2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

# Tracking Fast Moving Objects by Segmentation Network

Aleš Zita
The Czech Academy of Sciences
Institute of Information Theory and Automation
Pod Vodárenskou věží 4
Email: http://www.utia.cas.cz/people/zita

Filip Šroubek
The Czech Academy of Sciences
Institute of Information Theory and Automation
Pod Vodárenskou věží 4
Email: http://www.utia.cas.cz/people/sroubek

***Abstract*—Tracking Fast Moving Objects (FMO), which appear as blurred streaks in video sequences, is a difficult task for standard trackers, as the object position does not overlap in consecutive video frames and texture information of the objects is blurred. Up-to-date approaches tuned for this task are based on background subtraction with a static background and slow deblurring algorithms. In this article, we present a tracking-by-segmentation approach implemented using modern deep learning methods that perform near real-time tracking on real-world video sequences. We have developed a physically plausible FMO sequence generator to be a robust foundation for our training pipeline and demonstrate straightforward network adaptation for different FMO scenarios with varying foreground.**

## I. Introduction

Object tracking is a well-explored field of computer vision. The majority of object tracking algorithms starting from basic correlation trackers up to state-of-the-art deep network trackers utilize texture-based correlation or feature-based methods. Modern video capturing devices with built-in processing algorithms are capable of producing sharp images of moving objects. Moreover, the person capturing the object in motion typically tracks the moving object, hence it predominantly stays in the center of the image and in-focus. For such tasks, the correlation-based trackers are sufficient.

The situation changes dramatically when an object moves so fast, that it appears blurry on individual video frames. Such object in motion is called *'FMO'*, short for *Fast Moving Object* [1], and is loosely defined as an object traveling a distance larger than its diameter within one frame of the video sequence (Figure 1). The inter-frame object overlap is negligible, which causes problems to many conventional trackers.

A typical manifestation of the FMO in video frames is a prolonged streak without any particular texture, colored with the object prevailing color, or a combination of object colors; see Figure 1. The lack of any sharp texture of the object renders most of the texture-based correlation trackers inapplicable. Situation is even worse for very small objects moving fast relative to their sizes, such as ping-pong or squash balls.

The first tracking algorithm specifically designed for FMOs uses a method based on background subtraction [1]. This technique requires a static background, static camera, and large prominent foregrounds. It is also prone to object miss-tracking, which then requires a time-consuming object re-detection.

More recent approaches deal with the problem of FMO tracking by running a de-blurring algorithm [2], [3], [4]. These methods perform considerably better, but are extremely slow, as they require a full-blown de-blurring optimization pipeline. Therefore, they are not suitable for real-time video stream processing.

Our primary goal is to provide a method operating in real-world scenarios such as tracking of ping pong, squash balls, badminton shuttlecocks, and similar objects. This problem is very specific in sense, that the objects that need to be detected are very small and move very fast. This means that existing state-of-the-art tracking, object detection, or segmentation methods can not be used directly. Figure 2 shows the results from state-of-the-art semantic segmentation tool DeepLab3+ [5].

In this work we demonstrate that FMO tracking can be successfully solved with a machine learning approach. The proposed method uses a segmentation convolutional neural network (CNN) with real-time performance in videos with a resolution around 320x240. Network architecture is based on state-of-the-art tracking by segmentation methods rather than on object detection networks. We experimentally prove that tracking by segmentation outperforms tracking by correlation. In addition, we propose an on-demand synthetic FMO data generator to tackle the problem of producing annotated data automatically. Even though the network is trained solely on synthetic data, it can successfully be used in real-life applications, like processing YouTube sport video sequences. The proposed solution focuses predominantly on small bright foregrounds, yet we demonstrate the possibility of fast model fine-tuning for different foreground types.

Resulting segmentation can be further used for trajectory prediction and down the pipeline even for the trajectory estimation in de-blurring algorithms.

The method is evaluated on the FMO dataset [1] on which it shows competitive results. and investigate cases where the proposed algorithm outperforms or under-performs current methods both in precision and execution time.

978-1-7281-8808-9/20/$31.00 ©2020 IEEE


Fig. 1: Examples of Fast Moving Objects in real-world videos.

## II. Related Work

**Video Object Tracking**

Object tracking is a well-established field of research in computer vision. Many methods have been proposed for tracking single or multiple objects in video sequences. Namely tracking by detection [6], [7], tracking by features [8], [9] tracking by correlation [10] and others. All of the approaches mentioned are based on either object detection using texture information of the tracked object or features extracted from it. This assumes that the object image contains some minimum level of details. Also, many of the conventional trackers perform best when the tracked object bounding boxes largely overlap in the consecutive frames. Both of the mentioned assumptions do not hold in sequences containing a FMO.

