{Y}^{(0)}$ are initialized by per-pixel argmax over the logits $\mathbf{L}$, while the optimization converges within a few iterations.

Finally, severely reduced masks are removed: any label s in $\mathbf{Y}$, whose mask IoU with the initialization mask is smaller than a fixed threshold, is reassigned to the background, i.e., $s = S$. UGO uses a permissive threshold for trackers $\tau_{\text{lo}} = 0.2$ (since small overlap may be due to distractor correction), and, to ensure a high recall, we set the threshold for detector masks to a moderate level $\tau_{\text{md}} = 0.6$ (ablated in supplementary material).

3.4 Instance track lifecycle management

After consolidation, only two types of mutually-exclusive masks remain: (i) masks corresponding to instance trackers and (ii) masks corresponding to newly detected instances. Standard MOT rules are applied next for trajectory management.

Initialization & Update. New single-target trackers are initialized on detection masks, while the instance trackers are updated with their consolidated masks.

Termination. A tracker is flagged for termination when its mask collapses due to drift, or when the object leaves the field of view. In practice, the flag is raised when consolidation returns an empty mask. If the condition is met for ten consecutive frames, the tracker is terminated.

Initialized track validation. To remove tracks initialized on a false positive detection, they are classified as valid only after frequently confirmed by the detector. As a standard rule, tracks with 30% of their masks confirmed are declared valid, where a tracker mask $M_{t,k}$ is considered confirmed if its IoU with any of the detectors is greater than the moderate τ_{md}, i.e.,

$$\max_{d \in \mathcal{D}_t} \text{IoU}(M_{t,k}, d > 0) \geq \tau_{\text{md}}. \tag{7}$$

3.5 Hierarchical Adaptive Memory

UGO employs a hierarchical adaptive memory (HAM) to represent the targets at two levels of detail. The category-level memory is used by the detector, while per-instance-level memory is used by the tracker. The two are continually updated to improve detection capabilities and to ensure instance-level tracking robustness. Figure 5 overviews HAM.

Category-level memory $\mathcal{M}_t^D = \{\mathbf{b}_i^E\}_{i=1:N_B}$, always contains the user-provided exemplar $\mathbf{b}_1^E$ and holds a FIFO buffer with three slots, updated by the exemplars proposed from reliable per-instance trackers. At time-step t, we consider all trackers with at least ten frames long trajectories, whose current mask M_t highly overlaps with a detector, i.e., $\text{IoU}(M_t, d > 0) \geq \tau_{\text{hi}}$, where $\tau_{\text{hi}} = 0.9$. Among these, the current mask-fitted-bounding-box from the most frequently confirmed (7) trajectory updates the category-level memory $\mathcal{M}_t^D$.

Instance-level memory $\mathcal{M}_{t,k}^T$ stores segmented examples of k-th instance and is used for its localization by the tracker. As in [36], the memory is split into distractor-resolving memory (DRM), responsible for discriminating between the target and visually-similar objects, and recent appearance memory (RAM), responsible for frame-to-frame segmentation accuracy – each a FIFO buffer with three DRM and four RAM slots. Both are updated from non-empty tracker outputs, with RAM updated as in [36], while we introduce a new DRM update protocol. The standard assumption [36] that the initialization frame provides an iconic target with ground-truth segmentation does not hold in

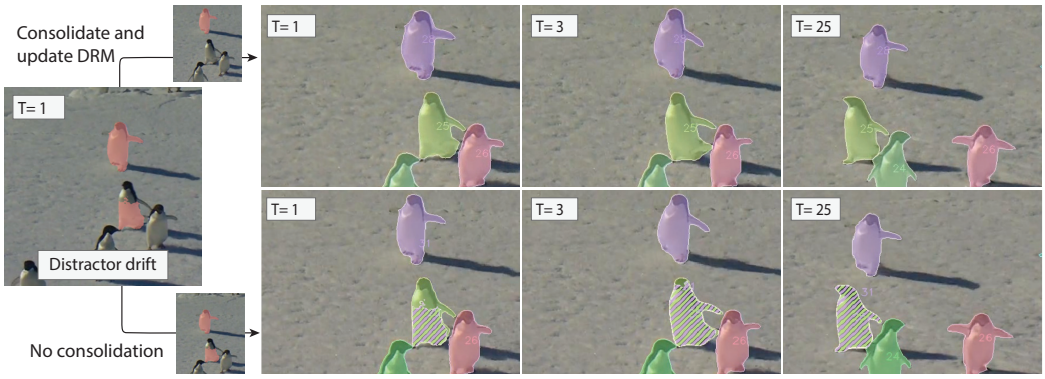

Figure 3: Oversegmented object at $T = 1$ and tracking in subsequent frames with and without consolidation-driven DRM update. Individual colors indicate an ID, with dashed pattern indicating overlapping IDs. Consolidation-based memory update leads to distractor resolution, while ignoring it leads to tracker failure.

GMOT, where trackers are initialized from potentially inaccurate detections. Therefore, the initialization frame should not be retained in DRM indefinitely [33; 36]. We instead leverage detector–tracker consolidation to identify frames with distractors. DRM is updated when the tracker mask before and after consolidation no longer reflect a high agreement (IoU$< \tau_{\text{hi}}$), while the consolidated mask is detector-confirmed (Eq. 7), indicating correction of distractor over-segmentation. Figure 3 visualizes the benefits of using the proposed DRM updating scheme.

4 Experiments

We follow the standard GMOT evaluation protocol [3], where a single target exemplar is provided in the first frame. Performance is evaluated by MOT measures, with the primary being MOTA, which integrates false positives, false negatives, and identity switches. Where possible, we also report HOTA [26], which jointly measures detection (DetA) and association (AssA) accuracy and provides a more reliable overall assessment than MOTA [26]. Several auxiliary measures are used. Identity preservation is measured by IDP, IDR, and their harmonic mean IDF1. Trajectory quality is quantified by MT, PT, and ML, denoting the number of mostly tracked ($> 80\%$), partially tracked (20-80$\%$), or mostly lost ($< 20\%$) identities, respectively. We also report false positives (FP), false negatives (FN), F1 score, identity switches (IDSw), and fragmentations (FM).

Implementation details. We use the pretrained SAM2.1 tracking module [33] with a pretrained GeCo2 [30] head, both operating on the same SAM2.1 Hiera-L backbone. The consolidation optimization (Section 3.3) runs for $N_{\text{iter}} = 3$ iterations with parameter $\theta_0 = 0.5$. The same thresholds for low, moderate and high overlaps ($\tau_{\text{lo}} = 0.2$, $\tau_{\text{md}} = 0.6$ and $\tau_{\text{hi}} = 0.8$) are used in all experiments. UGO tracks 100 objects at ~ 2.1 FPS on a single A100 GPU.

4.1 Comparison with general multi-object trackers

UGO is compared on GMOT-40 [3] benchmark with GMOT trackers, including the five versions of the current best state-of-the-art S-DETR [23] (Table 1). UGO consistently outperforms all competitors. It outperforms S-DETR [23] by 23$\%$ MOTA, reflecting joint improvements in detection quality, association accuracy, and long-term trajectory consistency. It also outperforms S-DETR [23] by 52$\%$ IDF1, indicating a substantially better instance identity preservation. We also evaluate SAM3 [9] with default parameters (see supplementary), which only supports prompt-based tracking – we thus initialize it with the category names of target objects. Despite this semantic advantage, UGO surpasses SAM3 by 6$\%$ in HOTA and 22$\%$ in MOTA, demonstrating markedly stronger instance coverage and overall tracking robustness.

UGO achieves the highest *mostly tracked rate* (MT), surpassing S-DETR [23] by 25$\%$ and the lowest (-39$\%$) *mostly lost rate* (ML), demonstrating a superior tracking stability. This highlights the effectiveness of the proposed pixel-level consolidation and evidence-gated memory mechanisms

Table 1: State-of-the-art comparison on the GMOT-40 [3] benchmark.

