MOTrack: Real-time Configuration Adaptation for Video Analytics through Movement Tracking

Fubao Wu, Lixin Gao, Tian Zhou and Xi Wang
Department of Electrical and Engineering
University of Massachusetts Amherst, Amherst, MA 01002, USA
Email: fubaowu@umass.edu, lgao@engin.umass.edu, tzhou@umass.edu, xwang4@umass.edu

***Abstract*—Video analytics has many applications in traffic control, security monitoring, action/event analysis, etc. With the adoption of deep neural networks, the accuracy of video analytics in video streams has been greatly improved. However, deep neural networks for performing video analytics are compute-intensive. In order to reduce processing time, many systems switch to the lower frame rate or resolution. State-of-the-art switching approaches adjust configurations by profiling video clips on a large configuration space. Multiple configurations are tested periodically and the cheapest one with a desired accuracy is adopted. In this paper, we propose a method that adapts the configuration by analyzing past video analytics results instead of profiling candidate configurations. Our method adopts a lower/higher resolution or frame rate when objects move slow/fast. We train a model that automatically selects the best configuration. We evaluate our method with two real-world video analytics applications: traffic tracking and pose estimation. Compared to the periodic profiling method, our method achieves 3%–12% higher accuracy with the same resource cost and 8–17x faster with comparable accuracy.**

***Index Terms*—Video Stream, Video Analytics, Object Tracking, Machine Learning**

I. Introduction

The proliferation of cameras deployed in many enterprises and cities [1] drives the demand for video analytics. Object detection and tracking as the common parts of video analytics focuses on detecting and tracking objects from video streams. Many applications in traffic control, business intelligence, action/pose analysis, and VR/AR are built on top of object detection and tracking [2], [3]. For example, a traffic control center in a city wants to detect and count cars over an intersection real-time with a good detection accuracy to estimate the traffic. Human poses and actions could be detected real-time for virtual games or dance guidance. Recently, object detection and tracking rely on deep neural network (DNN) models for more accurate inferences. Each DNN module corresponds to a frame rate and resolution, which is referred to as a configuration.

The selection of a configuration impacts the accuracy and resource consumption of object detection and tracking. A configuration with high frame rate and resolution is considered as an expensive configuration, and a configuration with low frame rate and resolution is considered as a cheap configuration. For example, an expensive configuration with 25 fps and 1080p generally leads to a higher accuracy than a cheap configuration with 1 fps and 240p, but consumes more resources. The "best" configuration is the configuration that achieves a desired accuracy with the least resource consumption.

The state-of-the-art configuration adaptation falls into two classes: one-time profiling and periodical profiling. One-time profiling [4]–[6] aims to profile all configurations only once during the beginning of a video (e.g., 10 seconds), and then chooses the best configuration above an accuracy requirement for the video. However, it uses a fixed configuration for the whole video and neglects the intrinsic dynamics of video contents. Periodic profiling [2], [7], [8] determines the configuration for every interval of a video through profiling the first few frames of the interval. However, profiling periodically costs computing resources and incurs more processing time.

Video content exhibits temporal and spatial characteristics. Due to these characteristics, objects keep the same or similar movement in a short period of time. Therefore, the video content could be quantified with object movement in this period. Objects moving fast usually need expensive configurations to track, and cheap configurations suffice for slow objects.

In this paper, we capture the object movement from past video analytics results to guide the selection of frame rate and resolution to adapt configurations over time. Leveraging this, a machine learning-based classification method, MOTrack, is utilized to obtain the relationship between object movement and the best configuration. We obtain the estimated object movement and corresponding configurations as labeled training data instances to automatically learn the mapping between them. Through extensive experiments on traffic tracking and pose estimation applications on large video datasets, we demonstrate the superiority and effectiveness of MOTrack. Compared with two state-of-the-art configuration adaptation approaches, MOTrack could achieve the accuracy threshold goal with 8–17x less computation resources.

Our main contributions are as follows:

1) We investigate the impact of object movement on configurations and propose to utilize the object movement to guide configuration adaptation.
2) We propose a machine learning-based classification method to predict the configuration for future frames, which significantly reduces the cost of configuration adaptation.
3) We experimentally demonstrate the effectiveness and efficiency of MOTrack on traffic tracking and pose estimation applications.

