2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Robust and efficient post-processing for video object detection

Alberto Sabater[1] Luis Montesano[1,2] Ana C. Murillo[1]

***Abstract*— Object recognition in video is an important task for plenty of applications, including autonomous driving perception, surveillance tasks, wearable devices or IoT networks. Object recognition using video data is more challenging than using still images due to blur, occlusions or rare object poses. Specific video detectors with high computational cost or standard image detectors together with a fast post-processing algorithm achieve the current state-of-the-art. This work introduces a novel post-processing pipeline that overcomes some of the limitations of previous post-processing methods by introducing a learning-based similarity evaluation between detections across frames. Our method improves the results of stat-of-the-art specific video detectors, specially regarding fast moving objects, and presents low resource requirements. And applied to efficient still image detectors, such as YOLO, provides comparable results to much more computationally intensive detectors.**

I. INTRODUCTION

Many application fields such as robotics, surveillance or wearable devices require object detection over their embedded camera video streams and efficient algorithms to process them. Deep learning based detection approaches, such as the well known YOLO [1] or MaskRCNN [2], have boosted image object detection performance in the recent years. However, there is still a large gap between object detection performance in video and images, mainly because video data is more challenging due to numerous artifacts and difficulties such as blur, occlusions or rare object poses.

Two main strategies have been explored to improve object detection in videos. On one hand, there are detection models specifically designed to work over video streams [3], [4]. They typically implement feature aggregation from nearby frames and achieve higher accuracy than still image detectors, but they are often slow and require heavy computation. This makes them not well suited for applications on low-resource environments, like wearable devices or for applications where near real-time computing is a requirement, such as robotics or monitoring video analysis. On the other hand, post-processing methods such as Seq-NMS [5] and Seq-Bbox-Matching [6] have been proposed to process the outputs of an image object detector evaluated on the video frames to improve the performance. They are mostly based on linking the predicted objects across frames and using these links to refine the detection results. This strategy is typically much faster than specific video object detection methods.

This research has been funded by FEDER/Ministerio de Ciencia, Innovación y Universidades/Agencia Estatal de Investigación RTC-2017-6421-7 and PGC2018-098817-A-I00, DGA T45 17R/FSE and the Office of Naval Research Global project ONRG-NICOP-N62909-19-1-2027.

[1] A. Sabater, L. Montesano and A.C. Murillo are with DIIS - I3A, Universidad de Zaragoza, Spain. {asabater, acm}@unizar.es
[2] L. Montesano is also with Bitbrain Technologies, Zaragoza, Spain. {luis.montesano}@bitbrain.com

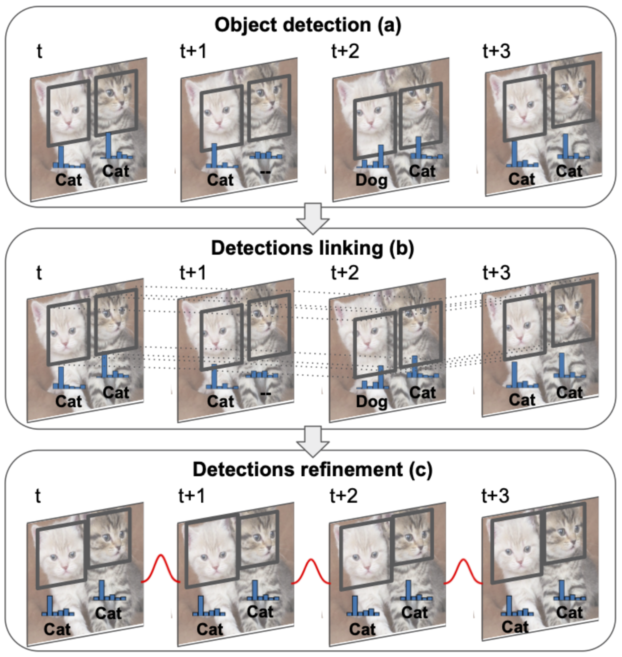

Fig. 1. Proposed post-processing pipeline to improve video object detection. A standard image detector predicts object instances in a sequence of video frames (a). Our approach links object instances across frames based on the learned similarity evaluation (b) and uses contextual information to refine the detections (c), both object classification and location.

