Motion Vector Extrapolation for Video Object Detection

Julian True
Ryerson University, Toronto ON
jtrue@ryerson.ca

Naimul Khan
Ryerson University, Toronto ON
n77khan@ryerson.ca

Abstract

Despite the continued successes of computationally efficient deep neural network architectures for video object detection, performance continually arrives at the great trilemma of speed versus accuracy versus computational resources (pick two). Current attempts to exploit temporal information in video data to overcome this trilemma are bottlenecked by the state-of-the-art in object detection models. We present, a technique which performs video object detection through the use of off-the-shelf object detectors alongside existing optical flow based motion estimation techniques in parallel. Through a set of experiments on the benchmark MOT20 dataset, we demonstrate that our approach significantly reduces the baseline latency of any given object detector without sacrificing any accuracy. Further latency reduction, up to 25× lower than the original latency, can be achieved with minimal accuracy loss. MOVEX enables low latency video object detection on common CPU based systems, thus allowing for high performance video object detection beyond the domain of GPU computing. The code is available at https://github.com/juliantrue/movex.

1. Introduction

Object detection has seen significant progress over the last several years [1] [8]. Each new iteration or approach promises higher accuracy at the cost of higher inference latency, or lower latency at the cost of lower accuracy when compared with high latency counterparts. Computing companies are in an arms race to provide hardware that offers the capability to use high accuracy models at low latencies, but this introduces, at best, a dependency on off-the-shelf GPUs and at worst a dependency on expensive niche co-processors. Despite the progress made, the trilemma persists; there is not a silver bullet to address the three constraints of accuracy, latency, and cost simultaneously.

Optical flow has also had recent performance gains through the use of CNN architectures [6] [4]. Through these methods and GPU hardware acceleration, dense optical flow techniques have become faster and more accurate over the last several years.

It is well established that image content varies slowly in video data as following the same object through time can be viewed as a different task all together compared to finding unique objects every frame. Though there have been attempts to exploit this temporal information redundancy through feature propagation based on optical flow methods in the past, performance remains bottle-necked by the latency associated with a CNN inference [11].

This work proposes MOVEX, an online, real-time, method of object detection in video. Through the combination of an arbitrary off-the-shelf object detection deep neural network (DNN) with a coarse approximation of optical flow and an optimistic sparse detection propagation parallelism strategy, we demonstrate that fast, accurate, and computationally inexpensive video object detection can be achieved. Furthermore, this work demonstrates the capability to accelerate object detectors up to 25 times the original performance with minimal (< 0.01AP) accuracy degradation. Since MOVEX does not require the use of a GPU, it enables models typically limited to the realm of GPU computing to be used on commodity CPU hardware with lower inference latency than found in GPU implementations of the same model.

2. Related Work

This work builds on the problem formulation provided by Zhu *et al.* known as Deep Feature Flow. It frames video object detection as a two-step algorithm consisting of expensive feature extraction at sparse key-frames and feature propagation for non-key frames through the use of a function they coin the sparse feature propagation [11].

$$f_k = \mathcal{W}(f_i, M_{i\to k}, S_{i\to k}) \tag{1}$$

Equation 1 provides the ground work for propagating features forward from frame I_i to frame I_k through the use of the sparse feature propagation function $\mathcal{W}$ which accepts as input the feature map from frame i, the 2D optical flow field $M_{i\to k}$, and a so-called scale field $S_{i\to k}$. The authors used a CNN based method known as FlowNet [4] to estimate the flow $M_{i\to k}$ and adding an extra channel to esti-

mate the scale field $S_{i \to k}$. Additionally they use a ResNet 50 and 101 network with classification layers removed to use for the feature extractor backbone [5] and turn it into an object detector through the use of an R-FCN head network on top [2] [11].

Through this work, Zhu *et al.* demonstrated that this approach was effective at reducing average latencies associated with video object detection. This technique however, did not fully redress the high latency operation of performing an inference with a CNN. Although inferencing less on a sequence of images does lower the average, it does not eliminate the necessity for a blocking high latency inference every k frames.