The rest of this paper is organized as follows. We introduce the motivation in Section II. Section III describes our algorithms in detail. Experimental evaluation is presented in Section IV. Related work is introduced in Section V. We conclude our paper in Section VI.

II. Motivation

Object detection and tracking in video analytics is to detect object positions and track objects for every frame. Traditionally, every frame is expected to be processed with an expensive DNN model for detection. However, when an object moves slowly in a short period of the video, the frames in that period could be skipped for an expensive DNN model. We just need to estimate the position of the object in the skipped frames from its previous location and momentum. Similarly, a low resolution can be adopted for a slow-moving object. This could save huge computation resources while still maintaining the required accuracy. Therefore, the object movement could help make a decision for the needed frame rate and resolution.

The object movement could be quantified with the object moving velocity, which could be measured based on the distances of detected object keypoints between frames over time (we call it keypoint movement velocity). Given a minimum accuracy requirement, if the movement velocity is high in a short period of the video, the high frame rate/resolution would be needed to satisfy the accuracy requirement, and vice versa.

Fig. 1 shows the (normalized) keypoint movement speed (velocity magnitude) for every 1 second interval, and the necessary frame rate and resolution in a 4-minute dancing video clip in the pose estimation application [9]. To show the motivation effectively, we assume there is only one high frame rate/resolution and one low frame rate/resolution available. We calculate the object moving speed for every second interval. Given a minimum accuracy requirement, a frame rate/resolution will be selected for each interval. We could see that when the movement speed increases to a high value in periods of around seconds 15, 30, 110, 130, 180, and 230 highlighted with grey backgrounds, high frame rates/resolutions are necessary to achieve the accuracy requirement. Conversely, we find most of the time the object moves at a low speed and thus a low frame rate/resolution is needed. It shows the strong correlation between the movement speed and the frame rate or resolution needed.

III. Movement Features and Prediction Model

We propose to quantify the correlation of target objects in different periods of a video with some movement features. A movement feature of one target object describes how the keypoints of the object move. In addition to the movement features describing each keypoint, we also consider some dense features for the dynamics of the whole object. Then, we develop a model that learns how to dynamically adjust the configurations based on the proposed features.

We utilize the movement information of target objects to dynamically decide which frames should be analyzed in detail and which frames can be skipped. The physical movement of an object is inertial, which leads to the velocity and acceleration of the object. Then for skipped frames, we estimate the location of the target object.

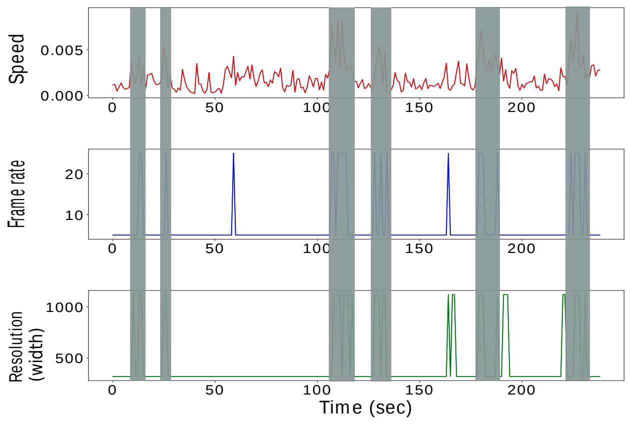


Fig. 1: The correlation between movement speed and the necessary frame rate/resolution for the accuracy requirement

We first introduce some key concepts and notations of our algorithm. Our algorithm tries to find small periods of a video for processing with different resolutions and frame rates. We attach a timestamp t_i to each frame i considering a real-time video playout rate of 25 frames/second. It can be derived easily by counting how many frames are skipped from the playout rate of the video. Similarly, our algorithm also memorizes the resolution of each frame. The width and height of a frame i are represented with W_i and H_i respectively. Therefore, a sequence of frames can be represented as follows:

$$[(t_1, W_1, H_1), \cdots, (t_n, W_n, H_n)] \tag{1}$$

Then the position $p(i)$ of one point with coordinate (x, y) at frame i is represented with the following vector (normalized):

$$\vec{p}(i) = [x(i)/W(i), y(i)/H(i)]^T \tag{2}$$

where $x(i)$ and $y(i)$ indicate X and Y coordinates of the point in frame i. When there are multiple keypoints such as a, b, c in one object, the corresponding positions are denoted as $p_a(i)$, $p_b(i)$ and $p_c(i)$, respectively.

