Real-time Spatio-Temporal Action Localization in 360 Videos

Bo Chen*, Ahmed Ali-Eldin†, Prashant Shenoy†, Klara Nahrstedt*
*University of Illinois at Urbana-Champaign
†University of Massachusetts Amherst

***Abstract*—Spatio-temporal action localization of human actions in a video has been a popular topic over the past few years. It tries to localize the bounding boxes, the time span and the class of one action, which summarizes information in the video and helps humans understand it. Though many approaches have been proposed to solve this problem, these efforts have only focused on perspective videos. Unfortunately, perspective videos only cover a small field-of-view (FOV), which limits the capability of action localization. In this paper, we develop a comprehensive approach to real-time spatio-temporal localization that can be used to detect actions in 360 videos. We create two datasets named UCF-101-24-360 and JHMDB-21-360 for our evaluation. Our experiments show that our method consistently outperforms other competing approaches and achieves a real-time processing speed of $15fps$ for 360 videos.**

I. Introduction

Spatio-temporal action localization of human actions in videos is a challenging task. It involves finding the bounding boxes belonging to an action, linking bounding boxes across frames together into action tubes, and predicting the class of it. Though many researchers have proposed systems to solve this problem [1]–[3], these systems only work for perspective cameras, which can only capture a limited field-of-view (FOV) and causes actions not in the FOV to be missed. Consider the case of performing action localization to find suspicious activities in a surveillance video. Missing detection of suspicious activities due to a limited FOV can cause serious omissions. To deal with this limitation, we propose to perform spatio-temporal action localization in 360 videos. The 360 video, constructed by stitching images from multiple cameras into equi-rectangular frames, captures actions in all directions in its line-of-sight and overcomes the limited FOV problem of perspective videos.

However, there are many challenges in performing real-time spatio-temporal action localization in 360 videos. First, the objects in the equi-rectangular frame are distorted depending on their locations in the frame, which makes it difficult for standard methods like the CNNs to extract image features. Second, running action localization in 360 videos in real-time is very computation intensive in a real-life setting where we do not have powerful GPUs everywhere.

In this paper, we design a real-time action localization system for 360 videos that addresses the above challenges. Our approach takes advantage of the speed and accuracy of SSD to perform frame-level detection. We experimentally show that, when combined with Feature Pyramid Networks (FPN), SSD has a better mean average precision (mAP) when compared to state-of-the-art neural network architectures such as VGG, SphereNet [4], KTN [5] and Cube Padding [6]. To generate action tubes from frame-level detection, we adapt the online tube linking algorithm in [3] tuning it for higher tube linking speeds. Since there is a discrepancy of the speed of frame-level detection and tube linking on the same device (the linking speed is much slower than the detection speed), we proposea way to split the processing on the edge device and the cloud.

We implement a prototype of our system and evaluate its efficacy via detailed experimental evaluation. In our evaluation, we compare our system with [3], the first work on online spatio-temporal action localisation for perspective videos, and variants of our system that are using state-of-the-art techniques, i.e., SphereNet, Kernel Transformer Networks, Cube Padding, and MobileNet [7]. By using two datasets transformed from UCF-101-24 [8] and JHMDB-21 [9], we demonstrate that the action can be predicted with a good accuracy at an early stage and the prediction accuracy improves as more frames are observed. We also show that our system achieves the best mAP among all competing approaches and real-time performance with a Jetson Nano at the edge and a RTX 2060 GPU on the cloud.

In summary, this paper makes the following contributions.

- We design a real-time spatio-temporal action localization system for 360 videos that works in an edge-cloud setting.
- We create two datasets called UCF-101-24-360 and JHMDB-21-360 for 360 video spatio-temporal action localization.
- We evaluate our system against baseline systems that use state-of-the-art techniques on UCF-101-24-360 and JHMDB-21-360, and demonstrate that our system achieves the best mAP and real-time performance.

In the remainder of this paper, we first review the related work in Section II. We then describe our system design in Section III. After that, we evaluate our system in Section IV, which is followed by our conclusions in Section V.

II. Related Work

A. Neural Network for 360 images

CNNs are widely used to extract features from perspective images. However, they are not suitable for 360 images since the image content in the equi-rectangular format are distorted. Several approaches [4], [5], [10] have been proposed to

mitigate the distortion in 360 images by adapting the neural network architectures. [6], [11] tries to solve this problem by using cube map projection, which map the equi-rectangular frame to the six faces of a cube and then apply standard CNNs on them. In this paper, we experimentally demonstrate that the above approaches are not as effective in action localization as they are in simpler tasks, and our proposed FPN-SSD architecture consistently outperforms them.

