Objects do not disappear:
Video object detection by single-frame object location anticipation

Xin Liu[1]   Fatemeh Karimi Nejadasl[2]   Jan C. van Gemert[1]   Olaf Booij[1]   Silvia L. Pintea[1]

Computer Vision Lab, Delft University of Technology[1]

Institute for Biodiversity and Ecosystem Dynamics, University of Amsterdam[2]

Abstract

Objects in videos are typically characterized by continuous smooth motion. We exploit continuous smooth motion in three ways. 1) Improved accuracy by using object motion as an additional source of supervision, which we obtain by anticipating object locations from a static keyframe. 2) Improved efficiency by only doing the expensive feature computations on a small subset of all frames. Because neighboring video frames are often redundant, we only compute features for a single static keyframe and predict object locations in subsequent frames. 3) Reduced annotation cost, where we only annotate the keyframe and use smooth pseudo-motion between keyframes. We demonstrate computational efficiency, annotation efficiency, and improved mean average precision compared to the state-of-the-art on four datasets: ImageNet VID, EPIC KITCHENS-55, YouTube-BoundingBoxes and Waymo Open dataset. Our source code is available at https://github.com/L-KID/Video-object-detection-by-location-anticipation.
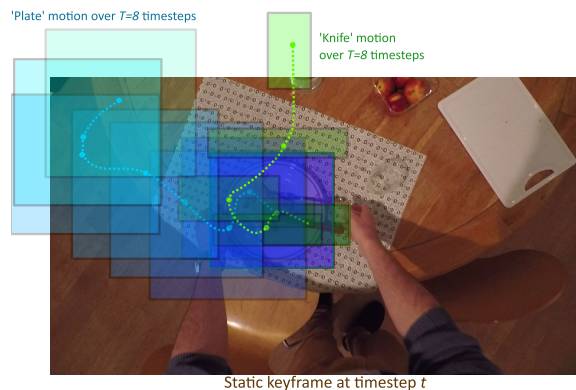
Figure 1. Anticipating future object locations from a static keyframe is *efficient*. We only do the expensive feature extraction on a small subset of keyframes, while still accessing bounding-box locations for all video frames. Moreover, exploiting motion cues as additional supervision *improves* object detection. By sampling a static keyframe at time t and anticipating the object locations over the next T timesteps, we incorporate temporal consistency and smoothness of object motion.

1. Introduction

Humans assume object permanence: blink your eyes, and the world is still there. Similarly, video frames are redundant, and missing some frames when watching a movie does not drastically change the scene. Actually, for the parts that *did* change, if these parts changed coherently, they might hint at a sense of objectness, as hypothesized by the Gestalt law of common fate.

As illustrated in Fig. 1, here we explore these observations in the context of video object detection in three ways: 1) Improved accuracy by exploiting an additional source of supervision: the coherent motion from the law of common fate; by predicting object motion from a static image. 2) Improved efficiency by exploiting redundancy to reduce computational cost by only processing sampled keyframes and predict object motion for the missing frames in-between. 3) Reduced annotation cost by only annotating sampled keyframes. Thus, we improve accuracy and save computation and annotation time by simply skipping the feature computation and/or the annotation for a large majority of the frames.

We make the following contributions: (i) A video object detection method that samples static keyframes and predicts object motion for unseen future frames. (ii) Computational efficiency, as our method only extracts features for sampled static keyframes; (iii) Data efficiency, as we use sparse annotations only at the sampled keyframes, hallucinating motion in-between these sparse annotations. (iv) Our extensive experimental results on ImageNet-VID [58], EPIC KITCHENS-55 [11], YouTube-BoundingBoxes [56] and Waymo Open dataset [15] show that our approach improves accuracy over the state-of-the-art methods, while being faster at both training and inference time.

2. Related work

Video object detection. Several methods do temporal modeling by post-processing still-image object detectors such as post-processing Faster-RCNN [27, 40] or post-processing Mask RCNN [4, 44]. Alternatives include recurrent blocks [49, 65] or optical flow [80, 78]. This can be further extended with instance and pixel-level calibrations over time [70], or using a space-time lattice [6]. More recently, the field has advanced by aggregating temporal information: either by defining detection correlations as a graph in SELSA [72], or by using global and local temporal pooling in MEGA [7]. MEGA is further extended by considering all pairwise frames in TF-Blender [9], while HVR-Net [24] integrates inter-video and intra-video object relations. IFF-Net[38] uses a feature flow estimating module to indicate the feature displacement. LWDN [36] does not simply aggregate features, but aligns the features between keyframes by adopting a memory mechanism. In contrast, we avoid computationally demanding optical flow and recurrent blocks, and we do not aggregate neighboring frames, or use temporal heuristics to post-process still-image detections. Instead, we anticipate future object locations over time from a single static keyframe, placing temporal prediction at the heart of our model.

Single image motion prediction versus tracking. Object tracking aims to predict object location in a video. A recent overview of object tracking is given in [8]. In classical methods, the bounding box around the object to track is given a priori [61], but can also be estimated by an object detector [3, 18, 32, 64]. The object location in the next frame can be estimated with siamese networks [3], or object detections at every frame are linked into tracks by motion regression [18], or by trajectory forecasting [12]. Our method is inspired by forecasting in tracking, yet we are different from tracking because we predict the object location in future frames from a single static input image, without actually using the image features of the future frames.

Anticipating motion from a static single image. A single static frame is rich enough to allow predicting of future appearance [33, 46, 52, 67], actions [1, 20, 50, 60], and motion [19, 54, 68]. Such motion predictions, in turn, can then be used for predicting pedestrians' behaviors [55, 76] or other road agents [23]. Moreover, motion prediction can be used as a self-supervision cue [17, 25, 26, 43, 71] to improve feature learning. Inspired by these works, we use location anticipation as an additional source of supervision to improve the accuracy of the still-image detector, while we also exploit it for efficiency as it allows us to only compute features on a subset of still-images, avoiding expensive feature computations on all frames.

Efficiency in video. Because videos typically sample several frames per second (FPS) it is important to have efficient video analysis methods. Successful prior work proposes network architecture adaptations to reduce computations [16, 41, 45, 47, 53, 81]. For video object detection, it is efficient to adapt the detector online [59] or to transfer an image detector to video [39]. Alternatively, computations can be reduced by focusing only on specific video regions [37, 63, 75]. Similarly, we also focus on efficiency in video object detection. We are efficient by predicting object trajectories a few frames ahead and therefore saving computations by not processing those frames.