**FMO Tracking**

FMO tracking has been attracting the attention of researchers lately. Initial work in this field was done in [1], where the authors introduced the theme and proposed the first tracker based on background subtraction. In the heart of the method lies a tracker capable of tracking the background changes. When the tracker fails a time-consuming re-detection is executed to resume tracking.

Recently, interesting work was done in [2] where the tracking problem was defined as a de-blurring optimization problem. In another similar approach [4], authors show intra-frame tracking capability of the de-blurring approach. Albeit the results are promising in both mentioned publications, these methods focus on videos with static camera and background, and additionally, their algorithm cannot be used in real-time due to the high processor time demands of the optimization algorithm.

**Deep Semantic Segmentation**

There are many deep segmentation methods currently available, mainly based on encoder-decoder architecture. In [11] researchers introduced an interesting approach by using multiple stacked deconvolution blocks. Impressive results were achieved in DeepLabv3+ [12], where authors use the depth-wise separable convolution to both Atrous Spatial Pyramid Pooling and decoder modules to achieve high scores in both the PASCAL VOC 2012 and the Cityscapes datasets. Because the solutions often incorporate very deep networks, many of them have longer inference times [11] or are GPU memory demanding [13].

## III. Method

First we give a brief introduction to the overall framework of the proposed method, then we investigate various strategies and finally, we describe the proposed method in depth.

### A. Overview

The work of [1] inspired us to tackle the problematic cases on which the method performs poorly, namely tracking of very small objects. Larger moving objects in sports videos are often sharp because modern acquisition devices have short exposure and the cameraman actively tracks the object of interest. However for small objects that are moving very fast, typically balls in sports such as tennis, softball, or badminton, this is not true.

After some failed attempts to solve the tracking of small FMOs by conventional means, we have turned our attention towards deep learning methods. Learning-based approaches achieve top results in many segmentation tasks in terms of both computation time and precision. We researched several state-of-the-art segmentation networks and we achieved the best results with u-net-type architecture with inception bottleneck modules called ENet [14]; refer to Figure 3 for more details.

We choose the publicly available FMO dataset as a benchmark, to be able to compare the performance of the proposed

