Segmenting Moving Objects via an Object-Centric Layered Representation Supplementary Material

Junyu Xie[1] **Weidi Xie**[1,2] **Andrew Zisserman**[1]
[1]Visual Geometry Group, Department of Engineering Science, University of Oxford, UK
[2]Coop. Medianet Innovation Center, Shanghai Jiao Tong University, China
`{jyx,weidi,az}@robots.ox.ac.uk`

Contents

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

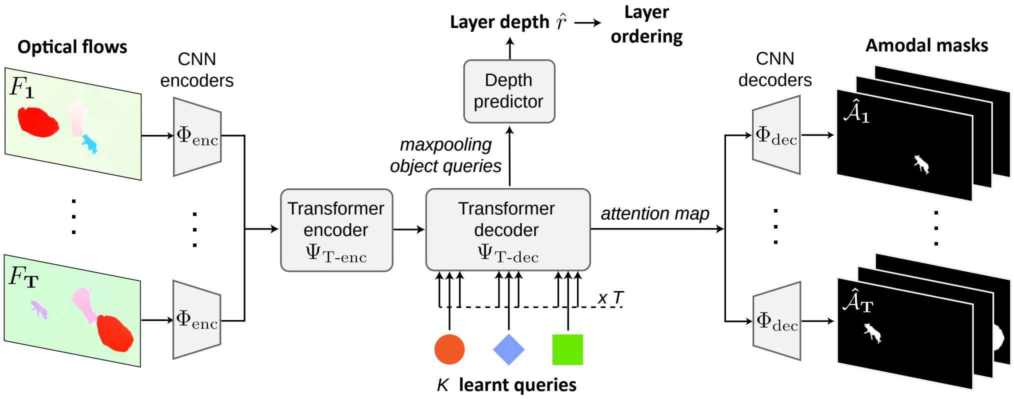


Figure A1: **OCLR Model architecture (copied from the main text).** The model is a U-Net architecture with a Transformer bottleneck. It takes optical flow as input and extracts spatial features by CNN encoders. A transformer encoder jointly processes spatio-temporal features across all frames, followed by a transformer decoder that determines the layer representations. Each learnable query vector in the transformer decoder is associated to one object and used to infer its layer depth. Additionally, the cross-attention maps from the last transformer decoder are extracted and upsampled by CNN decoders to infer amodal segmentation for the moving objects. Note, the skip connections from the CNN encoders to decoders are not shown.

A Architecture and implementation details

In this section, we describe the architecture of the OCLR model in detail.

A.1 CNN backbone

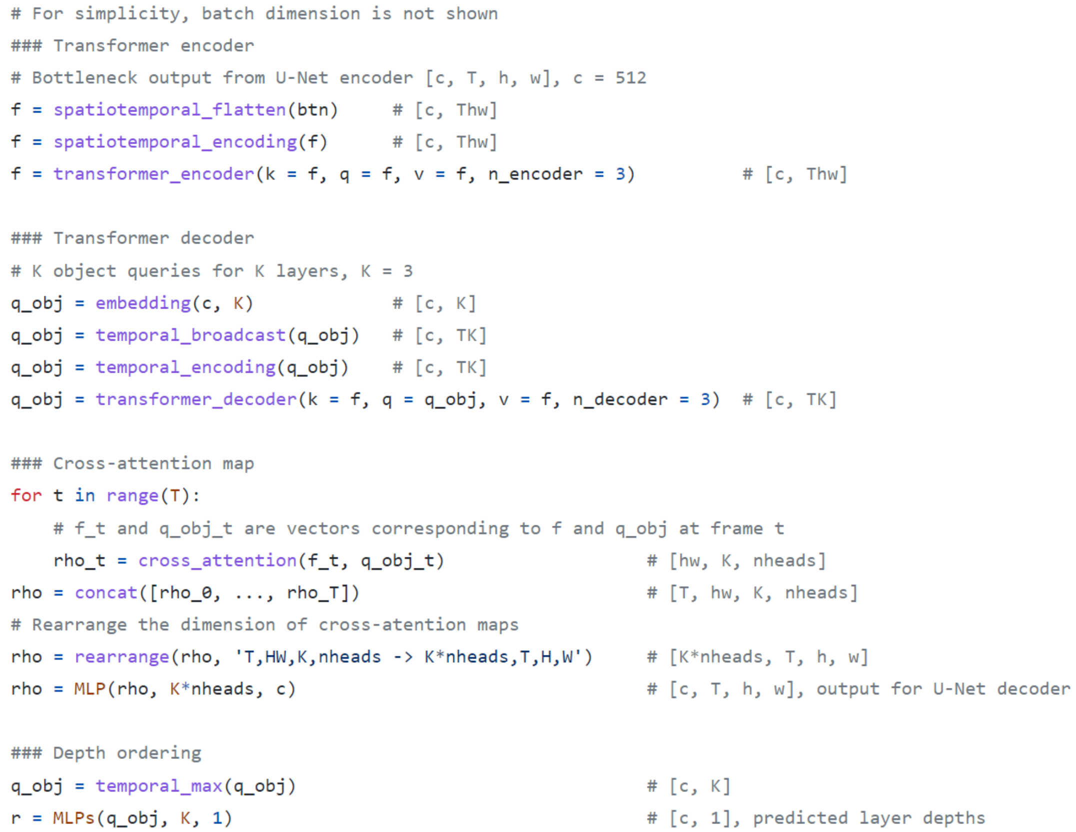
As shown in Figure A1, we adopt a U-Net architecture to extract the visual features for each input frame, and to upsample the feature map output from the transformer-based bottleneck. The details of the U-Net backbone are provided in Table A1.

Table A1: **U-Net architecture.** The U-Net uses a CNN as the backbone. All convolution operations are followed by an Instance Normalisation and a ReLU activation, except for the "de-output" stage in the U-Net decoder.

Encoder			Decoder		
stage	operation	output size	stage	operation	output size
en-input	–	$2 \times 128 \times 224$	de-input	–	$512 \times 8 \times 14$
en-conv1	$(3 \times 3, 32) \times 2$	$32 \times 128 \times 224$	de-conv^{T}4	stride $= 2, 256$	$256 \times 16 \times 28$
en-mp1	maxpool, stride $= 2$	$32 \times 64 \times 112$	de-sc4	en-conv4 skip connect.	$512 \times 16 \times 28$
–	–	–	de-conv4	$(3 \times 3, 256) \times 2$	$256 \times 16 \times 28$
en-conv2	$(3 \times 3, 64) \times 2$	$64 \times 64 \times 112$	de-conv^{T}3	stride $= 2, 128$	$128 \times 32 \times 56$
en-mp2	maxpool, stride $= 2$	$64 \times 32 \times 56$	de-sc3	en-conv3 skip connect.	$256 \times 32 \times 56$
–	–	–	de-conv3	$(3 \times 3, 128) \times 2$	$128 \times 32 \times 56$
en-conv3	$(3 \times 3, 128) \times 2$	$128 \times 32 \times 56$	de-conv^{T}2	stride $= 2, 64$	$64 \times 64 \times 112$
en-mp3	maxpool, stride $= 2$	$128 \times 16 \times 28$	de-sc2	en-conv2 skip connect.	$128 \times 64 \times 112$
–	–	–	de-conv2	$(3 \times 3, 64) \times 2$	$64 \times 64 \times 112$
en-conv4	$(3 \times 3, 256) \times 2$	$256 \times 16 \times 28$	de-conv^{T}1	stride $= 2, 32$	$32 \times 128 \times 224$
en-mp4	maxpool, stride $= 2$	$256 \times 8 \times 14$	de-sc1	en-conv1 skip connect.	$64 \times 128 \times 224$
–	–	–	de-conv1	$(3 \times 3, 32) \times 2$	$32 \times 128 \times 224$
en-bottleneck	$(3 \times 3, 512) \times 2$	$512 \times 8 \times 14$	de-output	$3 \times 3, 3$	$3 \times 128 \times 224$