3. Motion Vector Extrapolation (MOVEX)

3.1. Coarse Optimal Flow Approximation

Modern video codecs such as VP9 and H.265, as well as older codecs such as H.264 encode video with intra-frame and inter-frame coding techniques. In order to reduce the entropy between successive frames, these video codecs implement a macroblock (MB) structure that allows for pixel-patches to be translated within the frame before subsequently taking the difference between successive frames. These intra-frame translations that minimize the mean absolute difference (MAD) between image patches, and thus successive frames, are referred to as motion vectors (MVs) [9].

It is notable that these vectors do not specifically encode inter-frame object translations, despite their name, they encode the vector that minimizes the differences between successive macroblocks. However, it is often the case that they provide a reasonable approximation of such inter-frame object motion [10]. It is important to note that not all H.264 encodings are created equally. Lower quality settings for such video codecs often yield poor motion vector representations while achieving their goal of minimizing MAD in less search time or fewer vectors, if any. As such, the scope of motion vector encoding's applicability to optical flow approximation remains limited to only higher quality encodings.

Despite downstream applications getting these artifacts/features for free since the are pre-computed at encoding time, very few applications make use of them. When taking into account the quality considerations, H.264 motion vectors allow for extremely fast optical flow approximations.

3.2. Optimistic Sparse Detection Propagation

The objective of optimistic sparse detection propagation is to accept a prior set of detections and perturb them according to the sparse detection propagation function (SDPF) $\mathcal{W}$ which, similar to Zhu *et Al.* takes a 2D flow field as input and rather than passing a set of features, our function accepts a prior set of object detection bounding-boxes $\mathcal{D}_i$. This difference allows our approach to be entirely model agnostic, and thus is not tied to one particular object detector.

$$\mathcal{D}_k = \mathcal{W}(\mathcal{D}_i, M_{i \to k}) \tag{2}$$

The SDPF iterates over each detection d_j in the set $\mathcal{D}_i$ and applies an aggregation function ϕ to the enclosed flow vectors m_{uv}, resulting in a net flow vector to perturb the detection with. Each 2D flow field is stored in a temporary buffer $\mathcal{B}$.

$$d_k = \phi(d_j, m_{uv}) \tag{3}$$

In practice, the aggregation function is simply the mean or median in x and y, however more complicate aggregation functions that weight areas of the detection more than others can be considered. Figure 1 depicts the role of the aggregation function in propagating detections from the current frame to a consecutive frame.

The SDPF requires a starting set of detections to propagate forward through the video. This particular set is known as the prior detection set, which is an estimate provided from a key-frame inference. However, rather than evoke a computationally expensive and blocking DNN inference at a key-frame, this inference is computed in parallel. The object detector runs in parallel with another worker which simply iteratively applies an SPDF following equation 2 to the existing detection set at frame i.

Since there is no waiting for object detection to complete before proceeding, the process which iteratively applies the SDPF works several frames ahead of the object detector before receiving the computed detections. As such there is a discrepancy of several frames between the current set of detections and the returned detections from the object detector. However since the flow fields have been retained every frame in the flow vector buffer $\mathcal{B}$, the detections received from the inferencing process are propagated forward through iteratively applying the SDPF on said buffer of flow vectors, in order to update the prior. At the end of this update the detections at the current frame incorporate the computed detection information from the DNN worker. Articulated another way, when new information is returned for a frame that has already passed, the stored flow fields are used to re-propagate the new detection set forward to the current frame. The buffer is emptied in this update to allow for new flow vectors to be added. This process is outlined in the pseudocode algorithms 1 and 2 in the appendix.

Since there is no scheduling for key-frame prior updates based on elapsed time or frame index, existing detections will continue to be propagated forward in time until the prior is updated with new information from the object detector. As such, the object detector latency does not directly contribute to the computation time of predicting detections