The key for post-processing methods is the way they relate detected objects among consecutive frames. This linking is usually based on hand-made heuristics, a common one uses the Intersection over Union (IoU) between object detections. This approach has several limitations. First, the base detector does not always predict reliable bounding box coordinates. Second, IoU values strongly depend on displacements due to camera or object motion. For instance, fast moving objects may not present enough overlap to get a reliable linking. Note that the same effect occurs when the frame rate drops (e.g., due to computational constraints). Finally, the presence of multiple objects simultaneously in the scene also makes heuristics difficult to design and prone to fail.

This paper presents a novel post processing pipeline for video object detection (see Fig. 1) that can be used in conjunction with any video or image detector. The main novelty relies on the way the similarity is evaluated between object detections to link them across frames. We propose to use a learning-based similarity function that combines different descriptors and is designed to be more robust to varying speeds of object motions. Once all possible object instances are linked across frames, a refinement step is run to improve both the classification and location of object detections.

We evaluate our method against state-of-the-art post-processing methods for image detectors on the well known video dataset ImageNet VID [7], obtaining better results mainly due to more robust links for fast moving objects. We also show that our method improves the performance of specific video object detectors. Interestingly, the increased robustness to fast moving objects implies we can process frames more sparsely, and then allows us to use more computationally demanding object detectors even if time constraints are high and not all frames can be processed. Code, learned models and training data are available online[1].

II. Related Work

Object detection in videos often builds on top of image object detection. The latter is a well studied problem with current state-of-the-art methods based on deep learning architectures. Multi-stage detectors [8], [9] follow R-CNN [10] and split the prediction process in two stages: candidate selection and candidate classification. Single shot models, on the other hand, use a single Neural Network trained end-to-end to perform object detection in a single step. Many variants exist [11], [1], [12], [13], [14] with different object representations. They are in general faster but perform worse than multi-stage ones.

There are two types of approaches to extend object detection to video and cope with its specific challenges (blur, occlusions, rare poses) and to exploit the temporal information and consistency in video data:

a) Video Detectors: Video object detectors are designed to exploit the surrounding context of a frame and usually the model propagates or shares object features across frames. FGFA [3] aggregates nearby features along the motion path given by an optical flow estimation. D&T [15] trains a ConvNet end-to-end both for object detection and tracking by using correlation across feature maps and re-scores linked detections. SELSA [4] extracts object proposals from different frames of a video and aggregates their features depending on their semantic similarities. TSM [16] shifts features along the temporal dimension to perform object detection with 2D CNNs. TCD [17] conditions the output of a single image detector by the tracklets calculated in previous steps.

Video object detectors are usually more computationally expensive than detectors working on still images due to the increased network complexity, their need to process more data and often the requirement to calculate additional data such as optical flow.

b) Post-processing methods to improve video detection: Post-processing methods incorporate the temporal context information to the output predictions of either image or video object detectors. Applied to per-frame detections, these methods speed-up the inference pipeline with respect to specific video detectors while boosting the final detection performance with respect to their base detector.

[1]https://sites.google.com/a/unizar.es/filovi/

Some post-processing methods are based on Kalman Filter variations and use tracking ideas to make the detections more robust or consistent. These are often applied to specific domains like person re-identification, where persons have to be detected and tracked in videos [18], [19] extracted from cameras usually with a fixed position. Deep SORT [20] improves the SORT [21] algorithm, based on Kalman Filters and the Hungarian Algorithm, by adding appearance information to each predicted bounding box.

Other strategies applied in more general settings, are based on bounding box propagation or matching across frames. For example, T-CNN [22] uses context information to suppress false positives and optical-flow to propagate detections across frames to reduce false negatives. Seq-NMS [5] matches high overlapping detections across frames within the same clip to make detection results more robust. Seq-Bbox-Matching [6] links overlapping detections from continuous pairs of frames to create and re-score object instances and uses them to infer missed detections.

Our proposed approach is related to this last group of post-processing techniques. We perform bounding box linking across frames, but instead of hand-made heuristics, a learned classifier is used to distinguish whether two detections belong to the same object instance or not. This model exploits both intermediate features from the base object detector and additional properties of the bounding box.

III. Proposed Framework

Our proposed approach for video object detection runs the three modules summarized in Fig. 1, which are detailed next.