A.2 Transformer-based bottleneck

In Figure A2, we provide pseudo-code for obtaining cross-attention feature maps and the layer depth order in the Transformer-based bottleneck.



```
# For simplicity, batch dimension is not shown
### Transformer encoder
# Bottleneck output from U-Net encoder [c, T, h, w], c = 512
f = spatiotemporal_flatten(btn)     # [c, Thw]
f = spatiotemporal_encoding(f)      # [c, Thw]
f = transformer_encoder(k = f, q = f, v = f, n_encoder = 3)          # [c, Thw]

### Transformer decoder
# K object queries for K layers, K = 3
q_obj = embedding(c, K)             # [c, K]
q_obj = temporal_broadcast(q_obj)   # [c, TK]
q_obj = temporal_encoding(q_obj)    # [c, TK]
q_obj = transformer_decoder(k = f, q = q_obj, v = f, n_decoder = 3)  # [c, TK]

### Cross-attention map
for t in range(T):
    # f_t and q_obj_t are vectors corresponding to f and q_obj at frame t
    rho_t = cross_attention(f_t, q_obj_t)                   # [hw, K, nheads]
rho = concat([rho_0, ..., rho_T])                           # [T, hw, K, nheads]
# Rearrange the dimension of cross-atention maps
rho = rearrange(rho, 'T,HW,K,nheads -> K*nheads,T,H,W')     # [K*nheads, T, h, w]
rho = MLP(rho, K*nheads, c)                                 # [c, T, h, w], output for U-Net decoder

### Depth ordering
q_obj = temporal_max(q_obj)                                 # [c, K]
r = MLPs(q_obj, K, 1)                                       # [c, 1], predicted layer depths
```

Figure A2: Pseudo-code for Transformer-based bottleneck.

A.3 Training details

The model is trained on the synthetic dataset from about 120k frames. The input batch size is set to 2, and each input sequence has 30 frames (optical flow). During training, we apply a boundary loss weight $\lambda_{\text{bound}} = 0.2$ and an ordering loss weight $\lambda_{\text{order}} = 0.05$. The learning rate is linearly warmed up to 5×10^{-5} during the initial 40k iterations, followed by 50% decays every 80k iterations. All the models are trained on a single NVIDIA Tesla V100 GPU with 32G memory, and the full convergence takes approximately 5 days with 600k iterations.

B Synthetic data generation pipeline

In this section, we elaborate on our synthetic data generation process, with more implementation details on background and foreground objects, dataset distribution, together with several examples.

B.1 Background

Real background video. As shown in Figure B3, we select real-world background videos from copy-right-free sources, followed by frame-wise pre-processing including random cropping, colour jittering, flipping and reflections. Augmented video frames are then sampled as backgrounds, in a

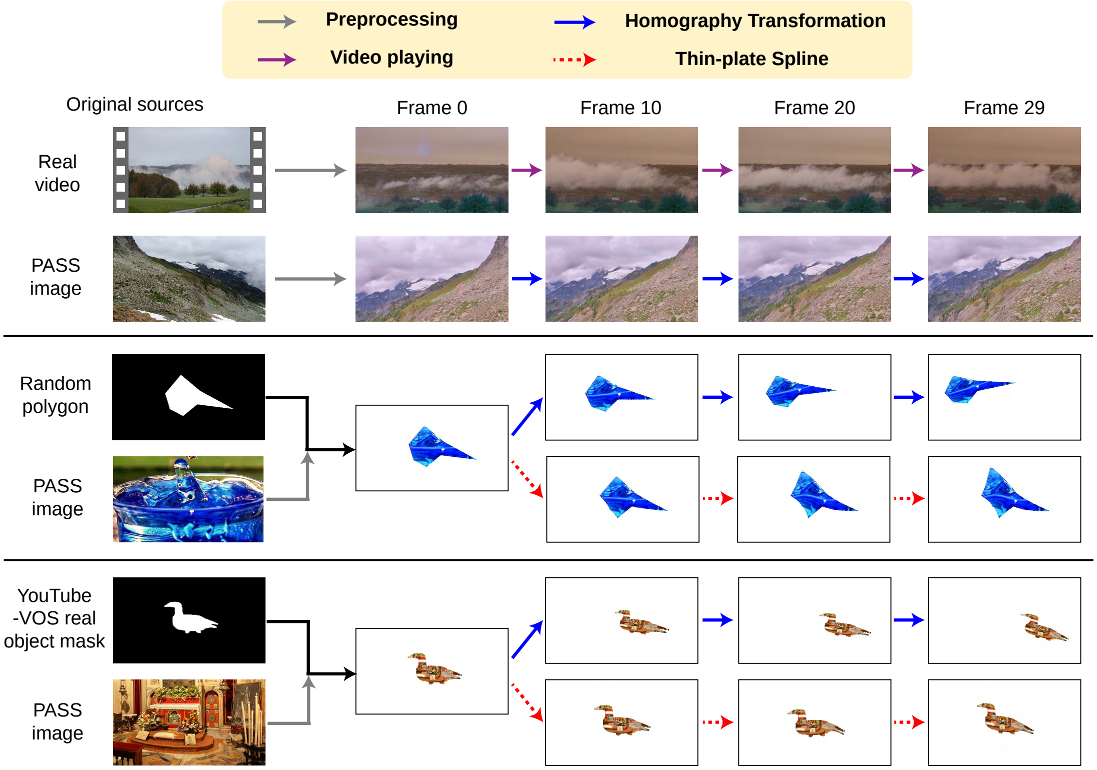


Figure B3: Synthetic data generation.

chronological or reverse chronological order, starting from a random frame with several different frame rates.

Homography-transformed image. Apart from using real videos, we generate background frames by randomly sampling images from the PASS dataset [1]. After a similar pre-processing to that for videos above, the augmented image is set as frame 0. Backgrounds for later frames are propagated by applying homography transformations to simulate the camera motion.

B.2 Foreground objects

Shapes and textures. As shown in Figure B3, the foreground objects are obtained from two sources, one is by generating polygon shapes by connecting 3 to 8 points with random coordinates as vertices, and the other is by taking real object masks from the YouTube-VOS 2018 training set, followed by random resizing and rotation. Both object shapes are textured by pre-processed PASS images to form object sprites.

Foreground object motion simulation. At frame 0, the sprites are initialised at random positions. (In Figure B3, they are initialised in the middle of the frame for demonstration purposes.) We further apply two transformations to simulate object motion, namely homography transformations and thin-plate spline mappings. Homographies include rotations, scaling, perspective distortions and spatial translations. Thin-plate splines transform coordinates according to the motion of control points and include elastic-like stretching. For the examples shown in Figure B3: for the polygon sprite, the top and bottom left vertices are moving control points; while for the real duck-shaped object, the moving point is at the "tail" of the duck.

Stationary object. In Figure B5, the first sequence (from the top) illustrates an example of a stationary object. Between frames t_1 and t_2, the object in layer 2 has no relative motion with respect to the background. As a result, it disappears in the flow field at frame t_1. In practise, stationary objects are achieved by matching object transformations to background motions (therefore only applicable for homography backgrounds).

B.3 Layer composition

In the previous sections, we have explained how objects and backgrounds can be simulated independently. The synthetic video sequence is obtained by a layer composition process that combines all the introduced components via a back-to-front blending process as shown in Figure B4. Examples of composited sequences are given in Figure B5. We also refer an animated layer composition demonstration and more example sequences to the supplementary video.

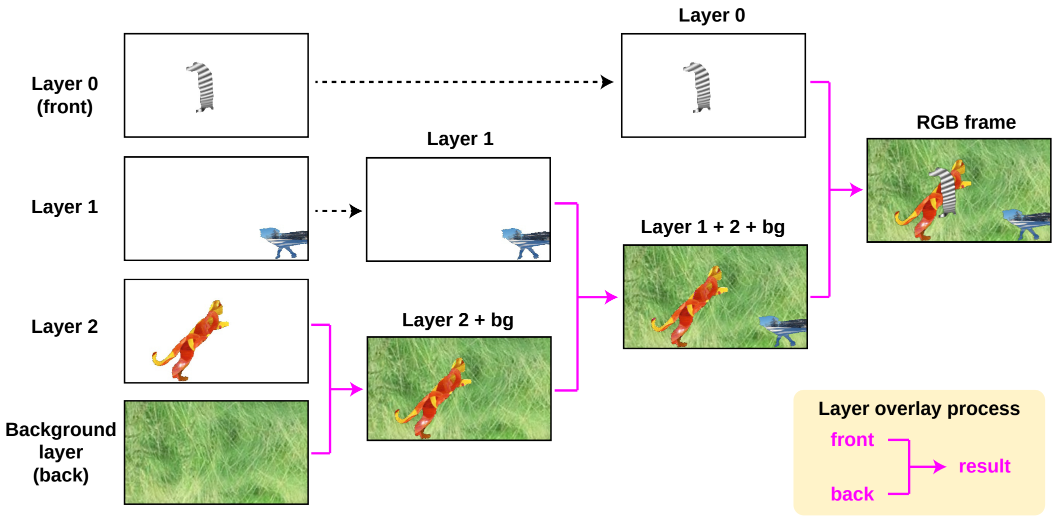


Figure B4: **Layer composition from back to front.** The order from front to back: Layer 0, Layer 1, Layer 2, Background layer.