Sparse annotations in video. Training a model on sparse video annotations can be done by iteratively updating the model in a boosting fashion [2], or propagating groups of pixels through time [35]. More recent work generates dense object masks in videos from sparse bounding boxes [74]. Rather than focusing on improving the model accuracy on sparsely annotated videos, prior work has analyzed what is the accuracy vs. annotation effort trade-off [51]. Specifically, for object locations in video, annotations can be generated through a combination of tracking, frame selection and active learning [10, 34, 42, 66]. Inspired by these works we also explore a variant of our model that allows sparse annotations. We require bounding-box annotations on the static keyframes, and we experimentally show that we can hallucinate the motion between keyframes, removing the need to annotate all frames.

3. Anticipating object locations

Object detection backbone. We illustrate our method in Fig. 2. We start from a standard static image object detection backbone. We input static frames uniformly sampled from the video with a time step T; we call such static frames "keyframes". For each keyframe at time t, the object detector gives a set of N proposal bounding boxes: $B_t = \{B_t^i\}_{i=1}^N$ where $B_t^i = (x_t^i, y_t^i, w_t^i, h_t^i)$, and (x_t^i, y_t^i) are the top-left corner and (w_t^i, h_t^i) are the width and height of the bounding box. Each B_t^i corresponds to a possible object and has an associated class c_t^i: $\{(B_t^i, c_t^i)\}_{i=1}^N$.

Trajectory subnetwork. Starting from the keyframe detection bounding boxes B_t, we anticipate object trajectories – defined as the future bounding box locations of an object over the following T frames. The trajectory subnetwork is highlighted in green in Fig. 2. For each keyframe indexed by t, we define a batch of length T and we input into the trajectory network three types of inputs: (a) a vector of time indices of the future trajectories relative to the keyframe index t batched from $[1, .., T]$; (b) the set of bounding boxes detected at the keyframe $B_t = \{B_t^i\}_{i=1}^N$ repeated over all T time steps: $[\{B_t^i\}_{i=1}^N] \times T$; and (c) the static keyframe featuremaps extracted via the network mapping $f(\cdot)$ from the the bounding boxes $\{f_t^i | f_t^i = f(B_t^i)\}_{i=1}^N$, also broadcast for
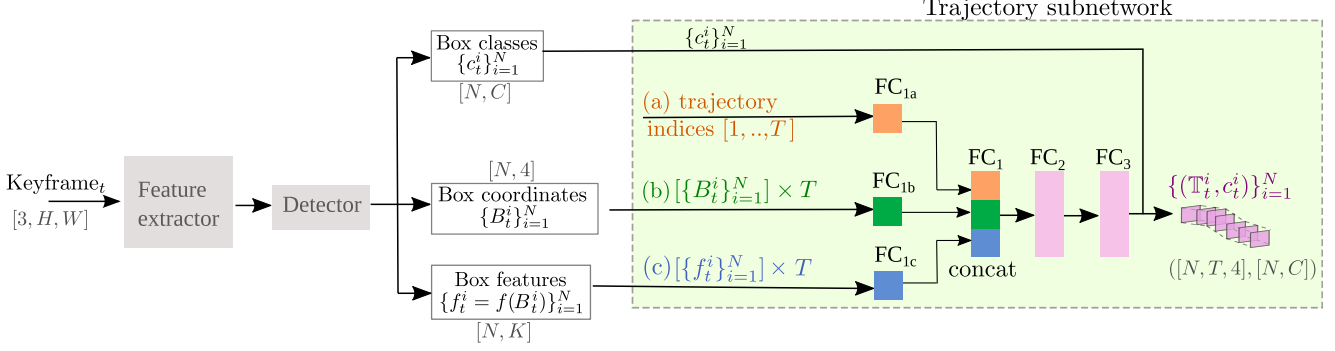
Figure 2. **Overview:** The network starts from a single sampled video keyframe at timestep t. A feature extractor backbone is followed by an object detector. The object detector outputs for the keyframe at time t a set of N bounding boxes $\{B_t^i\}_{i=1}^N$, together with their associated class probabilities $\{c_t^i\}_{i=1}^N$ and keyframe features extracted from each box area $\{f_t^i\}_{i=1}^N$. Next, our trajectory subnetwork (highlighted in green) takes as input: (a) a batch of trajectory indices $[1, .., T]$, where T is the trajectory length; (b) the keyframe bounding boxes repeated T times and batched, $[\{B_t^i\}_{i=1}^N] \times T$; and (c) the box features also repeated T times $[\{f_t^i\}_{i=1}^N] \times T$. These are projected through linear layers of equal sizes (FC$_{1a}$, FC$_{1b}$, FC$_{1c}$) and the output is concatenated and passed through two additional linear layers. The trajectory network predicts a set of N trajectories $\{\mathbb{T}_t^i\}_{i=1}^N$ of length T and their associated classes $\{c_t^i\}_{i=1}^N$.

each of the T time steps: $[\{f_t^i\}_t^N] \times T$. Note that by inputting into the trajectory subnetwork the future trajectory-frame indices in (a), we add temporal ordering information: each future box prediction has an associated frame index. We map all three inputs: boxes, features and time indices, through fully connected (FC) layers of equal sizes. We concatenate the output features, and pass them through two additional fully connected layers.

The output of the trajectory subnetwork is a list of trajectories, one trajectory for each of the N detected objects. Each trajectory starts at the keyframe indexed by t and extends up to frame $t+T$. Concretely, our trajectory predictions are: $\{\mathbb{T}_t^i | \mathbb{T}_t^i = (B_t^i, B_{t+1}^i, .., B_{t+T}^i)\}_{i=1}^N$, where we also add the keyframe bounding box B_t to the trajectory.

3.1. Associating trajectories to ground truth

To optimize the trajectories we need an associated object class for each trajectory. Each object trajectory $\mathbb{T}_t^i$, indexed by i, starts with a keyframe bounding box B_t^i, which has a corresponding object class c_t^i. Each trajectory corresponds to one object, thus we let each trajectory inherit the class of its starting keyframe bounding box, yielding: $\{(\mathbb{T}_t^i, c_t^i)\}_{i=1}^N$.

Moreover, we also need to associate ground truth boxes B^* with all predicted boxes along each object trajectory $\mathbb{T}_t^i = (B_t^i, B_{t+1}^i, .., B_{t+T}^i)$. Following the standard procedure [30, 57] we rank the predicted boxes $B_{t+l}, l \in \{0, .., T\}$ based on their overlap with the ground truth boxes B_{t+l}^* at each frame $t+l$. We associate each of the N predicted boxes with the best matching IoU score of the ground truth box.

