S2D: Sparse-To-Dense Keymask Distillation for Unsupervised Video Instance Segmentation

Leon Sick Ulm University
Lukas Hoyer Google
Dominik Engel KAUST
Pedro Hermosilla TU Vienna
Timo Ropinski Ulm University

Abstract

In recent years, the state-of-the-art in unsupervised video instance segmentation has heavily relied on synthetic video data, generated from object-centric image datasets such as ImageNet. However, video synthesis by artificially shifting and scaling image instance masks fails to accurately model realistic motion in videos, such as perspective changes, movement by parts of one or multiple instances, or camera motion. To tackle this issue, we propose an unsupervised video instance segmentation model trained exclusively on real video data. We start from unsupervised instance segmentation masks on individual video frames. However, these single-frame segmentations exhibit temporal noise and their quality varies through the video. Therefore, we establish temporal coherence by identifying high-quality keymasks in the video by leveraging deep motion priors. The sparse keymask pseudo-annotations are then used to train a segmentation model for implicit mask propagation, for which we propose a Sparse-To-Dense Distillation approach aided by a Temporal DropLoss. After training the final model on the resulting dense labelset, our approach outperforms the current state-of-the-art across various benchmarks.

1 Introduction

Video instance segmentation is a key task in perceptive computer vision, empowering a range of applications, such as autonomous driving, AR/VR applications, and video editing. However, unlike image segmentation, video instance segmentation is fundamentally more challenging because models must learn temporal dynamics and maintain identity-preserving masks across occlusions and scene changes. In recent years, a range of supervised video instance segmentation models have advanced rapidly [43, 4, 37, 38, 44], steadily increasing segmentation accuracy. However, many of these models rely on per-frame human annotations, which require immense annotation efforts and thus result in high costs. For example, SA-V [24] required crowdworkers for 190K manual annotations. While heavily supervised foundation models like SAM 2 [24] provide a strong upper-bound for video segmentation, they fundamentally rely on these massive annotation efforts for training and often require manual prompting at inference. To bypass these bottlenecks and costly efforts, the field of unsupervised video instance segmentation has emerged. Advancing this paradigm remains critical, as it answers the fundamental research question of how models can learn to track and separate instances in raw video without requiring any task-specific video annotations. This research area follows the same core motivation as unsupervised image segmentation: Enabling segmentation without target-task supervision, therefore entirely removing the need for human labeling anywhere in the pipeline [34, 27, 1, 35, 10, 28, 9]. The current state-of-the-art, VideoCutLER [35], trains on synthetic sequences by shifting copy-pasted ImageNet [6] instances. However, this copy-paste synthesis introduces unnatural boundary artifacts and simulates simplistic, single-object translations, failing to capture real-world multi-instance dynamics, occlusions, and complex trajectories.

To overcome these limitations, our goal is to learn unsupervised video instance segmentation exclusively from real-world videos. Transitioning from image to video domains poses challenges: applying

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

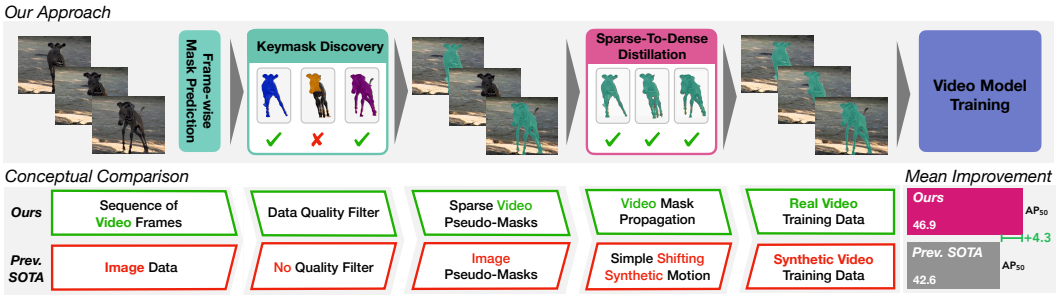

Figure 1: **Overview & Conceptual Comparison.** While the previous state-of-the-art [35] trains on synthetic videos generated by shifting image masks, we propose a method to **learn Unsupervised Video Instance Segmentation from real videos**. Given single-frame unsupervised instance masks, we first discover temporally-coherent, high-quality keymasks. The resulting sparse labelset is then propagated using Sparse-To-Dense Distillation. Finally, we train our model on the resulting dense labelset and achieve state-of-the-art results across the YouTube-VIS benchmarks [43, 42].

unsupervised image segmenters frame-by-frame yields spatial masks but fundamentally lacks temporal coherence for mask contours and instance identities. Consequently, raw pseudo-annotations extracted from real videos are heavily corrupted by temporal noise when instances undergo severe occlusions, motion blur, or enter and exit the scene.

To make learning from real-world videos viable, we propose **S2D**, an approach for learning video instance segmentation exclusively from real videos without human supervision. We achieve this by introducing technical contributions specifically designed to robustly filter temporal noise and establish implicit tracking. Our proposed method is illustrated in Figure 1. We first predict single-frame masks using an unsupervised image instance segmentation model [27]. However, we observe that the quality of these initial segmentations varies strongly throughout a video due to small perspective or lighting changes. While the model might fail to segment an object in one frame, it can produce a highly accurate mask of the exact same object in another. Identifying these reliable segmentations is challenging in an unsupervised setting, as no ground-truth annotations exist to evaluate mask quality. To overcome this and obtain clean video pseudo-annotations, we propose a novel Keymask Discovery algorithm to evaluate frame-wise mask quality and remove noisy segmentations. We do this by analyzing multiple temporal coherence signals per mask, utilizing instance point tracks as a deep motion prior. While this filtering process yields a temporally coherent annotation set, the resulting masks are inherently sparse. Although these keymasks serve as high-quality anchors, we empirically observe that a Video Instance Segmenter achieves stronger overall performance when provided with temporally dense supervision to learn continuous motion dynamics.

To bridge this gap and fully harness our sparse keymasks, we design a novel distillation-based learning framework for implicit mask propagation and geometric refinement. We employ a distillation setup that trains a VideoMask2Former [4] on these sparse annotations to infer the missing video instance masks, thereby producing dense video annotations. First, we eliminate the loss penalty for missing annotations by introducing a Temporal DropLoss. Second, we propose Sparse-To-Dense Distillation, where the student model learns to generate dense video mask predictions under the guidance of a teacher model. Crucially, this distillation process does not merely copy initial predictions but serves as a refinement mechanism that smooths temporal trajectories and corrects boundary artifacts inherent in single-frame predictions. This yields a dense, high-quality video annotation set, which we further refine in a subsequent round of training. Finally, because our pseudo-label generation process is applicable to any video dataset, we scale our training data beyond the in-domain videos and demonstrate that this yields strong zero-shot performance. In summary, our main contributions are as follows:

1. We propose a novel **Keymask Discovery** algorithm that filters noisy single-frame masks to construct a sparse pseudo-labelset of high-quality, temporally coherent video masks.

2. We introduce **Sparse-To-Dense Distillation** in combination with the **Temporal DropLoss**, a novel approach to train video models on sparse annotations that achieves implicit mask propagation and object discovery. This enables the conversion to a dense video labelset.

3. Building on these technical contributions, we present the first video instance segmentation model **trained exclusively on real videos** without task-specific labels, demonstrating state-of-the-art in-domain and zero-shot performance.

2 Related Work

Unsupervised Image Instance Segmentation. Early unsupervised instance segmentation methods leverage self-supervised features to extract masks. For example, MaskDistill [30] uses a MoCo pixel-grouping prior to train Mask R-CNN [11], while FreeSOLO [32] extracts DenseCL attention maps to train a SOLO model [33]. Another significant direction extends Normalized Cut-based segmentation techniques. Wang et al. propose CutLER [34] by generalizing the use of Normalized Cut on DINO [3] features for mask extraction, first introduced by TokenCut [36], to the multi-instance case. After extracting pseudo-masks with MaskCut, a detection network is trained for multiple rounds with self-training. CuVLER, by Arica et al. [1], enhances the pseudo-mask generation by using a six-DINO-model ensemble and a soft target loss. Finally, CutS3D [27] leverages 3D information to separate instances from semantics. LocalCut is used on a 3D point cloud to cut the instances, and the detector's training is augmented with Spatial Confidence components. We utilize CutS3D in our work to predict single-frame instance masks.
Unsupervised Video Instance Segmentation. Unsupervised video instance segmentation is a challenging task that requires not only the separation of foreground objects from the background but also the differentiation and tracking of multiple distinct instances over time, all without any human annotations [45]. Previous research on unsupervised video segmentation largely proposed approaches for unsupervised video object segmentation [15, 39, 40, 25]. VOS aims to detect all moving objects as a single foreground, without the necessity of distinguishing between individual object instances. A common practice in prior studies on this task [31, 39, 40, 41] is the dependence on optical flow networks. Notably, work by Karazija et al. [15] has leveraged model-predicted point tracking and optical flow as supervision for their segmentation model. MotionGroup [41] train a transformer to extract motion-grouped segmentations using only optical flow and no RGB frames. Furthermore, OCLR [39] propose an object-centric approach, also leveraging model-predicted optical flow. Another line of work tackles unsupervised label propagation throughout a video sequence, using human annotations only for the first frame [3, 13, 29] or by incorporating supervised learning with extensive external labeled data [19]. For unsupervised video instance segmentation, VideoCutLER [35] has proposed copy-paste shifting ImageNet [6] masks to generate synthetic videos. The scale of their synthetic dataset allows the method to scale to a zero-shot model.

3 Method

An overview of our method is presented in Figure 1: We first leverage CutS3D [27] to extract frame-wise instance masks for all videos in the dataset. We then filter those predictions to identify sparse, temporally-coherent keymasks (see Figure 2 and Section 3.1), which are then used to train our implicit mask propagation module using our proposed Temporal DropLoss (Section 3.2). To train this model, we employ a Sparse-To-Dense Distillation setup, which allows the model to smooth out ambiguities remaining in the pseudo-annotations in a dual-stage setup (Figure 3 and Section 3.3). The resulting model predicts high-quality video instance masks, as is shown in Figure 4.

3.1 Temporal Grouping for Keymask Discovery

We start with unsupervised single-frame predictions from CutS3D [27] as the basis for video pseudo-annotation. However, these predictions lack temporal coherence since they do not encode which mask in frame 1 belongs to frame 2 and often exhibit inconsistencies. To establish temporal coherence and filter noisy masks, we propose a novel unsupervised keymask discovery method.
Obtaining Mask-Point Trajectories. Using an off-the-shelf point tracker, CoTracker3 [14], we initialize a sparse point grid for each single frame mask and track it forward/backward through the video. Practically, we initialize a set of N_i instance points $\mathbf{P}_i^t = \{p_1^t, \ldots, p_{N_i}^t\}$ for a given instance mask m_i^t of instance i at time t. We track all points throughout the video for all time steps $t \in \{1, \ldots, T\}$, i.e. forward & backward, and record their coordinates and visibilities. This results in a set of trajectories $\mathcal{T}_i = \{\tau_1, \ldots, \tau_{N_i}\}$ for each instance i, where each trajectory $\tau_j = \{(x_j^t, v_j^t) \mid t = 1, \ldots, T\}$ encapsulates the coordinates $x_j^t \in \mathbb{R}^2$ and visibility $v_j^t \in \{0, 1\}$ of point j over the video duration.