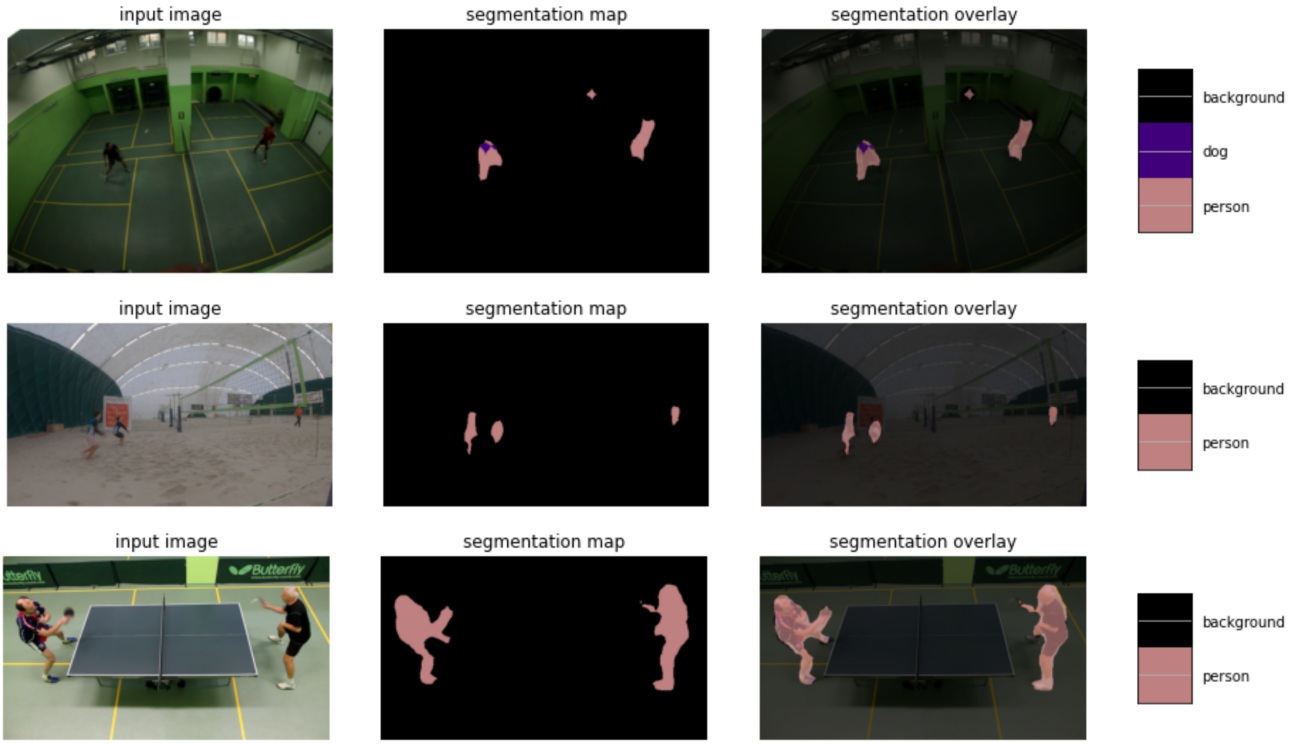


Fig. 2: Examples of FMO images evaluated on publicly available DeepLab3+ [5] semantic segmentation framework. State-of-the-art segmentation methods are not trained for detecting FMOs.

approach with the original method [1]. We perform preprocessing of the dataset in such a way that the size and color of the foreground resembles the foreground used for training.

Since our ultimate goal is tracking in real-world sports videos, we also include YouTube sports videos for performance evaluation. However, they are not annotated and so we provide only visual assessment.

| Network | mAP | mAR | F1 |
|---|---|---|---|
| Faster-RCNN with ResNet-50 | 33.2 | 15.5 | 21.2 |
| ENet | 36.5 | 52.7 | 41.2 |

TABLE I: Performance comparison between Enet segmentation and modified Faster-RCNN network as measured on bounding boxes. Metrics used in the table are standard Pascal mean Average Precision @0.5 and mean Average Recall @0.5

### *B. Network architecture*

To decide the main direction of our research, we tested two current machine learning approaches: Semantic Segmentation and Object Detection.

After running performance and metric assessment tests of several segmentation frameworks, we opted for U-Net architecture called ENet [14] consisting of inception blocks proposed in [15]. The initial choice of this network design was based on the inference speed and performance on our benchmark dataset.

The choice of Object Detection network was based on study published in [16] revealing the Faster-RCNN [17] network with ResNet-50 [18] feature extractor backbone as a well balanced framework in terms of speed and accuracy.

ENet provide binary segmentation masks and RCNN bounding boxes. Both networks were adjusted to facilitate 15-channel input images to be able to process 5-frame sequences, initialized with publicly available weights pre-trained on ImageNet [19] and trained using our synthetic data generator. The performance of both networks was evaluated on the FMO dataset using mean Average Precision and mean Average Recall with Intersection over Union (IoU) threshold set to 0.5. See the performance comparison Table I. To compare both approaches, results of ENet were converted to bounding boxes by calculating axis-aligned rectangles circumscribing connected components in the segmentation masks.

Since ENet outperforms RCNN, we decided to base our approach on semantic segmentation.

The basic idea behind the FMO trace segmentation is training the network to recognize prolonged objects with no apparent texture, typically of white color to resemble most common sports balls. This represented in our opinion the majority of the problematic sports videos.

Single image segmentation, which is the standard input scenario for most of the segmentation methods, performs poorly in detecting FMOs and produces a large number of false positives. The overall measured Precision, Recall and F1 score (as defined in Section IV-B) was 4.3, 4.3 and 3.6, respectively. This is expected, as the proposed network learns to recognize

