IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 14, 2021, accepted January 27, 2021, date of publication February 4, 2021, date of current version February 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057172

Online Spatio-Temporal Action Detection in Long-Distance Imaging Affected by the Atmosphere

ELI CHEN[1], OREN HAIK[2], AND YITZHAK YITZHAKY[1]
[1]Department of Electro-Optics Engineering, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Be'er Sheva 84105, Israel
[2]Research and Development Department, HP Inc., Ness Ziona 76101, Israel

Corresponding author: Yitzhak Yitzhaky (ytshak@bgu.ac.il)

ABSTRACT Current state-of-the-art approaches for spatio-temporal action detection deal with stable videos and quite sterilized environments, as seen in the UCF-101 benchmark. In addition, the objects of interest are typically relatively close to the camera, and therefore fairly clear and easily distinguished. This study presents an approach method for online human action detection in long-distance imaging affected by atmospheric distortions. We created a unique dataset of typical actions in long-range imaging. Various CNN frameworks were examined for the initial moving object detection phase, including 2D, 3D, one stream, and two-stream (RGB frames and optical flow). The basic object detection methods examined within these frameworks include the YOLOv3 and an extension of the inflated 3D ConvNet with a Feature-Fused Single Shot Multibox Detector (FFSSD) to improve small object detection. To cope with the harmful effect of the spatio-temporal random movements induced by atmospheric effects on motion estimation, we first fit the optical flow stream characteristics to a temporally noisy turbulent environment. A significant improvement of the action detection quality under such noisy conditions was obtained by constructing an online tracking algorithm that incrementally constructs and labels the objects' tracks from the network's frame-level detections. Experimental results show that our approach outperforms the state-of-the-art on our dataset in terms of the mAP measure.

INDEX TERMS Computer vision, action recognition, machine learning algorithms, atmospheric image distortion, remote sensing.

I. INTRODUCTION

Action detection focuses on classifying the actions present in a video and localizing them in space and time. It is a challenging problem, and it becomes even more difficult in the case of long-distance imaging (at about two kilometers and above) due to the effects of turbulence and aerosols in the air, which become more meaningful as the imaging path length increases [1]. The atmospheric path blurs the objects in the recorded video sequence and also adds random movements into the image of the scene, making it difficult to distinguish real moving objects, particularly when the objects are small. Action localization in long-distance imaging is a high-level vision task that can be very important in applications such as homeland security and the monitoring of large regions.

The associate editor coordinating the review of this manuscript and approving it for publication was Miaohui Wang.

Fig.1 illustrates the motivation for our work. Figs.1(a) and (b) are samples from UCF101 [2], a dataset commonly used for action detection. Fig.1(a) is an RGB image, and Fig. 1(b) represents the optical flow from the sequence. Figs.1 (c) and (d) are samples from our dataset, where (c) is an RGB image, and (d) is the corresponding optical flow. In our case, the object is much smaller and more difficult to detect compared to the samples from UCF101. In addition, the optical flow map is much noisier due to the random movements caused by the air turbulence.

With the rise of Convolutional Neural Networks (CNNs), impressive progress has been made in image classification [3] and object detection [4], motivating researchers to apply CNNs to action classification and detection. Although the resulting CNN-based action detectors [5]–[8] have achieved remarkable results, these methods are computationally expensive, and their detection accuracy is still quite far

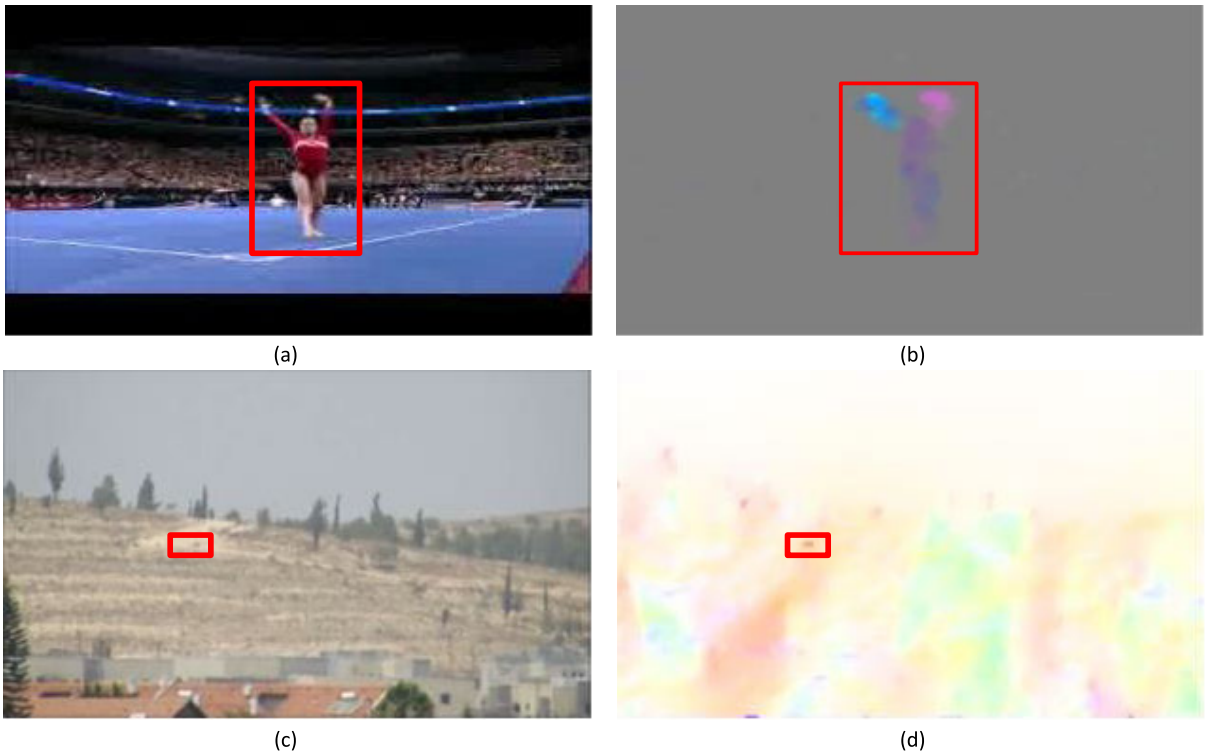


FIGURE 1. Sample comparisons of the challenges of our study and those of previous state-of-the-art studies of action recognition (red bounding boxes appear around the real moving objects). Here, (a) and (b) are samples from the UCF101 [2] dataset - an RGB and optical flow image, respectively, whereas (c) and (d) are samples from our dataset - an RGB and optical flow image, respectively.

from that of humans, limiting their real-world deployment. Most of these approaches [6], [7] are based on ''offline'' protocols, in which information from the entire video (taken as a whole observable quantity) is used to detect and recognize action instances. Recently, a method was proposed in [9] that relies on detection at the frame level. It performs action detection in real-time using (a) an efficient single-shot detector (SSD), (b) fast optical flow estimation [10] for the motion stream, and (c) an online tracking algorithm. The 3D CNNs in [11] have been shown to successfully extract spatio-temporal features, which can be used for action classification. Their 3D kernels allow these CNNs to learn temporal/motion information directly from the video frames. More recently, a two-stream I3D network was proposed [12] that takes advantage of ImageNet pretraining by inflating 2D ConvNets into 3D. The baseline presented in [13] extends the I3D network to action detection by having a region proposal network that selects spatio-temporal regions to be classified and refined. All these approaches do not deal with random background movements, such as those caused by atmospheric turbulence, that disrupt the detection process. In this paper, we present a method for human action detection in long-distance videos degraded significantly by the atmospheric path. The moving objects in these videos are quite small, have low signal-to-noise ratios (SNRs), and may be somewhat similar in appearance to the turbulence-induced motion of static regions, such as trees. We first created a data set of typical actions in long-range imaging (distance of about 2.5 km on a hot day with strong air turbulence that significantly affects the image). Then, for the detection stage, we combined and examined several state-of-the-art approaches. We trained two detection networks: one on RGB frames (appearance), and the other on optical flow images. The optical flow estimation process was modified to cope with the noisiness caused by spatio-temporal turbulence. The detection networks we used were YOLOv3 [14] and a slightly modified version of FFSSD [15], which is a variation of the SSD detector [16] for small objects. We adapted the I3D method to use FFSSD with Conv 3D, and examined several network architectures by combining 3D and 2D convolutions. Following the detection network, we fused the results from the two streams with the ''late fuse'' method [17]. In this paper we show that an important stage for increasing the precision of the action detection process can be a tracking process that takes into account the random spatio-temporal motions in the video, caused by the long atmospheric path. This process includes linking the detections to tracks via IoU (Intersection over Union) matching, and then determining and updating the classes of the tracks according to majority class voting of the tracks' histories, based on the turbulence characteristics. We also examined one-steam network architectures as baselines. Our main contributions can be summarized as follows: 1) a pre-processing algorithm for optical flow calculation based on characteristics of turbulence (random

motion), 2) an examination of various 2D and 3D networks combinations. In part of these architectures, we extended the I3D method [12] to support small-object action localization with a modified SSD detector, 3) a novel online tracking algorithm based on turbulence characteristics, 4) an algorithm based on majority voting that updates the labels of the actions dynamically during tracking, thus reducing false class predictions under turbulent conditions (we assumed that the action type can change during tracking, e.g., a person can alternate between walking and running), and 5) the creation of a unique dataset of typical actions undertaken in long-range imaging affected by the atmospheric path, which will be made publicly available. The rest of this paper is organized as follows: Related work is discussed in Section II. Section III describes the proposed method for action detection and classification in long-range videos degraded by the atmospheric path. Experimental results and a comparison with other methods are presented in Section IV. Discussion and Conclusions are presented in Sections V and VI, receptively.