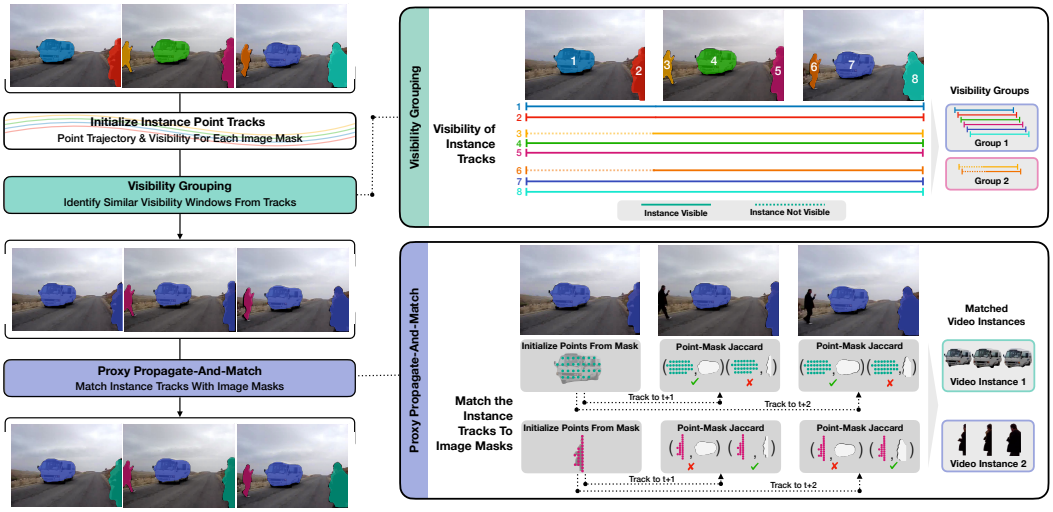

Figure 2: **Keymask Discovery.** Given single-frame unsupervised instance masks, we discover temporally-coherent keymasks by first generating instance tracks, then matching image instance masks based on temporal coherence concepts.
Visibility Grouping: Instance Tracks of Single-Frame Masks belonging to the same video instance have similar visibility windows.
Proxy Propagate-And-Match: In a Visibility Group, frame masks can be temporally associated if their instance tracks match other masks.

With trajectory and visibility data present for each single-frame mask in the video, we employ it as a proxy to validate our fundamental temporal coherence concepts.
Visibility Grouping. In the next step, we validate that single-frame mask tracks of the same temporal instance appear and disappear at the same time, and we verify this concept in the following section. For each single-frame instance mask m_i^t, we determine if its proxy-propagation, given by the instance trajectories $\mathcal{T}_i$, is visible at any given timestep t in the video. This is the case if the proportion of tracked points that are visible at a given frame, $\gamma_t = \frac{1}{N_i}\sum_{j=1}^{N_i} v_j^t$, exceeds a threshold γ_{thr}. In practice, we consider instances as occluded at $\gamma_{\text{thr}} < 0.3$. We collect these binary visibility tracks, displayed in Figure 2, for each single-frame mask across the video and cluster them with DBSCAN [8]. The input to the clustering algorithm is the set of binary visibility vectors $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_M\}$, where each vector $\mathbf{v}_i = \{\mathbf{1}(\gamma_t > \gamma_{\text{thr}}) \mid t = 1, \ldots, T\}$ represents the sequence of visibility states for a single mask m_i^t across the video duration and $\mathbf{1}(\cdot)$ is the indicator function that is 1 if the condition is true and 0 otherwise. The resulting clusters $\mathcal{C} \subset \mathcal{V}$ contain all single-frame masks whose proxy-propagations appear and disappear together, i.e., the instance tracks are visible at the same time. We find this provides a useful initial grouping to find masks of temporally-coherent instances. In the example in Figure 2, the human on the left appears later, hence it is a different video instance. In addition, the clustering returns outlier detections $\mathcal{O} \subset \mathcal{V}$ for masks that cannot be grouped by visibility.
Proxy Propagate-And-Match. Our Visibility Grouping correctly identifies instances with separate visibility windows. However, when multiple instances share the same visibility (e.g., the van and person in Figure 2), we further subdivide the groups using our proxy propagate-and-match approach. This validates our second temporal coherence concept by measuring spatial consistency across frames. Specifically, a mask's point tracks are used to proxy-propagate the instance to a target frame, requiring high overlap with the single-frame mask in that target frame. We restrict this search to masks within the same Visibility Group, which serves as an effective initial grouping.
To compute this overlap, we again leverage the instance trajectories $\mathcal{T}_i$ for each single-frame mask within a visibility cluster $\mathcal{C}$. For a reference instance i and its propagated point set $\mathbf{X}_i^t = \{x_1^t, \ldots, x_{N_i}^t\}$ at the frame of timestep t, we calculate the degree of overlap with every instance mask m_k^t in this frame. Specifically, we propose to compute the point-mask intersection between the point set $\mathbf{X}_i^t$ and the instance masks m_k^t using the Point-Mask Jaccard Index $\mathcal{J}(i, k, t)$, which measures the overlap of the propagated points of instance i and the mask of image instance k of the frame at time t. Given that $\mathbf{X}_i^t$ is a sparse set of N_i points and m_k^t is a dense set of pixels, the

Point-Mask Jaccard Index is formally expressed as:

$$\mathcal{J}(i,k,t) = \frac{\sum_{j=1}^{N_i} \mathbf{1}(x_j^t \in m_k^t)}{N_i}, \quad x_j^t \in \mathbf{X}_i^t \tag{1}$$

If $\mathcal{J}(i,k,t) > \lambda_{\mathcal{J}}$, we $\mathbf{X}_i^t$ matched with the mask m_k^t. In practice, we set $\lambda_{\mathcal{J}} = 0.5$.

We compute the matches for all instance point-tracks to all instance masks within a visibility cluster $\mathcal{C}$. The matching process yields a binary matching matrix $\mathbf{M} \in \{0,1\}^{N_{\text{masks}} \times N_{\text{masks}} \times (T)}$, where the match is defined by the following indicator function:

$$\mathbf{M}_{i,k}^t = \mathbf{1}(\mathcal{J}(i,k,t) > \lambda_{\mathcal{J}}) \tag{2}$$

We collect the set of all matching tracks where each element represents the temporal sequence of matches for a point track i . We then cluster the resulting matching tracks again with DBSCAN [8], which provides matched subgroups within a visibility cluster. In the case of the van-person visibility group in Figure 2, this leads to both instances being separated successfully. With this, we arrive at the final instance mask assignments over time, which meet the defined temporal coherence concepts. Since all masks which do not meet these criteria are filtered out, this pseudo-annotation set is inherently sparse. For cases where no temporal coherence can be established from the image instance masks, the video is discarded from the annotation set. Note that other unsupervised methods have also relied on model-predicted motion cues [15, 19, 41, 39, 40]. However, contrary to existing approaches, our method does not use point tracks or optical flow as supervision to the model.

3.2 Temporal DropLoss For Mask Propagation

Our goal is to obtain a model that makes temporally dense predictions by learning only from sparse annotations, thereby achieving implicit mask propagation. The Keymask Discovery process yields a sparsely annotated pseudo-label set where instances lack masks in certain frames. Crucially, because Keymask Discovery operates over the entire video duration, these sparse anchors provide long-term instance association, bridging occlusion gaps that short-window video models typically struggle with. To compensate for temporal sparsity, we first modify the dataloading process. When sampling snippets from a video, we prioritize windows with at least one temporally dense instance mask to construct pseudo-dense training samples. While training on this data provides samples with annotated temporal consistency, these video snippets often also contain sparsely labeled instances alongside the densely annotated ones. For example, a second instance in the video snippet might only have a mask for 1 of 2 frames. Computing a standard video instance segmentation loss on this sample would assign a penalty for predicting the second, un-annotated mask. To address this, we propose a Temporal DropLoss for video instance segmentation: For instances with sparse annotations, we limit the loss computation to the frame(s) where a given instance is annotated, and drop the loss for frames without an annotated mask. We find this encourages mask propagation to sparsely labeled instances since the model has learned to propagate temporally-coherent masks from the densely annotated instance samples in the data. Using VideoMask2Former [4], which employs bipartite matching between pseudo ground-truths and predictions, we formulate our Temporal DropLoss directly on the model's matched predictions. Given a set of sparse instance mask annotations $\mathbf{A}^t$ and matched model predictions $\hat{\mathbf{M}}^t$, we drop the segmentation loss $\mathcal{L}_{\text{mask}}$ for a given instance i at frame t if the sum of its pseudo-annotated binary mask $\mathbf{m}_i^t$ is zero, i.e. no matching mask is present at t. For a matched instance prediction, the loss is formally defined as:

$$\mathcal{L}_{\text{TempDrop}}(i) = \sum_{t=1}^{T} \mathbf{1}(\|\mathbf{m}_i^t\| > 0)\mathcal{L}_{\text{mask}}(\hat{\mathbf{m}}_i^t, \mathbf{m}_i^t) \tag{3}$$

where $\|\mathbf{m}_i^t\|$ denotes the total number of pixels in the pseudo ground-truth mask. $\mathcal{L}_{\text{mask}}$ is the standard Mask2Former [5] mask loss which combines a binary cross-entropy loss with a dice loss [21], i.e.

$$\mathcal{L}_{\text{mask}} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}} \tag{4}$$

Our approach can be understood as the temporal extension of Image DropLoss [34], which is a popular technique in unsupervised image instance segmentation to counteract the penalty of predicting masks that are completely disjoint from the pseudo-annotations.

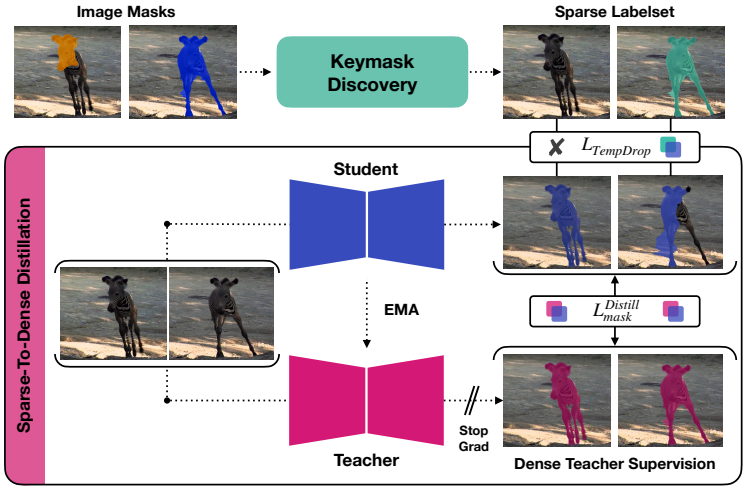

Figure 3: **Sparse-To-Dense Distillation.** We supervise the student model with our Temporal DropLoss on the sparse anchor labelset. Additionally, the teacher model provides dense supervision on-the-fly, which improves as training progresses.