B.4 Dataset distribution

In Table B2, we provide the details of the video datasets generated by the simulation pipeline. Overall, there are three major pipeline settings: background types, object types and object motions. For all synthetic datasets, an equal proportion of homography transformed backgrounds and real video backgrounds are generated. Similarly, we evenly split sequences with polygon and real object sprites. Regarding object motion simulations, we emphasize the object non-rigidness by applying a combination of homography transformations and thin-plate splines to most sequences (around two times more than sequences with only homography transformations.) Moreover, we randomly select one-third of all sequences and introduce stationary objects for 1 to 5 frames.

On the dataset splits: **Syn-train** is used as the main dataset to train the OCLR flow-based model, consisting of 4608 sequences with over 138k frames. **Syn-single** and **Syn-Val** are mainly used for validation in ablation studies. The former contains 256 single-object-only sequences with a total 7.7k frames, and is treated as a simplified dataset used for tuning synthetic pipeline settings. On the other hand, Syn-Val consists of 384 multi-object sequences (around 12k frames) with $1, 2, 3$ objects in equal proportions. It is mainly used for validating multi-object segmentation performance, with both modal and amodal results reported.

C RGB-based test-time adaptation

In this section, we detail the procedure for applying RGB-based sequences as test-time adaptation. Specifically, given the predicted modal segmentations from our **OCLR (flow-only)** model, we can adopt a similar mask propagation process to that in self-supervised tracking [6, 9, 27]. Overall, the test-time adaptation process proceeds in three stages: first, finetuning a DINO-pretrained vision

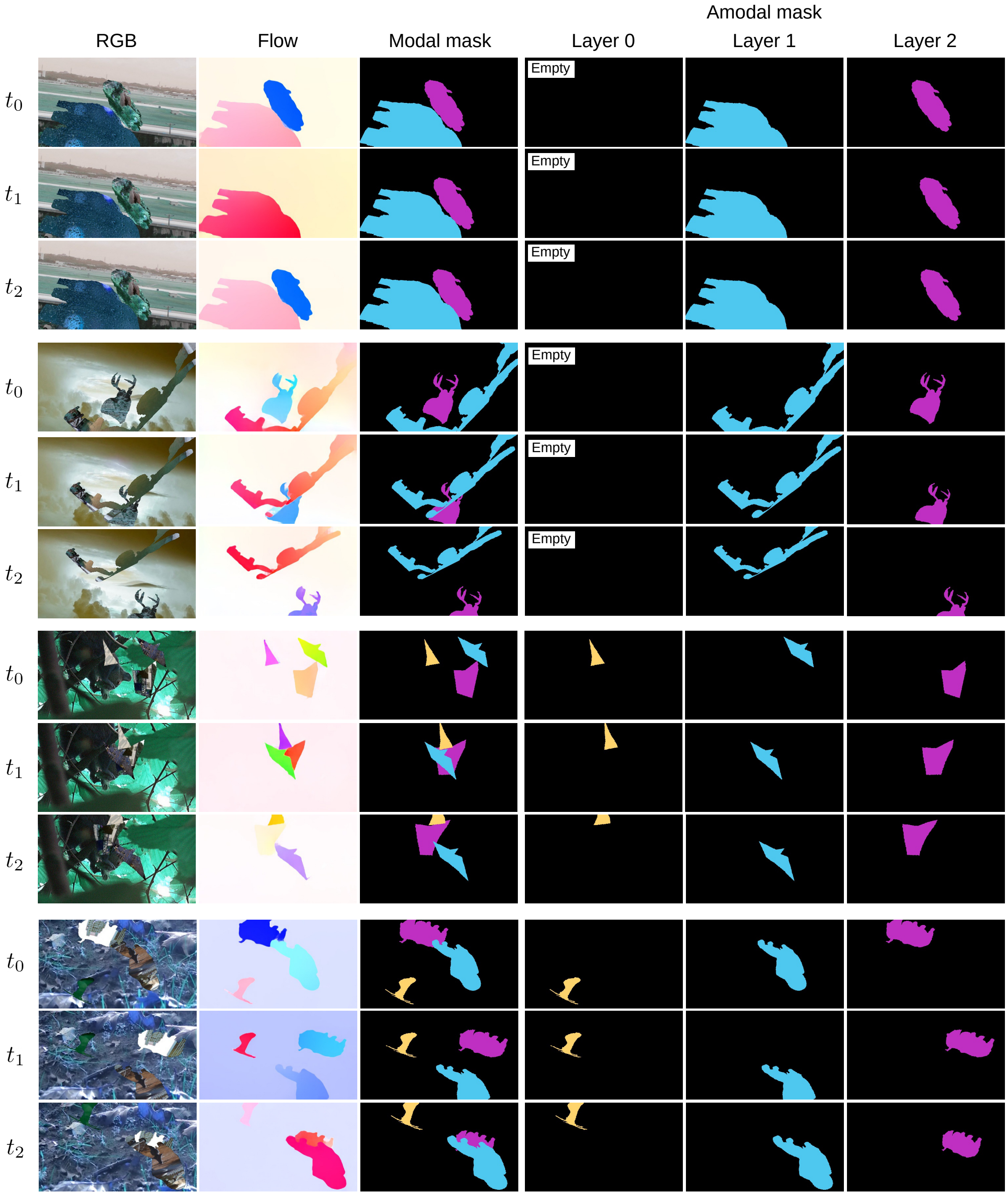


Figure B5: **Examples of synthetic data with ground-truth annotations.** From top to bottom: 2-objs sequence with stationary objects, 2-objs sequence with occlusions, 3-objs sequence with polygon sprites, 3-objs sequence with real object sprites. For demonstration purposes, the first sequence (from the top) shows three **consecutive** frames, whereas for the rest sequences, three frames are selected at fixed time intervals.

Table B2: **Synthetic dataset distribution. homo-bg:** homography transformed image background; **real-bg:** real video background; **homo:** homography transformations only; **homo+tps:** homography transformations and thin-plate spline; **polygon:** polygon sprites; **real-obj:** real object sprites. Unless specified otherwise, all values in the table denote the number of sequences (each with 30 frames).

Pipeline settings			Syn-Train				Syn-Single	Syn-Val			
Background	Obj. motion	Obj. type	1-obj	2-objs	3-objs	Total	1-obj	1-obj	2-objs	3-objs	Total
homo-bg	homo	polygon	96	96	96	288	16	8	8	8	24
homo-bg	homo	real-obj	96	96	96	288	16	8	8	8	24
homo-bg	homo+tps	polygon	288	288	288	864	48	24	24	24	72
homo-bg	homo+tps	real-obj	288	288	288	864	48	24	24	24	72
real-bg	homo	polygon	96	96	96	288	16	8	8	8	24
real-bg	homo	real-obj	96	96	96	288	16	8	8	8	24
real-bg	homo+tps	polygon	288	288	288	864	48	24	24	24	72
real-bg	homo+tps	real-obj	288	288	288	864	48	24	24	24	72
Total number of sequences			1536	1536	1536	4608	256	128	128	128	384
Total number of frames			46.8k	46.8k	46.8k	138.2k	7.7k	3.8k	3.8k	3.8k	11.5k

transformer (ViT) [2]; second, mask selection and propagation; and, third, dynamic refinement during mask propagations.