II. RELATED WORK

A. OBJECT DETECTION WITH CNNs

Object detection has advanced significantly in recent years. The current state-of-the-art CNN detectors can be divided into two categories. Detectors in the first category utilize a two-stage object detection approach and include R-CNN [4], Fast R-CNN [18], Faster R-CNN [19], mask R-CNN [20], and the Feature Pyramid Network (FPN) [21]. Using this approach, a detector first generates a sparse set of candidate object regions and extracts their feature information. Then the location and category of the candidates can be further predicted and identified. This approach is not suitable for some real-time situations. The second category of detectors use a one-stage object detection approach, as in YOLO (You Only Look Once) [22] and SSD (Single Shot Multibox Detector) [16], which classify and regress object locations directly without generating candidate targets first. This approach predicts the target location and category in a single pass through the network and is generally simpler and faster. In the basic YOLO method, all scores and regressions are computed from the last convolutional feature maps. Later improvements in this approach include YOLOv2 [23], YOLOv3 [14], and RetinaNet [24]. SSD employs multiple convolutional scales in the object detection process. Bounding boxes for predicting small-sized objects come from early layers, and boxes for predicting bigger objects come from the latter layers, which have larger receptive fields. Feature-Fused SSD(FFSSD) [15] introduces contextual information to the SSD via a multi-level feature fusion method in order to improve the accuracy of the SSD, especially for small objects. In our work, we adapt the YOLOv3 and FFSSD architectures to our action localization method.

B. OPTICAL FLOW ESTIMATION

Optical flow estimation has been dominated by the various approaches that followed [25]. Recently, approaches utilizing deep CNNs for optical flow estimation [26]–[28] have shown promise. Flownet [27], the first end-to-end trainable deep CNN for optical flow estimation, is trained using synthetic data to optimize the end-point error (EPE). The authors provide two architectures for estimating the optical flow. The first architecture is a standard CNN that takes the concatenated channels from two consequent frames and predicts the flow directly. The second is a two-stream architecture that attempts to find a good representation for each image before they are combined by a correlation layer. However, Flownet has fallen behind previous top methods due to inaccuracies concerning the small displacements present in realistic data. Flownet2 [28] addresses this problem by introducing a stacked architecture including a subnetwork for handling small displacements. It achieves a more than 50% improvement in EPE compared to Flownet. SpyNet [26] warps images at multiple scales to cope with large displacements, resulting in a compact spatial pyramid network. Recently, PWC-Net [29] and LiteFlowNet [30], which have warp features extracted from CNNs, have achieved state-of-the-art results with their lightweight frameworks. In our work, we examined PWCNet and LiteFlowNet for use in our action localization method.

C. SPATIO-TEMPORAL ACTION DETECTION

Inspired by the recent advances in image object detection, a number of efforts have been made to extend image object detectors (e.g., R-CNN, Fast R-CNN, and SSD) to perform the task of frame-level action detection. Several methods adopt 2D CNNs as backbones and classify videos by simply aggregating frame-wise predictions [31]. However, these methods only model the appearance (RGB image) features of each frame independently while ignoring the dynamics between frames, which results in inferior performances when attempting to recognize temporally dynamic videos. To handle this drawback, two-stream-based methods [5], [8], [17], [32]–[35] have been introduced that model the appearance and the dynamics (inter-frames motions) separately with two individual networks and then fuse them. This area of research can be divided into two categories: offline action detection and online action detection.

1) OFFLINE ACTION DETECTION

The goal of offline action detection is to detect the start and end times of action instances from fully observed long videos. The spatial location is often determined as well. In this offline setting, the entire action can be observed first. Moreover, calculation time is mostly a non-issue. As a result, the best performing methods are often far too complicated to be used in a real-time setting. The methods presented in [36], [37] claim that the context of the event is necessary and use several seconds of the video before and after the key frame as input. In [38], a spatio-temporal progressive learning framework is proposed, in which the initial coarser proposals are further refined by increasing the temporal context before it is fed into a classifier. This progressive way of increasing

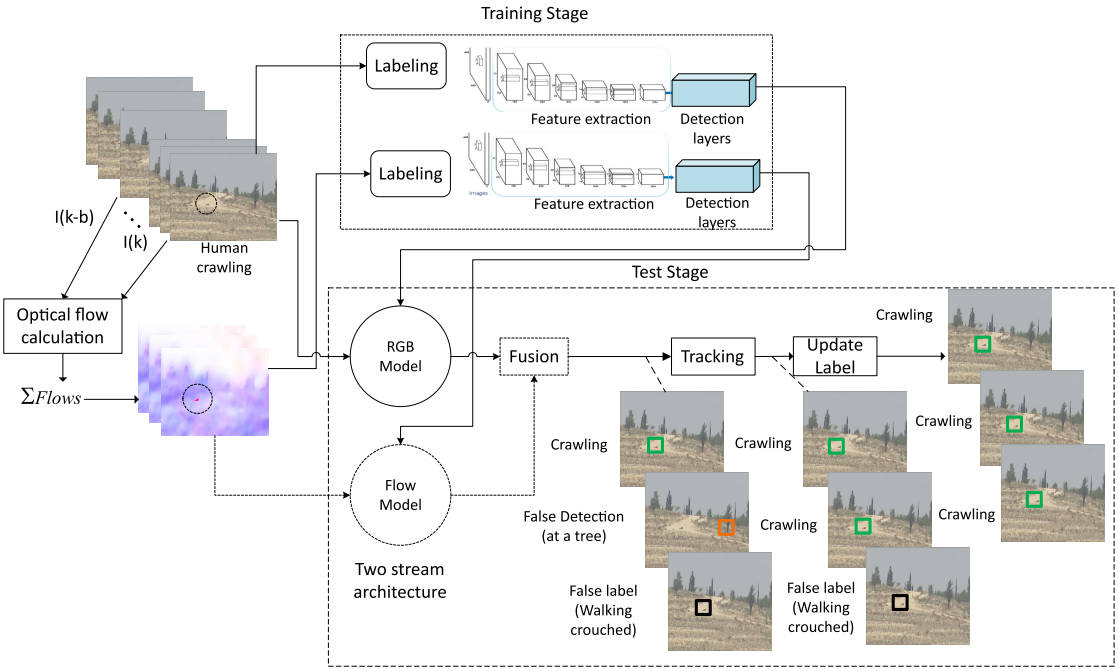


FIGURE 2. A schematic diagram of the proposed method and its training. A description from left to right according to the arrow flow: 1) A pre-processing algorithm for optical flow calculation based on turbulence characteristics; 2) labeling the input frames, RGB and optical flow separately; 3) training a moving object detection network; 4) in the test stage, using the trained model and inference frame by frame; 5) in the two-stream case, fuse the RGB and optical flow outputs before performing prediction; 6) An online tracking process based on turbulence characteristics; 7) an algorithm based on majority voting that updates the action labels dynamically during tracking, thus reducing false class predictions. An illustration of possible effects of the of Tracking and Update Label procedures on reducing false predictions is shown by the colored rectangles (true positive - green rectangle, false positive – orange rectangle, correct location but wrong class – black rectangle).

the temporal context helps overcome spatial displacement problems.

2) ONLINE ACTION DETECTION

Given a streaming video, online action detection aims to identify actions in video frames as they arrive without observing future video frames. A method termed ROAD [9] produces action bounding boxes for both the appearance and flow frames and uses an online algorithm to incrementally construct and label action tubes from boxes. To better leverage the temporal cues, several recent works have been proposed to perform action detection at the clip level. For instance, ACT [39] obtains a short sequence of frames (e.g., 6 frames) as input and outputs regressed tubelets, which are then linked by a tubelet-linking algorithm to construct action tubes. The importance of temporal information is further demonstrated in [13] via the use of longer clips (e.g., 40 frames) and taking advantage of I3D pre-trained on a large-scale video dataset [12]. Recently, in [40], a spatio-temporal action localization approach with an integrated optical flow sub-network has been proposed for better computational efficiency.

III. METHOD

In this section we describe the proposed method for action detection under spatio-temporal noisiness conditions that appear in long-distance imaging through the atmosphere. Such conditions set a challenge over current state of the art action detection methods that their strong performances are significantly reduced in such noisiness. An introductory schematic diagram of the proposed method and its training is presented in Fig. 2. This process can be divided into several components. First, a pre-processing algorithm is performed for optical flow calculation based on turbulence characteristics. Labeling the input frames (RGB in one-stream architecture cases, and RGB and optical flow separately, in two-stream cases) is performed, before training a moving object detection network (a variety of network types are examined, including 2D, 3D, one stream and two-stream (RGB and optical flow)). In the test stage, the trained model is used and inference online frame by frame. In the two-stream case, we fuse the RGB and optical flow outputs before performing prediction. Following the frame-level detections, we apply an online tracking algorithm based on turbulence characteristics, and an algorithm based on majority voting that updates the action labels dynamically during tracking, thus reducing false class predictions under turbulent conditions. Illustrations of false predictions (of the crawling action) before the Tracking and the Update Label stages are shown by the colored rectangles.

A. DATA COLLECTION

The most popular action recognition datasets, such as HMDB-51 [41], UCF101 [2], Kinetics-400 [42],

FIGURE 3. Sample frames from the examined video sequences: left and right columns were captured with the X100 and X50 zooms, respectively. Rows 1-4 exhibit examples of the action classes: crawling, running, walking, and crouch walking, respectively. Red boxes are added to clarify the location of the moving person. The raw video clips from our dataset are available in [44].

Kinetics-600 [43], and AVA [12], consist of video clips recorded in indoor or outdoor ''clean'' environments without moving background distractions. In our case, we dealt with moving background distractions caused by long atmospheric path degradations (blur and spatio-temporal image movements). We examined four types of actions during this study: walking, crouch walking, running, and crawling. A moving person was imaged from a distance of about 2.5 km, with a Cannon SX50 camera with X50 and X100 zooms on a hot sunny day (about 35^0C). Fig. 3 shows sample frames from the analyzed video sequences, each containing a single moving person (left X100, right X50).