A. Object detection and description

Our approach works on top of any initial object detector that can provide object bounding boxes and class confidence score vectors. For each video frame t, we get a set of object detections, and each object detection o_t^i is described by:

- Location and geometry, i.e., its bounding box information: $bb_t^i = \{x, y, w, h\}$
- Semantic information, i.e., the vector of class confidences cc_t^i provided by the detector. $cc_t^i \epsilon \mathbb{R}^C$, where C is the number of classes within the dataset.
- Appearance, i.e., a L2-normalized embedding, $app_t^i \epsilon \mathbb{R}^{256}$, representing the appearance of the patch. It is learned as detailed next.

Figure 2 summarizes how these descriptors are obtained. The first two are directly provided by the base object detection model. The appearance descriptor is computed from a set of feature maps generated by an intermediate layer of the base model. We propose a simple architecture to learn this appearance embedding as shown in the figure. A RoI Pooling Layer [8] is used to extract the feature map outputs that correspond to each of the predicted bounding boxes and scale them to fit a pre-defined shape. Since such intermediate descriptors are generally large, we use a single Fully Connected layer to learn a mapping to a lower dimensional embedding and limit the memory and

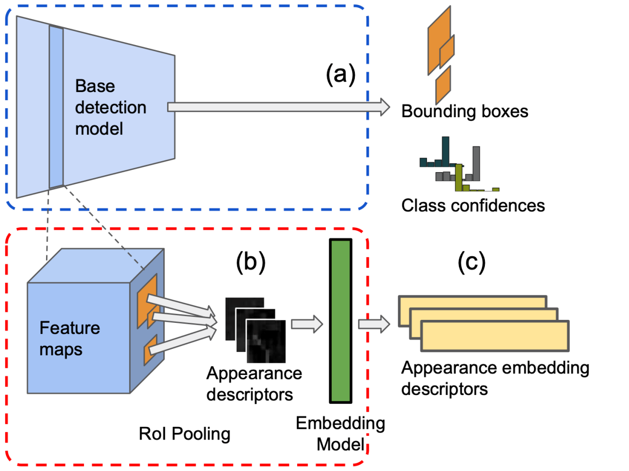

Fig. 2. **Object detection and description**. Each detection from the base object detector is represented by the bounding box and vector of class confidences provided by the base model (a) and by an appearance descriptor. The appearance descriptor is built from a set of feature maps pooled from the base detector (b) mapped into a lower-dimension embedding vector (c).

computational resources. This embedding model is trained minimizing the triplet loss proposed in [23]:

$$\sum_{i}^{N}\left[\|f(x_i^a)-f(x_i^p)\|_2^2-\|]f(x_i^a)-f(x_i^n)\|_2^2+\alpha\right]_+, \quad (1)$$

that compares one Anchor sample x_i^a to one Positive sample x_i^p and one Negative sample x_i^n. $f(x)\epsilon\mathbb{R}^d$ embeds a sample x in a d dimensional space and α is the margin enforced between positive and negative pairs. The loss minimizes the euclidean distance between the Anchor and the Positive samples, while maximizing the euclidean distance between the Anchor and the Negative sample. Section IV-A describes the essential step of creating the triplets training set as well as all the other implementation details.

B. *Object detection linking*

Our second module links detections into tracks by building a set of tubelets, i.e. sets of corresponding detections along the video, in a similar manner to [6]. This linking, summarized in Fig. 3, is a sequential process. We start building tubelets from the object detections between the first pair of frames and extend them as long as corresponding objects are still found in the next following frames. New tubelets can be initialized at any frame with those detections not included in existing tubelets.

To link detections between two consecutive frames, we propose a similarity function based on the following pair-wise features computed for each possible link of detections (o_t^i and o_{t+1}^j):

$$\begin{aligned} f_{loc} &= \{IoU, d_{centers}\}, \\ f_{geo} &= \{ratio_w, ratio_h\}, \\ f_{app} &= d_{app}, \\ f_{sem} &= f_{sem}^a \cdot f_{sem}^b, \end{aligned}$$