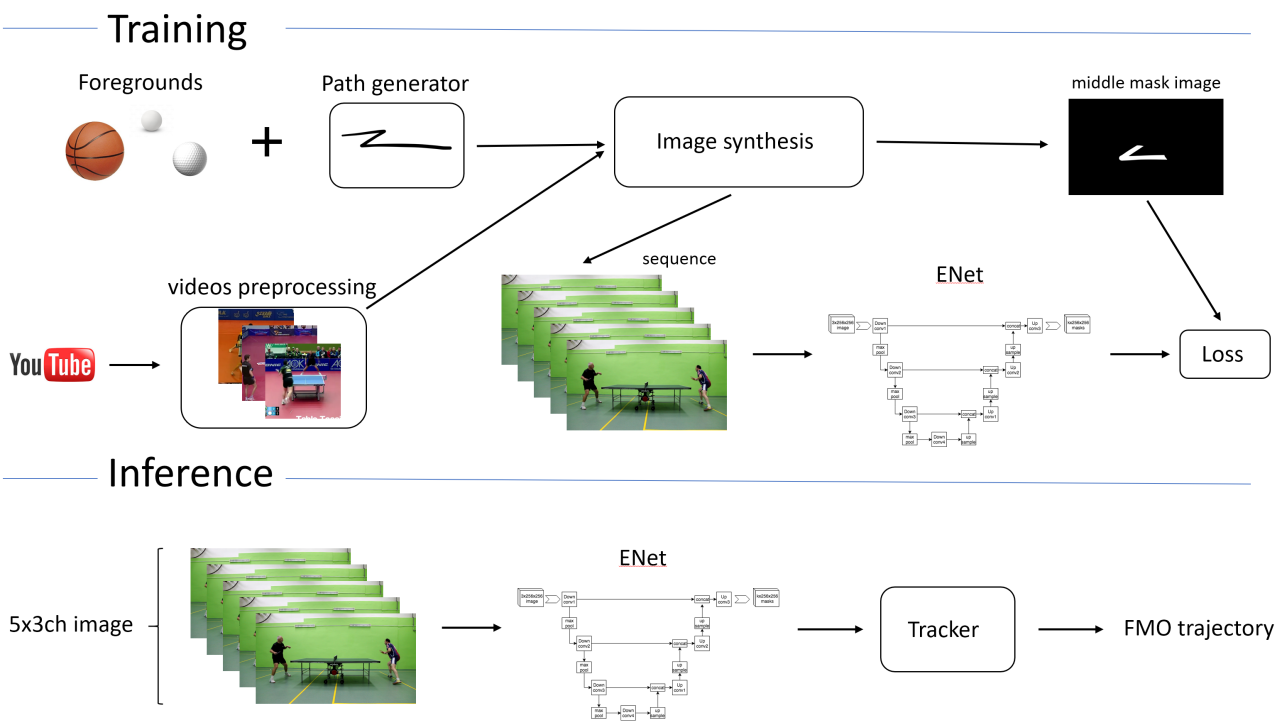


Fig. 3: Processing pipeline: During the training phase (top section) the sequences are dynamically synthesized using pre-processed video sequences, foregrounds, and path generator. Next, the frames are concatenated and input the network as a 15-channel image. During the inference phase (bottom part) the sequences are segmented by the network and Kalman based tracker is used for path prediction.

bright smears and therefore falsely segment any bright spots or lines in the image. In other cases, the model learns to ignore white lines, if they are in scene backgrounds, and does not detect FMOs at all. To overcome these limitations, we propose to use a sequence of consecutive frames as a network input. The idea is that image sequences improve trace consistency in time. We tested several multi-frame approaches, namely three and five frames either concatenated along color channels or as a full 4D input to the 4D network. Even though this approach is mathematically equivalent to the channel concatenation, it can provide faster learning and less false positives. The best results were achieved by using five consecutive video frames concatenated along color channels, i.e. the input to the network is a single 15-channel image; see Figure 3.

In our experiments, the original ENet architecture produced segmentation images with insufficient segment border precision. To address this problem, we have replaced the standard max-pooling with max-pooling-with-argmax and used the argmax values in corresponding upsampling unpooling layers. From this modification, more detailed segmentations were obtained.

The images used for training are synthetic FMO sequences based on real-world sports background images. Because every deep network is only as good as the dataset used for training, we have created a tool for generating synthetic sequences. This approach has proven to be very effective as the system is able to successfully segment fast-moving objects in real-world images, even though the network has never seen any during the training.

The majority of the state-of-the-art deep learning methods heavily depends on re-using the learned parameters from their successful predecessors. In our case, transfer learning led to worse performance. We hypothesize that this is due to the specificity of our task, which cannot exploit learned convolution kernels from other problems based on the extraction of texture features.

### C. *Dataset generator*

Due to the nonexistence of a large annotated FMO dataset for training, we propose our own FMO sequence generator that obeys Newton's laws of motion. First, we collected YouTube sports videos which we used as a background. To eliminate any false fast-moving object from the videos, we have generated sequences of median images. Every frame of such a sequence was calculated as a median of 5 consecutive frames. Next, we created a foreground generator based upon selected ball images from a variety of sports. Finally, we designed a physically plausible generator of trajectories, including random bounces or occlusions; see examples in Figure 5.

In the core of the image synthesizer is a random motion path generator that takes into account a fully simulated camera (including CCD size resolution and aperture properties) as well as motion of the simulated object in space. The generator