B. PROPOSED OPTICAL FLOW CALCULATION

As in previous works [5]–[8], we used a two-stream CNN approach [45], in which optical flow and appearance are processed separately. For the optical flow stream, we implemented several methods: the TV-L1 [46], which is based on the classic gradient approach, and deep-learning based methods, namely, LiteFlowNet [30], and PWC-Net [29]. TV-L1 is very accurate, but when confronted with a large image (as in our case), it is relatively slow (about 400ms per frame). LiteFlowNet and PWC-Net, on the other hand, are computationally efficient (near real-time in our case). When attempting to calculate the optical flow between consecutive frames

(with all the various methods examined), the true object's flow is usually non-visible due to the dominant turbulence movements. In order to improve the SNR, we calculate the optical flow as follows:

$$flow_i = \sum_{k=i-a+1}^{i} OFC(I(k), I(k-b))) \quad (1)$$

where $flow_i$ is the optical flow of frame i inserted later into the optical flow stream input of the detection network, OFC represents the method of optical flow calculation (e.g., TV-L1, LiteFlowNet, etc.) applied to a pair of frames, a is the number of previous optical flow maps that are summed (set here to 10), and b is a gap between the frames inserted into the OFC as pairs (set here to 30), instead of consecutive frames as done conventionally. The reason for summing the optical flow maps over a number of frames is derived from the unique characteristics of turbulence, i.e., the integrated turbulence-based movements across a large number of frames approach zero [1], so summing over a number of frames increases the SNR. Also, we calculate the optical flow with a gap of some number of frames between the two images (and not consecutive frames) because in long distance imaging, object velocity in the image plane is typically slow, and might be insignificant even at the lowest scale. For example, in the videos captured with the X50 zoom, the movements of the small objects are very slow (down to 0.1 pixels/frame). Fig. 4 shows the optical flow outputs for TV-L1 and LiteFlowNet in the 'crawl' action class, which is the most challenging case in our database in terms of optical flow calculations because the object moves very slowly, so that its movements are hardly observable across short time segments. The left column shows the optical flow results obtained according to Eq. (1), and the right column shows the results when consecutive frames are utilized, as conventionally done. Row 1 at the top contains the results for TV-L1 X50, row 2 shows the results for TV-L1 X100, row 3 contains the results for LiteFlowNet X50, and row 4 shows the results for LiteFlowNet X100. It can be seen that in the right column, for both methods and both zoom levels, only the turbulence-related movements are observed, whereas, on the left side, the objects' movements can be clearly seen.

C. ACTION DETECTION NETWORKS

In our study, we applied different combinations of various 2D and 3D networks (where 2D include YOLOv3 and FFSSD, and 3D include the FFSSD-3D that is our extension of I3D), and also one-stream and two-stream architectures. The proposed action detection frameworks are outlined in Fig. 6 in a manner similar to that used in [12].

FFSSD [15] is a variation of the SSD [16] that uses a multi-level feature fusion method to introduce contextual information to the SSD in order to improve the accuracy with regards to small objects [15]. We use this method because in long-distance imaging objects may appear very small. For our integrated detection network, we adopt the network design and architecture of the SSD [16] object detector, with an input image size of 300 x 300. We also use an ImageNet pretrained VGG16 network [16].

YOLOv3 [14] is an improved version of previous YOLO architectures. First, YOLOv3 uses multi-label classification (independent logistic classifiers) to adapt to more complex datasets containing many overlapping labels. Second, YOLOv3 utilizes three different scale feature maps to predict the bounding box. The last convolutional layer predicts a 3D tensor encoding class predictions, objectness, and the bounding box. Third, YOLOv3 proposes a deeper and robust feature extractor, called Darknet-53, inspired by ResNet [24]. We chose YOLOv3 because it has been used frequently of late, and according to [47], it achieved greater accuracy in detecting small-sized objects. For our integrated detection network, we adopt the network design and architecture, with an input image size of 416 x 416 and also use an ImageNet pretrained Darknet-53 network. We employ an architecture with an input size of 416 x 416 instead of the 320 x 320 base version because it achieves higher accuracy, and the running time is almost the same (29 milliseconds instead of 22 milliseconds).

For another object detector suitable for our case, we designed a three-dimensional FFSSD (FFSSD-3D) by combining two methods: Inflated 3D ConvNet(I3D) [12] and FFSSD [15]. 3D ConvNets seem like a natural approach to action recognition, as they are similar to standard convolutional networks, but with spatio-temporal filters (the additional dimension is temporal). This can be accomplished by starting with a 2D architecture, andinflating all the filters and pooling kernels with an additional temporal dimension; thus, typical square filters usually become cubic [12]. In practice, I3D inflated the Inception-V1 architecture [48] to 3D, and to combine it with FFSSD, we fused layers 3b and 4a. The idea behind this fusion is similar to what was done in FFSSD [15], passing the semantic information captured in convolutional forward computation back to the shallower layers can improve the detection performance of small objects. The last part of the architecture is similar to the SSD300 model, as can be seen in Fig. 6. We use the fused layers 3b and 4a as well as layers, 4b, 5b, *conv*6_2, *conv*7_2, and *conv*8_2 to predict both location and confidences. The input image has a size of 300 x 300, and we also use an ImageNet + Kinetics pre-trained Inception-V1 network provided by [12].

For combining the two streams at test time, we use the late fusion approach [17] because it obtained the best results. We average the scores from both streams when the score of the flow stream is greater than the score of the appearance stream; otherwise, we use the score of the appearance stream. We keep the regressed boxes from the RGB stream, as appearance is more relevant for regressing boxes.

D. ONLINE STABILIZING ACTION RECOGNITION VIA TRACKING

The inputs to the tracking algorithm are the fused frame-level detection boxes with their class specific scores. In the first

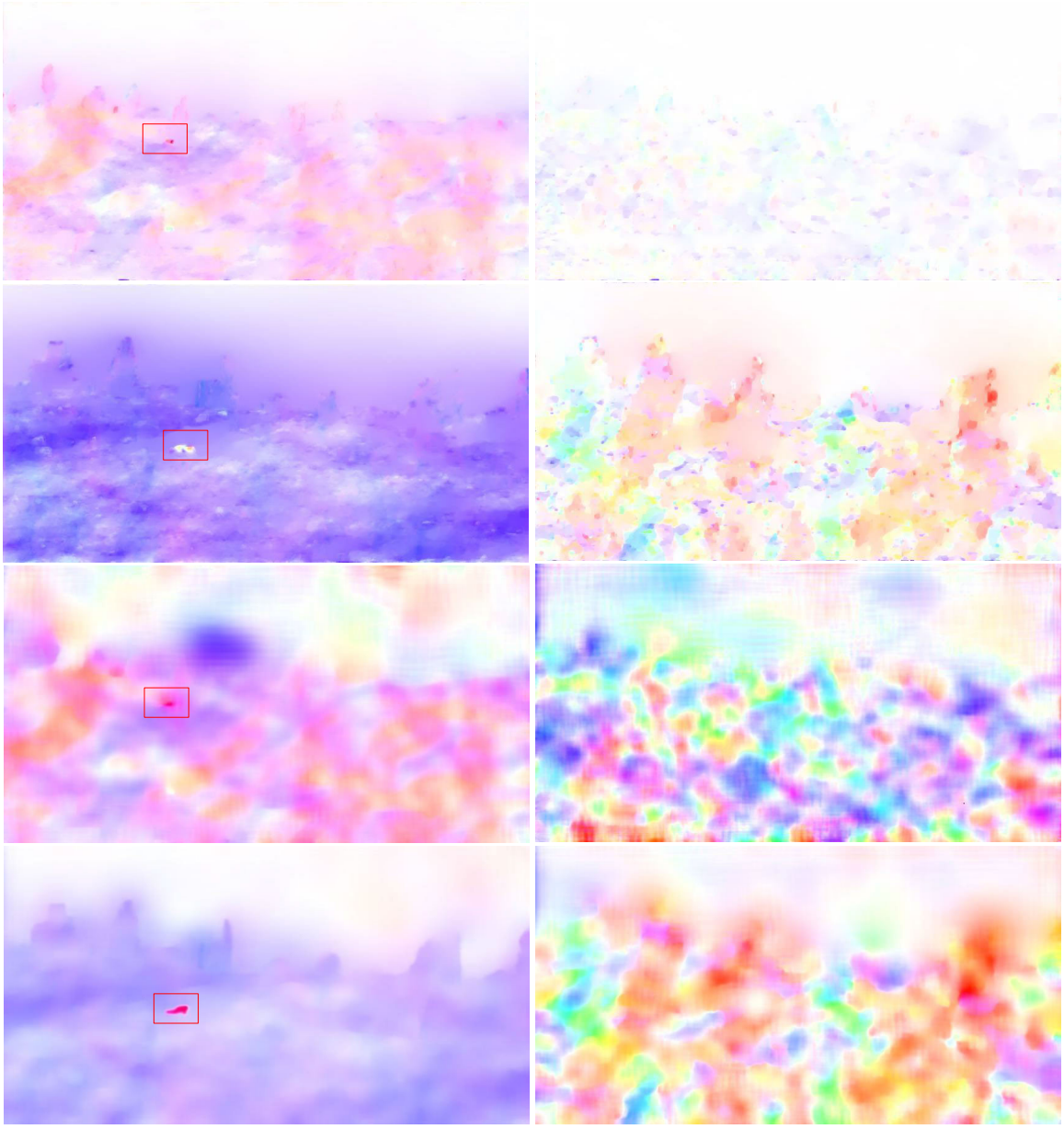


FIGURE 4. The optical flow outputs of TV-L1 and LiteFlowNet for the crawling class. The left column shows the optical flows calculated with Eq. 1, and the right column shows the optical flows calculated between consecutive frames. Row 1 (upper figures) shows the results for TV-L1 with the X50 zoom, row 2 shows the results for TV-L1 with the X100 zoom, row 3 shows the results for LiteFlowNet with the X50 zoom, and row 4 shows the results for LiteFlowNet with the X100 zoom.

frame of the video with any detection boxes, given N detection boxes, N tracks of the moving objects are initialized. The algorithm grows the track lengths incrementally over time by adding one box at a time according to IoU (Intersection over Union) matching, and updates the other parameters: location, width, height, velocity, class and score. The number of tracks varies with time, as new tracks are added and/or old tracks are terminated. At each time step (frame), we sort the existing tracks according to their lengths, and the detection boxes according their scores, so that the stable track (the longest one) can potentially match the best box from the set of detection boxes in the next frame (if the IoU between the detection box and current track box is above a threshold). We set two thresholds, a lower threshold for basic linking, and a higher threshold for the track update. When the IoU is above the lower threshold, the detection is related to the track but does not update the track parameters described above. Only when the IoU is above the higher threshold are the track parameters updated and the detection considered to be the current track box. The velocity of the track is also updated as follows: when the track length is less than the minimum number of frames to be considered as a stable track(λ_s),

FIGURE 5. Network architectures we examined: (a)-(d) networks that combine Conv 3D (FFSSD-3D) and Conv 2D (YOLOv3 and FFSSD) streams, and (e)-(h) networks with one stream. K is the number of 3D input frames (in our case, $K = 10$). After the detection network, results are fused in cases of two streams, and are improved by a tracking process based on turbulence characteristics.