3.2. Trajectory loss

Bag of boxes loss. We want to optimize for each object indexed by i its associated trajectory, starting at keyframe t:
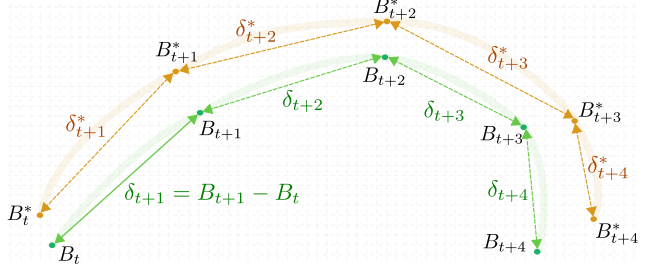

Figure 3. Object trajectories are piecewise continuous: *i.e.* an object cannot disappear between two neighboring frames t and $t+1$. We incorporate this continuity by noting that the loss between a box B_{t+l} along the trajectory and its associated ground truth B_{t+l}^* is defined as the sum of pairwise offsets of neighboring boxes, δ_{t+l}, starting from the keyframe box B_t. The orange line is the true trajectory and the green line is the predicted trajectory. (Here, for simplicity we discard the width and height of the bounding boxes and only show (x, y) coordinates and depict $l \in \{1, .., 4\}$.)

$\mathbb{T}_t^i$. For readability, we ignore the index i from here on. The standard loss L_{bag} for performing box regression, considers the trajectory boxes as an unordered bag and computes a smooth L_1 loss $\mathcal{L}_1(\cdot)$ [21, 57], for each predicted box B_{t+l}, to its associated ground truth box B_{t+l}^*:

$$L_{\text{bag}}(B^*, \{B_t, .., B_{t+T}\}) = \sum_{l=0}^T \mathcal{L}_1(B_{t+l}^* - B_{t+l}). \quad (1)$$

Trajectory cumulative loss. The downside of Eq. (1) is that it treats each prediction B_{t+l} as if it were independent of its neighboring predictions along the trajectory, B_{t+l-1} and B_{t+l+1}. Therefore, there is no temporal ordering enforced in the L_{bag} loss. Not enforcing the ordering of the

predictions along the trajectory could lead to discontinuous trajectories. We want to enforce smoothness in the predictions over time: objects cannot disappear or appear at random locations, between neighboring frames. Specifically, the trajectory is piecewise continuous: for an object to move from a location B_{t+1} to a location B_{t+4} it has to travel through the intermediate locations B_{t+2} and B_{t+3}, as illustrated in Fig. 3.

To add this insight, we define a loss that constrains the pairwise offsets along the trajectory from frame t up to every frame $t+l$: $\delta_{t+k} = (B_{t+k} - B_{t+k-1})$, $k \in \{1,..,l\}$, to add up to the offset from the ground truth at the frame $t+l$ to the keyframe prediction $(B_{t+l}^* - B_t)$. Concretely:

$$L_{\sum}(B^*, \overleftrightarrow{\mathbb{T}_t}) = \sum_{l=0}^{T} \mathcal{L}_1 \left((B_{t+l}^* - B_t) - \sum_{k=1}^{l} \delta_{t+k} \right), \quad (2)$$

where we redefine the trajectory to predict δ_{t+l} values describing pairwise offsets instead of bounding boxes: $\overleftrightarrow{\mathbb{T}_t} = (\delta_{t+1}, .., \delta_{t+T})$. Additionally, we ignore the calculation of the coordinates, if the ground truth bounding boxes B_{t+l}^* are not valid in Eq. (2). Note, that if we would predict bounding boxes B_{t+l} in the trajectory network, instead of offsets between pairs of bounding boxes δ_{t+l}, Eq. (2) would reduce to Eq. (1) and the temporal ordering would not be enforced (See the derivation in the supplementary material).

In Eq. (2) the inner loop over pairwise offsets δ_{t+k} accumulates the errors in the predictions over time from frame $t+1$ to frame $t+l$. To make sure the errors do not accumulate along the trajectory, and the change from frame to frame is smooth, we add another loss $L_{\text{bag}(\delta)}$ constraining the pairwise offsets δ_{t+l} at every timestep $t+l$ to map back to ground truth offsets: $\delta_{t+l}^* = (B_{t+l}^* - B_{t+l-1}^*)$:

$$L_{\text{bag}(\delta)}(B^*, \overleftrightarrow{\mathbb{T}_t}) = \sum_{l=1}^{T} \mathcal{L}_1 \left(\delta_{t+l}^* - \delta_{t+l} \right). \quad (3)$$

Our final loss L_{traj} is then a combination of the two losses where the cumulative trajectory loss $L_{\sum}$ enforces piecewise continuity in the predictions and the bag of offsets loss $L_{\text{bag}(\delta)}$ discourages errors from accumulating along the trajectory and makes the trajectory smooth:

$$L_{\text{traj}} = L_{\sum} + L_{\text{bag}(\delta)}. \quad (4)$$

We investigate the effect of each loss term in the experimental section, together with the effect of predicting offsets δ_{t+l} instead of box coordinates B_{t+l} in the trajectory network.

Sparse annotation loss. When there are no annotations in-between keyframes then we cannot optimize the anticipated trajectory. Since the task in the sparse annotation cases is object detection on annotated keyframes, we can hypothesize that the precise true location of an object along

a trajectory B_{t+l}^*, is not essential, as long as the trajectory is piecewise continuous and smooth, and the starting and ending points of the trajectory are known. Therefore, we can rewrite our losses in Eq. (4) to a sparsely-annotated variant $L_{\text{traj}}^{(\text{sa})}$ by changing the way in which we define the box-supervision. Explicitly, we replace the set of ground truth boxes $\{B_t^*, B_{t+1}^*, ..B_{t+T}^*\}$ with a pseudo box-trajectory. The pseudo-box trajectory is defined by a continuous function, $r_t(\cdot)$ describing the trajectory at every timestep as: $\mathbb{T}_{r_t} = (B_t^*, r_{t+1}(B_t^*), ..r_{t+T}(B_t^*))$, relative to the true keyframe location B_t^*. Because the next keyframe is the last trajectory location, we also constrain the pseudo trajectory to match the true bounding box at the end of the trajectory: $r_{t+T}(B_t^*) = B_{t+T}^*$.

