TVV: Real-Time Visual Identity and Tracking with Edge Computing

Xinpeng Zhang[1], Junchen Guo[1], Chunya Liu[1], Jie Zhou[1], Yao Luo[1], Long Liu[1],
Meng Jin[1], Ziqiang Zhou[2], Zhoubin Liu[2]
[1]School of Software and BNRist, Tsinghua University
[2]State Grid Zhejiang Electric Power Research Institute, Hangzhou, Zhejiang, 310000, China
{zhangxp16, gjc16, liuchuny17, zhouj15, luo-y17, liulong16}@mails.tsinghua.edu.cn,
mengj@mail.tsinghua.edu.cn, jx_zzq@sina.com, jxliuzb@qq.com

Abstract

Video surveillance today has become pervasive, making visual identification and tracking technology attractive to a broad class of applications like traffic counting, crime tracking, and Blockchain. However, visual tracking is also a victim of the ubiquity of surveillance camera: a huge amount of data that generated by the cameras leads to severe congestion problem, which decreases the frame rate and in turn affects the tracking accuracy. In this paper, we present TVV, a real-time visual tracking system that leverages edge computing to support accurate and continuous tracking in large scale areas. The design of TVV is based on a insight that almost 80% of frames in a video stream exhibit high quality, and such frames can be processed on the edge nodes using a lightweight filtering method named KCF. Based on this insight, TVV adaptively load the visual tacking task on the edge or the server, based on the quality of the currently generated frame. In this way, the traffic load is largely decreased, without sacrificing the tracking accuracy. Our experimental result show that the average frame rate of TVV achieves 45.75 fps, outperforming most state-of-the-art visual tracking approaches.

1 Introduction

We have entered a world where internet-enabled surveillance cameras pervade our daily life, providing security for the stores, airports, factories, campuses, roadways, etc. Visual tracking is a fundamental task in video surveillance system. Accurate and continuous tracking of the objects is prerequisite for many important functions such as traffic counting, crime tracking, and Blockchain.

However, achieving the requirement of both *accuracy* and *continuity* tracking is still a conundrum today, especially in large scale scenarios. Specifically, to cover a large scale area, such as a store, hundreds of cameras should be deployed at every corner of the store. Consider that the camera usually generate video frames with a 20-30 fps frame rate, the resulting huge amount of data will inevitably incurs severe congestion problem and further increases the network delay and packet loss rate. This finally decreases the frame rate and in turn affects the tracking accuracy.

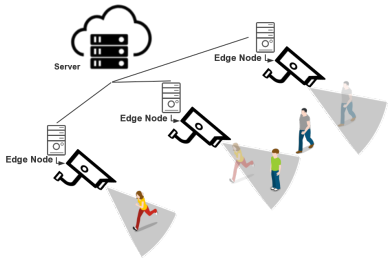


Figure 1. The cooperative processing of edge devices and server in the system.

So a key issue is how to reduce the high traffic load that generated by the ubiquitous cameras. Obviously, cutting down the number of camera brings too many blind areas; reducing the sampling rate at the camera or sacrificing the quality of the transmitted video frame reduces the tracking accuracy. In this circumstance, we ask why not leverage edge computing to preprocess the data on the edge nodes, and transmit only the analytic result (which is much smaller than the raw data) to the network.

However, a problem in realizing this vision is the limited computing capacity of the edge nodes. Specifically, to achieve high accuracy and robustness, today's visual tracking methods typically exploit deep convolutional neural network (CNN) to extract the target in the frame. The advantage of using CNN for target extracting is that it is able to handle many problems in real world videos such as occlusion, pose variations, illumination changes, fast motion, and background clutter. We terms such video frame as *polluted frame*, and others as *clear frame*. However, consider the limited computing capacity of the edge nodes, we cannot implement the resource-intensive CNN algorithm on them.

Fortunately, we find that: i) in 80% of the time, the generated video frames are clear frames; ii) in the clear frames,

International Conference on Embedded Wireless
Systems and Networks (EWSN) 2019
25–27 February, Beijing, China

ISBN: 978-0-9949886-3-8

the location of the target can be identified by a lightweight filtering method named KCF. That is to say, we can indeed process the majority frames (the clear frames) on the edge nodes, and transmit only the polluted frames to the server for more robust processing. In this way, the network load is reduced, without sacrificing the tracking accuracy.

Based on the above insights, we present TVV, a real-time visual tracking system that leverages edge computing to enable *accurate* and *continuous* visual tracking to scale to hundreds of cameras. The design principle of TVV is to adaptively load the visual tacking task on the edge or server, based on the quality of the currently generated frame (i.e., a clear frame or a polluted frame). A practical challenge we meet here is we can hardly distinguish between the polluted frame and the clear frame before the frame is processed. In this situation, how to ensure correct adapting between edge based tracking and server based tracking? In the design of TVV, all the generated frames are first processed by the edge node. To avoid incorrect adapting (i.e., processing the polluted frame on the edge node) which will incur large tracking error, the edge node will periodically verify whether the tracking on the edge has been successful. Once an unsuccessful tracking is detected, TVV will trigger a validating process, where the raw data will be send to the server for more robust tracking.