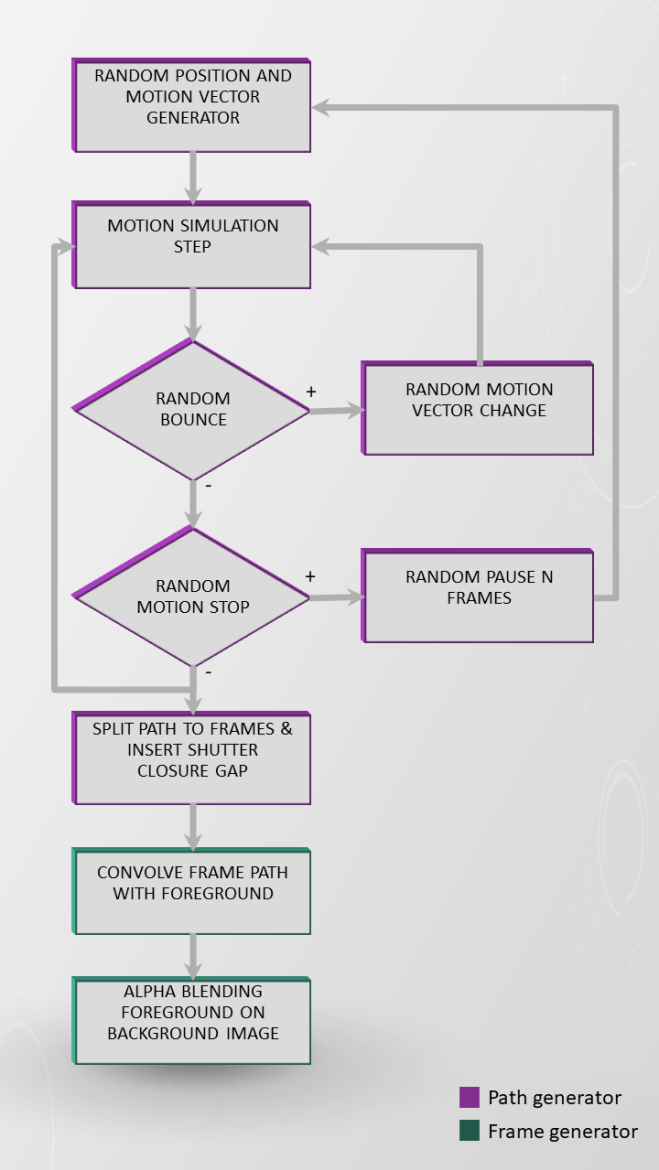


Fig. 4: Generator pipeline of synthetic images. The motion is generated in sub-frame steps, which are consequently split to 5 frames with shutter closure gap emulation. The trajectory for a given frame is convolved with the foreground to form the blurred motion and the resulting foreground image is alpha-blended with the background according to (1).

begins from a random initial speed vector and then iterates in time simulating the motion. For better plausibility, the gravitational acceleration is taken into account too. Sudden velocity changes (e.g. hit from a racket), bounces (from wall, ground or table) occlusions and sudden motion stops are simulated as well. The processing pipeline scheme is depicted in Figure 4.

The generated trajectory is then convolved with the foreground to create the motion trace and finally inserted as a weighted sum into the sequence of background images using the following formula.

$$I_t(x) = [P_t * b_f F](x) + (1 - [P_t * M])B(x), \qquad (1)$$

where $P_t$ is the path PSF normalized to sum to 1, $F(X)$ is the random foreground image, $b_f$ is the overexposure brightness factor (described in next paragraph), $M(x)$ is the foreground indicator function and $B(x)$ is the background image. The used foreground image is created as a random selection of real-world white ball images that are tinted in random bright color and resized to a pre-defined range of foreground sizes.

Another aspect that had been taken into account is fast-moving object overexposure. This is due to the 'HDR' effect of the moving object. The overall brightness of the object in one frame can, and often is, brighter than the maximum brightness point in the rest of the image. Typically what every camera has to solve is the conversion of high brightness range of the world to the quantized 255 brightness values. This is done by several techniques that are out of the scope of this article. This conversion usually includes some form of clipping of the brightness levels which are too high to optimize overall image brightness balance. In a typical image without any FMO the overexposed parts of the image are clipped to the maximum allowed brightness. But, in the case of a fast-moving object, the true brightness of the object when stopped is an integration of its brightness along the object trajectory. In other words, the overall brightness of the object is spread out along the object path so it does not exceed the maximum pixel brightness of any point in the image. Therefore, it is often the case, that the true brightness of the object, when aggregated along the path, exceeds the maximum brightness of the image, especially with the white ball. If this effect would not have be taken into the consideration, the rendered object would seem very dim in the resulting image. This led us to set the factor of absolute brightness of the foreground between 0.8 - 1.4 of the maximum brightness.

As the ground truth mask image used in the training phase, we use the foreground path mask corresponding to the middle frame of the sequence. It is calculated again as a foreground mask convolved with the trajectory corresponding to the middle frame ($[P_3 * M]$); see Figure 3 for illustration.

### *D. Tracking*

On top of the segmentation pipeline, we have implemented a simple tracker. The tracker is responsible for final object trajectory estimation. First, we select the blob which most likely represents the tracking object. This is achieved by simply selecting the largest connected component in the segmentation image. For sequences containing many false positives, more sophisticated logic should be applied. We used a weighted composition of two measures: connected component size and shape. Since we are looking for a prolonged object, we use second central moments of the connected components to estimate the prolongation.