Finetuning of DINO-pretrained vision transformer. Given a video sequence with T RGB frames, $\mathcal{V}_{\text{RGB}} = \{I_1, \dots, I_T\}$, $I_t \in \mathbb{R}^{480\times 832\times 3}$, we can compute their RGB features with a pre-trained self-supervised vision transformer, namely, DINO-ViT-S/8 [2] (patch sizes 8×8).

$$\mathcal{F}_{\text{RGB}} = \{f_1, \dots, f_T\} = \{\Phi_{\text{DINO}}(I_1), \dots, \Phi_{\text{DINO}}(I_T)\} \tag{1}$$

where $f_t \in \mathbb{R}^{60\times 104\times 384}$.

In order to adapt the DINO model for our purpose, we use the predicted modal masks from our **OCLR (flow-based) model** as noisy annotations to finetune the last two layers of the vision transformer (ViT) by noise contrastive estimation (NCE). In detail, for each object mask in frame t, we define a tri-map with positive P_t, negative N_t and uncertain U_t regions, where uncertain regions are normally a 5-pixel wide exterior to the object mask.

Considering two frames t and $t+n$ ($n \in [1, 4]$), for each pixel in the positive region in one frame, *e.g.* $f_{t,j}$ ($j \in P_t$), we can compute its cosine similarities to all features in the positive region of P_{t+n}, and treat this as the positive score:

$$PS_j = \sum_{k \in P_{t+n}} f_{t,j} \cdot f_{t+n,k} \tag{2}$$

Similarly, the negative sample is defined as

$$NS_j = \sum_{k' \in N_{t+n}} f_{t,j} \cdot f_{t+n,k'} \tag{3}$$

The process is repeated for all pixels in P_t region to give an averaged InfoNCE loss

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{|P_t|} \sum_{j \in P_t} \log \frac{PS_j}{PS_j + NS_j} \tag{4}$$

For each test sequence, we apply this contrastive loss to finetune the DINO-pretrained ViT by an Adam optimizer with a learning rate of 1×10^{-5} that linearly decays over 1k iterations.

Mask selection and propagation. After finetuning the DINO model, we can use it to propagate the predicted segmentation masks from our **OCLR (flow only)** model. Specifically, the predictions for each pixel at frame t can be obtained by computing its nearest neighbour pixels in previous frames, and copying their labels. Note that, this is exactly the same procedure as in [6], we simply denote the propagation process as "Mask-prop" here:

$$\hat{M}_t = \text{Mask-prop}(\hat{M}_{t-1}, \dots, \hat{M}_{t-n}) \tag{5}$$

where $\hat{M}_{t-1}, \ldots, \hat{M}_{t-n}$ refer to the previously obtained results within a temporal window size n.

In contrast to the conventional semi-supervised VOS scenario, where the groundtruth mask at frame 0 is given for propagation, in our work, **no** groundtruth segmentation mask is provided. Instead, we pick one key frame from our OCLR predictions, and then propagate the masks of those objects bi-directionally. The choice of the starting frame is based on the temporal coherence of the optical flow segmentation predictions. In the following, we describe the process for discovering frames with the most coherent predictions temporally:

1. We measure the temporal coherence by propagating our OCLR prediction ($\hat{M}^f_{t-1}$) to the next frame, *i.e.* $\hat{M}_t = \text{Mask-prop}(\hat{M}^f_{t-1})$, and compute the L1 distance between the propagation and our OCLR prediction at frame t, *i.e.* $L_t = |\hat{M}^f_t - \hat{M}_t|$.
2. We take a frame-wise average of L1 loss as $L_{\text{mean}} = \frac{1}{T}\sum_{t=1}^{T} L_t$, and find a set of temporally consistent frames $\{t_s\}$ with $L_{t_s} < L_{\text{mean}}$.
3. Among the frame set $\{t_s\}$, one frame t_k with the lowest L1 loss (*i.e.* $L_{t_k} = \min(\{L_t\})$) is then selected as the starting frame for mask propagations .

The motivation behind this selection process is that, at frames with lower L1 loss, the predicted mask by our OCLR model is more consistent with those in adjacent frames. We then start a bi-directional mask propagation from the key frame t_k with a temporal window size $n = 7$.

Dynamic refinement. Apart from using the prediction of frame t_k as starting frame, we also fuse the other selected frames $\{t_s\}$ into the propagation process for refinement. Specifically, if a new frame t belongs to the set $\{t_s\}$, the propagation result $\hat{M}_t$ would be averaged with the mask predicted by our flow-based model $\hat{M}^f_t$. The resultant mask, *i.e.* $(\hat{M}_t + \hat{M}^f_t)/2$, would then be applied to propagate later frames.

By utilizing masks predicted by our flow-based model, the temporally accumulated drifting issue can be largely alleviated, especially on the object boundaries, as they are usually clearly delineated in our OCLR flow-based model. We refer this process as dynamic refinement, short for **Dyn. Ref.**

Conditional random field (CRF). Following the common practices in video object segmentation tasks, we adopt CRF as the final post-processing step that utilizes RGB information to refine mask predictions

D Datasets

D.1 Dataset details

To evaluate our multi-layer model, we benchmark on multiple popular datasets on both single and multiple object segmentation tasks.

DAVIS2016 [21] consists of 50 high-resolution video sequences (30 for training and 20 for validation) with 3455 frames, the primary moving objects in the scene have been annotated at the pixel level. We report our model performance on the validation set at a 480p resolution.

SegTrackv2 [13] contains 14 sequences and 947 fully-annotated frames, with challenging cases such as occlusions, fast motion and complex shape deformations. Even though multiple objects may be annotated, the community often treats SegTrackv2 as a benchmark for single object segmentation [7, 30], by grouping objects in the foreground.

FBMS-59 [19] contains 59 sequences with a total of 720 pixel-level annotations provided every 20 frames. Similar to SegTrackv2, some multi-object sequences are relabelled for single object segmentation evaluations.

Moving Camouflaged Animals (MoCA) [12] focuses on segmenting camouflaged animals moving in natural scenes. It contains 141 high-resolution video sequences annotated by tight bounding boxes for every 5th frame. Following [29], we adopt a filtered MoCA dataset by excluding videos with predominantly no locomotion, resulting in 88 video sequences and 4803 frames

DAVIS2017 [22] extends DAVIS2016 dataset by introducing additional videos with multi-object contents, resulting in a total of 150 sequences with over 10k pixel-level annotations. In particular, DAVIS2017 is a common benchmark for semi-supervised video object segmentation (VOS) tasks (i.e. with ground-truth first-frame annotation provided). In this work, the model performance is not directly evaluated on DAVIS2017, where there are sequences with common motion objects that are indistinguishable based on purely flow information. Instead, we adopt a curated DAVIS2017-motion dataset, with details provided in the next section.

Overall, we adopt a Hungarian matching process to associate layer predictions with the ground-truth annotations in DAVIS2016, MoCA, DAVIS2017-motion and our synthetic datasets. For evaluations on SegTrackv2 and FBMS-59, we instead group all layers together as a single foreground object.

D.2 DAVIS2017-motion curation

As objects in common motion cannot be distinguished purely from motion cues, we re-annotate the original DAVIS2017 dataset by grouping jointly moving objects to form a new DAVIS2017-motion dataset for benchmarking motion-based object segmentation. In this section, we first provide a definition of common motion, followed by some curation details.

Common motion. In this work, the concept of object common motion is defined based on two necessary criteria: *First*, objects are or appear to be spatially connected throughout the whole sequence. More specifically, pixel-level masks for different objects are next to each other for all frames. *Second*, objects must share the same motion trends. A quick judgement can be made by observing if there is a noticeable flow discontinuity at the common boundary.

Curation details. As shown in Figure D6, based on the rules defined above, we group the annotations of jointly moving objects as a whole, resulting in the new DAVIS2017-motion dataset. Note that, as DAVIS2017-motion is adopted mainly for validation purposes, we re-annotate only the validation sequences in the original DAVIS2017. The full list of curated sequences includes: *bmx-trees, horsejump-high, india, kite-surf, lab-coat, mbike-trick, motocross-jump, paragliding-launch, scooter-black, shooting, soapbox*. We will release the curated dataset, together with the re-annotation codes for further research.