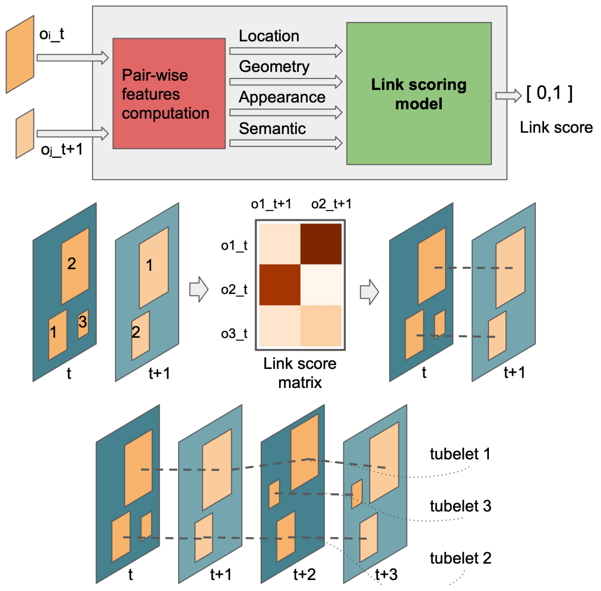

Fig. 3. **Object detection linking**. For all possible pairs of detections from consecutive frames (t and $t+1$), we build a set of features based on their location, geometry, appearance and semantics. These features are used to predict a linking (similarity) score. Links are established between consecutive frames and tubelets are composed of links as long as possible.

where IoU is the Intersection over Union of both detections, $d_{centers}$ is the relative euclidean distance between the two bounding box centers, $ratio_w$ and $ratio_h$ are the width and height ratio between the two bounding boxes, d_{app} is the euclidean distance between the appearance embeddings and f_{sem} is the dot product of class confidences vectors (cc_t and cc_{t+1}). The actual link score (LS) between two detections is computed as follows:

$$LS(o_t^i, o_{t+1}^j) = f_{sem}\,\mathbf{X}(f_{loc}, f_{geo}, f_{app}), \quad (2)$$

where $\mathbf{X}$ is a logistic regression trained to distinguish whether two detections, given their pair-wise features, belong to the same object instance (high linking score) or not (low linking score). More details on how this regression is trained based on triplets information can be found in Section IV-A.3.

A score matrix is created with the link scores of every pair of detections between two frames. We use a greedy approach to match them and extend the tubelet, i.e., the highest score is picked to be a link, and the corresponding row and column are suppressed from the matrix, repeating the process until no more links can be set. Although we tried more complex methods to solve the assignment problem, such as the Hungarian method, we did not find any improvements to the algorithm described above. For each pair of frames, a new tubelet is created if an assigned pair did not belong to an already existing tubelet.

Since object detectors do not always make accurate predictions, some instance linkings are prone to generate false positives when they do not belong to any real object. When this happens, our algorithm outputs a low linking score. Linking candidates that present a score under the established threshold are filtered out.

C. Refinement: re-scoring and re-coordinating

The final step of our method uses the linkings of each tubelet to improve its object classification and location:

1) Re-scoring: This step simply averages all class confidence vectors from each tubelet and assigns this average to all detections within the tubelet. This process is able to correct mislabeled detections in a subset of frames or disambiguate those with low confidence.

2) Bounding box coordinates: Object detectors predict highly accurate bounding box coordinates on still images or on low-motion frames, but this regression tends to be less accurate on the objects that present rare poses, defocus or fast motions. We treat each coordinate of a linked object over time as a noisy time series. Note that noisy bounding box detections cannot properly fit the real object along time. We use smoothing to remove or alleviate this noise. In particular, we convolve a one-dimensional Gaussian filter along each time series. The smoothed series are then used as the set of coordinates of the object in the tubelet.

IV. Experiments

This section describes our post-processing evaluation on different detectors and data conditions.

A. Experimental setup

1) Dataset: The main dataset used is ImageNet VID [7], so far the largest densely annotated dataset for video object detection. It consists of 3862 and 555 training and validation snippets densely labeled with multiple bounding boxes belonging to 30 different object classes. These classes are a subset of the 200 classes from ImageNet DET dataset [7]. ImageNet VID data includes a wide variety of conditions, such as different object movement types, blur, defocus or occlusions. There is an average of 1.59 objects per frame, each one with ground truth annotations of their bounding box and track id, both in train and validation sets.

2) Evaluation metrics: Mean Average Precision (mAP) is computed to evaluate video object detection performance in ImageNet VID, as established in the original paper, and to compare our approach with other methods. To provide deeper insight, we also report the mAP according to three groups of motion (slow, medium and fast) as defined in [3].