A. Movement Features

We extract the movement information of each object into a set of features from its past video analytics results. The features are described below.

1) Keypoint Movement Velocity: An object in a frame can be represented as a set of keypoints detected, such as 4 bounding box corner points in an object. This keypoint movement velocity feature captures how a keypoint moves across frames in the screen space. We measure the pixel velocity of a point p using consecutive frames. Assume that we have two consecutive frame $i-1$ and i taking at time t_{i-1} and t_i respectively. Then, the velocity of this point at frame i can be quantified as:

$$\vec{v_f}(i) = \frac{\vec{p}(i) - \vec{p}(i-1)}{t_i - t_{i-1}} \tag{3}$$

where $\vec{p}(i-1)$ and $\vec{p}(i)$ indicate the normalized position of the object at frame $i-1$ and frame i in the screen space. Therefore, for an object with n keypoints, we derive the keypoint velocity feature of the object on frame i as a vector of the $\vec{v_f}$ for all keypoints as follows:

$$\mathbf{V_f}(i) = [\vec{v_f^1}(i), \cdots, \vec{v_f^k}(i), \cdots, \vec{v_f^n}(i)] \tag{4}$$

where $\vec{v_f^k}(i)$ represents the velocity of keypoint k at frame i.

2) Object Movement Velocity: To capture more fine-grained features, we propose another object movement feature to complement previous movement features. Previous keypoint movement velocity feature only captures the corner's movement of an object across frames, which is also noisy. Here the object movement velocity is based on the robust optical flow [10] calculated only with pixel values of detected objects. It could recover the object's motion at each pixel and apparent velocities of movement of brightness pattern in a frame.

We capture the optical flow velocities of all the pixel points as our object movement velocity. To capture each pixel point's optical flow, we consider two consecutive frames $i-1$ and i taking at time t_{i-1} and t_i. For one pixel k at location (x, y, t) with its brightness $I(x, y, t)$ from frame $i-1$ to $(x+\Delta x, y+\Delta y, t+\Delta t)$ in frame i with its brightness $I(x+\Delta x, y+\Delta y, t+\Delta t)$, its brightness is assumed to keep constancy and transformed with the Taylor series, we obtain this equation,

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = -\frac{\partial I}{\partial t} \tag{5}$$

Where u and v are the optical flow of point (x, y) along x and y direction. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of the image intensity at (x, y, t) in the corresponding directions.

To get u and v with the one equation, we extend with more points over a small neighborhood by introducing nearby pixels within a window and solve the multiple equations with the least-squares principle [10]. Then, the optical flow velocity for that pixel k inside an object in frame i is $\vec{O}^k(i) = (u, v)^k(i)$.

Therefore, for one object with n pixels in the detected bounding box, we have movement feature $\mathbf{O}(i)$ at frame i,

$$\mathbf{O}(i) = [\vec{O}^1(i), \vec{O}^2(i), ..., \vec{O}^k(i), ..., \vec{O}^n(i)] \tag{6}$$

3) Keypoint Relative Movement Velocity: A highly relative movement among an object keypoints would indicate a drastic change. For some applications containing multiple correlated keypoints, such as pose estimation with 17 keypoints (e.g., arms, hands, etc.), the relative velocity can be adopted among different keypoints. We should use a higher resolution to check what happens to the object or even double-check whether there is something wrong with the previous tracking result. This feature captures the change by relative keypoint movements.

The relative velocity feature $\mathbf{V_r}$ of an object at the frame i is a triangle matrix of each pair of the keypoints among n keypoints as the following equation.

$$\mathbf{V_r}(i) = \begin{bmatrix} 0 & \vec{v_r}^{1,2}(i) & \vec{v_r}^{1,3}(i) & \cdots & \vec{v_r}^{1,n}(i) \\ 0 & 0 & \vec{v_r}^{2,3}(i) & \cdots & \vec{v_r}^{2,n}(i) \\ \cdots & & & & \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{7}$$

where $\vec{v_r}^{p,q}(i),\ p, q \in [1, n]$ indicates the relative velocity from point p compared to point q at frame i. It is formulated as the following equation.

$$\vec{v_r}^{a,b}(i) = \frac{\vec{r}_{a,b}(i) - \vec{r}_{a,b}(i-1)}{t_i - t_{i-1}} \tag{8}$$

where $\vec{r}_{a,b}(i) = \vec{p}_a(i) - \vec{p}_b(i)$ indicates the relative vector distance from point a to point b at frame i. $\vec{p}_a(i)$ and $\vec{p}_b(i)$ are the coordinates of two points a and b on the same object at frame i.