	Detector	Tracker	IDF1↑	MT↑	ML↓	FP↓	FN↓	F1↑	IDSw↓	MOTA↑	HOTA↑
Open-Vocabulary	OVTrack [21]	DeepSORT [38]	21.2	165	1367	49984	160378	47.7	1470	20.2	-
		ByteTrack [44]	20.6	164	1345	51356	156329	49.1	1669	19.9	-
		BoT-SORT [1]	20.3	167	1328	45721	163378	47.1	3278	20.0	-
		TbQ [23]-SwT	18.7	186	1304	50784	154893	49.7	6381	21.3	-
	GLIP-T [20]	DeepSORT [38]	41.6	401	877	46610	141330	55.0	2892	25.5	-
		ByteTrack [44]	45.1	447	746	52591	131759	57.5	2706	27.0	-
		BoT-SORT [1]	49.1	553	643	51308	133462	57.1	4675	27.3	-
		TbQ [23]-SwT	39.8	581	592	49470	136602	56.3	9972	27.5	-
	SAM3 [9]		72.8	1209	276	70994	56131	75.9	**990**	50.2	60.1
Template-Based	GTrack [15]	DeepSORT [38]	24.4	72	1363	**9000**	208818	30.4	1315	14.5	-
		ByteTrack [44]	32.1	178	1069	23881	181829	42.0	1791	19.1	-
		BoT-SORT [1]	34.0	251	978	22229	176991	44.3	7375	19.4	-
		TbQ [23]-SwT	27.4	213	1066	13507	182376	43.0	6407	20.6	-
	S-DETR [23]	DeepSORT [38]	41.8	382	773	47336	124257	60.6	5131	31.1	-
		ByteTrack [44]	41.4	331	764	53417	104765	65.7	4204	33.7	-
		BoT-SORT [1]	47.5	431	674	45769	119288	62.4	6775	34.1	-
		TbQ [23]-SwT	42.8	504	666	44882	107894	66.0	11664	35.9	-
		TbQ [23]-SwB	51.3	1083	278	44390	68189	77.0	11252	50.0	-
	UGO		**77.09**	**1349**	**170**	58165	**39670**	**81.3**	1136	**61.4**	**63.8**

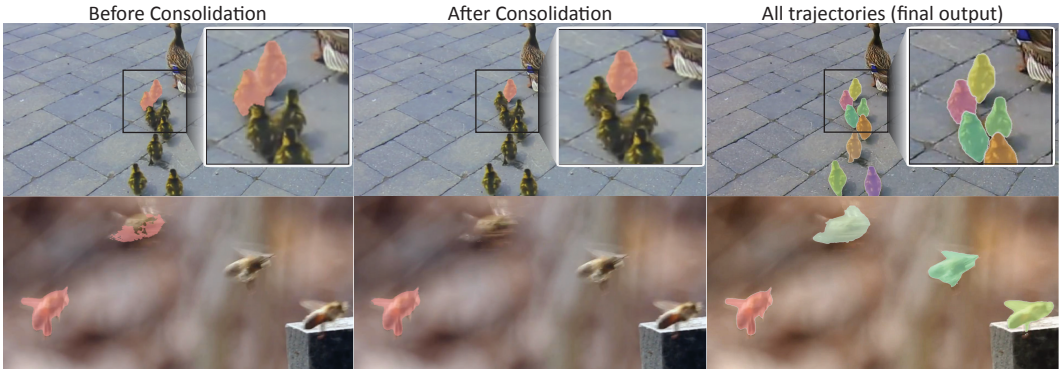

Figure 4: Incorrect masks spreading over several objects are corrected by the proposed detector-tracker consolidation method, enabling accurate memory updates and preventing drifts.

in enforcing consistent, globally coordinated identity assignment. As illustrated in Figure 4, the proposed consolidation resolves overlapping masks into mutually exclusive, pixel-consistent instance segmentations. By enforcing competition at the pixel level rather than relying on box-level matching (standard in prior SOTA), consolidation resolves ambiguous shared regions and corrects over-segmentation errors before they accumulate temporally. This prevents identity switches and trajectory fragmentations, especially in dense scenes with visually similar instances.

Compared to S-DETR, UGO reduces false negatives by 42% at a cost of 31% higher FP rate, while it overall produces a better temporal coverage of target-category instances, as observed in 6% higher F1 score. Inspection of S-DETR code revealed usage of sequence-specific thresholds to reduce false positives, while UGO relies on a fixed setting, indicating its gains come from architecture. Overall, UGO achieves state-of-the-art performance with strong cross-category generalization.

4.2 Comparison with specialist trackers

Next, we compare UGO on benchmarks with trackers specialized for individual object categories. These setups are particularly challenging for generalist UGO, since it has not been fine-tuned per category, unlike its competitors.

Evaluation on AnimalTrack. AnimalTrack benchmark [43] evaluates classical MOT trackers on 10 different animal species. The benchmark provides training sets with category-specific labels to train

Table 2: State-of-the-art comparison on AnimalTrack [43].

Method	HOTA↑	MOTA↑	IDF1↑	IDP↑	IDR↑	MT↑	PT	ML↓	F1↑	IDSw↓	FM↓
ByteTrack [44]	40.1	38.5	51.2	64.9	42.3	310	465	329	0.631	1309	3513
IOUTrack [6]	41.6	**55.7**	45.7	51.9	40.7	388	454	262	0.762	4639	5259
SORT [5]	42.8	55.6	49.2	58.5	42.4	333	470	301	0.749	2530	3730
OMC [22]	43.0	53.4	50.3	61.8	42.4	324	478	302	0.735	4938	7162
Tracktor++ [4]	44.2	55.2	51.0	58.5	45.1	364	472	268	0.751	1976	4149
TransTrack [35]	45.4	48.3	53.4	63.4	46.1	327	416	361	0.705	1978	6459
QDTrack [29]	47.0	**55.7**	56.3	65.6	49.3	367	420	317	0.752	1970	5656
SAM3 [9]	52.1	43.9	66.4	**68.4**	64.4	642	280	188	0.713	470	2588
UGO	**59.4**	53.8	**71.7**	67.6	**76.3**	**700**	254	**156**	**0.783**	**463**	**2490**

Table 3: Tracking performance on the MOT17. The colors denote UGO performing better/worse compared to the individual method, * denotes public detection setup.

Method	HOTA	DetA	AssA
MPNTrack17	46.6 (7%)	46.2 (-2%)	47.3 (18%)
eTC17	45.1 (11%)	44.1 (3%)	46.4 (19%)
Tracktor++v2	45.1 (11%)	45.3 (0%)	45.0 (24%)
ByteTrack [44]	63.1 (-21%)	**64.5** (-30%)	**62.0** (-10%)
OC-Sort [8]	**63.2** (-21%)	63.2 (-28%)	63.4 (-12%)
OC-Sort* [8]	54.6 (-8%)	47.8 (-5%)	57.6 (-3%)
UGO	50.0	45.3	55.6

Table 4: Video object counting results on ScienceCount [2]. Methods marked with * accept text prompts instead of exemplars.

	Penguins		Crystals	
Method	MAE↓	RMSE↓	MAE↓	RMSE↓
$\text{GDINO}_{\text{MASA}}$* [24]	9.0	11.5	72.3	87.9
$\text{CountGDB}_{\text{ByteTrack}}$* [2]	4.3	5.5	71.6	88.1
CountVid [2]*	4.0	5.3	69.1	86.0
$\text{CountGDB}_{\text{ByteTrack}}$ [2]	4.0	4.2	31.1	52.8
$\text{GeCo}_{\text{SAM2.1}}$ [31]	11.3	14.5	46.1	82.8
CountVid [2]	3.3	4.8	33.7	59.8
UGO	**2.7**	**3.5**	**24.6**	**43.4**

the MOT trackers for each test category. Since iconic exemplars are not provided, UGO randomly selects five ground-truth bounding boxes from the first frame to define the target category.