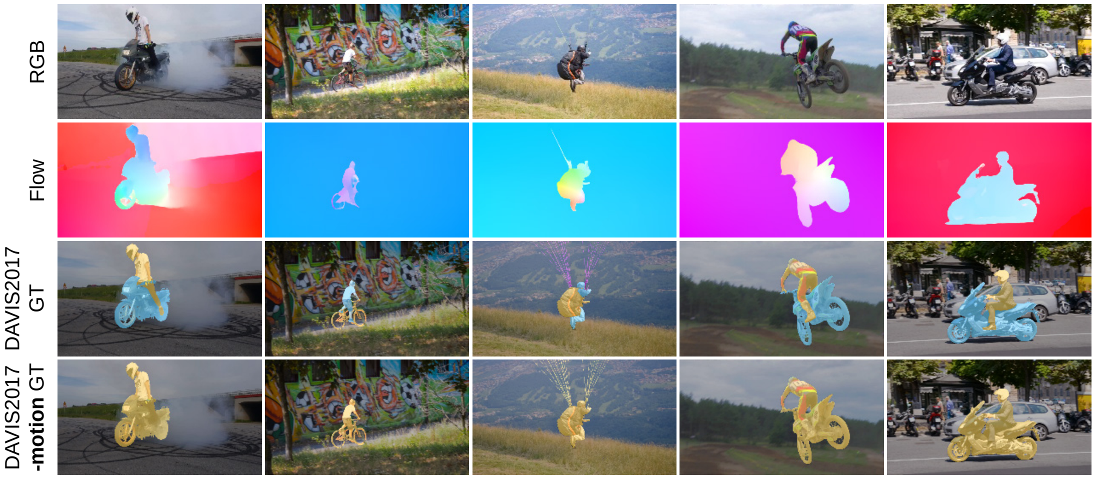


Figure D6: **Curated DAVIS2017-motion dataset.** Note that, in the original DAVIS2017 dataset (3rd row), the objects have been annotated based on their semantic categories, however, in our curated DAVIS2017-motion dataset, we join those objects with common motion as a whole, for example, in the 1st, 4th, 5th columns, the person and motorbike are originally annotated as two different classes, while in our curated dataset, they are labelled as a whole, as they follow the same motion.

E Ablation study

In this section, we present more details on our ablation studies, with a single parameter varied every time. Firstly, we conduct experiments on the pipeline for data simulation, to get the best Sim2Real performance on single moving object segmentation; Secondly, we extend the simulation procedure to multiple objects, and investigate more training details, for example, normalisation, loss function, number of frames, *etc*; Thirdly, we repeatedly train several motion segmentation models with the same optimal hyperparameters to verify our model stability. Fourthly, we demonstrate the scalability of our method by training on an increasing number of synthetic frames. Finally, we show the effectiveness of our model by comparing it with other supervised models under the same supervision.

E.1 Synthetic dataset

We simulate the videos with only a single object and conduct experiments to understand several key choices in the pipeline, for example, motion representation and background motion. As shown in Table E3, while comparing Ours-A and Ours-B, we observe that training on RAFT flows is beneficial, resulting in higher performance on all datasets. We conjecture that training on RAFT flows leads to a narrower Sim2Real domain gap, as flows in these test sets also come from RAFT. In addition, introducing real videos as backgrounds brings further performance gain, as shown in Ours-B and Ours-C. Therefore, we treat the settings in Ours-C as the default for our synthetic data pipeline.

Table E3: Settings for training in synthetic dataset pipeline. Note that, all the flows on test set are computed with RAFT.

	Synthetic Training Settings		$\mathcal{J}$ (Mean) ↑		
Experiment	Flow	Background motion	DAVIS 2016	SegTrackv2	Syn-Single
Ours-A	GT	Homography	67.4	52.3	78.0
Ours-B	RAFT	Homography	68.6	58.5	84.7
Ours-C	RAFT	Homography + Real video	**72.0**	**62.3**	**91.4**

E.2 Training setting

Instance normalisation. As shown in Table E4, while evaluating the multi-layer models ($N = 3$) on single object (DAVIS2016) and multi-object (DAVIS2017-motion, Syn-multi) segmentation datasets, the model with instance normalizations (Ours-G) consistently outperforms their counterparts (Ours-D), showing the importance of using instance normalization.

Hungarian matching. In our proposed architecture, the learnable object queries are permutation invariant, that is to say, each can correspond to different layers, and we use Hungarian matching to assign the layer to each object query. Here, we ablate the Hungarian matching procedure, and instead force each query to predict a layer at fixed order, *e.g.* 1st query is always associated with the front layer. As indicated by the result, such design significantly degrades the performance, as shown in Table E4 Ours-E.

Training by amodal segmentations. In Ours-F, we train the model only by modal masks in the synthetic dataset. The performance has dropped significantly, suggesting that explicit amodal supervision helps the network to learn object permanence within layers.

Boundary loss. While comparing between Ours-G and Ours-I, we observe a performance boost from applying boundary loss. This validates our assumption that focusing on object boundaries can help the model to learn the information on object shapes and layer orders from optical flows.

Number of frames. Lastly, we compare our default model (Ours-G) with a variant that takes in a reduced number of input frames (Ours-H, $T = 15$), not surprisingly, a longer temporal input tends to also give slightly higher overall performance.

Optical flow methods. Since our OCLR model takes optical flows as the only input, the resultant performance is largely influenced by the quality of optical flow estimations, as demonstrated in

Table E4: **Settings for training parameters. IN**: Instance Normalisation; **HM**: Hungarian matching; **Amodal**: Training on amodal mask (vs. modal mask); λ_{bound}: weight on boundary loss; T: number of input frames. Syn-Val ($\mathcal{M}|\mathcal{A}$) corresponds to modal and amodal results on synthetic dataset. Ours-G denotes a default baseline setting compared to others.

	Training settings					$\mathcal{J}$ (Mean) ↑		
Model	IN	HM	Amodal	λ_{bound}	T	DAVIS2016	DAVIS2017-motion	Syn-Val ($\mathcal{M}|\mathcal{A}$)
Ours-D	✗	✓	✓	0.2	30	67.6	48.7	83.5 | 83.0
Ours-E	✓	✗	✓	0.2	30	67.4	44.2	70.6 | 71.0
Ours-F	✓	✓	✗	0.2	30	69.2	50.5	81.1 | 76.9
Ours-G	✓	✓	✓	0.2	30	**72.1**	**54.5**	**85.6 | 84.7**
Ours-H	✓	✓	✓	0.2	15	71.3	53.5	82.8 | 83.0
Ours-I	✓	✓	✓	0	30	71.5	54.1	80.9 | 81.6

Table E5. The highest performance of Our-G verifies our choice of the RAFT method for optical flow predictions.

Table E5: Choice of optical flow methods.

		$\mathcal{J}$ (Mean) ↑	
Model	Optical flow	DAVIS2016	DAVIS2017-motion
Ours-G	RAFT [25]	**72.1**	**54.5**
Ours-J	ARFlow [15]	54.6	39.5
Ours-K	MaskFlownet [31]	66.0	49.0

E.3 Repeating experiments

In Table E6, we demonstrate multi-object segmentation results by re-training the model several times based on the default setting (Ours-G in Table E4). As can be observed, there are minimal performance differences between repeated experiments (within 1%), which validates the reliability of our results.

Table E6: Repeating experiments on multi-object segmentation tasks.

	$\mathcal{J}$ (Mean) ↑		
Experiment	DAVIS2016	DAVIS2017-motion	Syn-Val ($\mathcal{M}|\mathcal{A}$)
1	72.1	54.5	85.6 | 84.7
2	72.0	54.1	85.2 | 84.3
3	72.8	54.3	84.9 | 84.5
4	72.1	54.0	85.9 | 85.2
Mean	72.3	54.2	85.4 | 84.7
Std.	±0.3	±0.2	±0.4 | ±0.3

E.4 Scalability of model performance

Table E7 demonstrates how our model performance scales with the amount of synthetic data. As more synthetic frames are introduced during training, there is a clear increase in both DAVIS2016 and DAVIS2017-motion performance. Moreover, Ours-O and Ours-P correspond to models that are directly trained on real datasets (*i.e.* DAVIS2017-motion). Limited to the number of manual annotations, these models achieve lower performance than synthetic-supervised counterparts, particularly in multi-object segmentation.