B. Spatio-temporal Action Localization

Spatio-temporal action localization approaches can be categorized into unsupervised methods [12], [13] and supervised methods [1], [2]. For unsupervised methods, they do not assume bounding boxes annotations in videos. [13] exploits images downloaded from the internet using text-based queries to localize actions in images. For supervised methods, [1], [2] perform frame-level action detection based on Region Proposal Networks (RPN) [14] and faster R-CNN [15]. Unfortunately, they cannot provide real-time performance as our approach. For real-time spatio-temporal action localization, [3] is the first work to address this problem. It exploits the SSD detection framework for frame-level detection and builds action tubes incrementally. However, their approach only works for perspective videos and makes a too strong assumption of computing device. Instead, our approach is for 360 videos and considers a common edge-cloud setup for action localization.

III. System Design

Our system architecture is illustrated in Figure 1. At a high level, the 360 camera sends 360 images to the edge device via a USB cable. The edge device performs frame-level detection and sends the frame-level detection result, i.e., location and labels of bounding boxes, to the cloud server. The cloud server performs tube linking based on the frame-level detection result and produces action tubes.

A. Frame-level Detection

Frame-level detection exploits object detection methods to find the bounding boxes involved in an action. The most common approach towards this goal is to use CNNs to predict the location and the class label of bounding boxes. Therefore, we follow the SSD architecture in [3] to perform action localization. However, the feature extractor in [3], VGG16, does not work well on 360 images and produces low mAPs.

Inspired by [16], which demonstrates that the Feature Pyramid Network (FPN) is a good generic feature extractor, we replace VGG16 with FPN50 [16], and design a neural network architecture named FPN-SSD. This new architecture works quite well and achieves a frame-level mAP of 0.356, which is significantly better than that of VGG-SSD. To further improve the performance of FPN-SSD, we consider several state-of-the-art techniques like SphereNet [4], Kernel Transformer Networks (KTN) [5] and Cube Padding [6] to mitigate distortion in equi-rectangular images. We also test a lightweight version of FPN-SSD using MobileNet [7]. These models are presented below.

FPN-SSD-SphNet model replaces the standard CNN in FPN-SSD with spherical convolution in [4], where convolution kernels have different sampling positions compared with the standard CNN.

FPN-SSD-KTN model adds the KTN to the standard CNN in FPN-SSD, which adjusts the size of convolution kernel based on the position of kernel on the equi-rectangular frame.

FPN-SSD-Cube model maps the equi-rectangular frame to six faces of a cube before performing convolution.

FPN-SSD-Mobile model replaces the standard CNN in FPN-SSD with the MobileNet convolution.

The fmAP result of different variants of our model is presented in Table I. Our approach has the best fmAP compared with other neural network architectures we tested.

B. Tube Linking

Tube linking involves grouping bounding boxes in different frames together into action tubes. It is challenging because tube linking has to take into account the location change of bounding boxes and False Positives/False Negatives in the bounding boxes detection. In our system, we adopt the online tube generation approach in [3] based on an online Viterbi algorithm, but uses lower values of the threshold for better real-time performance.

C. Computation Placement

A critical question is where frame-level detection and tube linking should be performed. We cannot simply put both of them on the cloud or the edge due to the discrepancy of speed as mentioned in the previous section. Specifically, we consider the following three metrics to optimize the computation placement:

- the bandwidth usage: the size of data that needs to be transmitted over the Internet per frame,
- the image processing overhead: the maximum number of frames that can be processed for frame-level detection/tube linking on edge/cloud per second, and
- localization performance: fmAP of frame-level detection.

There are four placement options: 1) Edge-only: frame-level detection and tube linking are performed on the edge, 2) Edge-cloud: frame-level detection is performed on the edge and the bounding box information is sent to the cloud to be processed, 3) Cloud-only: frame-level detection and tube linking are performed on the cloud with raw 360 images sent from the edge to the cloud, and 4) Cloud-only (W/ c): frame-level detection and tube linking are performed on the cloud with compressed 360 images sent from the edge to the cloud.

In Table II, we compare the placement options based on three metrics we discussed above. For the **bandwidth usage**, both edge-only and edge-cloud have a low usage since the size of information of bounding boxes or action tubes in each frame is small. For the **image processing overhead**, the frame-level detection speed on the edge is 22.7 fps and that on the cloud is $152.8fps$, both of which can achieve real-time performance. Regarding the **localization performance**, Edge-only, Edge-cloud, and Cloud-only have the same fmAP of 35.6%, which