3.3 Dual-Stage Distillation Training

To train our final video instance segmentation model, we use a dual-stage setup. In the first stage, we propose a Sparse-To-Dense Distillation approach, illustrated in Figure 3, to overcome the temporal sparsity of our keymasks and achieve implicit mask propagation and refinement throughout the video. In the second stage, we utilize these new dense pseudo-annotations to train our final model.
Specifically, our Sparse-To-Dense Distillation establishes a student-teacher framework where both models are a VideoMask2Former. The student receives two learning signals. First, it is trained directly on the sparse keymask annotations using our proposed Temporal DropLoss. In this context, the sparse keymasks serve as an anchoring labelset, and the loss inherently encourages the student to learn temporally consistent mask propagation.
Concurrently, the teacher model's weights are updated via an exponential moving average (EMA) of the student's weights, controlled by the update rate μ. As training progresses, the teacher steadily improves at predicting temporally dense video instance segmentations. We harvest this learning signal by tasking the student to match the teacher's dense predictions on the fly. As a result, this distillation refines the student's temporal coherence capabilities, while the simultaneous Temporal DropLoss supervision ensures the model does not deviate too far from the pseudo-ground truth anchors. With this, the full loss can be formulated as:

$$\mathcal{L}_{\text{Full}} = \mathcal{L}_{\text{TempDrop}} + \mathcal{L}_{\text{mask}}^{\text{Distill}} \tag{5}$$

Beyond temporal propagation, this framework inherently serves as a mask refinement mechanism. Because the teacher averages the student weights via EMA throughout training, its predictions smooth over the frame-by-frame geometric artifacts and boundary inaccuracies present in the initial single-frame CutS3D masks. In the second stage, we employ the same student-teacher setup. Using the trained stage 1 Sparse-To-Dense model, we let the model predict on the training videos and utilize these newly refined and dense pseudo-annotations as our anchoring labelset. The new student model learns from these temporally dense pseudo-annotations and is additionally tasked to match the teacher's predictions. Through this second round of distillation, the model learns fine-grained instance segmentations by self-distilling its own predictions. We also observe that keeping the dense anchoring annotation set as direct supervision helps the stable convergence of the model.

4 Experiments

Training Datasets. We train our models under two distinct data regimes: in-domain datasets and scaled zero-shot dataset mix. For our in-domain experiments, we train exclusively on the training splits of YouTube-VIS (YTVIS) 2019 [43] and 2021 [42]. To investigate how our approach scales beyond a narrow target domain, we construct a large-scale, diverse video dataset to train our zero-shot

Table 1: **Unsupervised Video Instance Segmentation.** Our in-domain model is able to outperform the previous state-of-the-art across YouTubeVIS-2019 & YouTubeVIS-2021. We evaluate all models on the public validation split. *Results obtained using official checkpoint.

	YouTube-VIS 2019						YouTube-VIS 2021					
Method	AP_{50}	AP_{75}	AP	AP_S	AP_M	AP_L	AP_{50}	AP_{75}	AP	AP_S	AP_M	AP_L
MotionGroup [41]	0.5	0.0	0.1	0.0	0.4	0.1	0.3	0.0	0.1	0.0	0.2	0.1
OCLR [39]	5.5	0.3	1.6	0.1	1.6	6.1	4.3	0.1	0.9	0.1	1.0	4.9
CutLER [34]	36.4	13.5	16.0	3.5	13.9	26.0	28.5	9.3	12.0	2.7	12.7	27.2
VideoCutLER* [35]	48.2	22.8	24.5	**7.0**	17.5	36.1	37.0	16.1	18.0	4.9	17.0	36.8
S2D (Ours)	**51.2**	**25.2**	**26.2**	6.0	**21.2**	**39.4**	**42.5**	**17.6**	**20.1**	**5.1**	**21.7**	**41.0**
Δ vs. SOTA	**+3.0**	**+2.4**	**+1.7**	-1.0	**+3.7**	**+3.3**	**+5.5**	**+1.5**	**+2.1**	**+0.2**	**+3.1**	**+4.2**

model. We combine the training splits of VIPSeg [20], MOSEv1 [7], and all horizontal videos from SA-V [24]. After applying our Keymask Discovery to filter out unsuitable videos, we yield a dataset of 13,171 videos—far surpassing the 2,900 videos in YTVIS. This demonstrates our method's ability to scale much more efficiently than VideoCutLER [35], reaching superior performance with just 13K real-world videos, compared to 1.3 million synthetic videos.

Evaluation Protocols. We evaluate our approach across three protocols. First, for in-domain evaluation, models trained on YouTube-VIS 2019 and 2021 are tested on their respective validation splits. Second, we assess zero-shot generalization on out-of-domain benchmarks: the YouTube-VIS 2019, 2021 & 2022 validation splits, DAVIS-all [22], and challenging, highly occluded scenes from OVIS [23], and the "things" classes of VIPSeg [20] and Cityscapes-VPS [16] in Table 3. Finally, we evaluate our model as a pre-trained initialization by conducting semi- and fully-supervised fine-tuning on varying percentages of YouTube-VIS 2021 (Table 4). Further details are in the appendix.

Implementation Details. Our pipeline begins by predicting single-frame masks with CutS3D [27] on the training datasets, followed by the extraction of temporally coherent keymasks. For the YTVIS datasets, we apply our full Dual-Stage Distillation. For the zero-shot model, we only perform one distillation stage to train our model. We apply the same weight initialization as Wang et al. [35] and use a VideoMask2Former [4] with a ResNet-50 [12] backbone, trained with video snippets of 2 frames, as is standard for this model. After training, we keep the teacher model for evaluation.

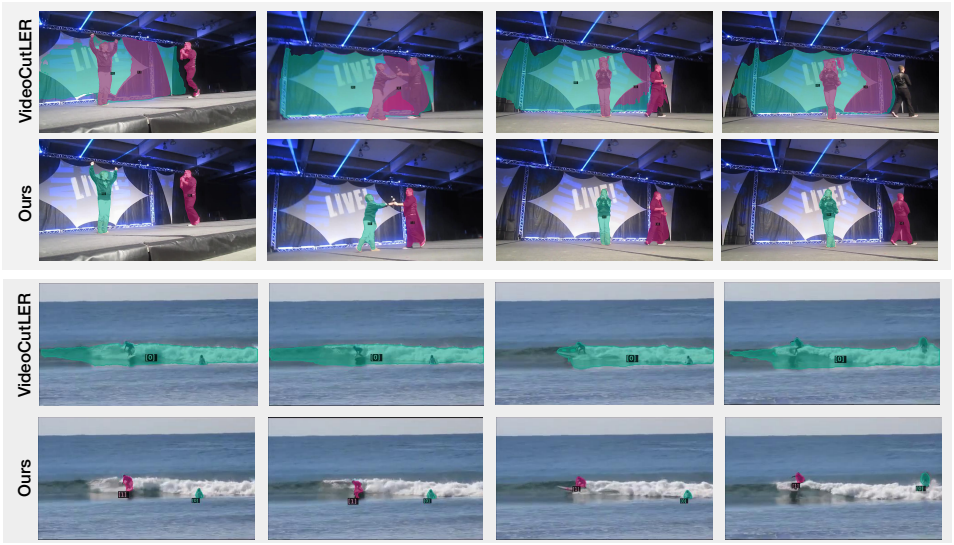

Figure 4: **Qualitative Results.** Our unsupervised model is able to segment more fine-grained structures in videos compared to VideoCutLER [35]. Furthermore, our predictions are less noisy.

Table 2: **Zero-Shot Unsupervised Video Instance Segmentation.** By scaling our dataset to 13K real videos, we demonstrate that our method yields a powerful zero-shot model, successfully outperforming the VideoCutLER model trained on 1.3 million synthetic videos.

Dataset		YouTube-VIS 2019		YouTube-VIS 2021		YouTube-VIS 2022		DAVIS-all	
Method	Training Data	AP_{50}	AP	AP_{50}	AP	AP_{50}	AP	$J\&F$	J-Mean
VideoCutLER [35]	1.3M Syn. Videos	48.2	24.5	37.0	18.0	31.7	15.0	44.9	44.8
S2D (Ours)	13K Real Videos	**51.4**	**25.6**	**41.8**	**19.8**	**35.3**	**16.4**	**49.8**	**49.7**
Δ vs. SOTA		**+3.2**	**+1.1**	**+4.8**	**+1.8**	**+3.6**	**+1.4**	**+4.9**	**+4.9**

Table 3: **Zero-Shot Evaluation on Challenging Datasets.** We evaluate our zero-shot model on datasets featuring high occlusion and complex dynamic scenes. By scaling up to 13K training videos, our method demonstrates robust generalization. *VIPSeg removed from the training dataset mix.

Dataset		VIPSeg-things		OVIS		Cityscapes	
Method	Training Data	AP_{50}	AP	AP_{50}	AP	AP_{50}	AP
VideoCutLER [35]	1.3M Syn. Videos	5.6	4.8	7.7	2.0	1.7	0.9
S2D (Ours)	13K Real Videos	**6.2***	**5.1***	**8.9**	**2.7**	**3.9**	**1.4**
Δ vs. SOTA		**+0.6**	**+0.3**	**+1.2**	**+0.7**	**+2.2**	**+0.5**

4.1 Unsupervised Video Instance Segmentation

In-Domain Evaluation. After training our model in-domain, we report our results in Table 1. Across all datasets, we compare our approach against baselines from VideoCutLER [35]. This includes the image-trained CutLER [34] model, and two approaches utilizing model-predicted optical flow estimations, OCLR [39] and MotionGroup [41]. For both YTVIS datasets, our model outperforms the best-performing baseline by up to **+2.1** AP. We re-evaluated the official open-source VideoCutLER checkpoint on the standard *validation* split. This rigorous protocol accounts for the numerical differences between our reported baselines and those in the original publication. We show qualitative results on YTVIS-2021 in Figure 4, demonstrating the capability of our model to make fine-grained predictions and follow the temporal dynamics of instances.

Zero-Shot Evaluation. Beyond in-domain evaluation, we assess the zero-shot generalization of our model trained on the combined 13K real-world video dataset. For this data-scaled regime, we perform a single round of Sparse-To-Dense Distillation on the extracted pseudo-annotations, which we empirically find to be sufficient. As reported in Table 2, our approach consistently outperforms VideoCutLER across all standard benchmarks, improving by **+3.2**, **+4.8**, and **+3.6** AP_{50} on YouTube-VIS 2019, 2021, and 2022, respectively. On the DAVIS-all dataset, we surpass the baseline by **+4.9** $\mathcal{J}\&\mathcal{F}$. Crucially, we achieve these results using only 13K real videos instead of 1.3M synthetic videos. While the synthetic videos of VideoCutLER contain 3 frames and 1.68 instances on average, our real videos contain 48.2 frames and 4.83 pseudo-annotated instances (Table 21). This gain in data efficiency demonstrates a fundamental effect of training on real video data. By learning from the complex temporal dynamics of real-world videos, our model effectively avoids the potential effects of copy-paste synthesis, which can introduce unnatural boundary artifacts and simplistic, single-object motion. Finally, we evaluate our model on more complex datasets in Table 3. These benchmarks are challenging for unsupervised methods due to prolonged occlusions, multi-instance deformations, and heavily crowded scenes. While absolute performance naturally drops for both methods under these conditions, highlighting that highly dynamic, heavily occluded scenes remain an open challenge for task-annotation-free methods, our approach still consistently outperforms VideoCutLER. We verify that these gains are stable over three random seeds in Appendix D.8.