For pose estimation application, we only consider the pairs in relative velocity features $\mathbf{V_r}$ that have high impacts on relative movements, such as the wrists to shoulders, and the ankles to hips.

4) Object Size Change Speed: In addition to object movement velocity features, we also capture the whole object morphing by the object size change. The object size in an image is a great indicator for shape morphing and Z-axis movement of the real-world object. The change ratio of the object size represents how fast the morphing or Z-axis movement happens.

The size can be directly acquired from the video analytics result. For simplicity, we use the size of the bounding box of the target object to approximate the object size. By representing the size of the object in frame i with S_i, we can formulate the object size change speed feature $\mathbf{SC}(i)$ at frame i as follows:

$$\mathbf{SC}(i) = \frac{S_i - S_{i-1}}{t_i - t_{i-1}} \tag{9}$$

When the bounding box is used, the size S_i can be computed as follows where $x_{max} = \max_k \vec{p^k}x$ represents the maximum X-coordinate among all the keypoints of the object.

$$\mathbf{S}(i) = \frac{(x_{max} - x_{min}) \cdot (y_{max} - y_{min})}{W_i H_i} \tag{10}$$

Similarly, x_{min}, y_{min} and y_{max} denote the minimum X-coordinate, minimum Y-coordinate, and maximum Y-coordinate respectively.

B. Feature Smoothing

The feature estimation using two frames is not reliable enough, as it can be affected by some random noises or a single failure in tracking. Since physical movement in the real-world is inertial, we use the exponential moving average to smooth the features as the final estimation. For a feature $\mathbf{F}$, we derive its final value $\hat{\mathbf{F}}(i)$ for frame i based on its instantaneous estimation $\mathbf{F}(i)$ and the smoothed value on the previous $\hat{\mathbf{F}}(i-1)$. We formulate this procedure as follows:

$$\hat{\mathbf{F}}(i) = \alpha\mathbf{F}(i) + (1-\alpha)\hat{\mathbf{F}}(i-1) \tag{11}$$

where $\mathbf{F}$ can be $\mathbf{V_f}$, $\mathbf{V_r}$, $\mathbf{SC}$ or $\mathbf{O}$, and α is the smooth factor in $[0, 1]$. Therefore, we use $X = [\hat{\mathbf{V}}_\mathbf{f}(i), \hat{\mathbf{V}}_\mathbf{r}(i), \hat{\mathbf{SC}}(i), \hat{\mathbf{O}}(i)]$ as our features.

C. Estimation of Skipped Frames

We estimate the positions of a target object in the skipped frames through their previous locations and movement information. A frame j's object position is assumed to be inferred with our DNN detector. Assuming that there are S skipped frames, we would estimate the keypoint p in the skipped frames, $\vec{p}(j+1)$, $\vec{p}(j+2)$,..., $\vec{p}(j+S)$ with the movement velocity vector $\hat{F}(j)$. If the current frame j has one object position with a vector coordinate $\vec{p}(j)$ in x and y dimension, and the estimated movement velocity for this position is $\hat{F}(j)$, then that position coordinate in the next skipped frame k is estimated as:

$$\vec{p}(k) = \vec{p}(k-1) + \frac{\hat{F}(j)}{S} \tag{12}$$

where $k = j+1, j+2, ..., j+S$.

D. Model for Configuration Prediction

We train a model that predicts which configurations are used for future frames. The frame rate determines the time interval until the next frame. Therefore, the training data is a set of features and configuration pairs generated from many video streams. The ground truth configuration is the configuration that requires the lowest computing resources, but still satisfies a certain accuracy requirement.

We model this problem as a multi-class and multi-output classification problem. Among many classical classification models, random forest classification is an efficient machine learning model, which minimizes both bias and variance on an ensemble of decision trees and deals with high dimensional data very well. A complex and expensive deep learning model is left for future work. We adopt the random forest classification model [11] for prediction and obtain accurate results. In our prediction model, frame rate and resolution are two targets, which are learned respectively with a classification model.