In practice, we choose $r_t(\cdot)$ to be either linearly interpolated box annotations, or a parabola as it is continuous and does not assume linear object trajectories. Here, we only need bounding-box annotations every T frames, for correcting the starting- and end-point of our predicted trajectory. Our sparsely-annotated loss $L_{\text{traj}}^{(\text{sa})}$ variants are useful when the dataset only has sparse annotations available.

4. Experiments

Datasets and evaluation setup. We test our hypotheses on a fully controlled MovingDigits dataset. We ablate model choices on a subset of ImageNet VID [58]. We show the practical benefits of our approach on the full ImageNet VID [58] and on EPIC KITCHENS-55 [11]. As a realistic sparsely annotated scenario, we evaluate on YouTube-BoundingBoxes [56] which has approximately 1 keyframe annotation per second, and the Waymo Open dataset [15]. For the ImageNet VID experiments, we train our model on a combination of ImageNet VID and DET datasets as is common practice in [9, 24, 72, 80]. To quantify detection accuracy we adopt the common approach [7, 9, 70, 72] by computing mean Average Precision (mAP), where a detection is correct if its Intersection over Union (IoU) with the ground truth is sufficiently large. Note that although our method samples keyframes during training, we evaluate on all frames at test time, unless stated otherwise.

Implementation details. We test either using a Faster RCNN [57] detector or a Deformable DETR [79] with an ImageNet pre-trained ResNet [31] or SwinB [48] base as the object detection backbone. Our trajectory prediction sub-network contains three fully-connected layers with 1024 dimensions for the middle layer, see Fig. 2. We train our network for 120k iterations on ImageNet VID and 173k iterations on Epic Kitchens with the SGD optimizer, on 4 GPUs. For ImageNet VID, the initial learning rate is 10^{-3} and is divided by 10 at 80K iterations. For Epic Kitchens, the initial learning rate is 5×10^{-4} and is divided by 10 at 120K iterations. For YouTube-BoundingBoxes, the initial learning rate is 5×10^{-4} and is divided by 10 at 100k iterations.

| Input frame | GT Bbox | Predicted Bbox |

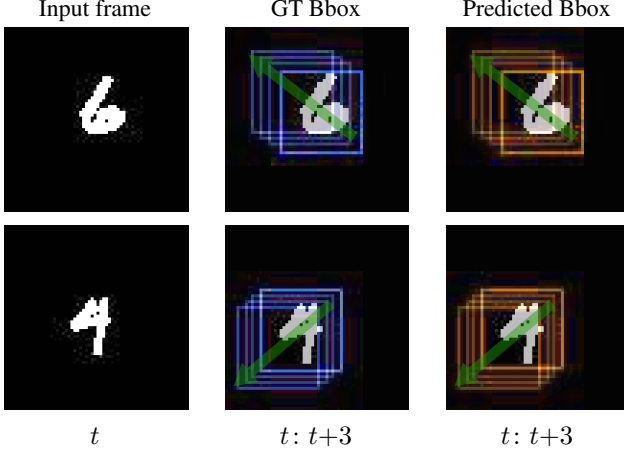| t | $t\colon t{+}3$ | $t\colon t{+}3$ |

Figure 4. **[H1]: Trajectory anticipation on MovingDigits.** We show a single static input frame of digits 6 and 4 and their time-accumulated ground truth and predicted bounding boxes in timesteps $t\colon t{+}3$. Each digit class has an associated linear motion (green arrow). The match of our predicted bounding boxes and the ground truth bounding boxes shows that our model can predict trajectories from a static frame.

4.1. Hypothesis testing

We test our hypotheses by creating our own fully controlled dataset, MovingDigits, where we pair each of the 10 MNIST digit classes with a unique, linear motion of 2 px per frame, see Fig. 4. Each video has 32 frames with a frame size of 64×64 px. We created 200 videos for training and 80 videos for testing, with an equal number of videos per class. We use a trajectory length of $T{=}8$, train for 1.25k iterations and use a ResNet-18 feature extractor and a Faster RCNN detector. We cannot use the existing MovingMNIST [62] because it does not provide detection bounding boxes, and it does not contain motion-appearance correlations.

[H1]: Can the model anticipate motion trajectories? To verify if our model can anticipate trajectories from a single static input frame t, we train on our MovingDigits, where each digit has its own linear motion. We calculate the average IoU over the predicted trajectories for the test videos. The IoU is 0.95, which is near-perfect compared to the IoU of 0.79 for no motion anticipation. We show an example of the predicted bounding boxes (Bbox) by our method and the ground truth bounding boxes (GT Bbox) from time steps t to $t{+}3$ in Fig. 4. Our model can successfully learn motion by anticipating trajectories from a static input frame.

[H2]: Anticipating improves static detection. The motion cues in-between the static keyframes offer an additional source of supervision. Here, we investigate how the motion anticipation affects the static object detector, when we evaluate at test-time only on keyframes. We consider four types of motion supervision for predicting box trajectories:

Motion	Keyframe mAP (%)
Randomized positions	62.68
No motion	73.51
Simulated smooth motion	76.57
Annotated motion	79.31

Table 1. **[H2] Influence of motion anticipation on static object detection.** Static keyframe detection mAP on MovingDigits for varying motion type supervision. For non-random motions, anticipating motion improves static object detection at keyframes.

(1) *Ground truth motion*: trained on true bounding-box trajectories annotated at every frame; (2) *Simulated smooth motion*: bounding boxes move between keyframes according to a smooth parabola. (Details and examples are in the supplementary material); (3) *Randomized positions*: there is motion, but it is not smooth, the boxes in-between the keyframes can occur at any random position in the image; (4) *No motion*: without motion prediction, *i.e.*, the static object detector baseline trained at every video frame.

Tab. 1 shows the keyframe mAP scores for IoU@ [0.50:0.05:0.95]. The *No motion* static object detector is the baseline, which uses no motion. The *Randomized positions* as supervision is detrimental for object detection because the motion is random, and unpredictable. Interestingly, both *Ground truth motion* and *Simulated smooth motion* supervision improve the static keyframe detection. We speculate that the anticipation loss encourages detecting static regions that are most likely to move coherently, with consistent motion offsets: i.e. same direction. Thus, for non-random motions, adding motion anticipation as additional supervision improves static object detection at keyframes, even without knowing the ground truth motion between the keyframes.