Table 2 shows that UGO outperforms all state-of-the-art methods. UGO surpasses the strongest specialist method QDTrack [29] by 26% HOTA and 27% IDF1. Remarkably, UGO tracks $2\times$ more instances than competing trackers, while producing $4\times$ fewer identity switches and half the number of fragmentations. We additionally evaluate SAM3 [9], following Sec. 4.1. Since SAM3 relies on text prompts rather than exemplar conditioning, it benefits from an explicit semantic specification of the target (e.g., "geese"), whereas UGO must infer the category purely from visual evidence in a few exemplars. Despite its semantic advantage and large-scale detection training, UGO surpasses SAM3 by 14% HOTA and 23% MOTA, demonstrating superior performance.

The primary limitation of UGO is a higher FP rate, resulting in a 3% lower MOTA than QDTrack [29]. Inspection reveals that a third of FPs originate from the *goose_3* sequence, where similar bird species are also detected, reflecting ambiguity in exemplar-based category specification. Nevertheless, UGO achieves superior identity quality and tracking performance, highlighting strong generalization without category-specific training.

Evaluation on MOT17. We next evaluate UGO on the human-centric MOT17 benchmark [28], again with the same setup as in AnimalTrack. Table 3 reports a comparison with three standard MOT baselines, and state-of-the-art OC-SORT [8]. UGO outperforms all MOT baseline specialists in HOTA, demonstrating strong joint detection-association reasoning, but lags behind the sota, which exploits in-domain category-specific training, in particular the excellent fine-tuned person detectors, which lead to superior detection accuracy (DetA in Table 3). This is evident from a 24% DetA drop of OC-SORT using a public human detector. Meanwhile, comparable association accuracy to that of OC-SORT* indicates the effectiveness of the consolidation module responsible for association.

4.3 Comparison with video object counters

We finally evaluate UGO in video object counting performance on the recent ScienceCount benchmark [2]. The MAE and RMSE [2], computed by counting trajectories without final confirmation-based filtering, are reported in Table 4. UGO outperforms the sota CountVid [2] by 18%/27% in MAE and 25%/27% in RMSE on the Penguins and Crystals subsets, respectively. Notably, CountVid is a video counting specialist, employing a multistage, forward-backward tracking pipeline with separate detection and tracking backbones. While UGO processes the video in a single pass, its dominance over the counting specialist emphasizes the strong generalization capabilities.

Table 5: UGO ablation on GMOT-40 [3].

Method	HOTA	DetA	AssA	MOTA	FN	FP	IDSw	MT	PT	ML	FM	IDF1
$\text{UGO}_{\overline{\text{CONS}}}$	50.98	49.11	54.36	43.12	61835	59794	24173	1084	671	189	8414	61.47
UGO_{HM}	52.35	39.42	70.55	5.88	34457	204395	2402	1460	328	156	4553	60.76
$\text{UGO}_{\overline{\Theta_1}}$	52.39	55.56	50.48	57.11	39900	64272	5773	1360	421	163	4478	56.86
UGO_{D4S}	61.00	53.80	70.49	57.33	51892	55349	2132	1252	430	262	4409	72.81
$\text{UGO}_{\overline{\Theta_2}}$	62.76	55.55	72.11	59.16	40059	63251	1375	1351	419	174	4633	75.66
$\text{UGO}_{\overline{\text{DRM}}}$	63.19	56.23	72.20	60.30	41057	59463	1244	1331	434	179	4554	76.10
UGO_{CAL}	63.73	56.57	72.98	61.03	40126	58646	1121	1344	423	177	4501	76.89
UGO_{1ex}	62.37	54.53	72.54	58.34	48908	56773	1113	1305	417	222	4406	75.07
$\text{UGO}_{\text{4-rand-GT}}$	63.17	55.52	73.13	58.84	37425	66933	1139	1372	429	143	4624	76.34
UGO	63.83	56.72	73.03	61.39	39670	58165	1136	1349	425	170	4473	77.09

4.4 Ablation study

UGO design choices are analyzed on GMOT-40 [3] in Table 5. Additional ablations in the supplementary further show graceful performance degradation with reduced detector and tracker capacity and robustness to variations of selected gating thresholds.

Consolidation module. Several variations are considered (Table 5). The first, $\text{UGO}_{\overline{\text{CONS}}}$, has the consolidation module removed, and each detection is associated by the instance tracker with highest corresponding IoU. $\text{UGO}_{\overline{\text{CONS}}}$ results in 20% HOTA and 30% MOTA drops compared to UGO and nearly $2\times$ as many trajectory fragmentations, due to missed detections and weak temporal consistency. Next, the consolidation module is replaced by Hungarian matching [18] (UGO_{HM}) for optimal assignment between tracker and detector masks. UGO_{HM} results in 18% HOTA and 90% MOTA performance drops. The results verify the importance and robustness of the proposed consolidation module, which standard MOT-style matching strategies cannot replace.

Calibration of detector and tracker logits. Consolidation relies on direct competition between tracker and detector logits, both computed with the shared SAM2 mask decoder; we verify their compatibility by scaling detector logits to match the tracker's mean amplitude (UGO_{CAL}), on GMOT-40. Performance remains unchanged (Table 5), indicating that no additional calibration is required.

Consolidation module potentials. The potentials $\Theta_1(\cdot)$ and $\Theta_2(\cdot)$ in (1) control the consolidation labeling dynamics. Removing $\Theta_1(\cdot)$, denoted by $\text{UGO}_{\overline{\Theta_1}}$ decreases HOTA by 18% and MOTA by 7%, while removing $\Theta_2(\cdot)$ leads to 2% HOTA and 4% MOTA reduction (denoted by $\text{UGO}_{\overline{\Theta_2}}$). $\Theta_1(\cdot)$ is thus central for resolving competition, favoring trackers over detections and reinforcing detector-supported tracks, while $\Theta_2(\cdot)$ suppresses uncertain or unstable masks. Together, they ensure stable, pixel-level consistent identities even in cluttered and highly dynamic scenes. This validates the importance of the proposed potentials for strong performance.

Instance-level memory. Replacing the proposed instance memory (Section 3.5) by the original SAM2 [33] ($\text{UGO}_{\overline{\text{DRM}}}$) results in 2% MOTA and 1% HOTA drops, 10% more identity switches and more fragmentations. This supports importance of the proposed memory for identity stability. Replacing our memory management with that of [36] (UGO_{D4S}) leads to 4% HOTA and 7% MOTA drops, confirming that the proposed memory management offers a more robust multi-object tracking.

Category-level memory. Removing FIFO and keeping only the initial exemplar in the memory (UGO_{1ex}), leads to 5% MOTA and 2% HOTA drop (Table 5), indicating that adaptive exemplar updates not only enhance instance discovery but also improve long-term identity consistency by stable trajectories mining. Constructing memory from four randomly selected ground-truth boxes from the first frame and keeping it fixed throughout the sequence ($\text{UGO}_{\text{4-rand-GT}}$) leads to 4% MOTA and 1% HOTA drops, further emphasizing the benefits of our on-the-fly trajectory mining.

5 Conclusion

We introduced UGO, a unified general multi-object tracker that integrates exemplar-conditioned detection and mask-based instance propagation within a single architecture. A novel consolidation module resolves conflicting and over-segmented outputs, yielding mutually exclusive masks, while hierarchical memory enables robust global and per-instance modeling. UGO achieves state-of-the-

art performance in GMOT and video counting, while remaining competitive with specialist MOT methods, thus confirming the potential of the new GMOT design paradigm.

6 Acknowledgements

This work was supported by the Slovenian Research Agency program P2-0214 and project J2-60054, as well as the supercomputing network SLING (ARNES, EuroHPC Vega - IZUM), and the Slovenian Ministry of MESY and EC/EuroHPC JU via the project SLAIF (grant number 101254461).