IV. Experimental evaluation

We evaluate our video analytics method, MOTrack, on two applications–pose estimation and traffic tracking, on one server with two Quadro RTX 5000 GPU, and compare it with two state-of-the-art configuration adaption methods.

A. Applications and Datasets

Traffic tracking localizes and tracks the pedestrian and vehicles indoors or outdoors from videos based on the DeepSort model [12]. The video analytics result in each frame for each object is represented by the object's bounding box. Pose estimation localizes anatomical keypoints or parts from a human body [9] based on the OpenPose model [13]. Each object in a frame has 17 keypoints as the video analytics result.

Existing video stream datasets do not have enough high resolutions or length of videos available for video analytics . We create video stream datasets from Youtube (dataset sources are public in: https://rb.gy/pofzby). Traffic tracking dataset contains 60 video clips of survelliance indoors, and mixed pedestrians and vehicles on the street. The objects in these videos move generally fast, but most of time are steady, with their moving speed difference between consecutive second averaging 15 pixels/second (p/s). However, the pose estimation dataset consists of 75 video clips of dancing, body conditioning and other workouts, in which an object moves dramatically with its moving speed difference averaging 23 p/s. Table I shows the dataset statistics.

TABLE I: Video dataset statistics

Dataset	Total length (min).	Object no. (each video)	Keypoint no. (each object)	Speed difference
Traffic tracking	600	Various	4	15 p/s
Pose estimation	750	Single	17	23 p/s

To train a random forest model, we have to obtain the training data ground truth–configuration for each feature, which is not practical to label manually. We utilize a heuristic approach to find the corresponding configuration for each interval in video analytics to generate labeled data instances. We explore all the available frame rate and resolution configurations. Given a minimum accuracy threshold, we select the lowest frame rate and resolution configuration as the ground truth to achieve the minimum processing time while maintaining the accuracy above that threshold in each estimated interval.

In our prediction model, we split the video dataset into a training dataset and test dataset for each application. Specifically, we use each video clip as the test dataset and other video clips as the training dataset. 5-fold cross-validation is utilized to obtain the best model, then it is applied to the test dataset. In our experiment, we adopt one minimum accuracy threshold of 0.92 as an example to show our results. Other thresholds show the similar results. The processing time shown in all the figures below is the average total time for processing each one-second interval of videos.

B. Metrics and Configurations

Traffic tracking uses detected bounding boxes' intersection over union to calculate the accuracy [14]. Pose estimation accuracy is calculated with the object keypoint similarity metric from COCO pose estimation dataset [15]. The ground truth for measuring video analytics accuracy is based on the most expensive configuration with the highest frame rate and resolution [2].

For the prediction model, we care about how accurate our prediction model predicts the correct configuration, so we use the "accuracy" to measure the average prediction accuracy for predicting the frame rate and resolution. There are 25 classes for 1 frame/second, 2 frame/second, ..., and 25 frame/second. The video resolutions considered are: 1120x832, 960x720, 640x480, 480x352 and 320x240. In total, we have 125 classes in the configuration space.

C. Impact of Features on Prediction Accuracy

Here we show the different features' impact on the prediction accuracy. We evaluate the impact of each feature on the prediction accuracy by removing the features one by one for each application. Table II shows the impact of different features on the prediction accuracy. "All" means we use all the proposed features for each application. "x-" symbol ahead of each feature name denotes we remove this feature from "All". It shows our prediction method achieves an accuracy from 0.794 to 0.865 with all the proposed features. When we remove one of the features, the accuracy has been degraded to a different extent. Each feature is indispensable and contributed in some way to predict the configuration.

TABLE II: Impact of features on prediction accuracy

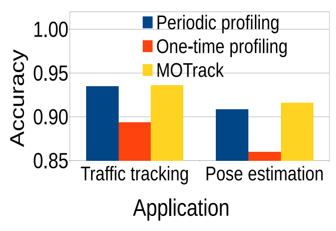
Feature\Application	Traffic tracking	Pose estimation
All	0.865	0.794
x- Keypoint movement velocity	0.838	0.785
x- Keypoint relative velocity	N/A	0.783
x- Object movement velocity	0.803	0.766
x- Object size change	0.846	0.789

D. Video Analytics Performance

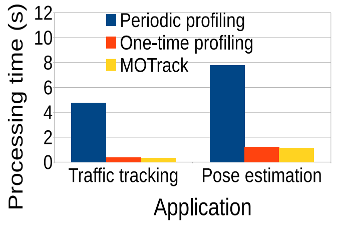
We show the performance of MOTrack on the whole video datasets and one-minute video clips, and compare it with two state-of-the-art configuration adaptation methods: one-time profiling and periodic profiling.