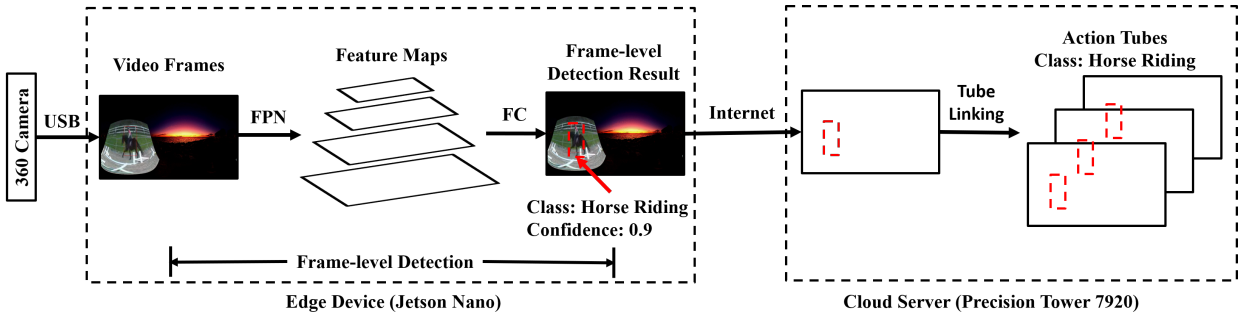


Fig. 1: System Architecture

TABLE I: Frame-level mean average precision (fmAP) of different NN architectures

Arch	FPN-SSD	VGG-SSD	FPN-SSD-Mobile	FPN-SSD-Cube	FPN-SSD-SphNet	FPN-SSD-KTN
fmAP(%)	35.6	14.6	0.0148	4.21	6.29	6.27

is higher than that of Cloud-only (W/ c) since there is no information loss in the input image. As a result, we decide to adopt the option of Edge-cloud, which has a good performance on the network usage, processing speed and fmAP.

IV. Evaluation

Our evaluation focuses on 1) early detection of actions, 2) spatio-temporal localization of actions, and 3) processing speed. According to the fmAP result in Table I, only VGG-SSD has competitive frame-level detection performance against our approach. Therefore, we only present results of our approach and VGG-SSD in this section.

Dataset. Since there are no existing large scale 360 video datasets that contain annotations of action tubes, we evaluate our approach on two datasets named UCF-101-24-360 and JHMDB-21-360 that are converted from UCF-101-24 [8] and JHMDB-21 [9], two datasets for action localization in perspective videos using the back-projection method [4].

UCF-101-24-360 has 3194 videos divided into training and testing sets according to split 1. Each video contains a single action class out of 24 total classes and there is one or multiple action instances of the same action class in one video. Every action comes with spatio-temporal localization annotation, i.e., the bounding box location in every frame.

JHMDB-21-360 has 928 videos divided into training and testing sets according to split 1. Each video contains one action instance out of the 21 action classes and each action has spatio-temporal localisation annotations.

Evaluation metrics. For real-time action localization, we use the mAP (mean average precision) criterion defined in the PASCAL VOC 2012 competition [17]. Note that the true positive (TP) in [17] is the bounding box while the TP in our case is the action tube. In addition to the mAP, we also use the metric of accuracy, i.e., the total number of TPs divided by the sum of the total number of TPs and FPs, to evaluate the correctness of action prediction.

Experimental setup. We use the NVIDIA Jetson Nano, a low-power edge GPU device, as the edge device and a Precision Tower 7920 with the RTX 2060 GPU as the cloud server.

A. Early Action Prediction

In this subsection, we present how our system predicts the action class of a video when only a portion of video is observed. Specifically, we have 10 different timestamps along each video, starting at 10% of the total number of video frames and with a step size of 10%. Figure 2 presents the accuracy of our approach and VGG-SSD at each timestamp. Our approach clearly outperforms VGG-SSD [3] and predicts the action class with a good accuracy when only a small portion of videos is observed.

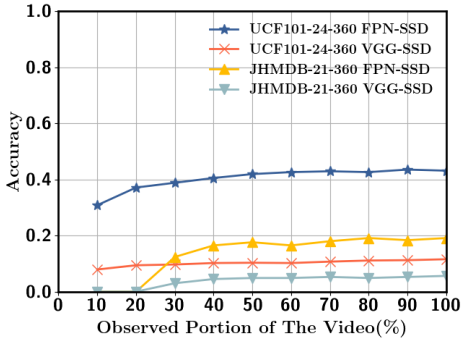


Fig. 2: Early action label prediction results (accuracy %)

B. Spatio-temporal Action Localization

In this subsection, we report the spatio-temporal action localization result of our system compared with the baselines. Examples of localized actions (basketball, cliff diving, fencing and horse riding) are shown in Figure 3.

Simliar to [3], we report the mAP when the STIoU threshold is set to 0.2, 0.5, 0.75, and the mAP averaged over thresholds from 0.5 to 0.95 in steps of 0.05.