References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[2] N. Amini-Naieni and A. Zisserman. Open-world object counting in videos. In *Association for Advancement of Artificial Intelligence Conference (AAAI)*, 2026.

[3] Hexin Bai, Wensheng Cheng, Peng Chu, Juehuan Liu, Kai Zhang, and Haibin Ling. Gmot-40: A benchmark for generic multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6719–6728, 2021.

[4] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3464–3468, 2016.

[6] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *AVSS*, 2017.

[7] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6247–6257, 2020.

[8] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9686–9696, 2023.

[9] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025.

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[11] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3151–3161, 2024.

[12] Peng Chu and Haibin Ling. Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6171–6180, 2019.

[13] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 129(4):845–881, 2021.

[14] Ruopeng Gao, Ji Qi, and Limin Wang. Multiple object tracking as id prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27883–27893, June 2025.

[15] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Globaltrack: A simple and strong baseline for long-term tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11037–11044, 2020.

[16] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016.

[17] Matej Kristan, Jiří Matas, Pavel Tokmakov, Alan Lukežič, Michael Felsberg, Luka Čehovin Zajc, Khanh-Tung Tran, Xuan-Son Vu, Johanna Björklund, Michal Neoral, et al. The third visual object tracking segmentation vots2025 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7422–7440, 2025.

[18] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[19] Kenneth Lange. *MM Optimization Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[20] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[21] Siyuan Li, Tobias Fischer, Lei Ke, Henghui Ding, Martin Danelljan, and Fisher Yu. Ovtrack: Open-vocabulary multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5567–5577, 2023.

[22] Chao Liang, Zhipeng Zhang, Xue Zhou, Bing Li, Yi Lu, and Weiming Hu. One more check: Making" fake background" be tracked again. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2022.

[23] Qiankun Liu, Yichen Li, Yuqi Jiang, and Ying Fu. Siamese-detr for generic multi-object tracking. *IEEE Transactions on Image Processing*, 33:3935–3949, 2024.

[24] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[25] Wenxi Liu, Yuhao Lin, Qi Li, Yinhua She, Yuanlong Yu, Jia Pan, and Jason Gu. Prototype learning based generic multiple object tracking via point-to-box supervision. *Pattern Recognition*, 154:110588, 2024.

[26] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021.

[27] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8844–8854, 2022.

[28] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831*, 2016.

[29] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition Conference (CVPR)*, 2021.

[30] Jer Pelhan, Alan Lukezic, and Matej Kristan. Generalized-scale object counting with gradual query aggregation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[31] Jer Pelhan, Alan Lukežič, Vitjan Zavrtanik, and Matej Kristan. A novel unified architecture for low-shot counting by detection and segmentation. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024.

[32] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *CVPR*, pages 3394–3403, 2021.

[33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[34] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *International Conference on Machine Learning*, pages 29441–29454. PMLR, 2023.

[35] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv:2012.15460*, 2020.

[36] Jovana Videnovic, Alan Lukezic, and Matej Kristan. A distractor-aware memory for visual object tracking with SAM2. In *Comp. Vis. Patt. Recognition*, 2025.

[37] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition Conference (CVPR)*, 2019.

[38] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference in Image Processing (ICIP)*, 2017.

[39] Stephen J. Wright. *Continuous Optimization (Nonlinear and Linear Programming)*. Princeton University Press, 2015.

[40] Tong Tong Wu and Kenneth Lange. The mm alternative to em. *Statistical Science*, 25(4), Nov. 2010.

[41] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision (ICCV)*, 2015.

[42] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. *Advances in Neural Information Processing Systems*, 34:2491–2502, 2021.

[43] Libo Zhang, Junyuan Gao, Zhen Xiao, and Heng Fan. Animaltrack: A benchmark for multi-animal tracking in the wild. *International Journal of Computer Vision*, 131(2):496–513, 2023.

[44] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Proceedings of the European Conference on Computer Vision*, pages 1–21, 2022.

A The mask consolidation algorithm

We provide further details on derivation of the UGO mask consolidation algorithm for the detector and tracker masks. We first introduce the notations, then define the original cost function and the surrogate loss, and then proceed to derive the corresponding minimization algorithm.

Let $\Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ be the pixel domain and let $S = N_{\mathcal{D}} + N_{\mathcal{T}} + 1$ denote the number of labels, i.e., potentially competing masks for explaining individual pixels (detections, trackers, and background). We form a logit tensor

$$\mathbf{L} \in \mathbb{R}^{H \times W \times S}, \tag{8}$$

by concatenating all detector logits, all tracker logits, and a constant background logit map (i.e., λ_{BG}). Let $\mathbf{L}(\mathbf{x}, s)$ denote the value of the tensor at pixel location $\mathbf{x}$ for label s.

We seek a mutually-exclusive pixel labeling $\mathbf{Y} : \Omega \to \{1, \dots, S\}$, with $\mathbf{Y}(\mathbf{x})$ reading out the label assigned to a pixel $\mathbf{x}$. Further, let $N_s(\mathbf{Y})$ be a function that counts the number of pixels assigned a label s in the labeling $\mathbf{Y}$,

$$N_s(\mathbf{Y}) = \sum_{\mathbf{x} \in \Omega} \mathbf{1}_{[\mathbf{Y}(\mathbf{x}) \equiv s]}, \tag{9}$$

where $\mathbf{Y}(\mathbf{x}) \equiv s$ verifies that the label at pixel $\mathbf{x}$ is equal to s in the labeling $\mathbf{Y}$.

A.1 The labeling cost function

To enforce a desired behavior of the optimization (as discussed in the paper, Section 3.3), we introduce two label-wise potentials $\Theta_1(s; \mathbf{Y})$ and $\Theta_2(s; \mathbf{Y})$ – for brevity we will omit $\mathbf{Y}$ in the following, i.e., $\Theta_1(s)$ and $\Theta_2(s)$. We define these potentials to encode: (i) a preference for trackers over detections, (ii) a preference for trackers with high detector support, and (iii) suppression of unstable labels whose final mask deviates from its initial mask. Concretely, we make the potentials depend on $\mathbf{Y}$ through IoU terms between masks induced by $\mathbf{Y}$ and (binarized) initial masks (see Section 3.3 in the paper for definitions). The labeling cost function to be optimized is thus

$$\hat{\mathcal{L}}(\mathbf{Y}) = - \sum_{\mathbf{x} \in \Omega; s = \mathbf{Y}(\mathbf{x})} \log \Big(\mathbf{L}(\mathbf{x}, s) \cdot \Theta_1(s) \cdot \Theta_2(s) \Big), \tag{10}$$

where $\mathbf{L}(\mathbf{x}, s) > 0$ always holds, as the label set includes a background label with a constant nonnegative logit $\lambda_{\mathrm{BG}} > 0$. Consequently, any detector or tracker label on a non-positive logit cannot be selected over the background. In practice, the IoU-based potentials are also lower-bounded by a small $\epsilon > 0$, ensuring $\Theta_1(s)\Theta_2(s) > 0$ and making the logarithm well-defined.

A.2 Reweighted surrogate objective

Due to nonlinear interaction between pixel labeling in $\Theta_1(s)$ and $\Theta_2(s)$, the objective (10) does not adhere to simple optimization over $\mathbf{Y}$. Thus, auxiliary variables $A_1(s)$ and $A_2(s)$ are introduced for $\Theta_1(s, \mathbf{Y})$ and $\Theta_2(s, \mathbf{Y})$ that decouple the logits from the latter and make pixel labeling fully separable. This leads to the following surrogate objective

$$\begin{aligned} &\mathcal{L}(\mathbf{Y}, A_1, A_2) = \\ &- \sum_{\mathbf{x} \in \Omega} \Big(\log(\mathbf{L}(\mathbf{x}, \mathbf{Y}(\mathbf{x})) A_1(\mathbf{Y}(\mathbf{x})) A_2(\mathbf{Y}(\mathbf{x}))) \Big) \\ &\quad + \sum_{s=1:S} N_s \Big(\big(\log A_1(s) - \log \Theta_1(s) \big)^2 + \\ &\qquad \big(\log A_2(s) - \log \Theta_2(s) \big)^2 \Big), \end{aligned} \tag{11}$$

where the N_s is introduced to remove the influence of the mask size, i.e., such that masks corresponding to large objects are not preferred over those of small objects.