One-time profiling is operated only once at the beginning of video analytics [5]. The profiling interval x is 10 sec. For periodic profiling [2], the profiling interval t for each time window is 1 sec, and the time window is 4 sec, which are same parameter settings from the references for comparisons.

Fig. 2(a) shows the accuracy and Fig. 2(b) shows the processing time. From these results, MOTrack achieves 3%-12% higher accuracy with the same processing time compared to one-time profiling and 8–17x faster with the similar accuracy compared to periodic profiling.



(a) Accuracy



(b) Processing time

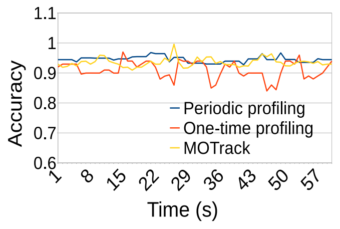
Fig. 2: Comparison of MOTrack with one-time profiling and periodic profiling on traffic tracking and pose estimation

To check the configuration adaptation effects in detail, we test on a one-minute video clip for each application, and show the comparison results in Fig. 3 and Fig. 4.

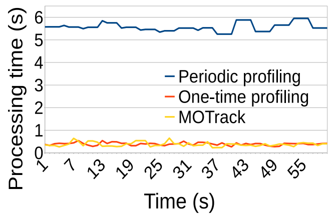
Fig. 3 shows the accuracy and processing time on traffic tracking. Compared to periodic profiling, we can see that MOTrack maintains a similar accuracy over time to periodic profiling, but much less processing time over time. This is because MOTrack has more frequent adaptation operations based on frames, but with an inexpensive adaption algorithm. Compared to one-time profiling, MOTrack maintains a much better accuracy over time and costs similar processing time.

For pose estimation on Fig. 4, MOtrack also demonstrates the similar accuracy over time compared to periodic profiling and shows more frequent configuration adaptation over time due to the drastic changes of human pose movements. Meanwhile, MOTrack maintains a much better accuracy over time and takes less processing time compared to one-time profiling.

Fig. 5 shows how close our configuration adaptation to the ground truth (configuration) on those video clips. The ground truth has the minimum processing time above the accuracy requirement. MOTrack obtains a very close result over time, with only an average of 5% more time than the ground truth.

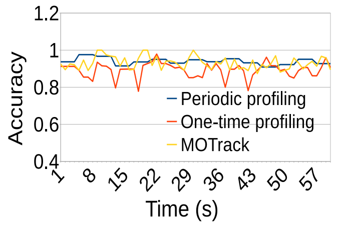


(a) Accuracy

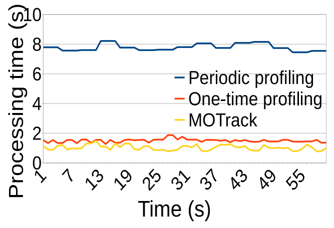


(b) Processing time

Fig. 3: Accuracy and processing time for one video clip on traffic tracking

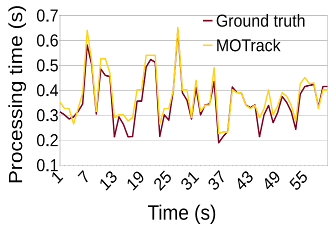


(a) Accuracy

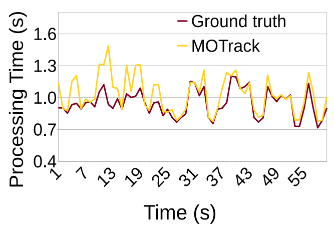


(b) Processing time

Fig. 4: Accuracy and processing time for one video clip on pose estimation



(a) Traffic tracking



(b) Pose estimation

Fig. 5: Processing time of MOTrack and ground truth configuration for one video clip on traffic tracking and pose estimation

E. Time for Configuration Adaptation

Here we compare MOTrack's configuration adaptation time with periodic profiling's on the test datasets. MOTrack adopts a machine learning-based prediction to do the configuration adaptation. Periodic profiling uses profiling configurations to adapt configurations.