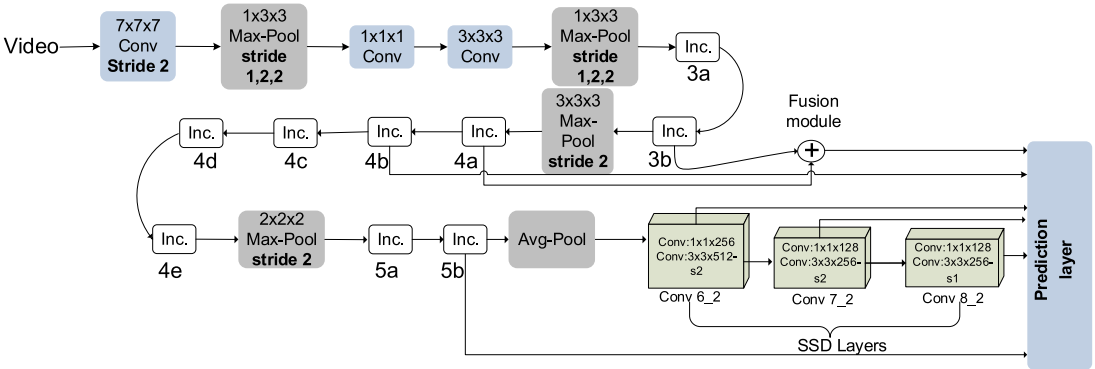


FIGURE 6. The I3D architecture extended with FFSSD. The network architecture begins with the I3D architecture, from which we fused layers 3*b* and 4*a*. The rest of the architecture is similar to that of the SSD300 model.

we calculate the mean velocity by dividing the distance the track has traveled by the number of frames at which it exists. If the track is stable, we update the mean velocity with a weighted average of the current track velocity and the mean velocity of the last λ_s frames. If there are tracks that were not updated in the current frame, we update their states according

to the mean velocity. In this way, tracks cannot simply drift off (which can lead to miss detection), and they can be terminated only if no matches are found for k consecutive frames, which we term a grace window, or if the track is too short and thus likely to be a turbulence-induced motion. Finally, each track updates its score and label (see Sec. 3.E). In our experiments, the lower threshold for the basic linking is 0.2, and the higher threshold is 0.35. We set the threshold of the number of frames for a stable track to $\lambda_s = 60$, the number of frames in the grace window to $\lambda_{GW} = 60$, and the minimum track length to $\lambda_{Turb} = 15$,were smaller tracks are suspected to be turbulence-induced motions. The tracking process is described in detail in Fig. 7.

E. ONLINE UPDATING ACTION LABEL BY MAJORITY VOTING

As a part of the tracking process described above, we suggest a robust method for determining the track action class. Due to the long turbulent path, our data is spatio-temporally noisy, and the objects are small and with low contrast. These conditions can cause the detection network to produce a relatively high False Alarm (FA) rate. Our updating method is based on the turbulence characteristics via the track linking history and a majority voting technique. This causes the label's determination to be more robust to FAs. In our study, we assumed that the behavior (action type of the object) can change during the tracking. In the initial λ_s frames, we search for the most common label of the track and calculate the mean score of the detection with this label. After the initialization stage, if the new detection has the same label, we only recalculate the score. Otherwise, we increase the counter of mismatched labels. If the counter is above a threshold λ_s, we change the label to the most common label in the last λ_s frames. The online temporal relabeling is summarized in Algorithm 1.

Algorithm 1 Majority Class Voting for Online Updating the Labels of the Tracks

Input
- T_k^i - Track i in frame k
- $T_k^i.label$ - Label of track i in frame k
- $T_k^i.score$ - Score of track i in frame k
- $T_k^i.count\_diff\_label$ - Counter of mismatched label of Track i in frame k
- D_k^j - Detection j in frame k
- $D_k^j.label$ - Label of detection j in frame k

Output
- T_k^i - Update track i in frame k

if $T_k^i.length > \lambda_s$ **then**
  if $T_k^i.label \neq D_k^j.label$ **then**
    if $T_k^i.count\_diff\_label > \lambda_s$ **then**
      $T_k^i.label = max(hist(\{T_{\tilde{k}}^i.label\}_{\tilde{k}=k-\lambda_s}^{k}))$
      $T_k^i.score = mean(\{T_{\tilde{k}}^i.score\}_{\tilde{k}=k-\lambda_s}^{k}, where\ T_{\tilde{k}}^i.label = T_k^i.label)$
      $T_k^i.count\_diff\_label = 0$
    else
      $T_k^i.count\_diff\_label = T_k^i.count\_diff\_label + 1$
      $T_k^i.score = mean(\{T_{\tilde{k}}^i.score\}_{\tilde{k}=k-\lambda_s}^{k}, where\ T_{\tilde{k}}^i.label = T_k^i.label)$
    end if
  else
    $T_k^i.count\_diff\_label = 0$
    $T_k^i.score = mean(\{T_{\tilde{k}}^i.score\}_{\tilde{k}=k-\lambda_s}^{k}, where\ T_{\tilde{k}}^i.label = T_k^i.label)$
  end if
else
  $T_k^i.label = max(hist(\{T_{\tilde{k}}^i.label\}_{\tilde{k}=k-\lambda_s}^{k}))$
  $T_k^i.score = mean(\{T_{\tilde{k}}^i.score\}_{\tilde{k}=k-\lambda_s}^{k}, where\ T_{\tilde{k}}^i.label = T_k^i.label)$
end if

IV. EXPERIMENTAL RESULTS AND COMPARISION

We tested the proposed method on two different goals: i) action detection, and ii) moving object detection only. We tested all the architectures in Fig. 5 to examine the effects of 3D CCNs vs 2D CNNs, two-stream vs one-stream, and RGB vs optical flow inputs. These different approaches were examined for leading object detectors (YOLOv3, FFSSD) and optical flow estimation techniques (TV-L1, LiteFlowNet, and PWCNet).We implemented our system using TensorFlow [49], with training executed on a single NVIDIA Titan X GPU of 12GB memory. We optimized the model's parameters with the Adam optimizer [50] and used a learning rate of 0.001 in a warm-up stage with 100 epochs and then trained for another 200 epochs with a learning rate of 0.0001. We also freeze the backbone layers in the warp-up stage. The batch size was 32 for FFSSD and Yolo3, and 8 for FFSSD-3D. The training lasted about 2 days for each stream (RGB,TVL1 and LiteFlownet) for Yolo3 and FFSSD and about 3 days for FFSSD-3D. The one-stream network architectures we propose are end-to-end trainable. In the two-stream network architectures, each stream is trained separately. Our code is available at [51].

A. DATASET

Our dataset contains 11k video frames (30 frames per second, 350 seconds) from four action classes: running, walking, walking crouched, and crawling. The dataset contains one action for each video. We split the dataset into three parts: training 6.3k frames, validation 1.3k frames, and test 3.2k frames. Results are reported for the test data. We annotated each part of the dataset for RGB, optical flow - TV-L1, and optical flow – LiteFlowNet separately. Our dataset is available at [52].

B. OPTICAL FLOW CALCULATION

Since, in our case, optical flows are not calculated between consecutive frames, we could not use the same annotation for the RGB images that other state-of-the-art spatio-temporal methods use. Instead, we had to annotate each output of the optical flow estimation method separately. We examined the following three optical flow estimation methods: TV-L1, LiteFlowNet, and PWC-Net. During the annotation process, we noticed that the turbulence-related movements made it difficult to identify the objects in the optical flow maps themselves, and that the observer who annotated the frames did not identify all the objects in them. In order to quantify the ability of the observer to identify the objects in the optical flow map for each method, we recorded the percentage of successful identifications across all the frames in our database.