3) Triplet dataset used for training our models: Both the embedding model (Sec. III-A) and the link scoring model (Sec. III-B) are trained with the same dataset organized in *triplets*. Each triplet is composed of one Anchor (sample point), one Positive (same object instance) and one Negative (different object instance) bounding box, all of them according to ImageNet VID data groundtruth. Note that for the link scoring model each triplet provides a positive and a negative training example.

We compiled a set of 50000 and 8000 triplet samples, for training and validation respectively, built as follows. For each triplet sample, we randomly sample a track id, i.e., an object track. The Anchor is obtained from a random frame of the track. The Positive example is taken from a frame sampled from $\pm$ 25 frames away of the Anchor, i.e. within a one second window. Note that most videos in the dataset are recorded with a frame rate between 20 and 30 frames per second. The Negative sample is simply an object with a different track id. It can be randomly obtained from either the same snippet as the Anchor or from another video. Negative examples may include objects from the same class but different instance.

4) Base detection model: Although our approach works with any object detector on video data, for comparison purposes we have defined a single baseline, YOLOv3 [24], as our per-frame base object detector. YOLO is a well known and broadly used single-shot detection model that uses a Fully-Convolutional Neural Network architecture to get predictions at different scales, achieving great time-accuracy trade-off.

This base model has been trained with data both from ImageNet DET and ImageNet VID using the same data split as FGFA [3]. We trained the model with data augmentation techniques: multi-scale input, horizontal flip, cropping, shifting, jitter and saturation. For the inference phase we fix the input image shape to 512x512 pixels and we replace the usual non-maximum-suppression (NMS) implementation, that works at a class detection level. Instead, we apply it at a bounding box level taking the maximum class confidence score for each detection in a frame as one of the inputs of NMS. Our NMS module finally outputs a set of bounding boxes for each frame along with a class confidence vector for each bounding box. Finally, we filter out those detections whose maximum class confidence is under a threshold of 0.005. The predictions obtained in this inference phase are the same data that we use in all the tests we perform within our ablation study and comparison with other post-processing techniques. With this configuration we are able to obtain predictions at 23 fps with a single NVIDIA GeForce RTX 2080Ti GPU.

5) Appearance embedding model configuration: The feature maps used for the appearance embedding correspond to the output of the Convolutional Layers Block from the base detection model, that downsamples the input image by a factor of 16. Our RoI Pooling extracts the feature patches from these maps and scales them to a shape of 5x5x256. These values are set to minimize the loss of information on the scaling phase while allowing to predict fast appearance embeddings. The Fully Convolutional layer that learns the final embedding contains 256 neurons and outputs $L2$-normalized embeddings of 256 values.

B. Performance & analysis of other post-processing methods

Table I shows object detection performance of our post-processing approach compared to two other well known approaches on the ImageNetVID dataset. We evaluate the improvements of the post-processing methods with respect to the predictions obtained from a common base model (YOLOv3). We measure the mAP for all test videos and detail specific results for slow, medium and fast moving objects. The average processing time per frame is also shown. Each row corresponds to one of the following approaches:

TABLE I

RESULTS OF DIFFERENT POST-PROCESSING APPROACHES FOR OBJECT DETECTION IN IMAGENETVID VALIDATION SET.

	mAP	**mAP - slow motion objects**	**mAP - medium motion objects**	**mAP - fast motion objects**	**avg. processing time (ms) per frame (detection + post-processing)**
Only base detector$^+$	68.59	76.79	66.45	45.79	44
Ours	**75.06**	**82.54**	**74.29**	**56.58**	46.58 (44 + 2.58)
Seq-NMS [5]	71.51	79.99	70.01	50.95	54.41 (44 + 10.41)
Seq-Bbox-Matching [6]	74.19	81.13	73.22	54.39	44.40 (44 + 0.40)

$^+$All other methods are run as post-processing of this base detector.