4.2 Semi- & Fully-Supervised Video Instance Segmentation

We experiment with semi-supervised & supervised fine-tuning with our models as weight initialization.
Evaluation Protocol. We re-use our model weights as unsupervised pre-trained initialization. For the semi-supervised evaluation, we select varying amounts of annotated videos as labeled subset to fine-tune the model. For the fully-supervised evaluation, we train the model on the entire training dataset. We provide further training details in the appendix.

Results. We report our evaluation results on the YouTube-VIS 2021 validation split in Table 4. For low-data regimes, such as using only the 10% of labeled training data, our model outperforms

Table 4: **Semi- & Fully-Supervised Video Instance Segmentation.** We show that in low-data regimes, our model trained on real videos outperforms VideoCutLER with ImageNet training across various data splits on YouTubeVIS-2021 [42].

Method	1%		5%		10%		25%		100%	
	AP_{50}	AP	AP_{50}	AP	AP_{50}	AP	AP_{50}	AP	AP_{50}	AP
DINO*[3]	20.0	11.1	18.9	9.4	21.8	11.5	20.9	12.3	43.3	26.3
VideoCutLER*[35]	23.9	14.5	22.8	14.3	25.7	16.2	32.8	20.9	51.0	32.4
S2D (Ours)	**25.1**	**15.2**	**25.4**	**15.4**	**28.4**	**17.4**	**33.1**	**21.1**	**53.1**	**33.0**

VideoCutLER [35] by **+2.7** AP_{50}. For fully-supervised training, our model remains slightly ahead of the baseline with **+0.6** AP. This is despite VideoCutLER being pre-trained on over a million synthetic videos, while we pre-train our model only on 13,171 real videos.

4.3 Computational Cost

We compare the computational cost of our approach to VideoCutLER on a single A100 GPU, treating publicly available pre-trained models as given for both, i.e., the shared weight initialization and the CutS3D image segmenter. Our full pipeline requires 1,302 GPU hours compared to 2,399 GPU hours for VideoCutLER, a reduction of 46%. For both methods, most of the cost comes from a one-time pre-processing step with reusable pseudo-annotations, i.e., point tracking with CoTracker3 [14] and Keymask Discovery for our approach (1,266 hours) and MaskCut extraction over ImageNet for VideoCutLER (2,347 hours). The training itself requires only 32 GPU hours for our approach compared to 52 GPU hours for VideoCutLER. While our student-teacher setup increases the time per iteration by 60% and the peak memory by 12%, the inference cost is unchanged. Our final model is architecturally identical to VideoCutLER, and both models run at $\sim$54 FPS. We provide a detailed breakdown in Appendix C.

5 Ablations

We ablate the components of Keymask Discovery (Tab. 5) and model training (Tab. 6), and compare our implicit mask propagation against an explicit propagation algorithm (Tab. 7). Finally, we analyze the robustness of our approach to its off-the-shelf components.

Keymask Discovery Components. In Tab. 5, we ablate all components of the Keymask Discovery. As a baseline, we train an Image Mask2Former [5] on the single-frame masks. We then compare training on video annotations from just visibility grouping to the sparse keymasks with proxy propagate-and-match. Finally, we compare to the model performance on the final dense labels. We report results on YouTube-VIS 2021, by training our video instance segmentation model on the resulting annotations in the dual-stage setup. While we find that training a model on single-frame masks results in performance competitive with VideoCutLER [35], the model strongly improves with temporally-coherent sparse annotations from our keymask discovery. Furthermore, training on the dense annotations, obtained by Sparse-To-Dense Distillation, further improves the results.

Model Training. We ablate the modifications to the VideoMask2Former. As a baseline, we train the model on the sparse pseudo-annotations without any modifications. We experiment with just adding pseudo-dense data loading, then we add the Temporal DropLoss. Finally, we compare against training the final model with dual-stage training. We again train and evaluate on YouTube-VIS 2021 and report the performance of our final dual-stage model training. We see improvements of our method from implicit mask propagation with our Temporal DropLoss. When employing Sparse-To-Dense Distillation in addition, our final model further improves to 20.1 AP. The first stage converts the sparse keymasks into a dense labelset, which then anchors the self-distillation in the second stage against drift. Since the EMA averages the student weights over training iterations, the teacher acts as a smoothed version of the student and provides more robust targets for noisy pseudo-masks. Compared to dual-stage training without distillation, our approach improves by **+0.7** and **+1.4** AP_{50} on YouTube-VIS 2019 and 2021 (Table 12), and it also slightly outperforms fine-tuning the stage-1 teacher on the dense labelset. We summarize both stages in Algorithm 2.

Table 5: **Keymask Discovery Components.** Metrics on YTVIS-2021. †Single-Frame Masks trained with an Image Mask2Former, the weights are compatible with the VideoMask2Former.

Components	AP_{50}	AP
Single-Frame Masks†	36.5	17.8
+ Visibility Grouping	38.9	18.9
+ Temporal Matching	40.9	19.5
+ Dense Labelset	**42.5**	**20.1**

Table 6: **Model Design.** We ablate the effect of our technical contributions to our training pipeline on YTVIS-2021. The baseline is a VideoMask2Former trained on sparse labels.

Components	AP_{50}	AP
VideoMask2Former	31.2	14.0
+ Pseudo-Dense Dataloading	39.9	19.3
+ Temporal DropLoss	41.9	19.8
+ Sparse-To-Dense Distillation	**42.5**	**20.1**

Table 7: **Sparse-To-Dense Distillation vs. DINO Propagation.** On DAVIS, we propagate every mask, then match with the ground-truth to calculate the J&F.

Method	$J\&F$	J-Mean	F-Mean
Propagation with DINO [3]	42.2	43.4	41.0
Sparse-To-Dense Distillation	**50.6**	**50.5**	**50.8**

DINO Propagation vs. Sparse-To-Dense Distillation. The purpose of Sparse-To-Dense Distillation and Temporal DropLoss is to enable implicit mask propagation. We experiment with this capability by performing Sparse-To-Dense Distillation directly on the DAVIS-all [22] dataset, separate from the zero-shot evaluation in Tab. 2. We compare our approach to off-the-shelf explicit propagation algorithms [13] that connect semantic encodings of individual frames using DINO [3] features. Since these algorithms propagate instance masks from a starting frame, we perform this propagation for all single-frame masks in the video and select the best result per instance for Tab. 7. Our approach outperforms this explicit propagation alternative.

Robustness to Off-the-Shelf Components. Replacing CutS3D with CutLER or CuVLER reduces the AP by only 0.3 (Table 16). Replacing CoTracker3, which is partly trained on synthetic data, with the self-supervised GMRW [26] costs only 0.7 $\mathcal{J}\&\mathcal{F}$ and removes synthetic data from our entire pipeline (Table 14). When we deliberately degrade the point grids and single-frame masks, AP_{50} stays within 2.6 points of our main result and above VideoCutLER, with small objects being affected most. Finally, Proxy Propagate-and-Match is highly stable under perturbations of its DBSCAN parameters, while Visibility Grouping is mainly sensitive to the neighborhood radius ϵ (Appendix D.6 and D.7).

6 Summary

In this paper, we have introduced S2D, a novel approach to learn unsupervised video instance segmentation from real video data. We have proposed a novel Keymask Discovery algorithm for identifying keymasks with temporal coherence from unsupervised image instance masks. To perform mask propagation for these sparse annotations, we have introduced Sparse-To-Dense Distillation with Temporal DropLoss. Finally, we have trained a self-distilled video instance segmentation model on the resulting dense annotations. Our final model outperforms VideoCutLER in both in-domain and zero-shot settings and achieves state-of-the-art results among unsupervised video instance segmentation methods that do not rely on synthetic training data.

Limitations. Our approach struggles with small objects, as reflected in the lower AP_S on YouTube-VIS 2019 (Table 1). Small instances receive only few points in our point grids, which makes their tracks less reliable, and our pseudo-annotations contain few small instances (Table 20). VideoCutLER partly mitigates this with downscaled copy-paste instances. Furthermore, the performance of both our approach and VideoCutLER drops under prolonged occlusions and in crowded scenes (Table 3), which are also overrepresented among the videos discarded by Keymask Discovery (Appendix I.1). Finally, our zero-shot model struggles with driving-domain data, which is underrepresented in our training mix.

Acknowledgments and Disclosure of Funding

We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LEONARDO, hosted by CINECA (Italy) and the LEONARDO consortium through an EuroHPC Development Access call. The authors declare no competing interests.

References

[1] Shahaf Arica, Or Rubin, Sapir Gershov, and Shlomi Laufer. Cuvler: Enhanced unsupervised object discoveries through exhaustive self-supervised transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23105–23114, 2024.

[2] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 DAVIS challenge on VOS: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737*, 2019.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[4] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G. Schwing. Mask2former for video instance segmentation, 2021.

[5] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. Mose: A new dataset for video object segmentation in complex scenes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 20224–20234, 2023. Data licensed under CC BY-NC-SA 4.0.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[9] Oliver Hahn, Christoph Reich, Nikita Araslanov, Daniel Cremers, Christian Rupprecht, and Stefan Roth. Scene-centric unsupervised panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24485–24495, June 2025.

[10] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. *Advances in neural information processing systems*, 33:19545–19560, 2020.

[14] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025.

[15] Laurynas Karazija, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Learning segmentation from point trajectories. *Advances in Neural Information Processing Systems*, 37:112573–112597, 2024.

[16] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9859–9868, 2020.

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[18] Kaixuan Lu, Mehmet Onurcan Kaya, and Dim P. Papadopoulos. Boosting unsupervised video instance segmentation with automatic quality-guided self-training. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7387–7397, March 2026.

[19] Jonathon Luiten, Idil Esen Zulfikar, and Bastian Leibe. Unovost: Unsupervised offline video object segmentation and tracking. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2000–2009, 2020.

[20] Jiaxu Miao, Xiaohan Wang, Yu Wu, Wei Li, Xu Zhang, Yunchao Wei, and Yi Yang. Large-scale video panoptic segmentation in the wild: A benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21033–21043, 2022. The data is released for non-commercial research purpose only.

[21] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[22] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. Data licensed under CC BY 4.0.

[23] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip HS Torr, and Song Bai. Occluded video instance segmentation: A benchmark. *International Journal of Computer Vision*, 130(8):2022–2039, 2022. Data licensed under CC BY-NC-SA 4.0.

[24] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. Data licensed under CC BY 4.0.

[25] Sadra Safadoust and Fatma Güney. Multi-object discovery by low-dimensional object motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 734–744, 2023.

[26] Ayush Shrivastava and Andrew Owens. Self-supervised any-point tracking by contrastive random walks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[27] Leon Sick, Dominik Engel, Sebastian Hartwig, Pedro Hermosilla, and Timo Ropinski. Cuts3d: Cutting semantics in 3d for 2d unsupervised instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21265–21275, 2025.