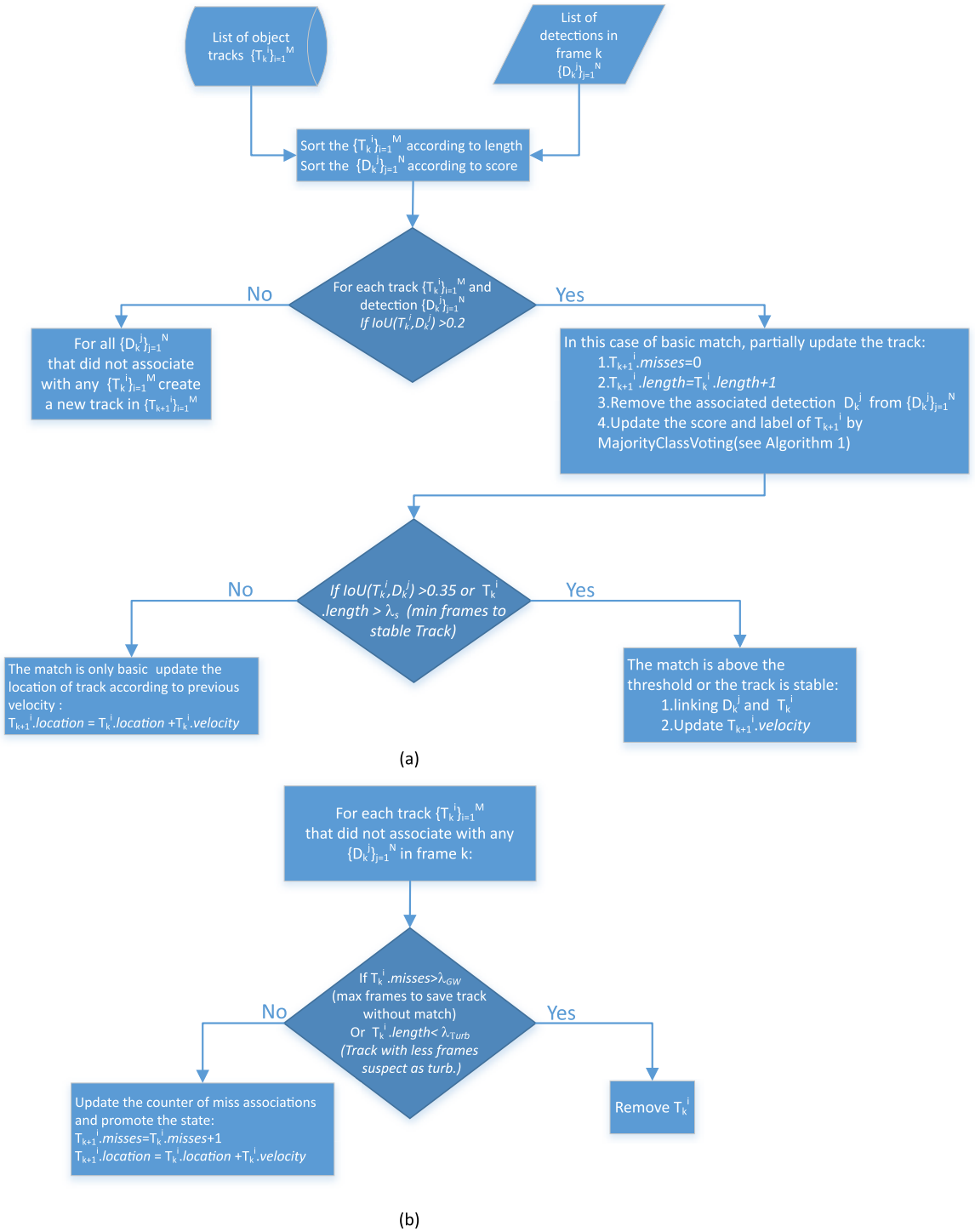


FIGURE 7. The tracking process: (a) the matching process in frame k between detections and associated tracks, and (b) the handling of tracks with no associations to detections.

PWC-Net was the fastest method (67ms per frame) but had poor results for our dataset, i.e., object movements were detected in only 11.2% (21.0% - X100, 2.8% - X50) of the frames. TV-L1 was the most accurate method, and the object movements were seen in 96% (99.8% – X100, 92.9% - X50) of the frames in the dataset. However, it was also the slowest method (400ms per frame). In addition, LiteFlowNet was more accurate than PWC-Net. In LiteFlowNet, the object movements were seen by an observer during the labeling process in 73% of the images (96.15% - X100, 54.4% - X50) from our dataset, and it was faster than TV-L1 (105ms for each frame optical flow calculation). Therefore, we used TV-L1 and LiteFlowNet in our network for the optical flow stream.

TABLE 1. Action detection results (mAP) for the two-stream 2D network architecture shown in Fig. 5(b), with and without the tracking process(denoted as TRK), and two different optical flow formation methods, depending on the IoU threshold (0.2 or 0.5) and imaging zoom (X100 or X50). The bottom row contains the results of [9] applied to our long-distance imaging database. The video clips marked results are available at Link.

Two-stream 2D architectures						
IoU threshold		0.2			0.5	
dataset zoom	all	X100	X50	all	X100	X50
FFSSD[RGB]+FFSSD[TV-L1]	52.79	58.96	46.61	46.53	53.96	39.09
FFSSD[RGB]+FFSSD[TV-L1]+TRK	91.22	97.25	85.18	75.65	83.68	67.62
FFSSD[RGB]+FFSSD[LiteFlowNet]	45.73	49.64	41.82	42.62	47.02	38.22
FFSSD[RGB]+FFSSD[LiteFlowNet]+TRK	59.86	68.86	51.46	52.18	60.67	43.69
YOLOv3[RGB]+YOLOv3[TV-L1]	72.07	69.37	74.77	62.73	65.66	59.80
YOLOv3[RGB]+YOLOv3[TV-L1]+TRK	**97.10**	**99.72**	**94.49**	**84.29**	**93.35**	**75.23**
YOLOv3[RGB]+YOLOv3[LiteFlowNet]	70.81	67.74	73.89	61.74	64.30	59.18
YOLOv3[RGB]+YOLOv3[LiteFlowNet]+TRK	97.05	**99.72**	94.39	84.20	93.24	75.16
ROAD [9]	57.61	59.87	57.82	47.44	56.39	37.39

TABLE 2. Action detection results (mAP) for the one-stream network architectures shown in Fig. 5(e-h), where we denote each architecture according to Fig. 5, with and without the tracking process(denoted as TRK), and two different optical flow formation methods, depending on the IoU threshold (0.2 or 0.5) and imaging zoom (X100 or X50). The last two rows contain the results of [9] applied to our long-distance imaging database. The video clips marked results are available at Link.

One-stream architectures						
IoU threshold		0.2			0.5	
dataset zoom	all	X100	X50	all	X100	X50
YOLOv3[RGB](f)	70.54	73.34	67.73	61.53	64.25	58.81
YOLOv3[RGB]+TRK(f)	**97.05**	**99.71**	**94.39**	**84.25**	**93.14**	**75.36**
FFSSD[RGB](f)	42.79	39.03	46.55	40.28	44.64	35.91
FFSSD[RGB]+TRK(f)	47.57	55.72	39.43	43.91	53.61	34.22
FFSSD-3D[RGB](e)	37.01	28.88	45.14	33.28	43.06	23.50
FFSSD-3D[RGB]+TRK(e)	63.68	85.90	41.46	58.20	80.20	36.20
YOLOv3[TV-L1](h)	54.82	53.27	56.37	15.34	9.60	21.07
YOLOv3[TV-L1]+TRK(h)	77.05	74.85	79.25	20.60	13.50	27.70
YOLOv3[LiteFlowNet](h)	8.62	16.79	0.45	2.21	4.18	0.24
YOLOv3[LiteFlowNet]+TRK(h)	10.98	21.16	0.81	2.97	5.68	0.27
FFSSD[TV-L1](h)	44.43	50.89	37.96	13.01	13.63	12.38
FFSSD[TV-L1]+TRK(h)	70.81	71.64	69.99	18.80	17.30	20.30
FFSSD[LiteFlowNet](h)	8.29	15.54	1.03	3.11	6.17	0.05
FFSSD[LiteFlowNet]+TRK(h)	11.71	21.04	2.38	4.22	8.37	0.08
FFSSD-3D[TV-L1](g)	34.6	39.77	29.43	7.87	8.52	7.23
FFSSD-3D[TV-L1]+TRK(g)	50.79	58.54	43.03	9.78	10.33	9.23
FFSSD-3D[LiteFlowNet](g)	7.62	14.92	0.33	3.29	6.58	0.00
FFSSD-3D[LiteFlowNet]+TRK(g)	11.12	21.89	0.34	4.50	9.00	0.01
ROAD [9]-RGB only	57.75	62.96	55.63	47.2	58.37	34.62
ROAD [9]-FLOW only	0.36	1.55	0.05	0.2	1.04	0.04

C. METRICS

Metrics in this study were measured at the frame level, and we report the mean Average Precisions (mAPs) along frames and classes. A detection is correct if its IoU with a ground-truth box is greater than some threshold (0.5 or 0.2) and its action label is predicted correctly. For each zoom level and class, we computed the average precision (AP), and report the average over all classes.

D. RESULTS AND COMPARISON

We divided the summarized results into four types of network architectures spread across four tables (as seen in Fig. 5): two-stream 2D in Table 1, one-steam (2D and 3D) in Table 2, two-stream 3D in Table 3, and two-stream combination of 2D and 3D in Table 4. For each architecture type, different detection methods, with and without the tracking stage, were examined and compared. Also, we present the action detection results for each action class in Fig. 8 and Fig. 9. These figures show the distribution of results according to the IoU threshold (Fig. 8 - 0.2 and Fig.9 - 0.5) and imaging zoom (X100 and X50). We compared our method to [9] (see Tables 1 and 2) and a recent method outlined in [40]. We adapted the method used in [40] to our dataset, based on authors' code. This method is based on the conventional optical flow calculation with consecutive frames and uses an "early fuse" between spatial and temporal features (concatenates the spatial and temporal features before the prediction layer). In our type of dataset, optical flow between consecutive frames is too noisy, as shown above (Fig. 4), and fusing the temporal "noisy" features with spatial features before the prediction layer is probably why their method did not achieve reasonable results. Table 5 summarizes a comparison with our previous work [53], which is not a network-based method and outperformed other such methods [54]–[56]. This comparison is only at the frame-level detection stage because

TABLE 3. Action detection results (mAP) for the two-stream 3D network architecture shown in Fig. 5(a), with and without the tracking process(denoted as TRK), and two different optical flow formation methods, depending on the IoU threshold (0.2 or 0.5) and imaging zoom (X100 or X50). The video clips marked results are available at Link.