Sequences of the bounding box positions are used by the tracker to extrapolate the object trajectory. For frames with

Fig. 5: Results of FMO synthetic data generator. The rightmost image shows example of small emulated bounce.

missing or too small blobs, we utilize a Kalman filter to estimate the missing trajectory or predicting trajectories in cases the object is lost or occluded. The output of the tracker is a sequence of coordinates representing the estimated object trajectory.

## IV. Experiments

In this section we present performance of the proposed method and compare it to the original work [1]. We focus our attention to real-world applications with both inference speed and accuracy for small ball-like object detection.

### A. Training

Initially, the network was trained on synthetic data with a wide range of foreground parameters. We used the modified ENet described in III using Adam optimizer, learning rate set to 0.01 and exponential learning rate decay; weight decay set to $2e-4$; average cross-entropy loss function; 200k iterations.

During the second stage of the training, the model was fine-tuned using the same architecture on a narrow size range of the synthetic foregrounds. The initial learning rate was lowered to 0.001 and optimizer switched to SGD. The training session ran for 50k iterations.

### B. Evaluation

The proposed method was evaluated on the FMO dataset [1], where it achieved comparable or better results than the previously published method.

The performance criteria correspond to evaluation statistics in the original paper. These are precision TP/(TP + FP), recall TP/(TP + FN) and F1-score 2TP/(2TP + FN + FP), where TP, FP, FN is the number of true positives, false positive and false negatives, respectively. A true positive detection has an intersection over union (IoU) with the ground truth polygon greater than 0.5 and an area larger than other detections. The second condition ensures that multiple detections of the same object generate only one TP. False negatives are FMOs in the ground truth with no associated FP detection.

The results for both the original method and the proposed approach are listed in Table II. We can conclude, that overall mean F1-score is slightly better for our method, as well as mean recall. We avoid significant under-sizing of the resulted segmentation of the FMO trace, which causes high precision values over small recall value. Therefore, we argue that our approach results are more balanced in terms of precision and recall performance metrics.

The performance of the method reflects the purpose of our algorithm. It performs well on sequences with small ball-shaped objects moving fast relative to their size (ping-pong, softball, tennis, and squash); see Figure 6. Poor performance was recorded on sequences with foregrounds different from balls (like darts or archery) and on sequences with low background-foreground contrast (darts window and blue ball). The method performs poorly on data with a larger size-to-velocity ratio (frisbee and volleyball). Even though these sequences are part of the FMO dataset, foregrounds on these sequences are larger and are not moving faster than their diameter, as per FMO definition in Section I.

Our approach is advantageous in the fact that the network can be easily fine-tuned with image synthesizer setup for another sequence type, such as particular background (i.e. tennis tournament), particular foreground (i.e. yellow ball), foregrounds of different sizes, etc. For comparison, we have re-trained the network to detect foregrounds of bigger size and slower motions. The results are in the most right section of Table II. The segmentation network stopped to be sensitive to smaller foregrounds, such as ping pong, squash, or tennis, and starts to perform in cases with larger foregrounds, like frisbee or volleyball.

### C. Computational time

Another benefit of the ENet neural network is the short inference time. The state-of-the-art approaches [2], [3], [4] are based on foreground de-blurring and therefore are inherently slow. In [4] authors state that the mean time is 4 seconds per frame. Our method is capable of near real-time execution while using a widely available graphics card, such as NVIDIA