Table E7: **Model performance scales with the size of training sets.** "with aug." stands for augmentations applied on the input flows including random cropping, rotations, jittering, dropouts, *etc.*

			$\mathcal{J}$ (Mean) ↑	
Model	Training set	No. of training frames	DAVIS2016	DAVIS2017-motion
Ours-L	Syn-train subset	43.2k	69.6	51.2
Ours-M	Syn-train subset	69.1k	70.2	51.5
Ours-N	Syn-train subset	115.2k	71.2	53.7
Ours-G	Syn-train	138.2k	**72.1**	**54.5**
Ours-O	DAVIS2017-motion	4.2k	66.7	42.8
Ours-P	DAVIS2017-motion	4.2k (with aug.)	69.4	45.3

E.5 Effectiveness of the OCLR model

Table E8 provides a comparison between our OCLR model and two other supervised models on multi-object segmentation. All methods take optical flow as the only input and are supervised by either the real dataset (DAVIS2017-motion) or our synthetic data (Syn-train). Motion Grouping (sup.) represents a supervised version of Motion Grouping [29], while the standard Mask R-CNN [5] model adopted follows the default settings in the original paper, with a ResNet-50-FPN backbone trained from scratch. In this case, the Mask R-CNN model takes optical flows as the only input, and is therefore referred to as Mask R-CNN (flow-only).

Table E8: **Comparison of different models under real or synthetic supervision**. All models take optical flow at the only input. During inference, the 1st frame GT is not available (*i.e.*, unsupervised VOS). In column Sup. (supervision), "Syn." and "Real" represent synthetic data supervision and real-data supervision, respectively.

				$\mathcal{J}$ (Mean) ↑
Model	Sup.	Training set	No. of training frames	DAVIS2017-motion
Motion Grouping (sup.)	Real	Syn-train	4.2k	32.7
Mask R-CNN (flow-only)	Real	Syn-train	4.2k	40.3
OCLR (flow-only)	Real	Syn-train	4.2k	42.8
Motion Grouping (sup.)	Syn.	DAVIS2017-motion	138.2k	44.9
Mask R-CNN (flow-only)	Syn.	DAVIS2017-motion	138.2k	50.4
OCLR (flow-only)	Syn.	DAVIS2017-motion	138.2k	**54.5**

From Table E8, it can be observed that: (i) Our OCLR outperforms both benchmark models under both supervision scenarios, particularly under synthetic supervision. When visualising the qualitative results, we found that Mask R-CNN demonstrates inferior performance in comparison to OCLR, particularly when there is noisy optical flow, temporally stationary objects, or heavy object deformations; while in contrast, OCLR is designed with the ability to infer amodal masks, and thus to handle situations with occlusion happening; (ii) Motion Grouping originally designed for self-supervision does not perform well when direct supervision is applied; (iii) Compared to supervision provided by a limited amount of real data (4.2k frames), scalable synthetic supervision (138.2k frames) leads to general performance improvements.

E.6 Settings for test-time adaptation

Mask propagation. According to Table E9, results obtained by RGB-based mask propagation from a key frame (Ours-Q) surpass the performance of the flow-only model (Ours-G). This validates the benefits of introducing RGB information during test time.

Dynamic refinement. In Ours-R, the dynamic refinement is applied to the propagation process by introducing segmentation outputs from our OCLR model. Consequently, this additional object mask information on average leads to a 2% improvement in performance.

Finetuning of vision transformer By comparing Ours-S with Ours-R, we notice a further performance boost by per-sequence finetuning of the DINO-pretrained vision transformer, and the settings in Ours-S contribute to the highest test-time adaptation result.

Conditional random field (CRF) Finally, by comparing between Ours-T with Ours-S, applying CRF as the post-processing step further improves the overall performance.

Table E9: Settings for test-time adaptation.

	Training Settings				$\mathcal{J}$ (Mean) ↑	
Model	Mask Prop.	Dyn. Ref.	Fine-tuning	CRF	DAVIS2016	DAVIS2017-motion
Ours-G	✗	✗	✗	✗	72.1	54.5
Ours-Q	✓	✗	✗	✗	76.5	60.9
Ours-R	✓	✓	✗	✗	77.4	63.5
Ours-S	✓	✓	✓	✗	78.9	63.9
Ours-T	✓	✓	✓	✓	**80.9**	**65.2**

F Quantitative results

Table F10 summarizes performance on single object video segmentation benchmarks. For camouflaged object detection on MoCA, Table F11 provides a more detailed quantitative comparison across different approaches. The multiple object video segmentation results on DAVIS2017-motion are shown in Table F12.

Table F10: Quantitative comparison single object video segmentation benchmarks. Results from more methods are quoted compared to the table in the main text. "HA" stands for human annotations. In column Sup. (supervision), "None", "Syn.", "Real" represent self-supervision, synthetic data supervision, and real data supervision, respectively. ***Bold*** represents the state-of-the-art performance (excluding our test-time adaptation results, which are labelled as *blue* instead).

	Training Settings				$\mathcal{J}$ (Mean) ↑		
Model	HA	Sup.	RGB	Flow	DAVIS2016	SegTrackv2	FBMS-59
SAGE [28]	✗	None	✓	✓	42.6	57.6	61.2
NLC [4]	✗	None	✓	✓	55.1	67.2	51.5
CUT [8]	✗	None	✓	✓	55.2	54.3	57.2
FTS [20]	✗	None	✓	✓	55.8	47.8	47.7
CIS [30]	✗	None	✓	✓	59.2	45.6	36.8
CIS (w. post-process.) [30]	✗	None	✓	✓	71.5	62.0	63.5
Motion Grouping [29]	✗	None	✗	✓	68.3	58.6	53.1
SIMO [11]	✗	Syn.	✗	✓	67.8	62.0	–
OCLR (flow-only)	✗	Syn.	✗	✓	**72.1**	**67.6**	**65.4**
OCLR (test. adap.)	✗	Syn.	✓	✓	80.9	72.3	69.8
SFL [3]	✓	Real	✓	✓	67.4	–	–
FSEG [7]	✓	Real	✓	✓	70.7	**61.4**	68.4
LVO [26]	✓	Real	✓	✓	75.9	57.3	65.1
ARP [24]	✓	Real	✓	✓	76.2	57.2	59.8
COSNet [16]	✓	Real	✓	✗	80.5	49.7	75.6
MATNet [32]	✓	Real	✓	✓	82.4	50.4	**76.1**
3DC-Seg [17]	✓	Real	✓	✓	84.3	–	–
D^2Conv3D [23]	✓	Real	✓	✗	**85.5**	–	–

G Qualitative results

Figure G7, G8 and G9 illustrate our model predictions on Syn-Val, SegTrackv2 and FBMS-59, respectively. We also demonstrates qualitative results on synthetic and real datasets in the supplementary video.