Two-stream 3D architectures						
IoU threshold		0.2			0.5	
dataset zoom	all	X100	X50	all	X100	X50
FFSSD-3D[RGB]+FFSSD-3D[TV-L1]	44.68	56.19	33.17	61.36	87.29	35.43
FFSSD-3D[RGB]+FFSSD-3D[TV-L1]+TRK	**70.89**	**99.72**	42.05	**79.02**	**87.29**	**70.75**
FFSSD-3D[RGB]+FFSSD-3D[LiteFlowNet]	41.10	50.60	31.60	36.04	45.60	26.47
FFSSD-3D[RGB]+FFSSD-3D[LiteFlowNet]+TRK	64.57	81.57	**47.56**	55.68	70.27	41.09

TABLE 4. Action detection results (mAP) for the two-stream combination of 2D and 3D network architectures shown in Fig. 5 (c,d), where we denote each architecture according to Fig. 5, with and without the tracking process(denoted as TRK), and two different optical flow formation methods, depending on the IoU threshold (0.2 or 0.5) and imaging zoom (X100 or X50). The video clips marked results are available at Link.

Two-stream combination of 2D and 3D architectures						
IoU threshold		0.2			0.5	
dataset zoom	all	X100	X50	all	X100	X50
FFSSD-3D[RGB]+FFSSD[TV-L1](c)	45.42	57.44	33.40	37.30	49.42	25.18
FFSSD-3D[RGB]+FFSSD[TV-L1]+TRK(c)	91.82	98.48	85.16	74.82	81.29	68.34
FFSSD-3D[RGB]+FFSSD[LiteFlowNet](c)	41.46	50.77	32.15	36.23	49.19	26.23
FFSSD-3D[RGB]+FFSSD[LiteFlowNet]+TRK (c)	62.71	83.97	41.46	50.91	65.80	36.02
FFSSD-3D[RGB]+YOLOv3[TV-L1](c)	41.54	50.63	32.45	78.31	86.90	69.72
FFSSD-3D[RGB]+YOLOv3[TV-L1]+TRK(c)	70.96	99.61	42.32	61.96	86.90	37.01
FFSSD-3D[RGB]+YOLOv3[LiteFlowNet](c)	38.40	47.31	29.48	33.68	43.68	23.68
FFSSD-3D[RGB]+YOLOv3[LiteFlowNet]+TRK(c)	70.40	99.30	41.51	61.72	87.18	36.25
FFSSD-3D[TV-L1]+FFSSD[RGB](d)	51.92	59.72	44.12	47.06	54.43	39.70
FFSSD-3D[TV-L1]+FFSSD[RGB]+TRK(d)	75.85	**99.72**	51.98	65.78	87.15	44.41
FFSSD-3D[LiteFlowNet]+FFSSD[RGB](d)	41.46	50.77	32.15	42.91	46.97	38.86
FFSSD-3D[LiteFlowNet]+FFSSD[RGB]+TRK(d)	68.75	94.99	42.50	59.38	81.39	37.37
FFSSD-3D[TV-L1]+YOLOv3[RGB](d)	70.54	67.73	73.34	61.53	64.25	58.81
FFSSD-3D[TV-L1]+YOLOv3[RGB]+TRK(d)	**97.05**	99.71	**94.39**	**84.25**	**93.14**	**75.36**
FFSSD-3D[LiteFlowNet]+YOLOv3[RGB](d)	70.54	67.73	73.34	61.53	64.25	58.81
FFSSD-3D[LiteFlowNet]+YOLOv3[RGB]+TRK (d)	**97.05**	99.71	**94.39**	**84.25**	**93.14**	**75.36**

TABLE 5. Frame-level detection results (mAP), devoid of action classification, for our test set across two IoU thresholds (0.2, 0.5) and imaging zooms (X100, X50). The bottom row contains the average per-frame results of our previous method [53]. The video clips marked results are available at Link.

IoU threshold		0.2			0.5	
dataset zoom	all	X100	X50	all	X100	X50
FFSSD[RGB]+FFSSD[TV-L1]	**99.52**	99.47	**99.57**	83.01	85.79	**80.24**
FFSSD[RGB]+FFSSD[LiteFlowNet]	97.45	97.21	97.68	81.30	84.00	78.61
YOLOv3[RGB]+YOLOv3[TV-L1]	97.13	**99.72**	94.54	**84.31**	**93.35**	75.27
YOLOv3[RGB]+YOLOv3[LiteFlowNet]	97.13	**99.72**	94.54	84.23	93.24	75.23
FFSSD-3D[RGB]+FFSSD[TV-L1]	92.50	98.48	86.02	75.20	81.29	69.11
FFSSD-3D[RGB]+YOLOv3[LiteFlowNet]	91.09	94.87	87.31	72.50	75.29	70.61
FFSSD-3D[RGB]+YOLOv3[TV-L1]	93.47	99.61	87.33	78.31	86.90	69.72
FFSSD-3D[RGB]+YOLOv3[LiteFlowNet]	93.47	99.62	87.21	78.86	87.51	70.21
FFSSD-3D[TV-L1]+FFSSD[RGB]	99.39	**99.72**	99.05	83.64	87.15	80.13
FFSSD-3D[LiteFlowNet]+FFSSD[RGB]	98.57	98.53	98.62	82.46	85.19	79.72
FFSSD-3D[TV-L1]+YOLOv3[RGB]	97.12	99.71	94.54	84.29	93.14	75.43
FFSSD-3D[LiteFlowNet]+YOLOv3[RGB]	97.12	99.71	94.54	84.29	93.14	75.43
FFSSD-3D[RGB]+FFSSD-3D[TV-L1]	93.52	**99.72**	87.31	79.02	87.29	70.75
FFSSD-3D[RGB]+FFSSD-3D[LiteFlowNet]	92.17	97.03	87.31	76.4	81.01	71.78
YOLOv3[RGB]	97.12	99.71	94.54	84.29	93.14	75.43
FFSSD[RGB]	98.85	99.70	97.99	83.90	90.35	77.46
FFSSD-3D[RGB]	93.51	**99.72**	87.31	79.66	88.78	70.54
Our previous method [54]	21.11	30.17	12.06	5.55	11.01	0.1

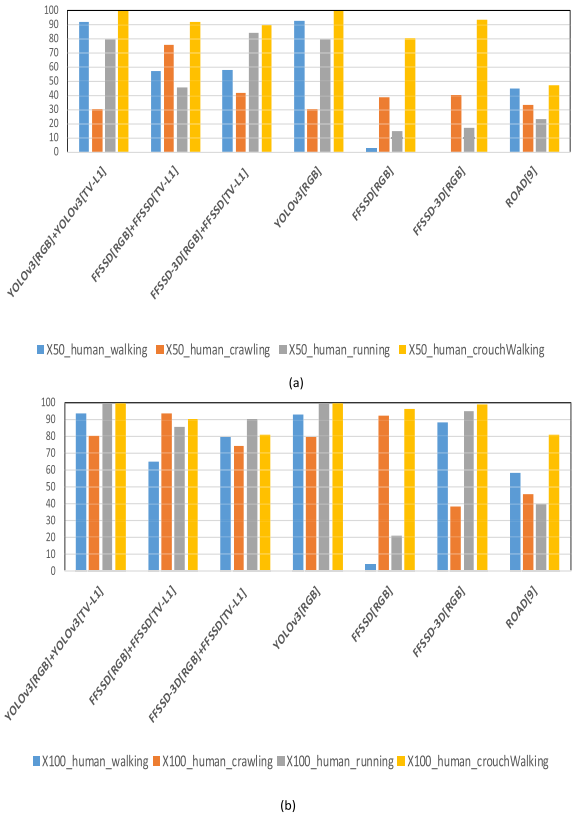


FIGURE 8. Action detection results (mAP) for our test set per action type, with an IoU threshold of 0.5 and imaging zooms X50 (upper row) and X100 (lower row).

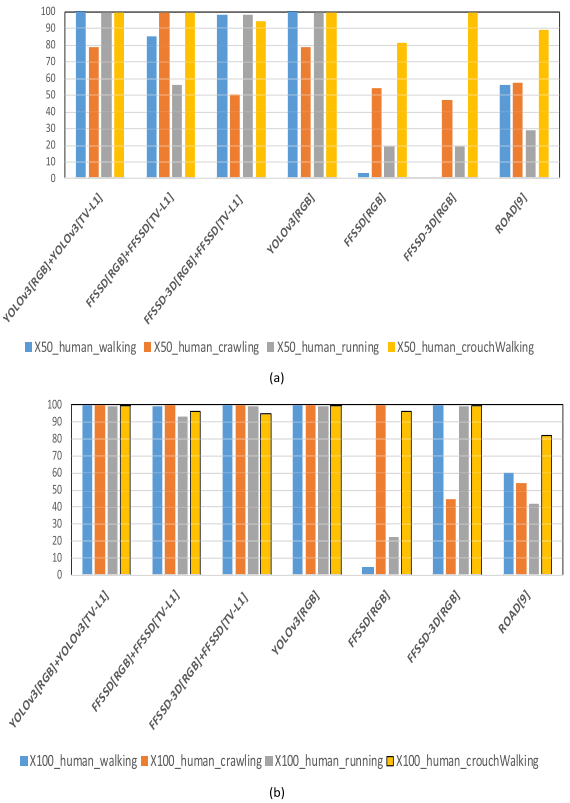


FIGURE 9. Action detection results (mAP) for our test set per action type, with an IoU threshold of 0.2 and imaging zooms X50 (upper row) and X100 (lower row).

the previous work dealt with detecting and tracking moving objects only, without action classification. Note that the results reported in this section in the case of 3D convolutional networks are with a number of frames, K =10, for both modalities (RGB and optical flow).