4.2. Ablation of model components

We run all ablation experiments on an ImageNet VID [58] subset containing the classes 'dog', 'giant_panda', and 'hamster'. We use a ResNet-101 for feature extraction and Faster RCNN for detection.

[A1]: The effect of the trajectory loss. We evaluate each term in our trajectory loss $L_{\text{traj}} = L_{\sum}{+}L_{\text{bag}(\delta)}$ in Eq. (4), compared to the standard loss L_{bag} in Eq. (1). The mAP scores for trajectory lengths $T{=}4$ and $T{=}8$ are in Tab. 2. The L_{traj} loss has a higher mAP than the bag loss L_{bag} for both trajectory lengths. L_{traj} enforces continuity and smoothness in the predictions over time thus leads to more precise predictions. Furthermore, the improvement of L_{traj} over L_{bag} is larger for longer trajectories, e.g. for $T{=}4$ and $T{=}8$, L_{traj} outperforms L_{bag} by 1.66% and 2.18% respectively. Our L_{traj} loss is defined using offsets, without offsets the $L_{\sum}$ would be equivalent to L_{bag}. We observe that the effect of predicting offsets between neighboring boxes δ_{t+l}, instead of bounding-box coordinates B_{t+l} gives

| | Trajectory length (mAP %) | |
Loss	T=4	T=8
L_{bag}	87.54 ± 0.16	83.97 ± 0.71
$L_{\sum}$	87.95 ± 0.27	84.09 ± 0.67
$L_{\text{bag}(\delta)}$	85.19 ± 0.66	79.73 ± 1.01
L_{traj}	$\mathbf{89.20 \pm 0.21}$	86.15 ± 0.75

Table 2. **[A1]: Loss choice.** Compared to L_{bag}, the $L_{\text{bag}(\delta)}$ loss does worse, whereas $L_{\sum}$ performs on par. Their combination in $L_{\text{traj}} = L_{\sum} + L_{\text{bag}(\delta)}$ outperforms L_{bag}. Predicting offsets instead of bounding-box coordinates in the trajectory network gives better results. These patterns are consistent over both trajectory lengths.

| | Keyframe mAP% | | |
| L_{traj} | Sparse annotation loss | | |
	$L_{\sum}^{(\text{sa})}$	$L_{\text{bag}(\delta)}^{(\text{sa})}$	$L_{\text{traj}}^{(\text{sa})}$
T=4	91.03 ± 0.19	90.35 ± 0.34	74.61 ± 0.72	90.76 ± 0.26
T=8	87.94 ± 0.66	86.98 ± 0.73	68.76 ± 0.99	87.12 ± 0.69

Table 3. **[A2]: Sparse annotation loss analysis.** We sub-sample keyframes of the fully annotated ImageNet VID subset to mimic the sparse annotations, and evaluate our sparse annotation loss on keyframe detection. The $L_{\text{bag}(\delta)}^{(\text{sa})}$ does not improve the detection accuracy, while $L_{\sum}^{(\text{sa})}$ performs on par with the $L_{\text{traj}}^{(\text{sa})}$. The sparse annotation loss achieves a comparable accuracy on keyframe detection with our fully-supervised loss L_{traj}.

an improvement of 0.81% and 0.89%. When considering $L_{\text{bag}(\delta)}$ or $L_{\sum}$ individually, their mAP is lower or on par with L_{traj}. This is because $L_{\text{bag}(\delta)}$ cannot enforce trajectory continuity, while $L_{\sum}$ cannot ensure that trajectories do not diverge by accumulating errors over time. Predicting offsets instead of box coordinates results in better accuracy.

[A2]: The effect of the sparse annotation loss. We test our model using the $L_{\text{traj}}^{(\text{sa})}$ loss for sparse annotations. To evaluate this in a controlled setting, we sub-sample keyframe annotations from the fully annotated ImageNet VID subset to mimic a sparse annotation scenario, and evaluate only on keyframes. We compare the ground truth motion with the result of anticipating linearly interpolated trajectories between keyframes. Tab. 3 shows that the $L_{\text{bag}(\delta)}^{(\text{sa})}$ does not contribute much for the keyframe detection. This is because $L_{\text{bag}(\delta)}^{(\text{sa})}$ constrains the predicted offsets to match the pairwise offsets of the pseudo trajectory at every frame, which is not useful here since the motion is simulated. The $L_{\sum}^{(\text{sa})}$ performs on par with the $L_{\text{traj}}^{(\text{sa})}$. And the keyframe detection accuracy with the sparse annotation loss $L_{\text{traj}}^{(\text{sa})}$ is close to that with the proposed fully-supervised loss L_{traj}.

[A3]: Inference speed vs. accuracy trade-off. We can control the inference speed by sampling fewer keyframes, and thus predicting longer trajectories. We analyse the speed vs. accuracy trade-off of our method on the subset
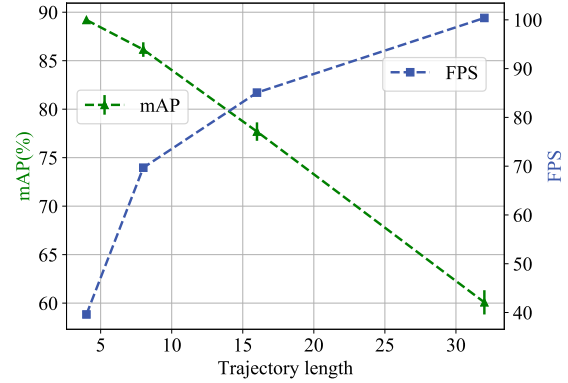

Figure 5. **[A3] Inference speed vs. accuracy trade-off.** We can increase inference speed (FPS) by sampling fewer keyframes and predicting longer trajectories. Yet, by increasing trajectory length the mAP decreases. The trajectory length can be selected according to the needed speed-accuracy trade-off.

of ImageNet VID. Fig. 5 shows we can reach an mAP of 89.2% with a runtime of 39.6 FPS. Moreover a noticeable drop in mAP occurs at trajectories longer than 10 frames. However, when the trajectory is too long, the object motion may vary or the object may leave the frame or a new object may enter the frame. These changes result in the decrease of our model's performance. The trajectory length can be chosen according to the desired speed-accuracy trade-off.

4.3. Comparison with state-of-the-art

[C1]: Experiments on ImageNet VID. We compare our method with state-of-the-art video object detection methods on ImageNet VID in Tab. 4. We mark methods without post-processing with ✓. Methods with post-processing add extra computational cost. Our method does not use any post-processing. We use a prediction trajectory of length $T=4$, as this already gives a considerable reduction in computation speed. For our model, we test using a ResNet-101 [31] and a SwinB [48] backbone with either a Faster RCNN [57] or a Deformable DETR [79].