[28] Leon Sick, Dominik Engel, Pedro Hermosilla, and Timo Ropinski. Unsupervised semantic segmentation through depth-guided feature correlation and sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3637–3646, 2024.

[29] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems*, 36:1363–1389, 2023.

[30] Wouter Van Gansbeke, Simon Vandenhende, and Luc Van Gool. Discovering object masks with transformers for unsupervised semantic segmentation. *arXiv preprint arXiv:2206.06363*, 2022.

[31] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro-i Nieto. Rvos: End-to-end recurrent network for video object segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5277–5286, 2019.

[32] Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. Freesolo: Learning to segment objects without annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14176–14186, 2022.

[33] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Solo: A simple framework for instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):8587–8601, 2021.

[34] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3124–3134, 2023.

[35] Xudong Wang, Ishan Misra, Ziyun Zeng, Rohit Girdhar, and Trevor Darrell. Videocutler: Surprisingly simple unsupervised video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22755–22764, 2024.

[36] Yangtao Wang, Xi Shen, Yuan Yuan, Yuming Du, Maomao Li, Shell Xu Hu, James L Crowley, and Dominique Vaufreydaz. Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[37] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8741–8750, 2021.

[38] Junfeng Wu, Yi Jiang, Song Bai, Wenqing Zhang, and Xiang Bai. Seqformer: Sequential transformer for video instance segmentation. In *European Conference on Computer Vision*, pages 553–569. Springer, 2022.

[39] Junyu Xie, Weidi Xie, and Andrew Zisserman. Segmenting moving objects via an object-centric layered representation. *Advances in neural information processing systems*, 35:28023–28036, 2022.

[40] Junyu Xie, Weidi Xie, and Andrew Zisserman. Appearance-based refinement for object-centric motion segmentation. In *European Conference on Computer Vision*, pages 238–256. Springer, 2024.

[41] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7177–7188, 2021.

[42] Linjie Yang, Yuchen Fan, Yang Fu, and Ning Xu. The 3rd large-scale video object segmentation challenge - video instance segmentation track, June 2021. Data licensed under CC BY 4.0.

[43] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019. Data licensed under CC BY 4.0.

[44] Tao Zhang, Xingye Tian, Yu Wu, Shunping Ji, Xuebo Wang, Yuan Zhang, and Pengfei Wan. Dvis: Decoupled video instance segmentation framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1282–1291, 2023.

[45] Tianfei Zhou, Fatih Porikli, David J Crandall, Luc Van Gool, and Wenguan Wang. A survey on deep learning technique for video segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 45(6):7099–7122, 2022.

A Additional Qualitative Results

We show additional qualitative examples of predictions from our S2D video instance segmentation model in Figure 5. Our approach shines on examples with multiple instances appearing and disappearing, such as the football players in the second row. Furthermore, our method handles the appearance of a car and a truck in the fourth row example, counting from the bottom. Our method can track instances that become partially visible, e.g., the skater in the first row.



Figure 5: **More Qualitative Results.** Our approach can robustly segment video instances across different scenes.

B Comparison to AutoQ-VIS

AutoQ-VIS [18] is a quality-guided self-training approach for unsupervised video instance segmentation. Given an already trained video instance segmentation model, it iteratively predicts pseudo-masks

Table 8: **Comparison to AutoQ-VIS.** In-domain results on YouTube-VIS 2019 & 2021. AutoQ-VIS refines the pre-trained VideoCutLER model with self-training on in-domain videos, while our approach trains from scratch without synthetic videos. *Results obtained using the official checkpoint. †Results as reported in [18]. ‡Reproduced with the official implementation.

Method	Training Data	AP_{50}	AP_{75}	AP	AP_S	AP_M	AP_L	AR_{10}
YouTube-VIS 2019								
VideoCutLER* [35]	1.3M Syn. Videos	48.2	22.8	24.5	**7.0**	17.5	36.1	42.3
AutoQ-VIS† [18]	1.3M Syn. Videos + YTVIS-2019	**52.6**	**28.2**	**28.1**	6.7	**21.2**	**40.7**	**42.5**
S2D (Ours)	YTVIS-2019	51.2	25.2	26.2	6.0	**21.2**	39.4	42.2
YouTube-VIS 2021								
VideoCutLER* [35]	1.3M Syn. Videos	37.0	16.1	18.0	4.9	17.0	36.8	31.0
AutoQ-VIS‡ [18]	1.3M Syn. Videos + YTVIS-2021	38.4	17.1	19.0	**5.1**	18.6	38.2	**31.4**
S2D (Ours)	YTVIS-2021	**42.5**	**17.6**	**20.1**	**5.1**	**21.7**	**41.0**	31.1

Table 9: **Combining S2D and AutoQ-VIS.** We apply AutoQ-VIS self-training to different initial models and evaluate on YouTube-VIS 2019. Our zero-shot model is trained on 13K real videos and has not been trained on YouTube-VIS 2019 before self-training. *Results obtained using the official checkpoint. †Results as reported in [18].

Initialization	Training Data	Self-Training	AP_{50}	AP_{75}	AP	AP_S	AP_M	AP_L	AR_{10}
VideoCutLER*	1.3M Syn. Videos	None	48.2	22.9	24.5	**6.7**	17.7	36.3	42.3
VideoCutLER	1.3M Syn. Videos	AutoQ-VIS (2 Rounds)†	52.6	**28.2**	28.1	**6.7**	21.2	40.7	**42.5**
S2D Zero-Shot (Ours)	13K Real Videos	None	51.4	25.2	25.6	5.9	20.6	38.9	41.9
S2D Zero-Shot (Ours)	13K Real Videos	AutoQ-VIS (2 Rounds)	**52.9**	28.0	**28.3**	6.3	**22.0**	**40.9**	42.3

on in-domain videos, selects high-quality pseudo-masks with a learned quality predictor, and re-trains the model on the selected pseudo-masks together with synthetic videos. In its original formulation, AutoQ-VIS initializes from the VideoCutLER model, which is trained on 1.3M synthetic videos. In contrast, our approach trains a video instance segmentation model from scratch using only real videos. This leads to two key differences. First, AutoQ-VIS requires an existing video model in which temporal coherence has already been learned, while our approach establishes temporal coherence from single-frame masks through Keymask Discovery. Second, AutoQ-VIS self-trains on the target domain by design, while we additionally evaluate our approach in a zero-shot setting across seven benchmarks.

In-Domain Comparison. We report the comparison in Table 8. On YouTube-VIS 2019, AutoQ-VIS outperforms our in-domain model by 1.4 AP_{50} and 1.9 AP. Since AutoQ-VIS does not report results on YouTube-VIS 2021, we train and evaluate it using the official implementation. On this more challenging benchmark, our approach outperforms AutoQ-VIS by **+4.1** AP_{50} and **+1.1** AP. We believe this is because VideoCutLER, which AutoQ-VIS starts from, struggles more to produce useful pseudo-masks on YouTube-VIS 2021, while our training on real videos is more robust. On AR_{10}, all three methods perform similarly. Among methods that do not use synthetic training data, our approach achieves state-of-the-art results across all benchmarks.

Combining S2D and AutoQ-VIS. Since AutoQ-VIS refines an existing video model, we investigate whether it can also be applied on top of our approach. For a fair comparison, we initialize AutoQ-VIS with our zero-shot model, which is trained on 13K real videos and has never been trained on YouTube-VIS 2019. We then run two rounds of AutoQ-VIS self-training on YouTube-VIS 2019 without using any synthetic videos. As shown in Table 9, this combination reaches 52.9 AP_{50} and 28.3 AP, compared to 52.6 AP_{50} and 28.1 AP for AutoQ-VIS initialized with VideoCutLER. Hence, our model can serve as the initial video model that AutoQ-VIS requires, which removes synthetic videos from the entire training pipeline.

C Computational Cost

We provide a detailed breakdown of the computational cost of our approach and VideoCutLER [35] in Table 10. All numbers are wall-clock hours on a single A100 GPU. For a like-for-like comparison, we re-run VideoCutLER on the same hardware with a proportionally lower batch size. For both

Table 10: **Computational Cost Breakdown.** Wall-clock hours on a single A100 GPU. VideoCutLER is re-run on the same hardware with a proportionally lower batch size.

Stage	Cost (h)
VideoCutLER [35]	
MaskCut pseudo-mask extraction over ImageNet	2,347
Training (80K iterations, on-the-fly copy-paste augmentation)	52
Total	2,399
S2D (Ours)	
CutS3D single-frame mask extraction	4
CoTracker3 & Keymask Discovery (13,171 videos)	1,266
Stage-1 Sparse-To-Dense Distillation (32K iterations)	16
Stage-2 distillation	16
Total	1,302

Table 11: **Training Throughput and Memory.** Measured with a batch size of 4 and clips of 3 frames on a single A100 GPU.

Training Setup	Time per Iteration	Videos per Second	Peak Memory
Vanilla VideoMask2Former	1.44 s	2.78	23.81 GB
Sparse-To-Dense Distillation	2.31 s	1.73	26.63 GB

methods, we treat publicly available pre-trained models as given, i.e., the shared weight initialization and the CutS3D [27] model that we use for single-frame mask prediction. Hence, we do not include their training cost.

The cost of VideoCutLER is dominated by the MaskCut pseudo-mask extraction over the 1.3M ImageNet images, while its training synthesizes the copy-paste videos on the fly. For our approach, the cost is dominated by a single offline pass of CoTracker3 [14] with subsequent Keymask Discovery over our 13,171 zero-shot training videos. Since Keymask Discovery operates on each video independently, this step is embarrassingly parallel and scales linearly with the number of videos. Furthermore, it results in a labelset that can be reused for further training runs. Overall, our pipeline requires 1,302 hours compared to 2,399 hours for VideoCutLER, a reduction of 46%. Note that the second distillation stage is only used for our in-domain models. Our zero-shot model uses a single stage and is therefore cheaper by 16 hours.

Training Throughput and Memory. We measure the training throughput and memory at matched settings, i.e., a batch size of 4 and clips of 3 frames on a single A100 GPU, and report the results in Table 11. Since both the student and the teacher perform a forward pass, our distillation increases the time per iteration by around 60% and the peak memory by around 12% compared to vanilla VideoMask2Former training. However, this overhead only applies to the 32 hours of distillation training out of the 1,302 hours of our full pipeline.

Inference. Keymask Discovery and distillation are only used during training. At test time, our model is a VideoMask2Former with a ResNet-50 backbone and is therefore architecturally identical to VideoCutLER. Both models run at approximately 54 FPS on a single A100 GPU with matching peak memory.

D Further Ablations

D.1 Distillation vs. Vanilla Training

We present Sparse-To-Dense Distillation as a key part of our method, followed by a second distillation round. To ablate the effect of the distillation process, we construct a dual-stage training method without distillation: In the first stage, the model is trained with Temporal DropLoss on the sparse annotations. The resulting model predicts dense pseudo-annotations, which are then used for a second round of vanilla training. We report the results in Table 12. We find that using distillation in our method consistently outperforms the vanilla VideoMask2Former [4] setup for both in-domain YouTube-VIS [43, 42] evaluations, improving by +0.7 AP_{50} and +0.5 AP on YouTube-VIS 2019 and

by +1.4 AP_{50} and +0.3 AP on YouTube-VIS 2021. Note that both variants use the same number of training stages, hence this comparison isolates the effect of the EMA teacher. We summarize both stages of our training in Algorithm 2.