Fig. 6 shows the average (configuration) adaptation time on the two applications. MOTrack takes about 97%-98% less time than the periodic profiling's adaptation time. The efficiency of configuration adaptation has been greatly improved with the MOTrack method.

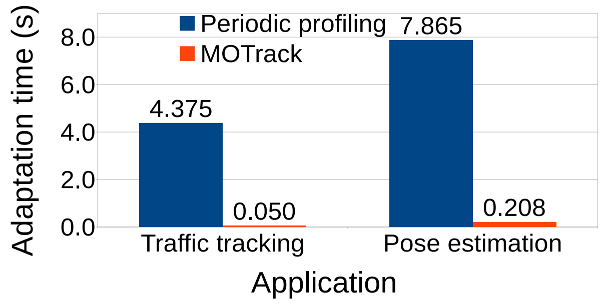


Fig. 6: Configuration adaptation time of MOTrack compared with periodic profiling

V. Related work

Existing research for video pipeline resource management and optimization mainly develops algorithms through specialized DNN models or through adjusting configurations with costly profiling [2], [5], [7], [16]. MCDNN [4], NoScope [17] and Focus [18] optimize video analytics with resource-light specialized DNN models to detect objects. The core of specialized DNN models is to train compressed DNN models with fewer layers or parameters on a few objects that typically appear on video streams. Some other researchers consider the temporal and spatio-temporal characteristics of videos. Samvit et al. [19] utilize the cross camera spatio-temporal characteristics to remain or improve inference for scalable camera deployment. Our work effectively utilizes the spatio-temporal characteristics to capture object movement features for configuration adaptation dynamically.

VideoStorm [5] processes live video streams over large clusters by profiling each video query in the cluster and change configurations to maximize the performance. AWStream [7] and JetStream [8], however, consider the wide-area network changing by profiling video query to achieve the trade-off of accuracy and bandwidth. Chameleon [2] considers profiling video segments periodically to obtain the best configuration. Our paper avoids the fixed period scheduling and overcomes the expensive profiling for configuration adaptation.

VI. Conclusion

In this paper, we propose a configuration adaption algorithm for video analytics through movement tracking. Considering estimating object movement information from past object tracking results, we devise a machine learning-based method to predict effectively and efficiently the configuration over time dynamically. This reduces the cost of expensive profiling and overcomes the fixed period of configuration adaptation. Our results suggest that our method can make smart decisions under different video analytics applications, which achieves better accuracy and less resource cost compared to state-of-the-art methods.

VII. Acknowledgment

This work was supported in part by National Science Foundation Grants CNS-1815412 and CNS-1908536.

References

[1] "Artificial intelligence surveillance cameras security." https://www.theverge.com/2018/1/23/16907238/artificial-intelligence-surveillance-cameras-security, 2018, accessed: 2018-1-23.

[2] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[3] L. Chen, H. Ai, R. Chen, and Z. Zhuang, "Aggregate tracklet appearance features for multi-object tracking," *IEEE Signal Processing Letters*, vol. 26, no. 11, pp. 1613–1617, 2019.

[4] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 123–136.

[5] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 377–392.

[6] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 426–438.

[7] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 236–252.

[8] A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman, "Aggregation and degradation in jetstream: Streaming analytics in the wide area," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 275–288.

[9] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[10] M. Kharbat, N. Aouf, A. Tsourdos, and B. A. White, "Robust brightness description for computing optical flow." in *BMVC*. Citeseer, 2008, pp. 1–10.

[11] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[12] X. Hou, Y. Wang, and L.-P. Chau, "Vehicle tracking using deep sort with low confidence track filtering," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–6.

[13] G. H. Martınez, "Openpose: Whole-body pose estimation," 2019.

[14] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, "Unitbox: An advanced object detection network," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 516–520.

[15] M. R. Ronchi and P. Perona, "Benchmarking and error diagnosis in multi-instance pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[16] W.-J. Kim and C.-H. Youn, "Lightweight online profiling-based configuration adaptation for video analytics system in edge computing," *IEEE Access*, vol. 8, pp. 116 881–116 899, 2020.

[17] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing deep cnn-based queries over video streams at scale." *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1586–1597, 2017.

[18] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 269–286.

[19] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, "Scaling video analytics systems to large camera deployments," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, 2019, pp. 9–14.