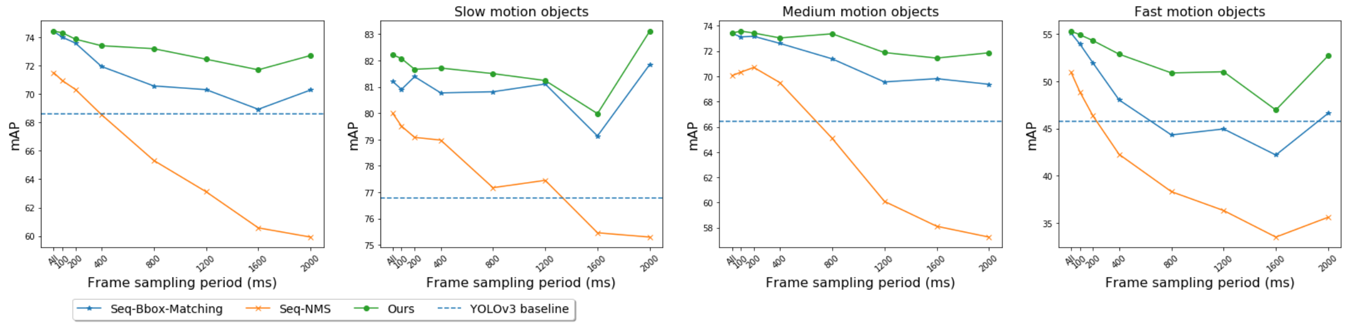

Fig. 4. Performance of different post-processing methods evaluated with different frame sampling period. (a) Shows mAP on all object instances. Following [3] (b), (c), and (d) shows the mAP of the objects according to their motion, slow, medium of fast. Dotted line shows the mAP baseline for each motion. All post-processing methods have been applied to our YOLOv3 baseline predictions. Seq-NMS has been calculated with the code released by [3]. Seq-Bbox-Matching has been calculated with code replicated from the paper, since the original one is not available.

- **Only base detector**. Corresponds to the execution of the base model detector, YOLOv3, with the configuration described in Sec. IV-A.4 with no post-processing.
- **Ours.** Corresponds to YOLOv3 detections post-processed with our proposed method configured with the following parameters: linking threshold of 0.7 to suppress low-scoring detection linkings, and standard deviation of the Gaussian filter in the re-coordinating module set to 0.6.
- **Seq-Bbox-Matching [6].** Corresponds to YOLOv3 detections post-processed with Seq-Bbox-Matching. Since it does not have a public implementation, we have replicated it from the paper (κ value of 12 for their tubelet linking module, as they specify).
- **Seq-NMS [5]**. Corresponds to YOLOv3 detections post-processed with Seq-NMS using the public implementation by FGFA [3], with the same parameters as in the original publication [5] but with our own confidence threshold (defined in Section IV-A.4).

The results show how post-processing is able to improve the baseline detector. Our method achieves better performance than the two state-of-the-art methods when applied to the same base detections as our approach. The computation times of our method are slightly higher than those of Seq-Bbox, but they are still in the order of milliseconds. Although our method overperforms all the groups, the difference between the methods is more noticeable for fast-moving objects, likely because both previous methods rely too much on the IoU between object detections in consecutive frames, while our approach has additional descriptors and a learned function that make the linking process more robust to changes in position. Our method improves $6\%, 8\%$ and 11% w.r.t. the baseline method, while the other methods improve around $6-7\%$ for all classes.

To get more insight about this increased robustness to speed, we simulated a lower processing frame rate. This is important in practice when limited computational resources impede the processing at the acquisition frame rate. This more challenging scenario shows how the approaches work when objects change more drastically both their location and appearance. We ran the same object detection task on the ImageNet VID validation set, but simulating a lower processing frame rate. Since the videos have different sampling rates, we fixed the time between frames and picked the closest frame for each video. We removed tracks composed by less than 2 frames, since post-processing methods are not useful there. The number of videos dropped from 555 to 522 in the smallest data configuration (2000 ms). Note that evaluating different frames and tracks for different sampling rates can lead the mAP to not decrease as expected when the dropped data belongs to objects that are difficult to detect (this effect is more noticeable between the frame sampling periods 1000 and 2000). Since linking scores drop their value for long time-distant detections due to their lower similarities, for this test we set a linking threshold of 0.05, more suitable to suppress spurious detections and being able to get long-term linkings. We also remove the tubelet linking module of Seq-Bbox-Matching and set the parameters of the method according to the paper recommendations for this type of experiment.