E. ABLATION STUDIES

In order to study the influence of different parts of the tracking process, we separated them into two stages: building the tracks (''linking'') and updating the labels with the majority voting technique, according to Algorithm 1. The label of a track in the ''linking'' stage is only updated by the latest class detection associated with the track. For this analysis, we chose several network architectures that achieved the highest accuracy (see Table 6).

F. RUN TIME DETECTION SPEED

In Table 7, we report the execution times of our pipeline for all the proposed architectures. The results were generated using a desktop computer with two Intel Xeon CPU@2.40GHz (8 cores) and two NVIDIA Quadro P5000 GPUs. Efficient run time capabilities can be achieved by computing the two CNN forward passes in parallel on two GPUs.

As can be seen in Table 7, the one-stream 2D architectures, FFSSD[RGB]+TRK and Yolo3[RGB]+TRK, are the fastest, and can be used for real-time applications with runtime of 21 milliseconds and 30 milliseconds, receptively. Thus, for the real-time case, the YOLOv3[RGB]+TRK architecture performs very well as can be seen in Table 2. The two-stream 2D architectures that use LiteFlowNet optical flow stream, have delays of 0.11-0.12 sec. The two-stream networks with 3D architectures have delays of 0.19 sec, and the delays of the two-stream architectures that use the TV-L1 optical flow stream are 0.4-0.5 sec. The longer delays in this case result from the heavier computational load of the TV-L1 optical flow estimation process, which resulted in 400ms per frame, as detailed in Sec. IV.B.

V. DISCUSSION

The results of Tables 1-4 show that our tracking process significantly improved the results of all the network architectures we examined with our turbulence-affected dataset. In addition, according to the ablation studies of the tracking process (Table 6), it is clear that our Online Majority Class Voting algorithm has made a major contribution to successfully classifying actions correctly. YOLOv3[RGB]+YOLOv3[TV-L1]+TRK achieved the best results for the action detection task with our dataset and posted a 39.49% improvement compared to [9] at an IoU threshold of 0.2 (36.85% at an IoU threshold of 0.5), and additionally compared to [40] which gave poor results on our database. Also, the turbulence-affected optical flow stream in [9] made a minor contribution to the final result, and in [40]

TABLE 6. Results of the ablation studies of the tracking process.

IoU threshold			0.2			0.5		
dataset zoom			all	X100	X50	all	X100	X50
	Linking	Majority Class Voting						
FFSSD[RGB]+FFSSD[TV-L1]			52.79	58.96	46.61	46.53	53.96	39.09
FFSSD[RGB]+FFSSD[TV-L1]	✓		55.30	61.10	49.50	47.91	55.00	40.82
FFSSD[RGB]+FFSSD[TV-L1]	✓	✓	91.22	97.25	85.18	75.65	83.68	67.62
FFSSD[RGB]+FFSSD[LiteFlowNet]			45.73	49.64	41.82	42.62	47.02	38.22
FFSSD[RGB]+FFSSD[LiteFlowNet]	✓		46.38	49.78	42.38	42.98	46.65	39.30
FFSSD[RGB]+FFSSD[LiteFlowNet]	✓	✓	59.86	68.86	51.46	52.18	60.67	43.69
YOLOv3[RGB]+YOLOv3[TV-L1]			72.07	69.37	74.77	62.73	65.66	59.80
YOLOv3[RGB]+YOLOv3[TV-L1]	✓		75.68	74.41	76.94	64.44	68.76	60.13
YOLOv3[RGB]+YOLOv3[TV-L1]	✓	✓	**97.10**	**99.72**	**94.49**	**84.29**	**93.35**	75.23
YOLOv3[RGB]+YOLOv3[LiteFlowNet]			70.81	67.74	73.89	61.74	64.30	59.18
YOLOv3[RGB]+YOLOv3[LiteFlowNet]	✓		74.42	72.78	76.05	63.39	67.33	59.44
YOLOv3[RGB]+YOLOv3[LiteFlowNet]	✓	✓	97.05	**99.72**	94.39	84.20	93.24	75.16
YOLOv3[RGB]			70.54	67.73	73.34	61.53	64.25	58.81
YOLOv3[RGB]	✓		74.14	72.78	75.51	63.15	67.23	59.07
YOLOv3[RGB]	✓	✓	97.05	99.71	94.39	84.25	93.14	**75.36**
FFSSD[RGB]			42.79	46.55	39.03	40.28	44.64	35.91
FFSSD[RGB]	✓		44.14	48.21	40.08	41.60	45.80	36.91
FFSSD[RGB]	✓	✓	47.57	55.72	39.43	43.91	53.61	34.22
FFSSD-3D[RGB]			37.01	45.14	28.88	33.28	43.06	23.50
FFSSD-3D[RGB]	✓		37.90	45.81	30.00	33.70	43.11	24.30
FFSSD-3D[RGB]	✓	✓	63.68	85.90	41.46	58.20	80.20	36.20
FFSSD-3D[RGB]+FFSSD[TV-L1]			45.42	57.44	33.40	37.30	49.42	25.18
FFSSD-3D[RGB]+FFSSD[TV-L1]	✓		46.64	57.71	35.56	38.59	49.32	27.85
FFSSD-3D[RGB]+FFSSD[TV-L1]	✓	✓	91.82	98.48	85.16	74.82	81.29	68.34

TABLE 7. Execution time (in milliseconds) for each module of our pipeline. Since we use two GPUs in parallel, the time delay of the CNN detection stage in the two-stream cases is the longer delay of each stream.

	Flow	Detection	Tracking	Overall
FFSSD[RGB]+TRK	0	20	1	**21**
YOLOv3[RGB]+TRK	0	29	1	30
FFSSD-3D[RGB]+TRK	0	95	1	96
FFSSD[TV-L1]+TRK	380	20	1	401
YOLOv3[TV-L1]+TRK	380	29	1	410
FFSSD-3D[TV-L1]+TRK	380	95	1	476
FFSSD[LiteFlowNet]+TRK	90	20	1	111
YOLOv3[LiteFlowNet]+TRK	90	29	1	120
FFSSD-3D[LiteFlowNet]+TRK	90	95	1	186
YOLOv3[RGB]+YOLOv3[TV-L1]+TRK	380	29	1	410
YOLOv3[RGB]+YOLOv3[LiteFlowNet]+TRK	90	29	1	120
YOLOv3[RGB]+FFSSD-3D[TV-L1]+TRK	380	95	1	476
YOLOv3[RGB]+FFSSD-3D[LiteFlowNet]+TRK	90	95	1	186
FFSSD[RGB]+FFSSD[TV-L1]+TRK	380	20	1	401
FFSSD[RGB]+FFSSD[LiteFlowNet]+TRK	90	20	1	111
FFSSD[RGB]+FFSSD-3D[TV-L1]+TRK	380	95	1	476
FFSSD[RGB]+FFSSD-3D[LiteFlowNet]+TRK	90	95	1	186
FFSSD-3D[RGB]+FFSSD-3D[TV-L1]+TRK	380	95	1	476
FFSSD-3D[RGB]+YOLOv3[TV-L1]+TRK	380	95	1	476
FFSSD-3D[RGB]+FFSSD[TV-L1]+TRK	380	95	1	476
FFSSD-3D[RGB]+FFSSD-3D[LiteFlowNet]+TRK	90	95	1	186
FFSSD-3D[RGB]+YOLOv3[LiteFlowNet]+TRK	90	95	1	186
FFSSD-3D[RGB]+FFSSD[LiteFlowNet]+TRK	90	95	1	186

the optical flow stream is probably the reason of its failure on our database. However, in FFSSD[RGB]+FFSSD[TV-L1]+TRK, for example, the optical flow stream contributed a 44.25% improvement (at an IoU threshold 0.2), most likely due to our unique pre-and post-processing algorithm for the optical flow estimation based on turbulence characteristics(Eq. (1)). As expected, the results of TV-L1 are better than those for LiteFlowNet, but, for some applications, LiteFlowNet can be a good choice. For example, the FFSSD[RGB]+FFSSD[LiteFlowNet]+TRK achieves reasonable results, i.e., 97.45% for the detection-only task (see Table 5). When speed is an important parameter, LiteFlowNet may be preferred. For network architectures that include YOLOv3[RGB], the optical flow stream makes a minor contribution because YOLOv3[RGB]+TRK achieved a 97.05% mAP due to the improved performance of the YOLOv3 object detector combined with a reduction in the false and miss detections in the proposed tracking process. According the run time for YOLOv3[RGB]+TRK (see Table 7), it can be used for real-time applications. As we mentioned in Sec. 1, it was shown in [13] that Conv 3D improved the action detection results for the common UCF-101 and HMDB-51 datasets. On our dataset, the best results were achieved by a combination of two streams of Conv 2D, possibly because the random movements due to turbulence makes it somewhat more difficult for the 3D network to learn the pattern of the action without learning and fusing the optical flow data explicitly, especially in the case of the X50 zoom, where the objects are very small and their movements are comparable to the turbulence movements. We can see in Table 4 that in the case of the X100 zoom, when the movements of the objects are larger, the best results are achieved by FFSSD-3D[TV-L1]+FFSSD[RGB]+TRK. In addition, Fig. 8 and Fig. 9 show the results per action class. The networks YOLOv3[RGB]+YOLOv3[TV-L1]+TRK and YOLOv3[RGB]+TRK achieved the highest accuracy for all classes except crawling, especially for the X50 zoom.