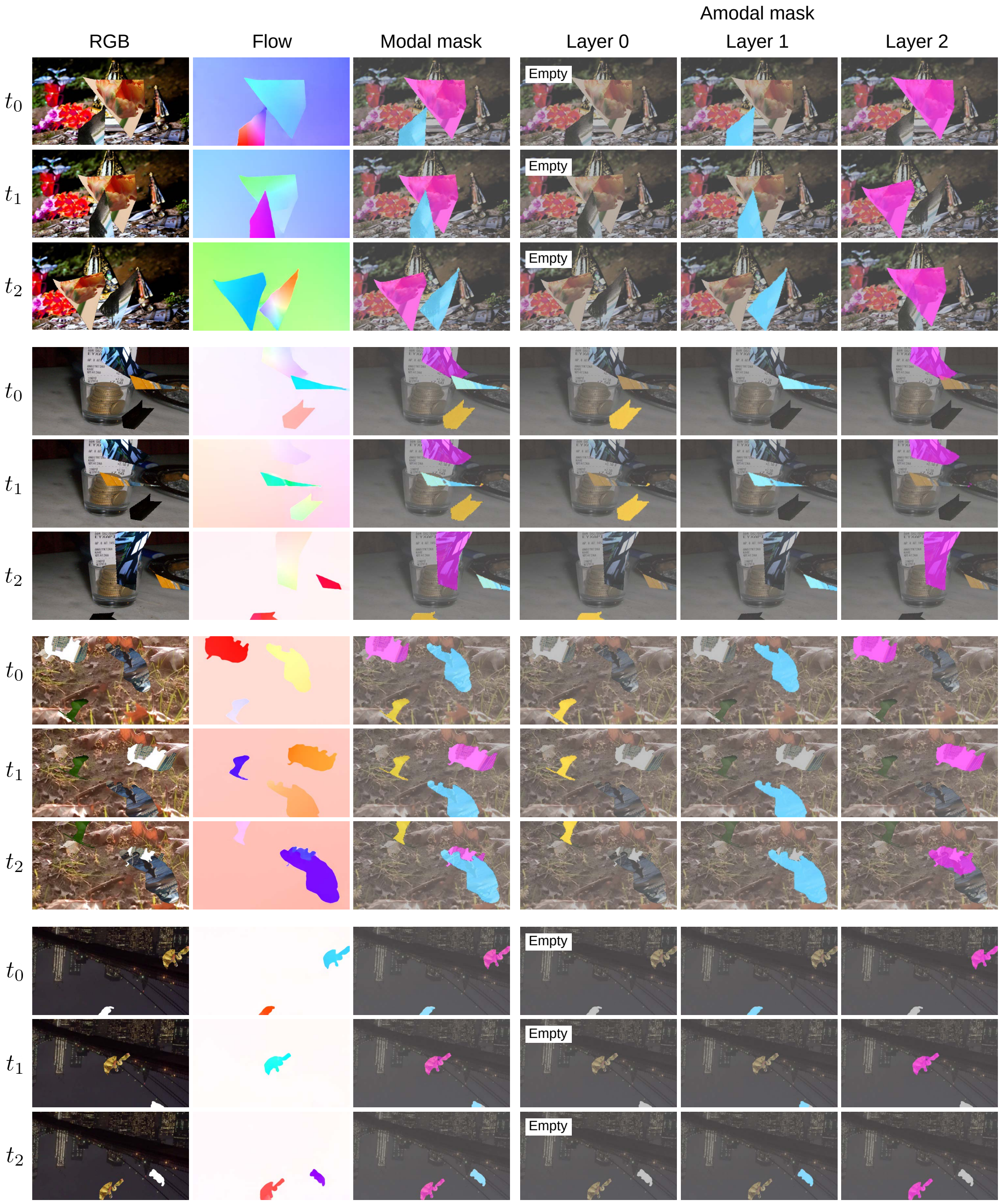


Figure G7: Qualitative results on our synthetic dataset Syn-Val. Both modal and layer-wise amodal predictions are shown.

Table F11: Quantitative comparison of camouflaged object detection on MoCA. “HA” stands for human annotations. In column Sup. (supervision), “None”, “Syn.”, “Real” represent self-supervision, synthetic data supervision, and real data supervision, respectively. ***Bold*** represents the state-of-the-art performance.

	Training settings					Detection Success Rate $\uparrow$					
Model	HA	Sup.	RGB	Flow	$\mathcal{J}\uparrow$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$	mean
CIS [30]	✗	None	✓	✓	49.4	0.556	0.463	0.329	0.176	0.030	0.311
CIS (w. post-process.)	✗	None	✓	✓	54.1	0.631	0.542	0.399	0.210	0.033	0.363
Motion Grouping [29]	✗	None	✗	✓	63.4	0.742	0.654	0.524	0.351	0.147	0.484
SIMO [11]	✗	Syn.	✗	✓	68.6	0.772	0.717	0.623	0.464	0.255	0.566
Ours (flow-only)	✗	Syn.	✗	✓	**70.9**	**0.795**	**0.743**	**0.658**	**0.508**	**0.289**	**0.599**
Ours (test. adap.)	✗	Syn.	✓	✓	67.5	0.789	0.717	0.615	0.445	0.230	0.559
COD [12]	✓	Real	✗	✓	44.9	0.414	0.330	0.235	0.140	0.059	0.236
COD (two-stream)	✓	Real	✓	✓	55.3	0.602	0.523	0.413	0.267	0.088	0.379
COSNet [16]	✓	Real	✓	✗	50.7	0.588	0.534	0.457	0.337	0.167	0.417
MATNet [32]	✓	Real	✓	✓	64.2	0.712	0.670	0.599	0.492	0.246	0.544

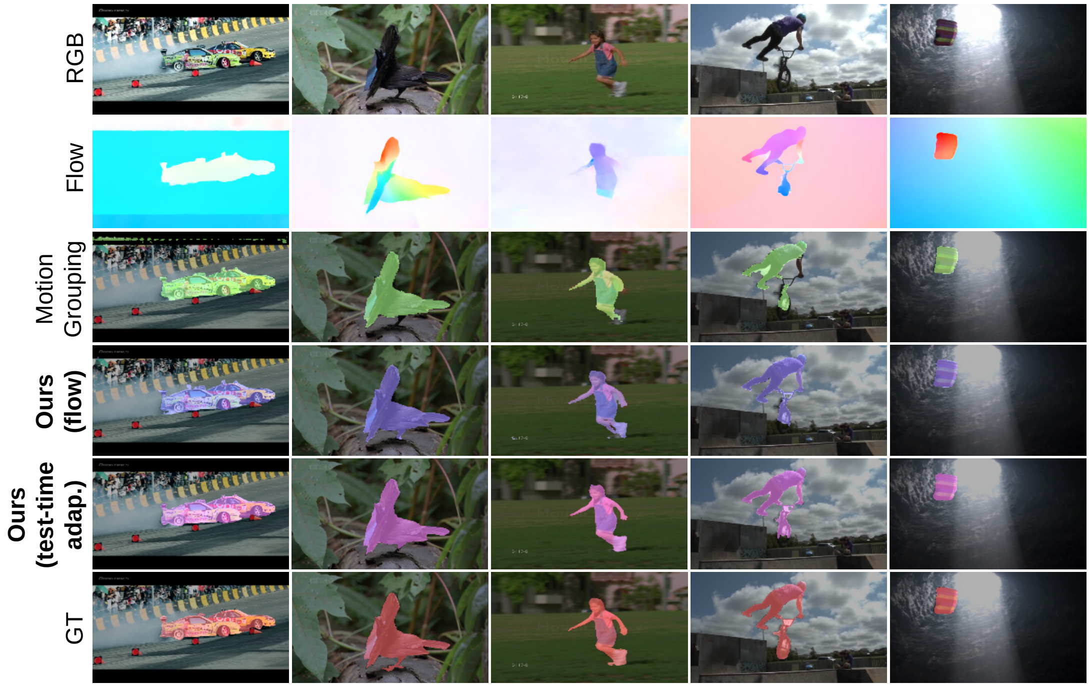


Figure G8: Qualitative results of single object video segmentation on SegTrackv2.

Table F12: Quantitative comparison of multi-object video segmentation on DAVIS2017-motion. Results for semi-supervised methods are obtained by re-running source codes on the DAVIS2017-motion. Note that, the compared methods here are trained without using any human annotations during training, in particular, Motion Grouping (sup.), Mask R-CNN (flow-only) and OCLR models are supervised by only synthetic data, and other approaches are trained with self-supervision. ***Bold*** represents the state-of-the-art performance (excluding our test-time adaptation results, which are labelled as *blue* instead).