Among all methods using a ResNet-101 backbone, our method is the most accurate with a 87.2% mAP, which has a 2.7% improvement over the leading MEGA [7]. We also report runtime (FPS) and train-time (hrs/epoch) to show that our method is fast and efficient. We measure the efficiency using the code provided by the original papers, and where the code is not available we mark this with '-'. Note that for the training time this is a rough estimate, when considering the same settings (batch-size, GPU) for all methods. We tested the inference runtime speed on a single NVIDIA GTX 1080 Ti. In terms of both training time and runtime our method is most efficient. Our method has fast inference time with 39.6 FPS, which is $\approx 1.8\times$ faster than existing fast methods like LWND [36] and ST-Lattice [6]. Our effi-

Methods	Backbone	No Post-proc.	mAP (%)	Train-time (hrs/epoch)	Runtime (FPS)
Faster-RCNN [57]	R101	✓	73.6	1.55	21.2
LWND [36]	R101	✓	76.3	-	20.0
FGFA [80]	R101		78.4	6.59	5.0
THP [78]	R101+DCN	✓	78.6	-	-
ST-Lattice [6]	R101		79.6	1.40	20.0
D&T [18]	R101		80.2	6.56	5.0
MANet [70]	R101		80.3	6.88	4.9
STSN [5]	R101+DCN		80.4		
STMN [73]	R101		80.5	2.49	13.2
TROI [22]	R101		80.5	5.18	6.4
SELSA [72]	R101		80.5	3.15	10.6
OGEMN [13]	R101+DCN		81.6	-	8.9
SparseVOD [29]	R101	✓	81.9	-	14.4
BoxMask [28]	R101	✓	83.2	-	6.1
RDN [14]	R101		83.8	-	-
HVRNet [24]	R101		83.8	-	-
TF-Blender [9]	R101	✓	83.8	-	4.9
MEGA [7]	R101		84.5	6.34	5.3
TransVOD [77] (Def. DETR)	R101	✓	81.9	-	32.3
PTSEFormer [69] (Def. DETR)	R101	✓	88.1	-	-
TransVOD [77] (Def. DETR)	SwinB	✓	90.1	-	14.9
Ours (Faster RCNN)	R101	✓	87.2	0.78	**39.6**
Ours (Def. DETR)	R101	✓	87.9	-	36.4
Ours (Def. DETR)	SwinB	✓	**91.3**	-	18.1

Table 4. **[C1]: Experiments on ImageNet VID.** We indicate the methods without video-level post-processing with a ✓. 'No Post-proc.' means no post-processing. R101 here is ResNet-101. The runtime is measured on a NVIDIA GTX 1080 Ti. Our method has the best performance and fastest runtime among all the methods using two-stage detectors (e.g. Faster RCNN). With a stronger detector and backbone, our method exceeds state-of-the-art.

ciency comes from predicting object locations for the next T frames, while processing only sub-sampled keyframes.

Because our method is detector agnostic, we also compared our method with methods using a more advanced detector, Deformable DETR [79], and a more advanced backbone SwinB [48]. In Tab. 4 our method is on-par with PTSEFormer [69] using the Deformable DETR detector and R101 backbone. And our method outperforms overall the state-of-the-art, when using a SwinB backbone.

Comparison on different motion speeds. We evaluate across different motion speeds in Tab. 5. The category of object motion speeds in ImageNet VID follows FGFA [80]. Our method improves mAP on slow and medium motion speeds and achieves comparable results to the previous best method on fast motion speed, which shows the effectiveness of our method across different motion speeds.

[C2]: Experiments on EPIC KITCHENS-55. In EPIC KITCHENS-55, each frame contains avg/max 1.7/9 objects, which is more challenging compared to ImageNet VID. The Epic Kitchens video object detection task consists of 32 different kitchens and 290 classes. The training set has 272

Methods	Backbone	mAP (%)	mAP (%) (slow)	mAP (%) (medium)	mAP (%) (fast)
FGFA [80]	R101	78.4	83.5	75.8	57.6
MANet [70]	R101	80.3	86.9	76.8	56.7
SELSA [72]	R101	80.5	86.9	78.9	61.4
OGEMN [13]	R101+DCN	81.6	86.2	78.7	61.1
HVRNet [24]	R101	83.8	88.7	82.3	**66.6**
IFFNet [38]	R101	79.7	87.5	78.7	60.6
Ours (Faster RCNN)	R101	**87.2**	**92.2**	**86.1**	66.5

Table 5. **[C1]: ImageNet VID across different motion speeds.** Our method improves mAP on different motion speeds.

Methods	S1		S2	
	mAP@.5	mAP@.75	mAP@.5	mAP@.75
EPIC [11]	34.2	8.5	32.0	7.9
Faster-RCNN [72]	36.6	9.9	31.9	7.4
SELSA [72]	37.9	9.8	34.8	8.1
SELSA-ReIm + TROI [22]	42.2	-	39.6	-
BoxMask [28]	44.3	18.5	41.3	15.7
Ours (Faster RCNN)	**44.9**	**18.7**	**41.7**	**16.0**

Table 6. **[C2]: Experiments on EPIC KITCHENS-55.** S1 and S2 represent Seen and Unseen splits, respectively. Our method achieves promising results for both test sets and IoU thresholds.

video sequences captured in 28 kitchens. For evaluation, 106 sequences collected in the same 28 kitchens (S1) and 54 sequences collected in 4 other unseen kitchens (S2) are used. We use a prediction trajectory length of $T=4$ and evaluate for two IoU thresholds of 0.5 and 0.75. As summarized in Tab. 6, our method is more accurate than previous state-of-the-art methods for both Seen/Unseen splits. Our method is applicable to complex video detection tasks.