| | | original work | | | Learning-based I | | | Learning-based II | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | n | Pr. | Rec. | F1 | Pr. | Rec. | F1 | Pr. | Rec. | F1 |
| volleyball | 50 | 100 | 45.5 | 62.5 | 0 | 0 | 0 | 33.3 | 42.9 | 37.5 |
| volleyball passing | 66 | 21.8 | 10.4 | 14.1 | **20** | **16.2** | **17.9** | **85** | **98.1** | **91.1** |
| darts | 75 | 100 | 26.5 | 41.7 | 37 | **62.5** | **46.5** | 33.3 | **100** | **50** |
| darts window | 50 | 25 | 50 | 33.3 | **33.3** | 33.3 | 33.3 | **33.3** | 33.3 | 33.3 |
| softball | 96 | 66.7 | 15.4 | 25 | **83.3** | **83.3** | **83.3** | 54.5 | **66.7** | **60** |
| archery | 119 | 0 | 0 | 0 | **25** | **20** | **22.2** | **18.8** | **100** | **31.6** |
| tennis serve side | 68 | 100 | 58.8 | 74.1 | 66.7 | **76.9** | 71.4 | 35.3 | **85.7** | 50 |
| tennis serve back | 156 | 28.6 | 5.9 | 9.8 | **35.3** | **69.2** | **46.8** | 26.4 | **70** | **38.4** |
| tennis court | 128 | 0 | 0 | 0 | **33.3** | **40.8** | **36.7** | **25.5** | **58** | **35.5** |
| hockey | 350 | 100 | 16.1 | 27.7 | 24.1 | **86.7** | **37.7** | 20 | **91.7** | **32.8** |
| squash | 250 | 0 | 0 | 0 | **26** | **84.4** | **39.7** | **21.6** | **75.9** | **33.6** |
| frisbee | 100 | 100 | 100 | 100 | 0 | 0 | 0 | 94.7 | 94.7 | 94.7 |
| blue ball | 53 | 100 | 52.4 | 68.8 | 40 | 26.7 | 32 | 58.3 | **43.8** | 50 |
| ping pong tampere | 120 | 100 | 88.7 | 94 | 58.6 | 66.7 | 62.4 | 0 | 0 | 0 |
| ping pong side | 445 | 12.1 | 7.3 | 9.1 | **45.4** | **79.1** | **57.7** | 0 | 0 | 0 |
| ping pong top | 350 | 92.6 | 87.8 | 90.1 | 56 | **98.9** | 71.5 | 0 | 0 | 0 |
| Average per frame | 2476 | 53.7 | 31 | 35.5 | 38.3 | **68.5** | **47.2** | 21.7 | **49.7** | 27.8 |

TABLE II: Performance of the original CVPR2017 method [1] in comparison to the proposed method (method I - trained for smaller foregrounds; method II - trained for bigger foregrounds). The results suggest the better overall performance of the trace segmentation in overall F1 performance score for method I.

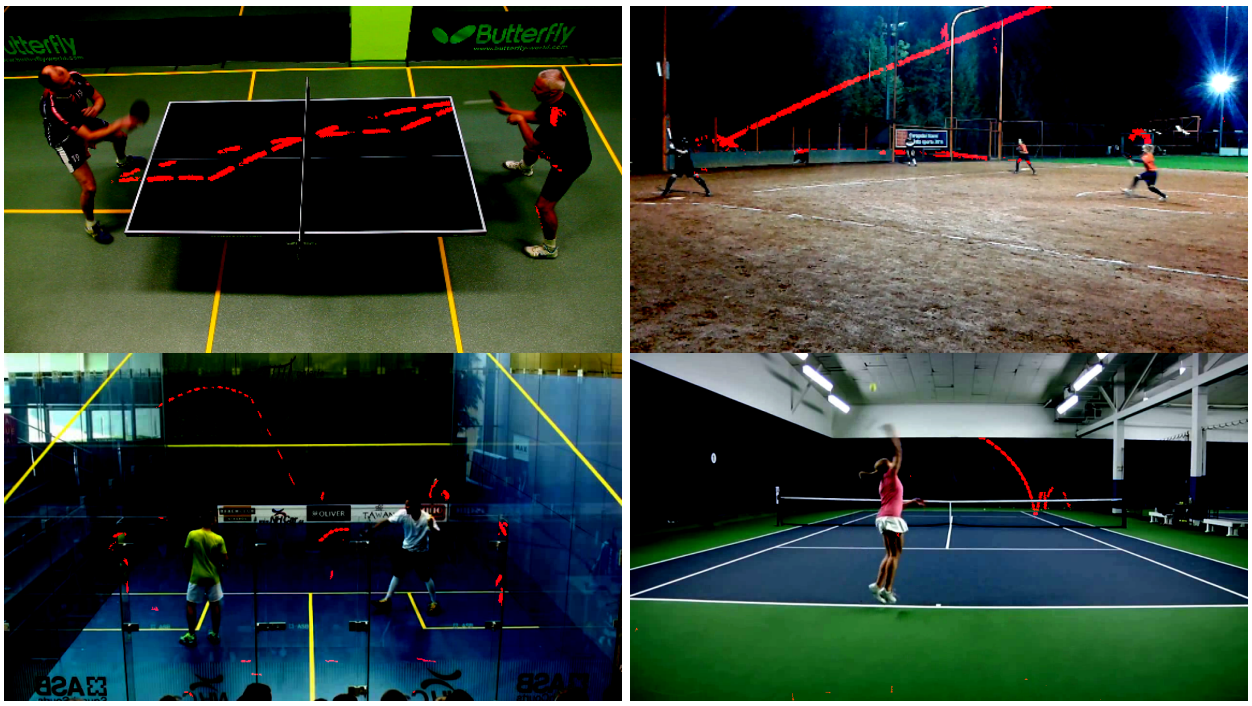


Fig. 6: Examples of segmentation results on FMO dataset. Notice false positives caused by racket or players movements or glass reflections.

| video resolution | average fps |
|---|---|
| **864 x 1536** | 2 |
| **576 x 1024** | 4.7 |
| **430 x 768** | 8.6 |
| **324 x 576** | 11.8 |
| **216 x 384** | 23.1 |

TABLE III: Some examples of video inference times achieved using NVidia Tesla X GPU.