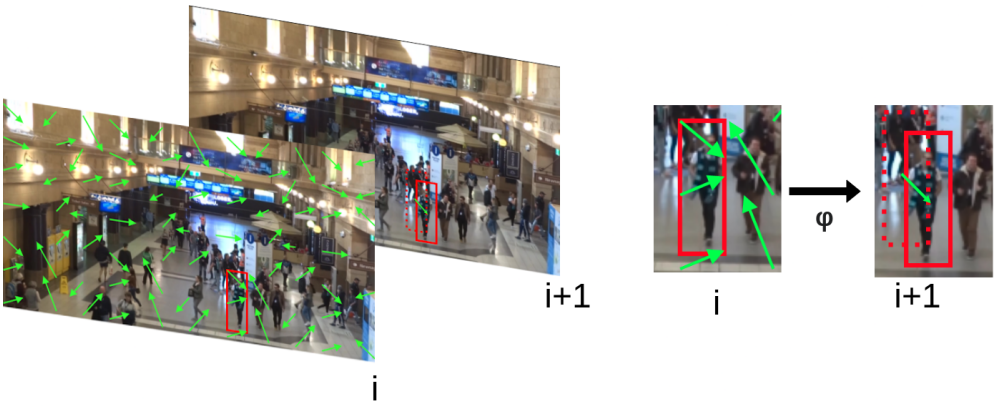

Figure 1. The only motion vectors considered in the source frame i are those which fall in the area of the bounding box. The median perturbation of those motion vectors is computed and applied to the source bounding box in order to predict the bounding box in frame $i + 1$.

at a frame i. However as the object detector latency increases, more frames will have passed during the elapsed computation time and thus will fill the motion vector buffer $\mathcal{B}$ to a greater capacity. As $\mathcal{B}$ fills with more frame data, the cost of a prior update becomes larger due to the number of frames for which detections need to be propagated forward to arrive back at the current frame i.

As the object detector latency increases, the interplay between updating detections based on image content versus updating based on flow vectors becomes apparent. New detection targets can only be detected with the object detector and thus higher latencies will ultimately determine performance in applications that have targets which enter and exit the image frame quickly.

4. Experiments

In order to evaluate the capabilities of this approach, two critical metrics were considered: average-precision (AP) and detection latency. The dataset used to evaluate these metrics was the MOT20 dataset [3]. The reason for using this dataset is because the data in this case is taken directly from video and maintains the temporal context between images.

All evaluations were conducted on an Intel i7-8700K CPU 3.70GHz with Nvidia GTX 1080Ti 12GB. Tests marked with "CPU" were evaluated solely on the CPU without exposing GPU capabilities to the test, otherwise the GPU was used to evaluate.

4.1. H.264 Motion Vectors and FlowNet2.0

Examining the performance results in Table 1, the latency of the original Faster R-CNN model is reduced by a factor of $10.3\times$ when run with MOVEX using the H.264 MVs. However, when using FlowNet2 as the source for the motion vectors, performance is greatly impacted, presenting a $2.0\times$ increase in latency. The FlowNet2-s model was used to compute the optical flow which claimed to have a runtime of approximately $7ms$ on a GTX 1080Ti [6], however performance when running it was no where near this as model forward passes were routinely reaching $100ms$. The AP differences between the three evaluated Faster R-CNN models demonstrates that the use of MOVEX decreases the AP of the baseline model by approximately 0.007 AP when using H.264 MVs but only a decrease of 0.006 AP when using the FlowNet2.0 model.

4.2. Hi-Resolution Versus Low-Resolution

The effect of increasing input resolution for CNN object detectors is known to increase their accuracy. Shown in Table 1, YOLOv4 trained on the COCO dataset [7] is compared against itself at two different resolutions 416×416 versus 960×960, resulting in APs of 0.261 and 0.402 respectively. This confirms the relationship between input resolution and accuracy, but also demonstrates an opportunity for the MOVEX augmentation. Consider that when using the higher resolution model with MOVEX, the AP drops by a mere 0.002 yet the latency falls below that of the original low resolution model. A latency decrease of $7.18\times$ compared to the original high resolution model.