Table 12: **Dual-Stage Distillation vs. Vanilla Training.** Both variants are trained in two stages and evaluated in-domain. The vanilla variant does not use the EMA teacher.

	YTVIS-2019		YTVIS-2021	
Method	AP_{50}	AP	AP_{50}	AP
Dual-Stage Vanilla Training	50.5	25.7	41.1	19.8
Dual-Stage with Distillation	**51.2**	**26.2**	**42.5**	**20.1**

D.2 Keymask Discovery Thresholds

Our proposed Keymask Discovery algorithm is comprised of two parts, Visibility Grouping & Proxy Propagate-and-Match. In the first part, we determine the visibility of an instance trajectory $\mathcal{T}_i$ at timestep t with $\mathbf{1}(\gamma_t > \gamma_{\text{thr}})$. Computing this for each timestep in the video returns a binary visibility vector $\mathbf{v}_i = \{\mathbf{1}(\gamma_t > \gamma_{\text{thr}}) \mid t = 1, \dots, T\}$, which is later used for clustering the instance visibilities. We ablate the γ_{thr} hyperparameter in Table 13a. In the second part, we determine the overlap for a reference instance i and its propagated point set $\mathbf{X}_i^t = \{x_1^t, \dots, x_{N_i}^t\}$ at the frame of timestep t by computing $\mathcal{J}(i, k, t)$, and determine matches with $\mathcal{J}(i, k, t) > \lambda_{\mathcal{J}}$. We ablate the choice of $\lambda_{\mathcal{J}}$ and evaluate the final models on YouTube-VIS 2021 in Table 13b. We provide the DBSCAN parameters of both parts in Appendix D.7.

Table 13: **Keymask Discovery Thresholds.** Ablation of γ_{thr} and $\lambda_{\mathcal{J}}$ with in-domain models on YouTube-VIS 2021.

(a) Different values for γ_{thr}.

γ_{thr}	0.1	0.3	0.5
AP	19.7	**20.1**	20.0

(b) Different values for $\lambda_{\mathcal{J}}$.

$\lambda_{\mathcal{J}}$	0.25	0.5	0.75
AP	19.7	**20.1**	**20.1**

Table 14: **Different Point Trackers for Keymask Discovery.** We compare the resulting models on DAVIS-all [22, 2]. GMRW is self-supervised on real videos and does not use synthetic data.

	CoTracker3 [14]	GMRW [26]
$J\&F$	**50.6**	49.9

D.3 Point Trackers

To establish temporal coherence in our keymasks, we employ an off-the-shelf point tracking model, i.e., CoTracker3 [14]. This model is originally trained on synthetic data. It then makes predictions on real videos to obtain pseudo-labels, which are used for further training. Since our instance trajectories are a core component of the Keymask Discovery algorithm, we ablate the choice of point tracking model. As an alternative, we use point tracks predicted by GMRW [26], a self-supervised point tracker that is trained on real videos without using synthetic data. GMRW takes inspiration from contrastive random walks [13] and performs a global matching random walk, using a global matching transformer to learn cycle-consistent point tracks from a palindrome sequence.

Table 15: **Model Scaling.** In-Domain Evaluation on YTVIS-2021.

Backbone	Params	AP	AP_{50}
ResNet-50	24M	20.1	42.5
ResNet-101	43M	**20.4**	**43.0**

Table 16: **Impact of Image Segmenters.** Comparison of utilizing different unsupervised models for single-frame mask prediction. In-domain results on YouTube-VIS 2021.

Model	CutLER [34]	CuVLER [1]	CutS3D [27]
AP	19.8	19.8	**20.1**

For the comparison, we extract keymasks with instance trajectories from GMRW on DAVIS-all [22, 2], perform Sparse-To-Dense Distillation, and compare the results in Table 14. We find that the performance of the resulting models is similar, with a difference of only 0.7 $\mathcal{J}\&\mathcal{F}$. Importantly, this configuration removes synthetic data from our entire pipeline, including the point tracker. We observe that GMRW can have some individual robustness deficits compared to CoTracker3. For example, an individual point might lose track, but the overall point grid tracking stays relatively accurate. However, this effect on the final performance seems to be marginal since temporal coherence can be established in the keymasks, nevertheless.

D.4 Model Scaling

To investigate how our framework scales, we conduct an ablation study replacing the default ResNet-50 [12] backbone with a higher-capacity ResNet-101 architecture. Both models are trained and evaluated in-domain on the YTVIS-2021 dataset under identical training configurations. As shown in Table 15, increasing the backbone capacity yields consistent improvements across the evaluation metrics. By scaling the parameters from 24M to 43M, we observe improvements in AP and AP_{50}. These results indicate that richer feature representations provided by larger backbones can improve segmentation quality.

D.5 Choice of Unsupervised Image Instance Segmentation Model

To evaluate the sensitivity of our framework to the quality of the underlying single-frame masks, we conduct experiments using various unsupervised image instance segmentation models. Specifically, we compare the downstream video performance when initializing our pipeline with masks generated by CutS3D [27] against those from CutLER [34] and CuVLER [1].

As presented in Table 16, substituting our default segmenter with less accurate models results in only a marginal performance degradation of 0.3 AP. This resilience underscores a core strength of our architecture: the Keymask Discovery module effectively functions as a noise-filtering mechanism, cleaning the input data and mitigating the adverse effects of lower-quality initial masks. Furthermore, this demonstrates the modularity of our framework, confirming that it is compatible with a variety of image segmenters and well-positioned to directly benefit from future advancements in unsupervised 2D segmentation models.

D.6 Sensitivity to Degraded Inputs

Our approach relies on off-the-shelf models to predict single-frame masks and point tracks. Beyond replacing these models (Tables 14 and 16), we analyze how our approach behaves when their outputs are deliberately degraded. For all experiments, we re-run our full pipeline and evaluate the resulting in-domain models on YouTube-VIS 2021.

Point-Grid Density. We reduce the number of points that we sample within each single-frame mask to initialize the point tracks for CoTracker3.

Corrupted and Subsampled Masks. We apply two independent degradations to the CutS3D single-frame masks. First, we corrupt the mask boundaries by randomly applying either morphological erosion or dilation to each mask. The radius of the structuring element is set to a percentage of the effective mask radius $r = \sqrt{A/\pi}$, where A denotes the mask area. Hence, higher percentages remove or add a larger band around the mask boundary. Second, we randomly discard a fraction of the single-frame masks across each video.

We report the results in Table 17. As expected, the performance decreases with stronger degradations. However, even under severe degradations, such as a point-grid density of 25% or a boundary

Table 17: **Sensitivity to Degraded Inputs.** We degrade the point-track initialization and the single-frame masks, re-run our full pipeline, and evaluate in-domain on YouTube-VIS 2021. VideoCutLER achieves 37.0 AP_{50} on this benchmark.

Condition	AP_{50}	AP	AP_S
Default (Ours)	**42.5**	**20.1**	**5.1**
Point-Grid Density			
50%	41.8	19.5	4.5
25%	40.1	18.6	3.7
Single-Frame Masks			
Boundary Corruption 10%	41.6	19.8	4.9
Boundary Corruption 30%	39.9	18.7	4.1
Subsampled, 50% Kept	40.9	19.1	4.5

corruption of 30%, AP_{50} stays within 2.6 points of our main result and remains above VideoCutLER (37.0 AP_{50}). In relative terms, AP_S is the most sensitive metric under every degradation. With a lower point-grid density, small objects receive only very few points, which makes their trajectories unreliable. Similarly, small masks consist of only few pixels, which leaves less margin before they vanish under boundary corruption. This is consistent with the dependency on object scale discussed in our limitations in Section 6.

D.7 Clustering Parameters and Stability

Both stages of our Keymask Discovery cluster binary vectors with DBSCAN using the Hamming distance. For Visibility Grouping, we cluster the visibility vectors of all single-frame masks in a video with a neighborhood radius of $\epsilon = 0.2$ and a minimum number of samples of 5. For Proxy Propagate-and-Match, we select the parameters adaptively based on the number of candidate masks in the matching matrix. For more than 50 candidate masks, which is the most common case, we use $\epsilon = 0.05$ with a minimum of 5 samples. For fewer than 10 candidate masks, we use $\epsilon = 0.1$ with a minimum of 3 samples, and $\epsilon = 0.1$ with a minimum of 5 samples otherwise. We keep all values fixed across all datasets and experiments.

A full sweep over these parameters with re-training is not feasible, since every setting requires re-extracting the pseudo-annotations for the full training set. However, the parameters only affect the final model through the set of masks that the clustering selects. Hence, a stable clustering under parameter perturbations is a necessary condition for a stable final performance, which we can measure directly. We sample 10 videos from YouTube-VIS 2021 and compute the Adjusted Rand Index (ARI) between the clustering of each perturbed setting and our default clustering. An ARI of 1.0 indicates identical clusterings, while an ARI of 0 corresponds to chance-level agreement. Since the Adjusted Mutual Information follows the ARI within about 0.02 to 0.03 across both grids, we only report the ARI.

We report the results in Table 18. Proxy Propagate-and-Match is highly stable across a 6.7× range of ϵ and a 5× range of the minimum number of samples. The ARI stays at or above 0.8 in 28 of 30 settings and never drops below 0.79. Visibility Grouping shows a clear maximum at our default setting. At $\epsilon = 0.2$, the ARI stays at or above 0.90 for up to 7 minimum samples. Moving ϵ in either direction decreases the agreement, which matches the two typical failure modes of DBSCAN. An ϵ that is too small fragments coherent instances, while an ϵ that is too large merges distinct instances. While the choice of ϵ matters for Visibility Grouping, we use the same value for all datasets and experiments without any per-dataset tuning. Together with the small effect of the thresholds γ_{thr} and $\lambda_{\mathcal{J}}$ on the final AP (Table 13), this indicates that our approach does not require careful tuning of these parameters. We release the script for this analysis together with our code.

D.8 Variance over Random Seeds

Some of our improvements on the challenging zero-shot benchmarks in Table 3 are small. To verify that they are not within noise, we train our zero-shot model with three random seeds and report the mean and standard deviation in Table 19. The standard deviation is at most 0.07 AP, and all improvements over VideoCutLER exceed four standard deviations. Note that a low variance is

Table 18: **Clustering Stability.** ARI between the DBSCAN clusterings of perturbed and default parameters on 10 videos from YouTube-VIS 2021. Our default settings are highlighted in bold.

(a) Proxy Propagate-and-Match.

	Min. Samples				
ϵ	2	3	5	7	10
0.030	0.99	0.99	0.99	0.91	0.91
0.050	1.00	1.00	**1.00**	0.92	0.92
0.075	0.99	0.99	0.99	0.92	0.92
0.100	0.98	0.98	0.98	0.91	0.91
0.150	0.92	0.92	0.87	0.80	0.80
0.200	0.88	0.88	0.83	0.83	0.83

(b) Visibility Grouping.