GeForce 2080ti or similar. For more details refer to Table III, where we summarize mean frame evaluation times for NVidia Tesla P100 GPU with different image resolutions.

### *D. YouTube sport videos*

We have created a tool that automatically downloaded more than 900 000 YouTube sports videos to create a base of our synthetic data generator backgrounds. Over 1800 of these sequences contain ping-pong matches, which we used for visual assessment of our framework. Although we measured our performance on the FMO dataset, we also aim for good performance on real wold sequences. Examples of ping-pong sequence evaluation can be seen in Figure 7.

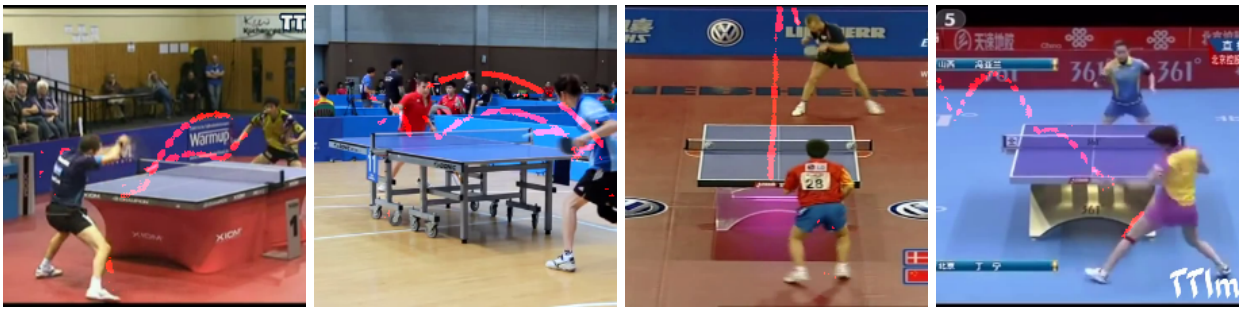

Fig. 7: Images show evaluation examples of YouTube real-world ping pong sequences.

## V. Conclusion

We have implemented a learning-based method that performs near real-time detection and tracking of real-world fast moving objects. The proposed approach overcomes limitations of previous methods in this field, namely the long computation time and difficulty to detect small and very fast objects. We have introduced a synthetic physically plausible fast moving object sequence generator, which we use for network training. The simplicity of adapting the generator to another type of foreground followed by network fine-tuning allows us to detect foregrounds of different sizes and colors.

In the future work, we would like to focus on optimizing the processing pipeline with respect to speed in order to achieve true real-time performance in high-resolution videos and automatically track all kinds of sports balls in video streams. This can be further utilized in various applications such as instantaneous ball speed detection, ball misses, or detection of balls out of bounds.

## Acknowledgment

This work was supported by Czech Science Foundation grant GA18-05360S and by the Praemium Academiae awarded by the Czech Academy of Sciences. We thank Prof. Jiri Matas for valuable comments and suggestions.

## References

[1] D. Rozumnyi, J. Kotera, F. Sroubek, L. Novotny, and J. Matas, "The world of fast moving objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5203–5211.

[2] D. Rozumnyi, J. Kotera, F. Šroubek, and J. Matas, "Non-causal tracking by deblatting," in *German Conference on Pattern Recognition*. Springer, 2019, pp. 122–135.

[3] J. Kotera and F. Šroubek, "Motion estimation and deblurring of fast moving objects," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2860–2864.

[4] J. Kotera, D. Rozumnyi, F. Sroubek, and J. Matas, "Intra-frame object tracking by deblatting," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[6] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2015.

[7] J. Zhang, S. Ma, and S. Sclaroff, "Meem: robust tracking via multiple experts using entropy minimization," in *European conference on computer vision*. Springer, 2014, pp. 188–203.

[8] K.-W. Chen and Y.-P. Hung, "Multi-cue integration for multi-camera tracking," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 145–148.

[9] S. Gladh, M. Danelljan, F. S. Khan, and M. Felsberg, "Deep motion features for visual tracking," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1243–1248.

[10] H. Liu, Q. Hu, B. Li, and Y. Guo, "Long-term object tracking with instance specific proposals," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 1628–1633.

[11] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *IEEE Transactions on Image Processing*, 2019.

[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[13] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 603–612.

[14] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[16] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[17] 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 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.