In crawling at X50 zoom, the movements of the object are small relative to the turbulence movements; thus, it was more difficult for the network to distinguish between them. However, FFSSD[RGB]+FFSSD[TV-L1]+TRK achieved an improvement of 45.21% (at an IoU threshold of 0.5) compared to YOLOv3[RGB]+YOLOv3[TV-L1]+TRK. This suggests that FFSSD might be more accurate in the case of slow and small objects. In order to compare our method to our previous work [53], Table 5 shows the results of our modules for detection only (without referring to the correctness of the action classification). It is clear from this table that the improvement is significant for all our proposed architectures with respect to the detection task, and the best results are achieved with FFSSD-3D [TV-L1]+FFSSD[RGB]. Furthermore, it can be seen that at X50 zoom, FFSSD-3D[TV-L1]+FFSSD[RGB] achieved an improvement of 5.03% (at an IoU threshold of 0.2) compared to YOLOv3[RGB]+YOLOv3[TV-L1]. This result strengthens the claim, that FFSSD is more accurate in the case of small objects.

VI. CONCLUSION

We presented a novel online framework for human action detection in long-distance imaging affected by the atmospheric path. The air turbulence in such a case causes random spatio-temporal image movements that may be interpreted by methods developed for relatively stable environments, as moving objects. We use an efficient deep-learning strategy for the detection and classification of actions and combine techniques that consider the unique properties of the atmospheric path. These include the pre- and post-processing of the optical flow and a new online tracking algorithm based on the turbulence characteristics. The pre- and post-processing of the optical flow which reduces many false alarm motions is very important when there is a temporally dynamic ''noisy'' image background, mainly when the detection network is not sufficiently robust. We examined different types of network architectures for moving object detection. These types are combinations of various 2D and 3D networks (where 2D include YOLOv3 and FFSSD, and 3D include the FFSSD-3D that is our extension of I3D), and also one-stream and two-stream architectures. The architecture that achieved the best results was YOLOv3[RGB]+YOLOv3[TVL1]+TRK, and in case of small and slow object the FFSSD[RGB]+FFSSD[TV-L1]+TRK is preferred. Also, real-time application the one-stream architecture YOLOv3[RGB]+TRK is the most suitable. The tracking process that considers the turbulence properties, significantly increases the precision in all the network architectures we examined. Our approach achieved better performances compared to the state-of-the-art baseline for our unique dataset.

REFERENCES

[1] O. Haik and Y. Yitzhaky, ''Effects of image restoration on automatic acquisition of moving objects in thermal video sequences degraded by the atmosphere,'' *Appl. Opt.*, vol. 46, no. 36, p. 8562, Dec. 2007. [Online]. Available: https://www.osapublishing.org/abstract.cfm?URI=ao-46-36-8562

[2] K. Soomro, A. R. Zamir, and M. Shah, ''UCF101: A dataset of 101 human actions classes from videos in the wild,'' 2012, *arXiv:1212.0402*. [Online]. Available: http://arxiv.org/abs/1212.0402

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ''Rich feature hierarchies for accurate object detection and semantic segmentation,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[5] S. Saha, G. Singh, M. Sapienza, P. Torr, and F. Cuzzolin, ''Deep learning for detecting multiple space-time action tubes in videos,'' in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 58.1–58.13

[6] G. Gkioxari and J. Malik, ''Finding action tubes,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 759–768.

[7] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, ''Learning to track for spatio-temporal action localization,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3164–3172.

[8] X. Peng and C. Schmid, ''Multi-region two-stream R-CNN for action detection,'' in *Computer Vision—ECCV*. Amsterdam, The Netherlands: Springer, 2016, pp. 744–759.

[9] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin, ''Online real-time multiple spatiotemporal action localisation and prediction,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3657–3666.

[10] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, ''Fast optical flow using dense inverse search,'' in *Computer Vision—ECCV*. Amsterdam, The Netherlands: Springer, 2016, pp. 471–488.

[11] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, ''Learning spatiotemporal features with 3D convolutional networks,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.

[12] J. Carreira and A. Zisserman, ''Quo vadis, action recognition? A new model and the kinetics dataset,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4724–4733.

[13] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, ''AVA: A video dataset of spatio-temporally localized atomic visual actions,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6047–6056.

[14] J. Redmon and A. Farhadi, ''YOLOv3: An incremental improvement,'' 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[15] X. Xie, G. Cao, W. Yang, Q. Liao, G. Shi, and J. Wu, ''Feature-fused SSD: Fast detection for small objects,'' in *Proc. 9th Int. Conf. Graphic Image Process. (ICGIP)*, Apr. 2018, Art. no. 106151E.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, ''SSD: Single shot multibox detector,'' in *Computer Vision—ECCV*. Amsterdam, The Netherlands: Springer, 2016, pp. 21–37.

[17] C. Feichtenhofer, A. Pinz, and A. Zisserman, ''Convolutional two-stream network fusion for video action recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1933–1941.

[18] R. Girshick, ''Fast R-CNN,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[19] S. Ren, K. He, R. Girshick, and J. Sun, ''Faster R-CNN: Towards real-time object detection with region proposal networks,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[20] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, ''Mask R-CNN,'' in *IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[21] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, ''Feature pyramid networks for object detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ''You only look once: Unified, real-time object detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[23] J. Redmon and A. Farhadi, ''YOLO9000: Better, faster, stronger,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, ''Focal loss for dense object detection,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[25] B. K. P. Horn and B. G. Schunck, ''Determining optical flow,'' *Artif. Intell.*, vol. 17, nos. 1–3, pp. 185–203, Aug. 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370281900242

[26] A. Ranjan and M. J. Black, ''Optical flow estimation using a spatial pyramid network,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2720–2729.

[27] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.

[28] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1647–1655.

[29] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8934–8943.

[30] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8981–8989.

[31] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.

[32] L. Wang, Y. Xiong, Z. Wanga, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Computer Vision—ECCV*. Amsterdam, The Netherlands: Springer, 2016, pp. 20–36.

[33] Y. Wang, M. Long, J. Wang, and P. S. Yu, "Spatiotemporal pyramid network for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1529–1538.

[34] Z. Yang, J. Gao, and R. Nevatia, "Spatio-temporal action detection with cascade proposal and location anticipation," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[35] Y. Ye, X. Yang, and Y. Tian, "Discovering spatio-temporal action tubes," *J. Vis. Commun. Image Represent.*, vol. 58, pp. 515–524, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047320318303468

[36] R. Girdhar, J. Joao Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 244–253.

[37] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick, "Long-term feature banks for detailed video understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 284–293.

[38] X. Yang, X. Yang, M.-Y. Liu, F. Xiao, L. S. Davis, and J. Kautz, "STEP: Spatio-temporal progressive learning for video action detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 264–272.

[39] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action tubelet detector for Spatio-temporal action localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4415–4423.

[40] D. Zhang, L. He, Z. Tu, S. Zhang, F. Han, and B. Yang, "Learning motion representation for real-time Spatio-temporal action localization," *Pattern Recognit.*, vol. 103, Jul. 2020, Art. no. 107312.

[41] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2556–2563.

[42] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," 2017, *arXiv:1705.06950*. [Online]. Available: http://arxiv.org/abs/1705.06950

[43] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," 2018, *arXiv:1808.01340*. [Online]. Available: http://arxiv.org/abs/1808.01340

[44] E. Chen and Y. Yitzhaky. *Raw Videos*. Accessed: Feb. 1, 2021. [Online]. Available: https://drive.google.com/drive/u/1/folders/1rBawiZ6soX9TQN5X4P_BHknrbZu%4vXua

[45] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576. [Online]. Available: http://papers.nips.cc/paper/5353-two-stream-convolutional

[46] C. Zach, T. Pock, and H. Bischof, "A duality based approach for real-time tv-l1 optical flow," *Pattern Recognit.*, vol. 4713, pp. 214–223, Sep. 2007.

[47] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS J. Photogramm. Remote Sens.*, vol. 159, pp. 296–307, Jan. 2020.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016, pp. 265–283.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2015, pp. 1–15.

[51] E. Chen and Y. Yitzhaky. *Our Code*. Accessed: Feb. 1, 2021. [Online]. Available: https://github.com/bgulab/Action-Detection-Long-Distance-Imaging

[52] E. Chen and Y. Yitzhaky. *Our Database*. Accessed: Feb. 1, 2021. [Online]. Available: https://drive.google.com/drive/folders/1ODs0vVZaJnxBNMM-qJiDHCYYT9-Im_3%6

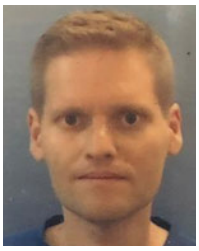
[53] E. Chen, O. Haik, and Y. Yitzhaky, "Detecting and tracking moving objects in long-distance imaging through turbulent medium," *Appl. Opt.*, vol. 53, no. 6, p. 1181, Feb. 2014. [Online]. Available: https://www.osapublishing.org/abstract.cfm?URI=ao-53-6-1181

[54] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[55] O. Oreifej, X. Li, and M. Shah, "Simultaneous video stabilization and moving object detection in turbulence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 450–462, Feb. 2013.

[56] E. Chen, O. Haik, and Y. Yitzhaky, "Classification of thermal moving objects in atmospherically degraded video," *Opt. Eng.*, vol. 51, no. 10, Jun. 2012, Art. no. 101710.

ELI CHEN received the M.S. degree in electro-optics from Ben Gurion University, Israel, in 2008, where he is currently pursuing the Ph.D. degree with the Electro-Optics Unit. His research interest includes computer vision in severe imaging conditions.



OREN HAIK received the Ph.D. degree in computer vision from Ben Gurion University, Israel, in 2008. He currently works at HP Indigo for the last ten years with the Computer Vision Team. During his work, he has developed real time inspection systems, printing quality measures, and halftoning algorithms. He is also developing various deep learning-based solutions for HP Indigo presses.

YITZHAK YITZHAKY received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Ben Gurion University, Israel, in 1993, 1995, and 2000, respectively. From 2000 to 2002, he was a Postdoctoral Research Fellow with the Harvard Medical School, Schepens Eye Research Institute, Boston, MA. He is currently with the Department of Electro-Optics Engineering, Ben Gurion University. His main research interests include computer vision, 3-D image analysis, image restoration, and understanding, mostly in severe imaging conditions.