4.4. Sparsely annotated videos

[S1] Sparsely annotated YouTube-BoundingBoxes. The YouTube-BoundingBoxes dataset [56] has sparse annotations: the video frame rate is 30 fps, and on average it only has annotations at 1 fps. We compare to the Faster RCNN [57] on YouTube-BoundingBoxes in Tab. 7, when using a Faster RCNN detector in our method. The inputs are keyframes sampled with a step 60 and 30, which is every 60 frames and 30 frames in the video, respectively. During training, for Faster RCNN we use the labels at the keyframes, while for our method we use labels with either a step of 30 or 1. For a label step of 30 and a keyframe step of 60, we use $2\times$ more labels than input frames, while for a label step of 1 and keyframe step of 30, we use $30\times$ more labels than frames. Since the video annotations are sparse, we do not have labels at every frame, therefore for a label step of 1, we use our sparse annotation loss, $L_{\text{traj}}^{\text{(sa)}}$. In the $L_{\text{traj}}^{\text{(sa)}}$ we define the box trajectories to be linearly interpolated pseudo trajectories. For keyframes sampled with a step of 60, our method has higher accuracy than Faster RCNN by using $2\times$ more labels and processing the same

Methods	Keyframe step	Label step	mAP (%)
Faster RCNN [57]	60	60	47.6
Faster RCNN [57]	30	30	58.7
Ours L_{traj}	60	30	51.3
Ours $L_{\text{traj}}^{(\text{sa})}$	30	1	**59.8**

Table 7. **[S1] Sparsely annotated YouTube-BoundingBoxes.** mAP on the sparsely annotated YouTube-BoundingBoxes. The sparse annotation loss $L_{\text{traj}}^{(\text{sa})}$ using interpolated pseudo trajectory labels improves keyframe detection.

Methods	AP/L1 (%)	AP/L2 (%)
Faster RCNN	55.66	49.63
Ours $L_{\text{traj}}^{(\text{sa})}$	64.53	59.28

Table 8. **[S2]: Experiments on the Waymo Open dataset**. We report average precision over the $L1$ and $L2$ instance difficulty levels. Even when learning from object annotations at 1 fps, our method using interpolated pseudo-trajectories outperforms the Faster RCNN baseline.

input keyframes. With input keyframes every 30 frames, our method achieves a 1.1% higher mAP than Faster RCNN while optimizing a simulated motion between these frames.

[S2] Results on Waymo Open Dataset. We are the first to perform video object detection on the Waymo Open dataset [15], and thus we can only compare to a static detector, Faster RCNN [57]. We use a Faster RCNN detector as well in our method, and the sparse annotations loss with linearly interpolated pseudo-trajectories. The Waymo Open dataset contains sparse object annotations at 1 fps. Nonetheless, the results in Tab. 8 show that we outperform Faster-RCNN.

4.5. Method limitations

Despite our method's successful predictions as in Fig. 6(a), we also identify several limitations. In practice, multiple motion patterns can be associated with the same appearance: e.g people can walk or jump. In this case, our method may fail to predict the correct object locations. Another failure case is if objects appear or disappear in the middle of a trajectory. In these cases, we will either miss the objects or over-predict. Yet another limitation of our method is the assumption that the motion changes smoothly. If the object trajectories have large variations in a short time because of the video frame rate, our trajectory network may not be able to learn this. One such example is the dog in Fig. 6(b) which suddenly changes its moving direction. Even if we miss some detections of intermediate frames, as shown in Fig. 6(b), our method can recover at the next keyframe detection. While these limitations exist, they only influence our method minimally, since we miss only a few hundred milliseconds, which is reflected in our state-of-the-art video object detection results.

(a) Success case   (b) Failure case

Figure 6. **Example predictions.** White boxes are the ground truth, blue boxes are the predictions. We show one success case in (a) a 'dog' running. In (b) the failure case shows a 'dog' suddenly changing direction. We miss the detection in the second and third frames, yet we recover at the keyframe detection. When the motion has large variations, is unpredictable, and objects enter and leave the frame, our method fails.

5. Conclusion

We propose a method to efficiently detect objects in videos by predicting their future locations from a static input keyframe and the ground truth locations of all frames. Our method associates appearance with motion. Different motion contexts have different appearances, thus we can model various motion patterns from the keyframes with different appearances. Because we predict the future object locations over multiple frames, we do not need to process every frame of the video, but only a subset of the keyframes, which makes our method efficient. Moreover, by learning to predict object trajectories we improve the object detection accuracy when compared to the state-of-the-art on multiple datasets. Finally, by using pseudo object trajectories defined by smooth continuous functions, we can improve object detection accuracy on sparsely annotated videos.

References

[1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5343–5352, 2018. 2

[2] Karim All, David Hasler, and Frangois Fleuret. Flow-boost—appearance learning from sparsely annotated video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1433–1440. IEEE, 2011. 2

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 941–951, 2019. 2

[4] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9739–9748, 2020. 2

[5] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 331–346, 2018. 7

[6] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing video object detection via a scale-time lattice. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7814–7823, 2018. 2, 6, 7

[7] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10337–10346, 2020. 2, 4, 6, 7

[8] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020. 2

[9] Yiming Cui, Liqi Yan, Zhiwen Cao, and Dongfang Liu. Tf-blender: Temporal feature blender for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8138–8147, 2021. 2, 4, 7

[10] Kenan Dai, Jie Zhao, Lijun Wang, Dong Wang, Jianhua Li, Huchuan Lu, Xuesheng Qian, and Xiaoyun Yang. Video annotation for visual tracking via selection and refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10296–10305, 2021. 2

[11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018. 1, 4, 7

[12] Patrick Dendorfer, Vladimir Yugay, Aljosa Osep, and Laura Leal-Taixé. Quo vadis: Is trajectory forecasting the key towards long-term multi-object tracking? *Advances in Neural Information Processing Systems*, 35:15657–15671, 2022. 2

[13] Hanming Deng, Yang Hua, Tao Song, Zongpu Zhang, Zhengui Xue, Ruhui Ma, Neil Robertson, and Haibing Guan. Object guided external memory network for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6678–6687, 2019. 7

[14] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7023–7032, 2019. 7

[15] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021. 1, 4, 8

[16] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–213, 2020. 2

[17] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3299–3309, 2021. 2

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3038–3046, 2017. 2, 7

[19] Ruohan Gao, Bo Xiong, and Kristen Grauman. Im2flow: Motion hallucination from static images for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5937–5947, 2018. 2

[20] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13505–13515, 2021. 2

[21] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 3

[22] Tao Gong, Kai Chen, Xinjiang Wang, Qi Chu, Feng Zhu, Dahua Lin, Nenghai Yu, and Huamin Feng. Temporal roi align for video object recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1442–1450, 2021. 7

[23] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15303–15312, 2021. 2

[24] Mingfei Han, Yali Wang, Xiaojun Chang, and Yu Qiao. Mining inter-video proposal relations for video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 431–446. Springer, 2020. 2, 4, 7