This objective can be minimized by an iterated scheme that exchanges between estimation of $A_1(s)$ and $A_2(s)$ and estimation of $\mathbf{Y}$, closely following the standard Majorize-Minorize [40] approach. In

particular, the following steps are exchanged:
Step 1 assumes an estimate $\mathbf{Y}^{(k-1)}$ from previous iteration and solves the following minimization:

$$A_1^{(k)}, A_2^{(k)} = \arg\min_{A_1, A_2} \mathcal{L}(\mathbf{Y}^{(k-1)}, A_1, A_2). \tag{12}$$

Step 2 then fixes $A_1^{(k)}$ and $A_2^{(k)}$, and solves the following minimization:

$$\mathbf{Y}^{(k)} = \arg\min_{\mathbf{Y}} \mathcal{L}(\mathbf{Y}, A_1^{(k)}, A_2^{(k)}). \tag{13}$$

A.3 Step 1 minimization problem

The objective (11) is rewritten into

$$\mathcal{L}(\mathbf{Y}^{(k-1)}, A_1, A_2) = \sum_{s=1:S} \Big(\tag{14}$$

$$- \sum_{\mathbf{x}\in\Omega} \Big[\log(A_1(s)A_2(s)) 1_{[\mathbf{Y}^{(k-1)}(\mathbf{x})\equiv s]} + \varepsilon(s) \Big] \tag{15}$$

$$+ N_s^{(k-1)} \Big((\log A_1(s) - \log \Theta_1^{(k-1)}(s))^2 \tag{16}$$

$$+ (\log A_2(s) - \log \Theta_2^{(k-1)}(s))^2 \Big) \Big), \tag{17}$$

where $\varepsilon(s)$ absorbs the terms not depending on A_1 and A_2, while $\Theta_1^{(k-1)}$, $\Theta_1^{(k-1)}$ and $N_s^{(k-1)}$ are evaluated at $\mathbf{Y}^{(k-1)}$.

Then the updates for $A_1^{(k)}(s)$ and $A_2^{(k)}(s)$ are obtained by differentiating the objective and setting to zero, i.e.,

$$\frac{\partial \mathcal{L}(\mathbf{Y}^{(k-1)}, A_1(s), A_2(s))}{\partial \log A_1(s)} \equiv 0 \tag{18}$$

$$\frac{\partial \mathcal{L}(\mathbf{Y}^{(k-1)}, A_1(s), A_2(s))}{\partial \log A_2(s)} \equiv 0. \tag{19}$$

The derivative in (18) is defined as

$$\frac{\partial \mathcal{L}}{\partial \log A_1(s)} = -N_s + 2N_s \log A_1(s) - 2N_s \log \Theta_1^{(k-1)}(s), \tag{20}$$

which gives the following update for $A_1^{(k)}(s)$

$$A_1^{(k)}(s) = \Theta_1^{(k-1)}(s) \cdot e^{1/2}, \tag{21}$$

while a similar derivation from (19) with $\frac{\partial \mathcal{L}}{\partial \log A_2(s)} \equiv 0$ gives

$$A_2^{(k)}(s) = \Theta_2^{(k-1)}(s) \cdot e^{1/2}. \tag{22}$$

A.4 Step 2 minimization problem

In the second step, the auxiliary variables are fixed, the labeling $\mathbf{Y}$ can be obtained from (11) by minimizing

$$\begin{aligned} \mathbf{Y}^{(k)} &= \arg\min_{\mathbf{Y}} \mathcal{L}(\mathbf{Y}, A_1^{(k)}, A_2^{(k)}) \\ &= \arg\max_{\mathbf{Y}} \sum_{\mathbf{x}\in\Omega; s=\mathbf{Y}(x)} L(\mathbf{x}, s) A_1^{(k)}(s) A_2^{(k)}(s), \end{aligned} \tag{23}$$

which decomposes over pixels, thus the labels can be computed for each pixel $\mathbf{x}$ separately:

$$\mathbf{Y}^{(k)}(\mathbf{x}) = \arg\max_{\mathbf{Y}(\mathbf{x})} L(\mathbf{x}, \mathbf{Y}(\mathbf{x})) A_1^{(k)}(\mathbf{Y}(\mathbf{x})) A_2^{(k)}(\mathbf{Y}(\mathbf{x})). \tag{24}$$

Table 6: State-of-the-art comparison on AnimalTrack [43].

Method	HOTA↑	MOTA↑	IDF1↑	IDP↑	IDR↑	MT↑	PT	ML↓	FP↓	FN↓	Pr↑	Re↑	F1↑	IDSw↓	FM↓
JDE	26.8	27.3	31.0	51.0	22.0	106	414	584	17887	155623	0.830	0.360	0.502	3187	5031
FairMOT	30.6	29.0	38.8	62.8	28.0	143	462	499	17653	152624	0.837	0.372	0.515	2335	5447
Trackformer	31.0	20.4	36.5	40.9	32.8	230	491	383	70404	118724	0.639	0.512	0.568	4355	3725
TADAM	32.5	36.5	37.2	44.4	32.0	258	495	351	41728	110048	0.761	0.547	0.637	2538	4469
DeepSORT	32.8	41.4	35.2	49.7	27.2	213	452	439	**14131**	124747	0.893	0.487	0.630	3503	4527
ByteTrack	40.1	38.5	51.2	64.9	42.3	310	465	329	31591	116587	0.800	0.521	0.631	1309	3513
IOUTrack	41.6	55.7	45.7	51.9	40.7	388	454	262	25206	77847	0.868	0.680	0.762	4639	5259
SORT	42.8	55.6	49.2	58.5	42.4	333	470	301	19099	86257	0.891	0.645	0.749	2530	3730
OMC	43.0	53.4	50.3	61.8	42.4	324	478	302	15910	92570	**0.904**	0.619	0.735	4938	7162
Tracktor++	44.2	55.2	51.0	58.5	45.1	364	472	268	25477	81538	0.864	0.665	0.751	1976	4149
TransTrack	45.4	48.3	53.4	63.4	46.1	327	416	361	28553	95212	0.838	0.608	0.705	1978	6459
QDTrack	47.0	**55.7**	56.3	65.6	49.3	367	420	317	22696	83057	0.876	0.658	0.752	1970	5656
SAM3	52.1	43.9	66.4	68.4	64.4	642	280	188	60979	74880	0.734	0.692	0.713	470	2588
UGO	**59.4**	53.8	**71.7**	**67.6**	**76.3**	**700**	254	**156**	71695	**40157**	0.739	**0.835**	**0.783**	**463**	**2490**

B HAM Architecture Details

The hierarchical adaptive memory (HAM), described in Section 3.5, consists of two complementary components: a category-level memory $\mathcal{M}_t^D$ used for exemplar-conditioned detection, and per-instance memories $\{\mathcal{M}_{t,k}^T\}$ used for instance propagation, see Figure 5. The category-level memory maintains a compact set of reliable exemplars, while each instance-level memory is decomposed into distractor-resolving (DRM) and recent appearance (RAM) buffers, enabling robust discrimination and temporal adaptation. The two levels are tightly coupled through consolidation: reliable tracks update $\mathcal{M}_t^D$, while consolidation-driven corrections trigger updates of $\mathcal{M}_{t,k}^T$, ensuring consistent global and instance-level representations.