	Min. Samples			
ϵ	3	5	7	10
0.10	0.66	0.65	0.53	0.47
0.15	0.73	0.73	0.68	0.63
0.20	0.90	**1.00**	0.97	0.78
0.25	0.62	0.62	0.62	0.61
0.30	0.53	0.53	0.53	0.57
0.40	0.30	0.30	0.30	0.30

Table 19: **Variance over Random Seeds.** Zero-shot AP of our model over three random seeds (mean±std) on the benchmarks with the smallest margins. *VIPSeg removed from the training dataset mix.

Benchmark	VideoCutLER [35]	S2D (Ours)	Δ
VIPSeg-things	4.8	5.12±0.07*	+0.3
OVIS	2.0	2.7±0.06	+0.7
Cityscapes	0.9	1.39±0.05	+0.5

expected in our setting, since all runs start from the same weight initialization and use the same pseudo-annotations. Hence, the only source of variation is the stochasticity of the training.

E Discussion

We discuss the limitations of our approach in Section 6 and outline possible directions to address them here. Our approach struggles with segmenting smaller objects, as does VideoCutLER. AutoQ-VIS also does not improve AP_S despite improving AP_{50} by 4.4 points (Appendix B), which shows that small objects remain an open problem for unsupervised video instance segmentation. A potential solution could be implementing copy-paste augmentation for videos and scaling down copied objects. Alternatively, the point-grid density could be adapted to the object scale, so that small instances receive more points (see Appendix D.6). Furthermore, our zero-shot model struggles for driving-domain data. We believe adding such domain-specific data to our dataset-mix can help the model improve. While our method relies on deep motion cues to establish temporal coherence, future methods could improve this aspect by using deep temporal features.

F Code Release

We release the code for our approach, including Keymask Discovery, Sparse-To-Dense Distillation, and the clustering stability analysis, along with the model checkpoints for our in-domain and zero-shot models at `https://github.com/leonsick/s2d`.

G Algorithms

To make our method easier to follow and to reproduce, we summarize Keymask Discovery in Algorithm 1 and our dual-stage training in Algorithm 2.

Algorithm 1 Keymask Discovery for a single video.

Require: Single-frame masks $\mathcal{M} = \{\mathbf{m}_i^t\}$ predicted by CutS3D, visibility threshold γ_{thr}, Point-Mask Jaccard threshold $\lambda_{\mathcal{J}}$
Ensure: Sparse keymask annotations $\mathcal{A}$

for each single-frame mask $\mathbf{m}_i^t \in \mathcal{M}$ **do**
  Sample a point grid $\mathbf{P}_i^t = \{p_1^t, \ldots, p_{N_i}^t\}$ inside $\mathbf{m}_i^t$
  Track $\mathbf{P}_i^t$ forward and backward to obtain the trajectories $\mathcal{T}_i$ for all frames
  $\mathbf{v}_i \leftarrow \{\mathbf{1}(\gamma_t > \gamma_{\text{thr}}) \mid t = 1, \ldots, T\}$ ▷ Binary visibility vector
end for
Cluster $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_M\}$ with DBSCAN into visibility clusters $\{\mathcal{C}\}$ and outliers $\mathcal{O}$
Drop all masks in $\mathcal{O}$ ▷ Visibility Grouping
$\mathcal{A} \leftarrow \emptyset$
for each visibility cluster $\mathcal{C}$ **do** ▷ Proxy Propagate-and-Match
  for each mask $\mathbf{m}_i \in \mathcal{C}$, each frame t, and each mask $\mathbf{m}_k^t \in \mathcal{C}$ at frame t **do**
    $\mathbf{M}_{i,k}^t \leftarrow \mathbf{1}\big(\mathcal{J}(i,k,t) > \lambda_{\mathcal{J}}\big)$ ▷ Propagated points $\mathbf{X}_i^t$ vs. mask $\mathbf{m}_k^t$
  end for
  Cluster the matching tracks $\{\mathbf{M}_i\}$ with DBSCAN into matched subgroups
  Add each matched subgroup to $\mathcal{A}$ as one video instance and drop unmatched masks
end for
return $\mathcal{A}$ ▷ The video is discarded if $\mathcal{A} = \emptyset$

Algorithm 2 Dual-stage distillation training. For our zero-shot model, we only perform the first stage.

Require: Training videos $\mathcal{D}$, sparse keymask annotations $\mathcal{A}_{\text{sparse}}$, initial weights θ_{init}, EMA update rate μ
Ensure: Final model weights θ_{T}

$\mathcal{A} \leftarrow \mathcal{A}_{\text{sparse}}$ ▷ Anchoring labelset
$\theta_{\text{S}} \leftarrow \theta_{\text{init}}, \quad \theta_{\text{T}} \leftarrow \theta_{\text{init}}$
for stage $s = 1, 2$ **do** ▷ Stage 2 continues from the stage-1 weights
  for each training iteration **do**
    Sample a snippet x with annotations a from $\mathcal{A}$ (pseudo-dense dataloading)
    $\hat{y}_{\text{T}} \leftarrow$ teacher predictions on x with a score > 0.75 ▷ Stop-gradient
    $\hat{y}_{\text{S}} \leftarrow$ student predictions on x
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{TempDrop}}(\hat{y}_{\text{S}}, a) + \mathcal{L}_{\text{mask}}^{\text{Distill}}(\hat{y}_{\text{S}}, \hat{y}_{\text{T}})$ ▷ Eq. (5)
    Update θ_{S} with $\nabla_{\theta_{\text{S}}}\mathcal{L}$
    $\theta_{\text{T}} \leftarrow \mu\,\theta_{\text{T}} + (1-\mu)\,\theta_{\text{S}}$ ▷ EMA over training iterations
  end for
  if $s = 1$ **then**
    $\mathcal{A} \leftarrow$ teacher predictions on all videos in $\mathcal{D}$ ▷ Dense labelset
  end if
end for
return θ_{T}

H Training Details

All experiments were conducted on university clusters with A6000 or A100 GPUs.
Unsupervised Training. For unsupervised instance segmentation, we use the same hyperparameters for all trainings of the in-domain and zero-shot models. Specifically, we train with a learning rate of 5×10^{-7} for 32,000 iterations using the AdamW[17] optimizer. We update the teacher with an EMA update rate of $\mu = 0.995$. We set the loss weights for the loss on the pseudo-annotations and the distillation loss to the vanilla VideoMask2Former config, i.e., $\lambda_{ce}, \lambda_{dice} = 5.0$ and $\lambda_{class} = 2.0$. Furthermore, as is common in the vanilla model, we train on video snippets with length $T = 2$, but find that increasing the snippet duration does neither hurt nor help performance. For the student-teacher distillation supervision, we keep only teacher predictions with a score > 0.75 to provide high-quality video instance mask supervision.

Table 20: **YouTube-VIS 2021 & Zero-Shot Dataset Statistics.** Comparison of YouTube-VIS 2021 statistics for ground-truth annotations of the training split, as well as our unsupervised sparse and dense pseudo-annotations. Furthermore, we report details for our zero-shot dataset mix.

	YouTube-VIS 2021			Zero-Shot Dataset
	Ground-Truth	Sparse	Dense	Sparse
Video Structure Metrics				
Total Number of Videos	2,985	2,830	2,985	13,171
Instance Quantity Metrics				
Total Number of Video Instances (Tracks)	6,283	7,778	11,122	63,645
Avg. Instances Per Video	2.10	2.75	3.73	4.83
Min/Max/Median Inst. Per Video	1 / 25 / 2	0 / 12 / 2	0 / 42 / 3	0 / 17 / 5
Coverage Metrics				
Total Mask Segments	175,384	128,449	320,732	1,202,868
Total Frames w/ Any Annotation	89,750	71,380	89,772	457,985
Mask Segments to Total Frames Ratio	1.94	1.51	3.55	1.42
Unique Annotated Frame Coverage (%)	99.55%	84.09%	99.57%	80.06%
Mask Size Distribution				
Small	6.31%	0.50%	5.23%	0.17%
Medium	25.24%	14.87%	19.92%	11.58%
Large	68.44%	84.63%	74.85%	88.25%

Table 21: **Real vs. Synthetic Training Data.** Comparison of our zero-shot sparse pseudo-annotations and the synthetic training videos of VideoCutLER [35].

	Ours (Sparse)	VideoCutLER (Synthetic)
Videos	13,171	1,281,167
Instance Tracks	63,645	2,153,170
Mask Segments	1,202,868	4,306,340
Avg. Instances Per Video	4.83	1.68
Avg. Frames Per Video	48.2	3

Supervised Trainings. Our supervised training hyperparameters follow the official implementation for the VideoMask2Former [4]. Specifically, we train the model at a base learning rate of 0.0001 with a batch size of 8 for 16,000 steps. After 10,100 iterations, we drop the learning rate by 10x for the rest of the training.

Semi-Supervised Training. To obtain subsets of labeled video data from the YouTube-VIS 2021 [42] dataset, we randomly sample labeled videos across different percentages with a fixed seed for reproducibility. We use the same sub-sampled videos for all models for a fair comparison. Across all data splits, we fine-tune the models with a base learning rate of 0.0001 and a batch size of 8 on one GPU for 4,000 steps. Especially for the smallest data splits, which, for example, contain only 29 videos at 1%, we find the models start to overfit with the supervised schedule.

I Pseudo-Annotation Statistics

We extract multiple sets of pseudo-annotations as part of our method. First, we construct a sparse annotation set with our Keymask Discovery Algorithm. Second, we perform Sparse-To-Dense Distillation, resulting in model predictions that create a dense annotation set. To investigate how the dataset statistics of our pseudo-annotation sets compare to the ground-truth annotations, we report statistics for the YouTube-VIS 2021[42] training split and our zero-shot dataset mix in Table 20. In the case of YouTube-VIS 2021, our Keymask Discovery algorithm filters out around 5% of videos as noisy. This leads to a lower amount of total mask segmentations and annotated frame coverage. Additionally, fewer small instances are found. After Sparse-To-Dense Distillation, we obtain a dense annotation set, with pseudo-labels for each video. Using the model's predictions, the total number of mask segments increases beyond the ground-truth. This shows our model is not confined by the class

Table 22: **Retained vs. Discarded Videos.** Statistics of the YouTube-VIS 2021 training videos that Keymask Discovery retains or discards.

	Retained	Discarded
Number of Videos	2,830	155
Avg. Instances Per Video	2.75	3.9
Avg. Occlusion Rate ($\gamma_t < 0.3$)	0.19	0.34
Avg. Video Length (Frames)	27.4	28.1

boundaries of the human annotated ground-truth, since it segments also instances not labeled in the dataset. Furthermore, the number of small objects increases from 0.5% to 5.2%.

In addition, we compare our zero-shot pseudo-annotations to the synthetic training data of VideoCutLER [35] in Table 21. VideoCutLER synthesizes videos of 3 frames by shifting copy-pasted masks from static images. Hence, its videos do not contain realistic motion or occlusions. In contrast, our real videos contain 48.2 frames and 4.83 instances on average. Therefore, although our dataset contains around 100 times fewer videos, each video provides substantially richer temporal supervision.