Figure 4 shows the mAP for all test videos, including separate plots for different object motion speeds, for varying values of frame sampling periods. We can observe similar effects to before. Our method gets comparable results to the

TABLE II

COMPARISON WITH VIDEO OBJECT DETECTORS ON IMAGENETVID VALIDATION SET.

	base object detector (backbone)	**mAP backbone**	**mAP ALL**	**mAP slow**	**mAP medium**	**mAP fast**	**avg. processing time (ms) per frame (detection + post-processing)**
FGFA [3]*	R-FCN (ResNet101)	74.1	77.1	85.9	75.7	56.1	128
TCD [17]	Faster R-CNN (ResNet101)	74.6	83.5				
SELSA [4]*	Faster R-CNN (ResNet101)	73.62	82.69	88.02	81.34	67.17	458
YOLOv3 + Ours	YOLOv3 (Darknet-53)	68.59	75.06	82.54	74.29	56.58	46.58 (44 + 2.58)
FGFA + Ours	R-FCN (ResNet101)	74.1	80.09	87.42	79.1	61.38	149 (128 + 21)
SELSA + Ours	Faster R-CNN (ResNet-101)	73.62	84.21	88.72	83.32	71.09	466.6 (458 + 8.6)

*Inference times have been calculated by running their official code on a single NVIDIA GEOFORCE RTX 2080Ti GPU
All models make use of both Imagenet DET and VID data for training (or pretraining).

others when working with continuous frames, but it manages to get more robust results when frames are processed more sparsely. Interestingly, our method is always able to improve over the baseline detector performance, while the other methods cannot.

C. *Post-processing of video object detection methods*

Table II compares the results of state-of-the-art specific Video Object Detectors with our post-processing method applied to an Image Object Detector (YOLOv3 [24]) and to two Video Object Detectors (SELSA [4] and FGFA [3]). The results with YOLOv3 are the same as in the previous Table I and are included to ease comparisons. TCD [17] results can not be fully analyzed, since there is no code available up to our knowledge. We show their overall result as reference, as reported in the original paper. SELSA and FGFA results have been obtained using their official code implementation and parameters. They use a ResNet-101 as backbone and train with data augmentation and a mix of ImageNet VID and DET. Since extracting low-level features from a Video Object Detector architecture is not trivial, appearance features have not been used in SELSA and FGFA tests.

Table II shows that post-processing the YOLO single image detections cannot beat the best more complex detectors that work over multiple frames. But it is interesting to note that it does achieve slightly better performance than the corresponding much more complex ResNet based models when performing per-frame detections. However, when applied to state-of-the-art video object detectors, our approach is still able to boost the performance of SELSA and FGFA between 2 and 3%. This gain is mainly due to medium and fast object movements, where we get a 5% gain. This is not always the case for other post-processing methods that failed to improve video detectors [4].

Finally, it is worth stressing again that applying our post-processing approach is always a computationally cheap operation when compared to the detection time. Post-processing time depends on the number of detections per frame, but even with many detections it is still in the order of tens of milliseconds. Our results show that simple and efficient post-processing methods can boost performance of many state-of-the-art detectors with minimum computational requirements and, when time constraints are important, can be a useful approach to trade-off time performance and computational resources.

D. *Qualitative results on EPIC-KITCHEN Dataset*

This experiment tests our proposed post-processing pipeline in a more challenging scenario. We perform a qualitative evaluation on the EPIC-KITCHENS dataset [25]. It is an egocentric video dataset where different users perform daily activities in a kitchen. Multiple objects of different categories appear in the scenes with uncommon perspectives and movements compared to traditional video data. Groundtruth includes user action annotations, multi-language narrations and *active* object detections (note this means not all objects are annotated in the videos).

In order to prove the generalization ability of our approach, we apply it to predictions on these egocentric videos. Predictions are obtained with a YOLOv3 model trained only with the COCO dataset [26], just still images from 80 different categories involved.

Figure 5 shows a scene (frames 18812, 18832, 18868 and 18908 from the video ”P04_24”; note that KITCHENS videos are recorded at 60 fps) that involves a fast camera motion, where objects drastically change their location and get out of focus. The YOLO predictions (first row) suffer from many misdetections and misclassifications due to the mentioned video artifacts. The application of our approach (second row) manages to correct many of them. Partially occluded objects are detected (tracks 14314 and 14364), out of focus objects are correctly detected (track 14436) and classified (track 13020) and false positives are suppressed (observe detected knife, mouse and toothbrush from the base predictions). By using neighboring frame information, object coordinates show a smoother evolution along the full video sequence thanks to our re-coordinating module, removing wrong shapes and flickering.