Results on UCF101-24-360 are reported in Table III. Our approach achieves an mAP of 0.241 when the STIoU threshold is 0.2 which is significantly higher than that of VGG-SSD. For a higher STIoU threshold of 0.5, FPN-SSD achieves an mAP of 0.149 while that of VGG-SSD is 0.0283. Our approach outperforms VGG-SSD by a large margin. For other STIoU thresholds, the mAP of our approach is higher than that of VGG-SSD in most cases.

Results on JHMDB-21-360 are reported in Table IV. We

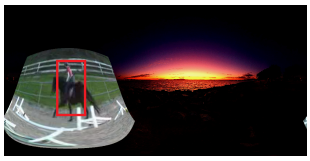
TABLE II: Comparing placement options

Option	Edge-only	Edge-cloud	Cloud-only	Cloud-only (W/ c)
Bandwidth usage (KB)	<1	<1	6441	190.5
Frame-level detection (fps)	22.7	22.7	153.8	153.8
Tube linking (fps)	3.55	48.1	48.1	48.1
Detection performance (%)	35.6	35.6	35.6	12.6

(a) Basketball

(b) Cliff Diving

(c) Fencing



(d) Horse Riding

Fig. 3: Examples of Frame-level Detection

TABLE III: Spatio-temporal action localisation results (mAP) on UCF101-24 dataset in split1

IoU threshold δ	0.2	0.5	0.75	0.5:0.95
FPN-SSD	0.241	0.149	0.0584	0.0638
VGG-SSD	0.0526	0.0283	0.108	0.0119

can still find that our approach is better than VGG-SSD for most STIoU thresholds.

TABLE IV: Spatio-temporal action localisation results (mAP) on JHMDB-21 dataset in split1

IoU threshold δ	0.2	0.5	0.75	0.5:0.95
FPN-SSD	0.101	0.0733	0.00183	0.0179
VGG-SSD	0.0336	0.0278	0.00498	0.00826

C. Processing Speed

To demonstrate that our system can perform real-time action localization, we test the speed of our system running in an edge-cloud setting. For frame-level detection, we measure the average time spent on detecting bounding boxes of each frame to be $0.044s$. For tube linking, we measure the time spent on tube linking divided by the number of frames in each tube to be $0.021s$. Thus, the average time spent on each frame is $t = 0.044 + 0.021 = 0.065s$, which results in a frame rate of roughly $15fps$.

V. Conclusion

We present a framework for real-time action localization in 360 videos. We take advantage of SSD and FPN for detecting bounding boxes of actions in 360 videos. The processing is split in an edge-cloud setting based on extensive analysis of the trade-offs. We compare our approach to state-of-the-art approaches and show that our approach achieves the best mAP for most spatio-temporal IoU thresholds. Besides, our approach achieves real-time performance of roughly $15fps$.

Acknowledgement

This work was partially supported by the US Army Research Laboratory under cooperative agreement W911NF17-2-0196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the US government.

References

[1] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin, "Deep learning for detecting multiple space-time action tubes in videos," *arXiv preprint arXiv:1608.01529*, 2016.

[2] X. Peng and C. Schmid, "Multi-region two-stream r-cnn for action detection," in *European conference on computer vision*. Springer, 2016, pp. 744–759.

[3] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin, "Online real-time multiple spatiotemporal action localisation and prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3637–3646.

[4] B. Coors, A. Paul Condurache, and A. Geiger, "Spherenet: Learning spherical representations for detection and classification in omnidirectional images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 518–533.

[5] Y.-C. Su and K. Grauman, "Kernel transformer networks for compact spherical convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9442–9451.

[6] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, T.-L. Liu, and M. Sun, "Cube padding for weakly-supervised saliency prediction in 360 videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1420–1429.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[8] K. Soomro, A. R. Zamir, and M. Shah, "A dataset of 101 human action classes from videos in the wild," *Center for Research in Computer Vision*, vol. 2, 2012.

[9] H. Jhuang, H. Garrote, E. Poggio, T. Serre, and T. Hmdb, "A large video database for human motion recognition," in *Proc. of IEEE International Conference on Computer Vision*, vol. 4, no. 5, 2011, p. 6.

[10] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360 videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 488–503.

[11] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, "Salnet360: Saliency maps for omni-directional images with cnn," *Signal Processing: Image Communication*, vol. 69, pp. 26–34, 2018.

[12] M. Sapienza, F. Cuzzolin, and P. H. Torr, "Learning discriminative space–time action parts from weakly labelled videos," *International journal of computer vision*, vol. 110, no. 1, pp. 30–47, 2014.

[13] W. Sultani and M. Shah, "What if we do not have multiple videos of the same action?–video action localization using web images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1077–1085.

[14] 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.

[15] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.