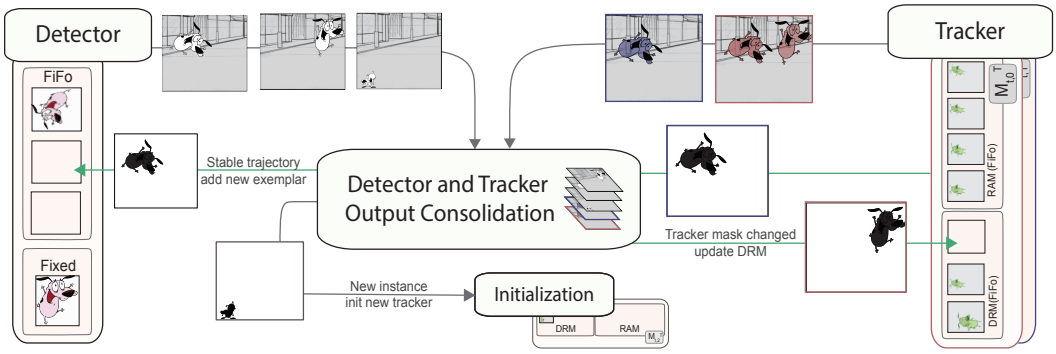

Figure 5: HAM overview: stable tracks provide robust updates for detector category-level memory, while instance-level memories are updated robustly by the result of consolidation analysis.

C Experimental Evaluation on Animal Track

Table 6 provides extended results on AnimalTrack, including false positives (FP), false negatives (FN), precision, recall, and F1 score (Table 6). The results show that UGO besides achieving the highest HOTA, it achieves the highest recall and F1 score among all compared trackers. In particular, UGO reduces FN (40157 FN) at least two-fold compared to all specialist methods, including QDTrack (83057 FN), demonstrating significantly better temporal coverage of target instances. Although UGO produces more false positives than some specialist detectors (e.g., OMC or QDTrack), it achieves the strongest overall F1 performance. This confirms that UGO favors consistent instance coverage and identity preservation over overly conservative detection suppression.

Comparison with SAM3 (text-prompted MOT). We additionally evaluate SAM3 [9] on AnimalTrack. Unlike UGO, SAM3 does not support exemplar-conditioned multi-object tracking in the GMOT sense, where a single first-frame exemplar defines the category for the entire video. While SAM3 allows image exemplars as prompts, these are applied within the same frame (e.g., for the

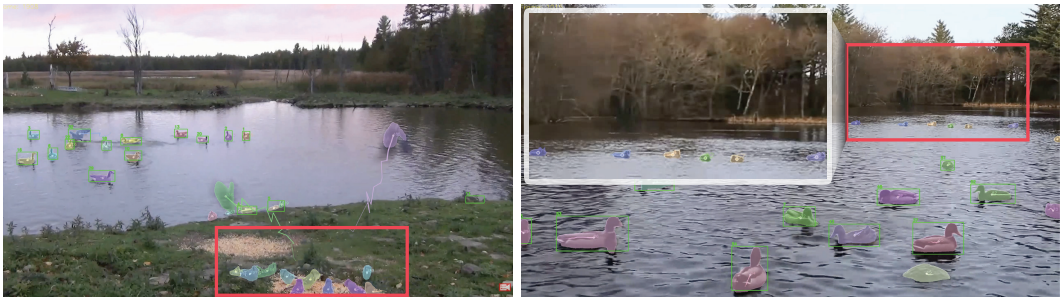

Figure 6: Two sequences from AnimalTrack where UGO produces the most false detections. Red bounding boxes are overlaid to highlight the relevant regions. Ground-truth bounding boxes are shown in green – note that some instances are missing in the ground truth annotations.

detection of all objects of the same category in that frame) and are not designed, nor do they work, as a persistent cross-frame category specification. We therefore evaluate SAM3 using a text prompt specifying the ground-truth category for each sequence, which aligns with its standard usage in video tracking.

Text prompting offers a semantic advantage over exemplar-based specification, particularly for fine-grained categories. A textual label such as "duck" or "goose" explicitly defines the target class, whereas a few visual exemplars may not clearly indicate whether only that species or also visually similar birds should be tracked. For example, bird species can appear highly similar in crowded scenes, making the category boundary ambiguous from a few exemplars. Moreover, SAM3 is trained on large-scale image and video segmentation data spanning a broad set of visual concepts, which supports strong generalization under semantic (text) prompting. In contrast, the few-shot detector used in UGO is not explicitly trained for the target categories and must rely solely on limited exemplar supervision. Despite this advantage, UGO outperforms SAM3 on the primary tracking metrics. This exposes an important constraint/limitation of UGO. Since the target category is specified only through visual exemplars, UGO cannot explicitly control the semantic granularity of the category to be tracked, which is inherited from GECO2 detector [30].

In contrast, UGO must infer the category solely from visual evidence contained in a few exemplars, without access to explicit semantic supervision, which is a hard task in fine-grained classification Figure 6. Despite this advantage, UGO outperforms SAM3 on the primary tracking metrics. UGO outperforms SAM3 in HOTA by 14% and a remarkable 23% MOTA. While SAM3 produces fewer identity switches and fragmentations, it is because of lower detection recall (20% lower relatively) and reduced overall instance coverage. These results highlight that the proposed exemplar-driven consolidation and memory mechanism provides stronger identity consistency and more complete tracking, even without access to explicit semantic category labels.

For evaluating SAM3 [9], we use the default parameters of the official video predictor. Table 7 summarizes these thresholds. SAM3 uses a loose association threshold to avoid duplicate masklet creation, while a stricter threshold determines whether an existing masklet is sufficiently supported by detector evidence. Newly initialized masklets are further filtered during a hotstart period, in which unmatched masklets or those repeatedly duplicating existing trajectories are removed. In contrast, UGO omits many of these thresholds by using the proposed consolidation module, since duplicate handling, new instance discovery, even mask correction, and instance memory update logic are handled in a single unified step.

Table 7: Threshold summary for SAM3 [9].

Parameter	**Default**	**Role**
`assoc_iou_thresh`	0.1	Loose detection-to-track IoU threshold for deciding whether a detection is new.
`trk_assoc_iou_thresh`	0.5	Stricter IoU threshold for deciding whether a tracker masklet is matched.

Parameter	Default	Role
new_det_thresh	0.7	Minimum detector score required to spawn a new object.
HIGH_CONF_THRESH	0.8	Threshold for high-confidence detector reconditioning.
iou_thresh_recondition	0.8	Default IoU threshold for detector-to-tracker memory refresh.
suppress_overlapping_based _on_recent_occlusion_threshold	0.7	IoU threshold for suppressing highly overlapping tracker masks based on recent occlusion history.
detector_tracker_confirmation	0.5	Ratio of detector-confirmed tracker masks required to keep a masklet; otherwise, the masklet is removed.
hotstart_delay	15	Number of frames used to judge early masklet stability.
hotstart_unmatch_thresh	8	Number of unmatched frames after which a new masklet is removed.
hotstart_dup_thresh	8	Number of duplicated frames before hotstart removal.
max_trk_keep_alive	30	Maximum keep-alive counter.
recondition_every_nth_frame	16	Periodic detector-based tracker refresh interval.
masklet_confirmation _consecutive_det_thresh	3	Number of detections required if confirmation is enabled.
mask logit threshold	0	Detector/tracker mask logits are binarized with threshold > 0.
NO_OBJ_LOGIT	-10	Logit assigned to suppressed masks before memory encoding.

D Ablation Study

This section analyzes the sensitivity of UGO to the *conservative gates* used by the consolidation module and the memory/lifecycle rules. These gates control when masks are accepted, suppressed, or used for memory updates. All experiments are conducted on GMOT-40 [3] by varying one gate at a time while keeping the remaining gates fixed to the default configuration. Results are reported in Table 8.