V. CONCLUSIONS

In this work, we present a novel post-processing method for video object detection. By using a novel set of detection features we study the similarity between frame detections from a learning-based approach as a preliminary step to a prediction refinement. With a light computational overhead, we boost the performance of state-of-the-art video object detectors and applied to efficient still image object detectors we achieve comparable results to more complex models. As demonstrated, our novel post-processing solution allows to overcome the main video challenges such as misdetections

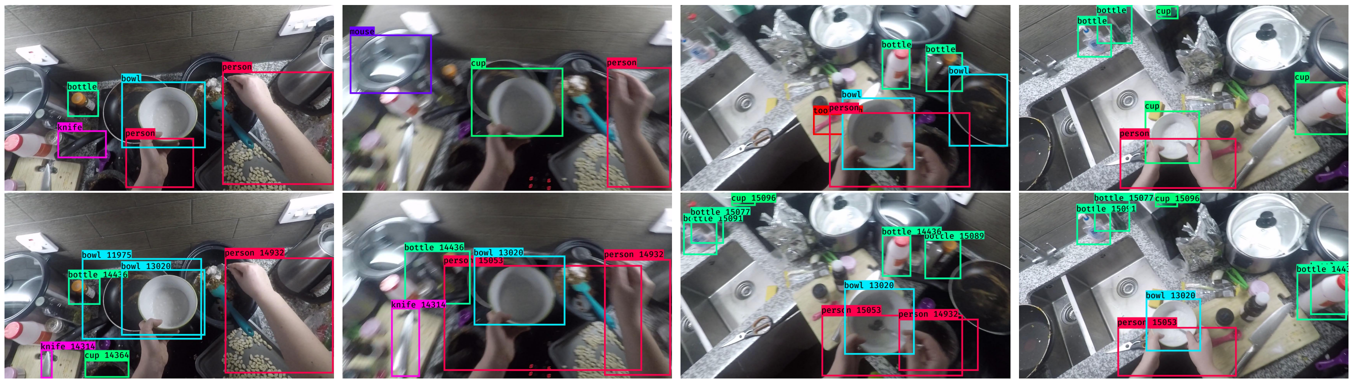

Fig. 5. Object detection on EPIC-KITCHENS data using a YOLOv3 model (top row) and improvements obtained applying our post-processing (bottom row). Predicted bounding boxes are shown along with their track_id when our post-processing is applied. To see more varied results these four frames are not consecutive (18812, 18832, 18868 and 18908), watch the supplementary video for results on the whole sequence[3]. Low-scoring detections have been removed for a better visualization

and misclassifications due to fast motion objects, occlusions or defocus, and also proves its robustness on environments where lower processing frame rates are a restriction.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[3] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[4] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," *ICCV 2019*, 2019.

[5] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *CoRR*, vol. abs/1602.08465, 2016. [Online]. Available: http://arxiv.org/abs/1602.08465

[6] H. Belhassen, H. Zhang, F. Virginie, and E.-B. Bourennane, "Improving Video Object Detection by Seq-Bbox Matching," in *14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Prague, Czech Republic, Feb. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02050829

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[9] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[12] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *The European Conference on Computer Vision (ECCV)*, 2018.

[13] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[15] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *IEEE International Conference on Computer Vision*, 2017.

[16] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[17] Z. Zhang, D. Cheng, X. Zhu, S. Lin, and J. Dai, "Integrated object detection and tracking with tracklet-conditioned detection," *CoRR*, vol. abs/1811.11167, 2018. [Online]. Available: http://arxiv.org/abs/1811.11167

[18] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *CoRR*, vol. abs/1603.00831, 2016. [Online]. Available: http://arxiv.org/abs/1603.00831

[19] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret, "Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[21] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[22] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-cnn: Tubelets with convolutional neural networks for object detection from videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, Oct 2018.

[23] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] D. Damen, H. Doughty, G. Maria Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, *et al.*, "Scaling egocentric vision: The epic-kitchens dataset," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 720–736.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[3]https://youtu.be/QXL2FlyO1jA