I.1 Discarded Videos

Keymask Discovery discards a video if no temporally coherent instance can be established from its single-frame masks, i.e., if no single-frame mask remains after both Visibility Grouping and Proxy Propagate-and-Match. Hence, the discarding directly follows from the visibility threshold $\gamma_{\text{thr}} = 0.3$, the Point-Mask Jaccard threshold $\lambda_{\mathcal{J}} = 0.5$, and the DBSCAN parameters in Appendix D.7, and does not require an additional threshold. On YouTube-VIS 2021, this is the case for 155 of the 2,985 training videos (5.2%).

We compare the discarded and the retained videos in Table 22. The discarded videos contain more instances and are more often occluded, while their length is similar. Hence, the discarding mostly affects crowded and occluded scenes. We believe that part of the discarded videos also contains objects that are difficult for the image segmenter, so that no coherent mask can be tracked. We therefore consider these cases as upstream failures of the single-frame masks rather than of Keymask Discovery. Importantly, the discarding only affects the construction of the sparse labelset. Since our first-stage model predicts pseudo-annotations for every training video, the dense labelset again covers all 2,985 videos (Table 20). Furthermore, all evaluations are performed on the full validation splits.

J Further Visualizations

J.1 Failure Cases

While our approach works well across a wide range of scenarios in video instance segmentation, it can struggle in some cases. We show examples of failure cases in Figure 6 and group them by their mode and by the stage of our pipeline they originate from.

Missed Instances. If the image segmenter does not produce reliable masks for an instance, no keymask can be discovered for it, and the instance is never supervised. This mostly affects small objects, which are harder to find due to their scale and receive only few points in our point grids. It is also reflected in the lower AP_S. In the first example, our model fails to find the leopard.

Identity Loss. Our model is trained on short video snippets, while the long-term association is only provided by the keymasks during training. Under prolonged occlusions or strong appearance changes, the model can therefore lose track of an instance, such as the biker in the second example. Furthermore, we find that egocentric videos can present a challenge.

Mask Merging. When instances interact closely, the model can merge or drop one of them. In the third example, the mask of the bull disappears after it comes closer to the person. We believe this is caused by the overlapping point tracks of closely interacting instances, which make it harder for Proxy Propagate-and-Match to separate their masks.

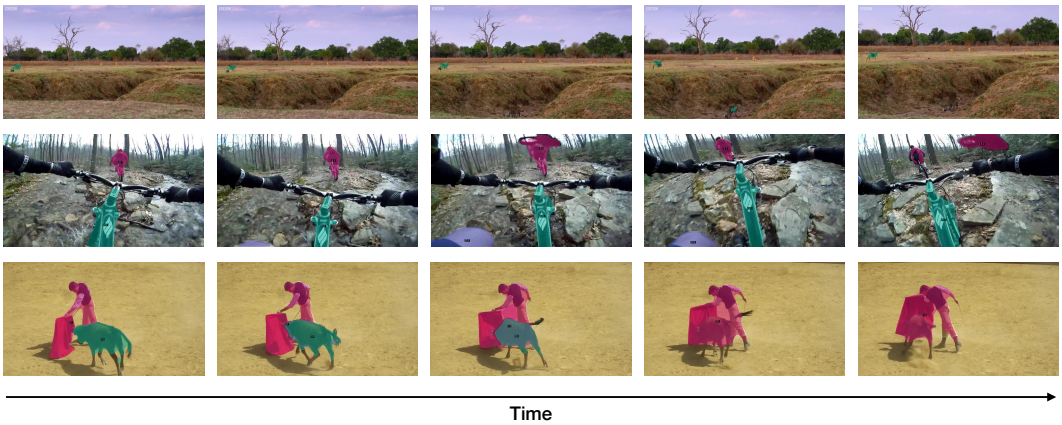

Figure 6: **Failure Cases.** We show failure cases of our method on YouTube-VIS 2021 [42]. In the first example, our model fails to find the leopard. In the second, the tracking of the biker is lost. And in the third, the bull mask disappears after coming closer to the person.

J.2 Image Masks - Keymasks - Dense Masks

Our approach starts by establishing temporal coherence within a temporally unconnected set of single-image masks. Using the Keymask Discovery algorithm, our method filters out masks that do not have matches in the Visibility Grouping and Proxy Propagate-and-Match clustering. We then convert the resulting sparse annotation set to dense labels using Sparse-To-Dense distillation. Figures 7 & 8 display the entire process of pseudo-annotation extraction.

In Figure 7, we show how our Keymask Discovery acts as a data quality filter for single-frame masks, and how Sparse-To-Dense Distillation further refines the masks. In the example depicting a person walking a dog, the initial single-frame predictions contain a temporally incoherent, noisy mask that inaccurately covers multiple instances. Our Keymask Discovery successfully identifies and filters out this unreliable prediction. Subsequently, Sparse-To-Dense Distillation implicitly propagates the remaining high-quality keymasks to the unannotated frames. Furthermore, as highlighted in the detail panels on the right, this distillation process smooths and boundary-refines the initially noisy single-frame predictions, yielding precise and temporally dense video masks.

In the series depicting a skater in Figure 8, some noisy background masks are predicted. However, since they only appear in 2 of 30 video frames, the algorithm classifies them as outliers. In the third series, the unsupervised image instance segmentation model misses the small zebra in frame 1. However, after implicitly propagating the other small-zebra masks to this frame, the zebra becomes densely annotated.



Figure 7: **Qualitative Example of Keymask Discovery & Sparse-To-Dense Distillation.** We show that our Keymask Discovery can effectively filter out noisy masks without temporal coherence, then generate and boundary-refine dense video masks with Sparse-To-Dense Distillation.

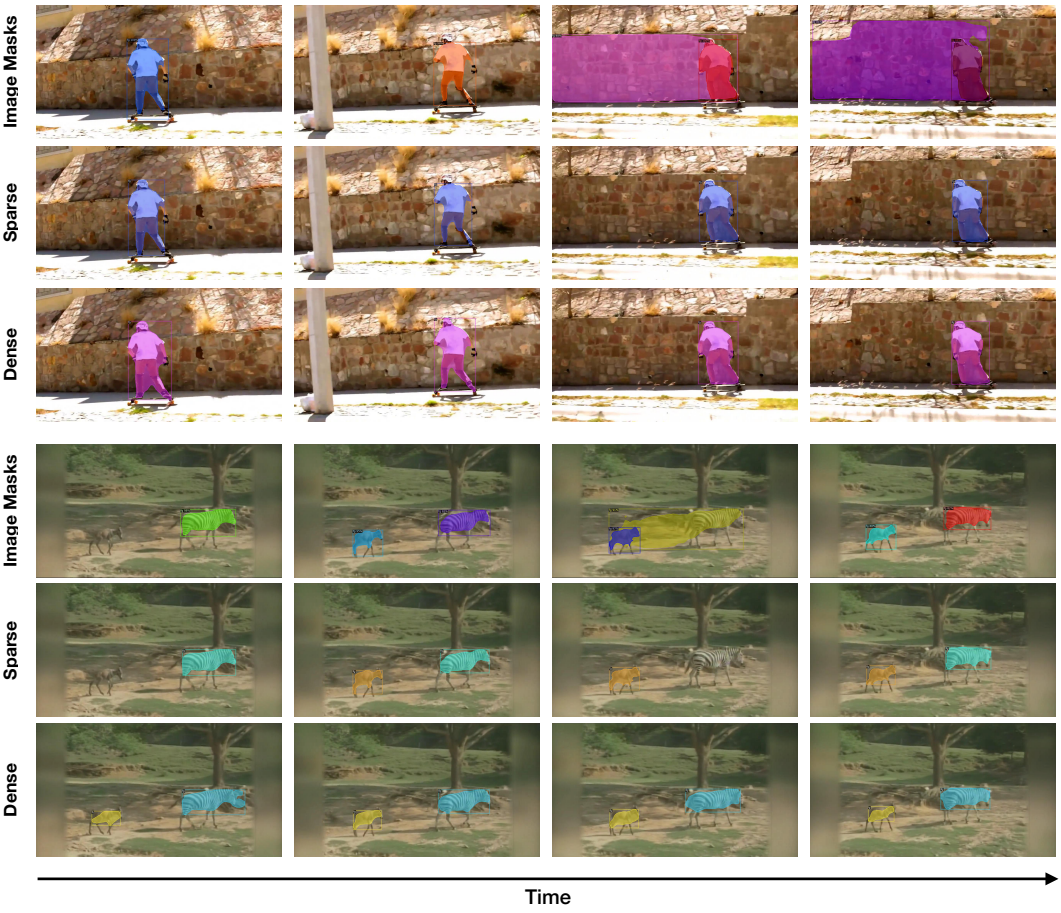

Figure 8: **From Image Masks to Sparse to Dense Annotations.** We showcase how our approach starts from single-frame masks, performs Keymask Discovery to obtain sparse annotations, then propagates them to obtain a dense labelset.

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 the abstract and introduction accurately reflect the paper's contributions, specifically the introduction of the S2D model, Keymask Discovery, and Sparse-To-Dense Distillation, which are supported by the experimental results in Section 4.

   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: Section 6 discusses small objects, prolonged occlusions and crowded scenes, the reliance on off-the-shelf components, the cost of the one-time point tracking, and driving-domain data.

   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.
   - 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: [N/A]

Justification: The paper does not include theoretical results or proofs; it focuses on empirical methods for video instance segmentation.

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: The paper provides implementation details, including the architectures used (CutS3D, VideoMask2Former, CoTracker3), hyperparameter thresholds, and dataset splits in Section 4, with further details available in the supplementary material.

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: We release the code, including Keymask Discovery, Sparse-To-Dense Distillation and the clustering analysis, together with the model checkpoints at `https://github.com/leonsick/s2d` (Appendix F). All datasets we use are publicly available.

   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: Section 4 specifies the training datasets, evaluation benchmarks, protocols, and implementation details necessary to understand the results.

   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: We report the mean and standard deviation over three random seeds for the zero-shot benchmarks with the smallest margins in Appendix D.8. The standard deviation is at most 0.07 AP, and all improvements exceed four standard deviations.

   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: Section 4.3 and Appendix C report the wall-clock GPU hours of every stage on a single A100, the training throughput and memory, and the inference speed.

   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: The research conforms to the NeurIPS Code of Ethics, utilizing standard publicly available datasets for video instance segmentation without raising evident ethical concerns.

   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: S2D is a foundational methodology for unsupervised video instance segmentation and does not introduce new generative capabilities, process sensitive personal data, or pose direct dual-use risks. Positive downstream applications include autonomous driving, AR/VR, and video editing. We do not foresee direct negative societal impacts beyond the general considerations inherent to all video tracking and perception research (e.g., integration into surveillance systems), which fall outside the scope of our specific methodological contributions.

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: The paper does not release models or datasets that have a high risk for misuse, so safeguards are not applicable.

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: The authors properly cite the datasets used and explicitly mention their respective licenses in the references section.

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: [Yes]

Justification: We release our code and model checkpoints at `https://github.com/leonsick/s2d` together with documentation. We do not introduce new datasets.

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: The paper does not involve crowdsourcing or research with human subjects.

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: The research does not involve human subjects, making IRB approval inapplicable.

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 as any important, original, or non-standard components.

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.