Tracker preference strength θ_0. The constant θ_0 controls the preference of trackers over detector logits in $\Theta_1(\cdot)$, and it also sets the scale of the detector-support term that boosts tracker masks consistent with detections. We evaluate $\theta_0 \in \{0.3, 0.4, 0.6, 0.7\}$ and compare it with the default setting 0.5 in UGO. As shown in Table 8, performance is essentially unchanged across this range: HOTA varies by less than 0.21 points around the default, and MOTA remains within ≈ 0.41 points of the default (with the largest drop observed at $\theta_0 = 0.8$ due to increased FP). This indicates that the consolidation labeling is insensitive to the exact strength of preference, provided that trackers are moderately favored.

Overlap threshold τ_{md}. The moderate threshold τ_{md} is used for: (i) deciding whether a tracker mask is *confirmed* by any detector that gates trajectory verification, and (ii) initialization of new tracks. We evaluate $\tau_{\mathrm{md}} \in \{0.45, 0.5, 0.55, 0.65, 0.7\}$. Lowering τ_{md} relaxes confirmation, which increases false positives; this is reflected by the drop of 2% MOTA at $\tau_{\mathrm{md}} = 0.45$. Increasing τ_{md} makes confirmation stricter; at $\tau_{\mathrm{md}} = 0.7$ it starts to reduce MT and increase ML, rejecting correct trajectories as invalid. Overall, $\tau_{\mathrm{md}} = 0.6$ is the stable balance, yielding the least ML.

Tracker-collapse threshold τ_{lo}. After consolidation, labels whose final mask is inconsistent with the initial mask are removed. For trackers, a permissive threshold τ_{lo} is used to avoid terminating

Table 8: UGO ablation on GMOT-40 [3].

Method	HOTA	DetA	AssA	MOTA	FN	FP	IDSw	MT	PT	ML	FM	IDF1
$\theta_0 = 0.3$	63.71	56.60	72.91	61.12	40195	58347	1130	1348	426	170	4494	76.85
$\theta_0 = 0.4$	63.75	56.62	72.96	61.15	40000	58416	1165	1345	431	168	4500	76.92
$\theta_0 = 0.6$	63.82	56.65	73.09	61.20	39769	58605	1092	1353	422	169	4463	77.03
$\theta_0 = 0.7$	63.77	56.64	72.98	61.20	39695	58666	1107	1344	433	167	4487	77.00
$\theta_0 = 0.8$	63.62	56.57	72.72	60.98	39996	58942	1082	1347	431	166	4492	76.78
$\tau_{\text{md}} = 0.45$	63.45	56.33	72.70	60.37	38370	61941	1269	1359	419	166	4575	76.49
$\tau_{\text{md}} = 0.5$	63.66	56.45	73.01	60.74	38617	60793	1219	1357	422	165	4532	76.80
$\tau_{\text{md}} = 0.55$	63.67	56.51	72.96	60.76	39616	59756	1212	1354	420	170	4506	76.75
$\tau_{\text{md}} = 0.65$	63.73	56.66	72.86	61.32	40446	57620	1091	1342	426	176	4492	76.90
$\tau_{\text{md}} = 0.7$	63.56	56.57	72.59	61.45	41299	56408	1103	1331	434	179	4466	76.73
$\tau_{\text{lo}} = 0.1$	63.78	56.67	72.99	61.14	39461	58990	1171	1351	427	166	4469	76.98
$\tau_{\text{lo}} = 0.3$	63.71	56.72	72.76	61.36	39771	58148	1137	1350	427	167	4469	76.88
$\tau_{\text{hi}} = 0.75$	63.65	56.65	72.69	61.29	40050	58023	1162	1341	437	166	4488	76.81
$\tau_{\text{hi}} = 0.85$	63.72	56.66	72.87	61.19	39762	58613	1118	1346	426	172	4482	76.82
$\tau_{\text{hi}} = 0.95$	63.26	56.25	72.32	60.43	40144	60185	1108	1356	405	183	4340	76.19
UGO	63.83	56.72	73.03	61.39	39670	58165	1136	1349	425	170	4473	77.09

tracks whose masks were corrected by consolidation (e.g., distractor removal reduces overlap with the pre-consolidation mask). We evaluate $\tau_{\text{lo}} \in \{0.1, 0.3\}$ around the default $\tau_{\text{lo}} = 0.2$. A smaller value (0.1) retains more corrected tracker masks, slightly increasing FP, whereas a larger value (0.3) prunes more aggressively but may remove some valid corrected tracks. The differences are minor (Table 8), confirming that consolidation already effectively suppresses unstable masks.

High overlap threshold τ_{hi}. The high threshold τ_{hi} is used to identify *high-agreement* tracker–detector pairs for category-level memory updates and to detect pre/post-consolidation changes that trigger distractor-removed-driven DRM updates. We evaluate $\tau_{\text{hi}} \in \{0.75, 0.85, 0.95\}$ around the default $\tau_{\text{hi}} = 0.8$. Lowering τ_{hi} increases the number of candidate updates (more frequent memory refreshes) with a cost of a modest FN increase. Increasing τ_{hi} to 0.95 makes updates rare and reduces the opportunity to adapt memory, leading to a small drop in HOTA/MOTA/IDF1 (Table 8). The default $\tau_{\text{hi}} = 0.8$ provides a conservative update regime.

Overall robustness to thresholds. Across all ablations in Table 8, varying any single conservative gate leads to only marginal performance changes. The largest observed deviation from the default configuration is observed when we set $\tau_{\text{md}} = 0.45$, corresponding to merely **2%** decrease in MOTA. Over the same sweeps, HOTA varies by at most less than **1%**. These results demonstrate that the proposed thresholds function as *robust gates* rather than delicately tuned hyperparameters. Performance remains stable across a wide operating range, indicating that UGO does not depend on dataset-specific calibration or fine-grained threshold optimization.

Sensitivity to detector and tracker capacity. We evaluate how performance of the proposed method depends on the capacity of its detection and tracking components on GMOT-40 [3]. The reference model uses SAM2.1 Hiera-L. We consider two settings: (i) replacing the shared backbone with Hiera-B+ or Hiera-S and retraining the GeCo2 detection head for compatibility, and (ii) retaining the original Hiera-L detector while replacing the tracker with SAM2.1 Hiera-S, SAM2.1 Hiera-T, or EdgeTAM. Results are reported in Table 9.

Using Hiera-B+ reduces the complete model from 234M to 87M parameters, a 63% reduction, while retaining 96% of the reference HOTA and 96% of its IDF1. The smaller Hiera-S configuration reduces the parameter count by 76%, retaining 89% of HOTA and 89% of IDF1. For Hiera-S, AssA retains 94.8% of its reference value, compared with 84% for DetA, indicating that detection accuracy is more affected by reduced backbone capacity than association accuracy.

The Hiera-B+ model also uses 20% fewer parameters than S-DETR Swin-B while improving MOTA by 7.45 points and IDF1 by 22.60 points. The Hiera-S model uses 49% fewer parameters, achieves comparable MOTA, and improves IDF1 by 16.89 points. These results show that UGO remains effective with substantially smaller detector and tracker components.

Table 9: Effect of detector and tracker capacity on GMOT-40. For UGO, configurations are specified as detector/tracker; L, B+, S, and T denote Hiera backbone variants.

Configuration	Params. (M)	HOTA↑	DetA↑	AssA↑	MOTA↑	IDSw↓	FM↓	IDF1↑
UGO (L/L)	234	63.83	56.72	73.03	61.39	1136	4473	77.09
UGO (B+/B+)	87	61.44	53.94	71.11	57.45	1219	4080	73.90
UGO (S/S)	56	57.02	47.80	69.24	49.88	1469	3809	68.19
UGO (L/S)	–	61.56	54.35	70.90	55.92	1574	4240	73.72
UGO (L/T)	–	59.73	52.99	68.54	53.60	1353	4450	71.33
UGO (L/EdgeTAM)	–	51.80	49.31	55.55	45.82	5335	6912	60.00

Keeping the original detector and substituting Hiera-S or Hiera-T tracking reduces HOTA by 4% and 6%, respectively. The Hiera-S and Hiera-T configurations outperform S-DETR Swin-B by 12% and 7% in MOTA, and by 44% and 39% in IDF1, respectively. The more aggressive EdgeTAM substitution produces a larger decrease, particularly in association accuracy: relative to the reference UGO, AssA decreases by 24%, while identity switches increase by 367%. UGO therefore tolerates reductions in tracker capacity, although substantial tracker compression remains challenging.