	Training settings			DAVIS2017-motion performance		
Model	1st-frame-GT	RGB	Flow	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}$ (Mean) $\uparrow$	$\mathcal{F}$ (Mean) $\uparrow$
Motion Grouping [29]	✗	✗	✓	35.8	38.4	33.2
Motion Grouping (sup.)	✗	✗	✓	39.5	44.9	34.2
Mask R-CNN (flow-only)	✗	✗	✓	50.3	50.4	50.2
OCLR (flow-only)	✗	✗	✓	**55.1**	**54.5**	**55.7**
OCLR (test. adap.)	✗	✓	✓	64.4	65.2	63.6
CorrFlow [10]	✓	✓	✗	54.0	54.2	53.7
UVC [14]	✓	✓	✗	65.5	66.2	64.7
MAST [9]	✓	✓	✗	70.9	71.0	70.8
CRW [6]	✓	✓	✗	73.4	72.9	74.1
MAMP [18]	✓	✓	✓	75.8	76.4	75.2
DINO [2]	✓	✓	✗	**78.7**	**77.7**	**79.6**

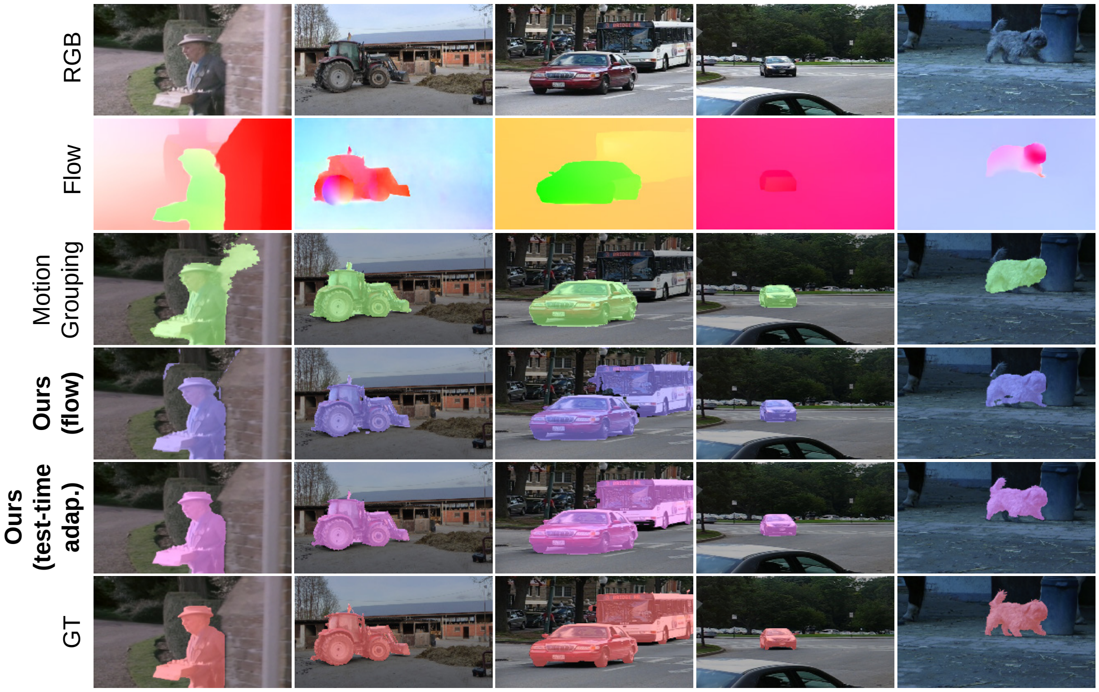


Figure G9: Qualitative results of single object video segmentation on FBMS.

H Discussions on ethic guideline

H.1 Potential negative societal impacts

In our work, the main source of information is from our synthetic dataset, which is generated based on textures from the PASS dataset (without any personally identifiable information) and only shapes from the YouTubeVOS2018 training sets. These procedures ensure that almost no human information is introduced to our network. Furthermore, we advocate a segmentation method mainly utilizing motion cues, represented by optical flows. The textures in optical flow are only related to the motion fields and object shapes, without any real-world semantic information. By filtering out possible human-related information, we have made largely eliminated any social negative impacts on the environment, human rights, economics, personal security, *etc.*

H.2 General ethical conduct guideline

Does the work contain any personally identifiable information or sensitive personally identifiable information? No. The dataset we benchmark and curate, *e.g.* DAVIS, SegTrackv2, FBMS and MoCA, do not contain sensitive personally identifiable information. For the generation of our synthetic dataset, we obtain all RGB texture information from the PASS dataset, which does not contain any personally identifiable information. For shape information, we use randomly generated polygon or only real-object shapes from the YouTubeVOS2018 training sets, without any textural details, which greatly eliminates possible sensitive information. If there is any personally identifiable information later found in any of the above datasets, we will make immediate corresponding changes.

Does the work contain information that could be deduced about individuals that they have not consented to share? No. As described above, all information provided to train and test our model does not contain sensitive personally identifiable information. Therefore, no personal privacy information could be deduced.

Does the work encode, contain, or potentially exacerbate bias against people of a certain gender, race, sexuality, or who have other protected characteristics? No. As explained above, we tried to eliminate personally identifiable information in our synthetic dataset, which in turn minimises possible human-related biases.

Does the work contain human subject experimentation and whether it has been reviewed and approved by a relevant oversight board? No. We did not include human subject experimentation in our work.

Have the work been discredited by the creators? No. All datasets we used and curated are under the CC-BY license, and we make necessary references to the original source.

Consent to use or share the data. Explain whether you have asked the data owner's permission to use or share data and what the outcome was. As explained above, all adopted datasets follow the CC-BY license, and we have made the necessary references.

Do you have domain specific considerations when working with high-risk groups. No. We minimise personally identifiable information in our synthetic dataset, therefore not raising any domain-specific issues regarding high-risk groups.

Have you filtered offensive content. Yes. As mentioned above, we filter out offensive content in our synthetic dataset.

Can you guarantee compliance to GDPR? Yes.

References

[1] Yuki M Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. Pass: An imagenet replacement for self-supervised pretraining without humans. In *NeurIPS Track on Datasets and Benchmarks*, 2021.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[3] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, 2017.

[4] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[6] Allan Jabri, Andrew Owens, and Alexei A Efros. Space-time correspondence as a contrastive random walk. In *NeurIPS*, 2020.

[7] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *CVPR*, 2017.

[8] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, 2015.
[9] Zihang Lai, Erika Lu, and Weidi Xie. Mast: A memory-augmented self-supervised tracker. In *CVPR*, 2020.
[10] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. In *BMVC*, 2019.
[11] Hala Lamdouar, Weidi Xie, and Andrew Zisserman. Segmenting invisible moving objects. In *BMVC*, 2021.
[12] Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. Betrayed by motion: Camouflaged object discovery via motion segmentation. In *ACCV*, 2020.
[13] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.
[14] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019.
[15] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *CVPR*, 2020.
[16] Xiankai Lu, Wenguan Wang, Chao Ma, Jianbing Shen, Ling Shao, and Fatih Porikli. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *CVPR*, 2019.
[17] Sabarinath Mahadevan, Ali Athar, Aljoša Ošep, Sebastian Hennen, Laura Leal-Taixé, and Bastian Leibe. Making a case for 3d convolutions for object segmentation in videos. In *BMVC*, 2020.
[18] Bo Miao, Mohammed Bennamoun, Yongsheng Gao, and Ajmal Mian. Self-supervised video object segmentation by motion-aware mask propagation. In *ICME*, 2022.
[19] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *TPAMI*, 2014.
[20] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.
[21] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 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.
[23] Christian Schmidt, Ali Athar, Sabarinath Mahadevan, and Bastian Leibe. D^2conv3d: Dynamic dilated convolutions for object segmentation in videos. In *WACV*, 2022.
[24] Hongmei Song, Wenguan Wang, Sanyuan Zhao, Jianbing Shen, and Kin-Man Lam. Pyramid dilated deeper convlstm for video salient object detection. In *ECCV*, 2018.
[25] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.
[26] Pavel Tokmakov, Cordelia Schmid, and Karteek Alahari. Learning to segment moving objects. *IJCV*, 2019.
[27] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *ECCV*, 2018.
[28] Wenguan Wang, Jianbing Shen, Ruigang Yang, and Fatih Porikli. Saliency-aware video object segmentation. *TPAMI*, 2018.
[29] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *ICCV*, 2021.
[30] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *CVPR*, 2019.
[31] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *CVPR*, 2020.
[32] Tianfei Zhou, Shunzhou Wang, Yi Zhou, Yazhou Yao, Jianwu Li, and Ling Shao. Motion-attentive transition for zero-shot video object segmentation. In *AAAI*, 2020.