4.3. CPU Versus GPU

Continuing in the vein of accelerating typically high latency object detectors, consider CPU versus GPU object detection latency. It is well known by practitioners that hardware accelerators such as GPUs or TPUs are needed to achieve low latency computation with CNNs. This point is further articulated by the latency data point given by running YOLOv4 with an input resolution of 416×416 on a CPU. This yields a latency of $190.41ms$, which is far too

Method	Avg Latency (*ms*) ↓	AP ↑
FRCNN [3]	131.52	63.0
FRCNN [3] w/ MOVEX + FlowNet2	263.25	62.4
FRCNN [3] w/ MOVEX + H.264 MVs	12.79	62.3
YOLOv4 [1] (960x960)	79.46	40.2
YOLOv4 [1] (960x960) w/ MOVEX + H.264 MVs	11.06	39.8
YOLOv4 [1] (416x416)	26.34	26.1
YOLOv4 [1] (416x416) on CPU	190.41	26.1
YOLOv4 [1] (416x416) on CPU w/ MOVEX + H.264 MVs	7.53	25.2

Table 1. Evaluation of MOVEX with H.264 MVs or FlowNet2 optical flow against baseline Faster R-CNN model used for MOT20 public detections without augmentation [3]. As expected, FlowNet2 flow vectors are more accurate than the approximation provided by the H.264 motion vectors, however this better accuracy comes at a cost of high inference latency. Varying the input resolutions of two YOLOv4 models [1] trained on the COCO dataset [7] demonstrates the accuracy gains possible without sacrificing inference latency.

large for any real-time application. When using MOVEX in conjunction with this model however, the latency falls lower than the original GPU computation latency, resulting in a latency reduction by $25.29\times$ and falling 0.009 AP.

Such latencies for large model such as YOLOv4 on a CPU have yet to be achieved with existing inference acceleration methodologies. GPU computational resources are orders of magnitude more expensive than standard CPU based systems. Employing MOVEX in systems looking to perform object detection on video data would lead to large cost savings by switching from GPU to CPU focused computing. Furthermore, emerging applications in edge computing where cost, space, and computing capabilities are typically limited would greatly benefit from using this technique since modern GPU centered computing often clashes directly with these constraints.

5. Conclusion

We presented MOVEX, a technique that can be applied to an arbitrary off-the-shelf object detector and reduce its inference latency on video data by large margins while sacrificing minimal accuracy. We have demonstrated that MOVEX improves performance for existing object detection models, for which, online real-time video object detection would not have been possible prior. Additionally, we have shown that accuracy improvements are possible without sacrificing latency through increasing the resolution of models and using these models with MOVEX. Lastly, MOVEX allows for models typically restricted to the domain of GPU or TPU computing, due to latency concerns, to expand to less expensive CPU devices.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 1, 4

[2] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 379–387, Red Hook, NY, USA, 2016. Curran Associates Inc. 2

[3] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes, 2020. 3, 4

[4] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. 1

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[6] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017. 1, 3

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 3, 4

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 1

[9] Joint Video Team. Draft itu-t recommendation and final draft international standard of joint video specificaton. *ITU-T Rec. H.264*, 2003. 2

[10] Wonsang You, Houari Sabirin, and Munchurl Kim. Moving object tracking in h.264/avc bitstream. volume 4577, pages 483–492, 06 2007. 2

[11] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4141–4150, 2017. 1, 2

6. Appendix

input: $image$ of resolution $w \times h$
begin
  img $\leftarrow preprocess\_image(img)$;
  detections $\leftarrow object\_detection(img)$;
  return $detections$
end

Algorithm 1: Object detection remote procedure call (RPC) psuedocode.

input: N length sequence of $images$ of resolution $w \times h$
begin
  m $\leftarrow 1$;
  $send\_img\_to\_object\_detection\_RPC(img_m)$;
  detections$_m \leftarrow wait_for_detections$;
  initialize MV buffer $\mathcal{B}$
  for $i \leftarrow 2$ **to** N **do**
    add $MV_{i \rightarrow i+1}$ to $\mathcal{B}$;
    if $detections\_received\_from\_RPC$ **then**
      prior$_m \leftarrow$ RPC_detections;
      for m **to** i **do**
        apply MV_m from $\mathcal{B}_m$ to $prior_m$ using $\mathcal{W}$
      end
      m $\leftarrow$ i;
      empty MV buffer $\mathcal{B}$;
      detections$_i \leftarrow prior_m$;
      $send\_img\_to\_object\_detection\_RPC(img_i)$;
    else
      detections$_i$ $\leftarrow \mathcal{W}(prior_m, MV_{m \rightarrow m+1})$;
    end
  end
end

Algorithm 2: SDPF worker process psuedocode.