Runtime and memory scaling. We measure end-to-end inference on GMOT-40 using a single A100, including frame encoding, detection, instance propagation, consolidation, and memory updates. Table 10 reports throughput and maximum sampled GPU memory across object-count ranges. UGO is faster than SAM3 under the same evaluation setting, but processing highly crowded sequences remains a challenge for which UGO was not optimized.

E Application to Single-target Tracking

Although UGO is primarily designed for tracking multiple objects within a specified category, it can be naturally adapted to the single-object tracking (SOT) setting, where only a single instance annotated in the first frame must be tracked throughout the sequence. To further test UGO's cross-task generalization capability, we evaluate it on a recent challenging dataset for single-target tracking DiDi [36].

Only a minimal modification of the original algorithm is required: we disable instance termination for the initialized target. This adjustment ensures that the tracker remains active even during extended periods of occlusion or temporary disappearance - scenarios that occur more frequently in SOT than in gMOT - to enable target re-detection.

Table E compares UGO with the tracking foundation model SAM2.1 [33] and its recent state-of-the-art SOT extension DAM4SAM [36]. UGO surpasses SAM2.1 by 5.5% and achieves performance comparable to DAM4SAM, with only 1.3% lower tracking quality. Considering that UGO is fundamentally a multi-target tracker, this result highlights its strong generalization capability across different tracking paradigms.

F Qualitative Results

Additional qualitative results of multi-object tracking with UGO are presented in Figure 7. Each tracked instance is visualized using a semi-transparent colored mask, accompanied by a trajectory line indicating its motion over past frames. We show 16 video sequences from different datasets, displaying two representative frames per sequence. The examples span a wide range of object categories – including airplanes, balls, balloons, and various animals (e.g., birds, penguins, ducks, deer, bees, fish), as well as people – highlighting the strong cross-category generalization capability of UGO.

Table 10: Runtime and memory usage on GMOT-40. Memory denotes maximum sampled GPU allocation.

Objects per sequence	UGO FPS	GPU memory (GiB)	SAM3 FPS
0–20	5.02	4.0	2.58
21–40	3.31	6.5	1.36
41–80	2.46	7.9	0.84
81–140	1.89	10.9	0.49
> 140	1.12	14.9	0.40

Table 11: Single-target tracking performance on the DiDi dataset [36] measured using the following standard performance measures: tracking quality, accuracy and robustness.

Method	Quality	Accuracy	Robustness
SAM2.1	0.649	0.720	0.887
DAM4SAM	0.694 ①	0.727 ①	0.944 ①
UGO	0.685 ②	0.724 ②	0.932 ②

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in abstract and introduction are reflected in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Although the submitted paper does not have a separate Limitations section, they are discussed in the experiments section elaborating the performance of UGO, and further discussed in the suplementary material.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

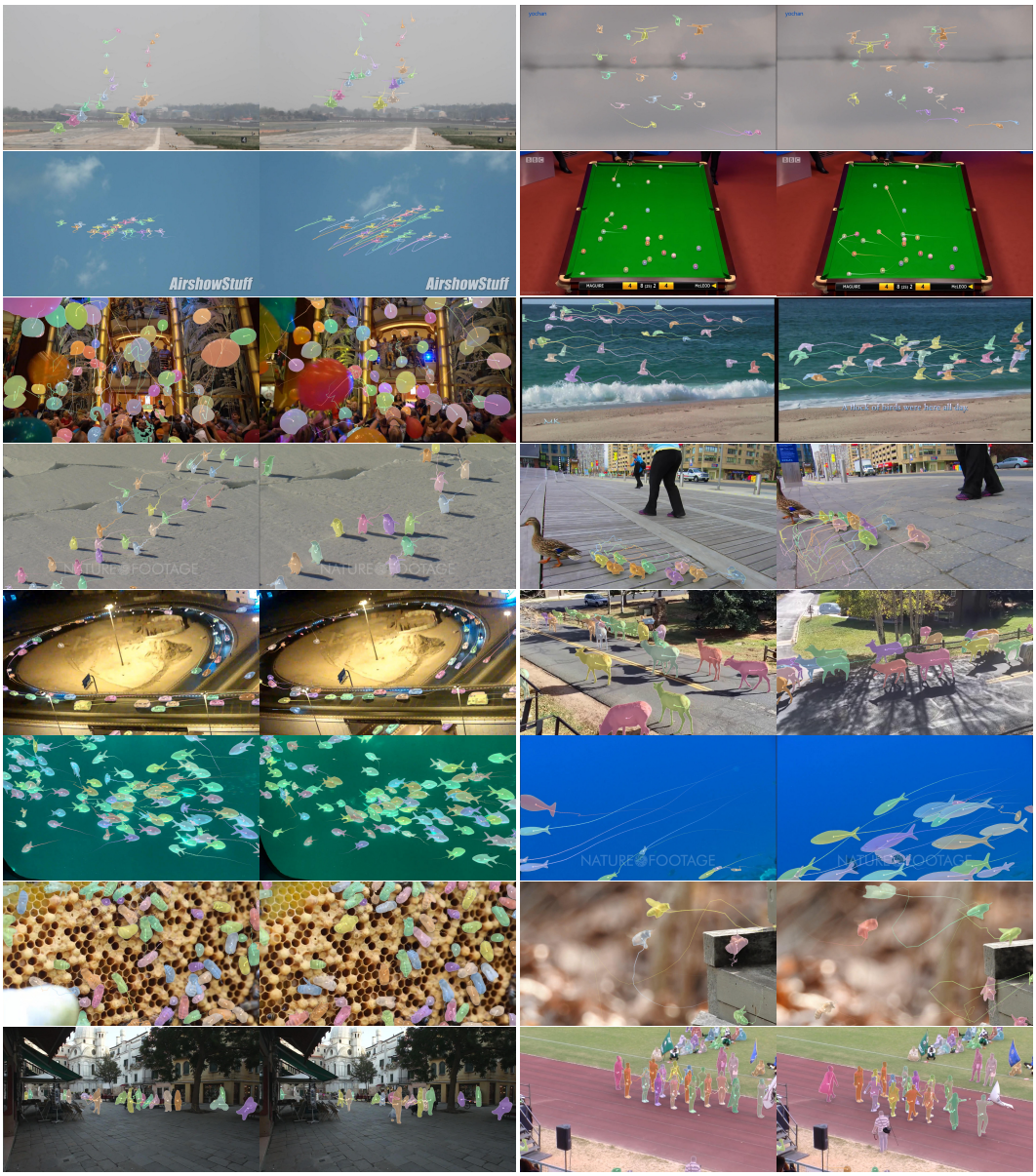

Figure 7: Qualitative examples of tracking with UGO. Each instance is represented by a colored mask and line, denoting its trajectory.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Full proof is included in the supplementary paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, all the details are included in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The method is clearly described, making it possible to re-implement. We will make the code publicly available upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the hyperparameters are ablated in supplementary.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Since the contribution is a training-free module, we include ablations of all hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Described in the Experiments section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed NeurIPS Code of Ethics, and paper is in agreement with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: Our work will not make any societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite all the creators of the used assets.

The identified licenses are: MOT17/MOTChallenge: CC BY-NC-SA 3.0; AnimalTrack annotations: CC BY 4.0, with videos provided for non-commercial research use; GMOT40: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. DiDi:

CC BY 4.0; SAM 2.1 checkpoints/code: Apache-2.0; GeCo/GeCo2 code: MIT, Science-Count: MIT.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing is involved in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No crowdsourcing is involved in the paper.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.