[25] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 0–0, 2019. 2

[26] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 312–329. Springer, 2020. 2

[27] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *CoRR*, 2016. 2

[28] Khurram Azeem Hashmi, Alain Pagani, Didier Stricker, and Muhammamd Zeshan Afzal. Boxmask: Revisiting bounding box supervision for video object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WAVC)*, 2023. 7

[29] Khurram Azeem Hashmi, Didier Stricker, and Muhammamd Zeshan Afzal. Spatio-temporal learnable proposals for end-to-end video object detection. *arXiv preprint arXiv:2210.02368*, 2022. 7

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017. 3

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4, 6

[32] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 749–765. Springer, 2016. 2

[33] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. 2

[34] Kutalmis Gokalp Ince, Aybora Koksal, Arda Fazla, and A Aydin Alatan. Semi-automatic annotation for visual object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1233–1239, 2021. 2

[35] Suyog Dutt Jain and Kristen Grauman. Supervoxel-consistent foreground propagation in video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 656–671. Springer, 2014. 2

[36] Zhengkai Jiang, Peng Gao, Chaoxu Guo, Qian Zhang, Shiming Xiang, and Chunhong Pan. Video object detection with locally-weighted deformable neighbors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8529–8536, 2019. 2, 6, 7

[37] Zhengkai Jiang, Yu Liu, Ceyuan Yang, Jihao Liu, Peng Gao, Qian Zhang, Shiming Xiang, and Chunhong Pan. Learning where to focus for efficient video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 18–34. Springer, 2020. 2

[38] Ruibing Jin, Guosheng Lin, Changyun Wen, Jianliang Wang, and Fayao Liu. Feature flow: In-network feature flow estimation for video object detection. *Pattern Recognition*, 122:108323, 2022. 2, 7

[39] Vicky Kalogeiton, Vittorio Ferrari, and Cordelia Schmid. Analysing domain shift factors between videos and images for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(11):2327–2334, 2016. 2

[40] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–825, 2016. 2

[41] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16020–16030, 2021. 2

[42] Alina Kuznetsova, Aakrati Talati, Yiwen Luo, Keith Simmons, and Vittorio Ferrari. Efficient video annotation with visual interpolation and frame selection guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WAVC)*, pages 3070–3079, 2021. 2

[43] Rui Li, Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Motion-focused contrastive learning of video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2105–2114, 2021. 2

[44] Chung-Ching Lin, Ying Hung, Rogerio Feris, and Linglin He. Video instance segmentation tracking with a modified vae architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13147–13157, 2020. 2

[45] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7083–7093, 2019. 2

[46] Bowen Liu, Yu Chen, Shiyu Liu, and Hun-Seok Kim. Deep learning in latent space for video prediction and compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 701–710, 2021. 2

[47] Xin Liu, Silvia L Pintea, Fatemeh Karimi Nejadasl, Olaf Booij, and Jan C van Gemert. No frame left behind: Full video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14892–14901, 2021. 2

[48] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 4, 6, 7

[49] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In *Proceedings of*

the IEEE/CVF International Conference on Computer Vision (ICCV), pages 2344–2352, 2017. 2

[50] Tahmida Mahmud, Mahmudul Hasan, and Amit K Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5773–5782, 2017. 2

[51] Pascal Mettes, Jan C van Gemert, and Cees GM Snoek. Spot on: Action localization from pointly-supervised proposals. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 437–453. Springer, 2016. 2

[52] Medhini Narasimhan, Shiry Ginosar, Andrew Owens, Alexei A Efros, and Trevor Darrell. Strumming to the beat: Audio-conditioned contrastive video textures. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3761–3770, 2022. 2

[53] SL Pintea and J van Gemert. Making a case for learning motion representations with phase. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, pages 55–64. Springer, 2016. 2

[54] SL Pintea, J van Gemert, and AWM Smeulders. Deja vu: Motion prediction in static images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 2

[55] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6262–6271, 2019. 2

[56] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5296–5305, 2017. 1, 4, 7

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015. 3, 4, 6, 7, 8

[58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 4, 5

[59] Pramod Sharma and Ram Nevatia. Efficient detector adaptation for object detection in a video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3261, 2013. 2

[60] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip H. S. Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[61] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(7):1442–1468, 2013. 2

[62] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 843–852. PMLR, 2015. 5

[63] Guanxiong Sun, Yang Hua, Guosheng Hu, and Neil Robertson. Efficient one-stage video object detection by exploiting temporal consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–16. Springer, 2022. 2

[64] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *CoRR*, 2020. 2

[65] Subarna Tripathi, Zachary C Lipton, Serge Belongie, and Truong Nguyen. Context matters: Refining object detection in video with recurrent neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 2

[66] Carl Vondrick and Deva Ramanan. Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011. 2

[67] Vedran Vukotić, Silvia-Laura Pintea, Christian Raymond, Guillaume Gravier, and Jan C van Gemert. One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network. In *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, pages 140–151. Springer, 2017. 2

[68] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 835–851. Springer, 2016. 2

[69] Han Wang, Jun Tang, Xiaodong Liu, Shanyan Guan, Rong Xie, and Li Song. Ptseformer: Progressive temporal-spatial enhanced transformer towards video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 732–747. Springer, 2022. 7

[70] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. Fully motion-aware network for video object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 542–557, 2018. 2, 4, 7

[71] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8052–8060, 2018. 2

[72] Haiping Wu, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Sequence level semantics aggregation for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9217–9225, 2019. 2, 4, 7

[73] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 485–501, 2018. 7

[74] Yuzheng Xu, Yang Wu, Shohei Nobuhara, Ko Nishino, et al. Video region annotation with sparse bounding boxes. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2020. 2

[75] Zhujun Xu, Emir Hrustic, and Damien Vivet. Centernet heatmap propagation for real-time video object detection. In

Proceedings of the European Conference on Computer Vision (ECCV), pages 220–234. Springer, 2020. 2

[76] Zhishuai Zhang, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. Stinet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11346–11355, 2020. 2

[77] Qianyu Zhou, Xiangtai Li, Lu He, Yibo Yang, Guangliang Cheng, Yunhai Tong, Lizhuang Ma, and Dacheng Tao. Transvod: end-to-end video object detection with spatial-temporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022. 7

[78] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7218, 2018. 2, 7

[79] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 4, 6, 7

[80] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 408–417, 2017. 2, 4, 7

[81] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712, 2018. 2