The contribution of this paper is summarized as follows:

- We propose a novel edge computing based architecture for accurate and continuous visual tracking in large scale areas. In the architecture edge based tracking and server based tracking are alternated to achieve both high tracking accuracy and large monitoring area with limited bandwidth.
- We design a validation method to estimate the confidence of the edge based processing and judge whether the tracking has been successful based on a classification algorithm. This ensures correct adapting between edge based tracking and server based tracking.
- We evaluate the performance of TVV in DUKE Dataset, the experimental results show that the average frame rate of TVV achieves 45.75 fps, which satisfies the requirement of most real-time tracking applications and outperforms most state-of-the-art visual tracking approaches like KCF and DET.

The rest of this paper is organized as follows. Section 2 discusses the related work. In Section 3 we elaborate on the design of Hubble. We present the evaluation results in Sections 4. Section 5 concludes this work.

2 Related Work

2.1 Edge Computing

Real-time video processing is widely used in the areas of transportation, surveillance and security. The requirement of low latency and high computing power poses challenges to existing computing frameworks such as cloud computing, while the transaction of raw video blocks will cause bandwidth shortage. Edge computing can optimize both processing capability as well as latency for applications requiring real-time communication between the cameras and a cloud server. Many works[4][14][3] perform different video analysis algorithms on the edge server, but the edge server can not meet the computing capacity requirements of the deep learning algorithm or the exact algorithm can not meet the real-time requirements of tracking. Some works[1][16] propose an edge computing framework for video analysis. However, these architectures can not meet both the real-time and precise requirements. Therefore, the collaboration between the edge server and the central server has become a challenge that must be considered.

2.2 Visual Tracking

Compared with RF-based tracking[10] visual based tracking exhibit higher accuracy and reliability and thus attracts much attentions these year. Recent visual tracking works mainly focus on deep-learning based or correlation-filters based methods. Wang et at. proposed the Deep Learning Tracker algorithm[17] in 2013, appling the deep learning method in the field of visual tracking for the first time. MDNet[12] and siamese-fc[2] focus more on the distinction between the foreground and background of the object level. In 2014, the original author of CSK[8] improved on the basis of this algorithm and proposed the KCF [9] tracking algorithm. KCF takes the advantage of the ability of HOG describing characteristics while maintains the speed advantage of CSK, making the algorithm more robust.

In general, the correlation filtering method is much faster, but the accuracy is not as high as the deep learning method. We run KCF on the edge node to achieve real-time tracking performance and do validation with deep learning method on the server node to refine the tracking result.

2.3 Re-identification

Re-identification is a important problem of how to establish the correspondence between target identities in different video sequences. Simonnet et al.[15] transformed the problem of re-identification into the distance measurement between two sequences. They use dynamic time warping to measure the distance of video sequences. McLaughlin[11] combined CNN and recurrent neural network (RNN) to process video sequences. It uses neural network to extract information of motion from video frames. Zhou et al.[18] proposed using deep neural network to unity appearance feature learning and measurement learning. We utilize re-identification as a method to refine the loss and drift in tracking.

3 Architecture

In this section, we first introduce the system overview both in hardware and software in subsection 3.1, and show more detail of edge tracker, verifier and server validator separately in subsection 3.2, 3.3 and 3.4.

3.1 System Overview

Nowadays, edge devices are abundant in computing and communication capabilities, which makes it possible to perform light-weight video processing algorithm. When the processing algorithm meets the situation that fail to be handled on the edge nodes, raw video frames will be sent to the remote server. Under actual situation, only a small amount of raw video frames need to be processed on the server node. The cooperation between edge nodes and server avoids

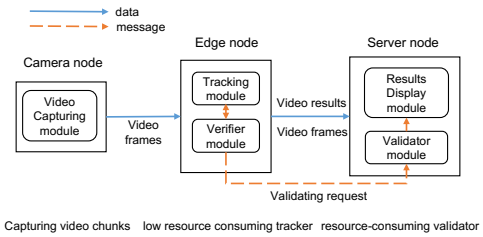


Figure 2. Architecture of System.

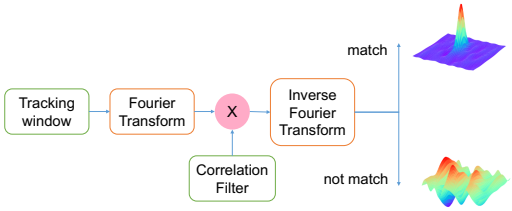


Figure 3. Correlation-filters based tracker.

the problem of insufficient and cloud computing resources caused by full video transmission.

As shown in Figure 2, our system consists of three parts at the hardware level: cameras, edge nodes and server node.

The camera nodes are responsible for capturing video images and sending video blocks to edge nodes.

The edge nodes are the devices with low computing resources and storage capacity and the edge tracker runs on them. A high-speed tracker runs on the edge node which meets the real-time requirement. However, this light-weight edge tracker cannot guarantee the accuracy of tracking result, so the edge node also runs a light-weight verifier, which will verify the accuracy of the result after every tracking operation. If the verification result shows that tracking is inaccurate, the edge node will transmit the current frame to the server node.

The server node is a video workstation with a huge amount of computing resources and powerful computing capacities. A resource-consuming validator runs on the server node. When the validator finishes processing, it will send the accurate tracking result back to the edge node.

3.2 Edge Tracker

It is necessary for the tracker to meet the requirements of light weight and fast processing speed considering the limited resource and real-time requirements on the edge node. We choose KCF (Kernelized Correlation Filters)[4] as the tracker running on the edge node which is a discriminative correlation-filters based tracker.

Originating from the idea of template matching, the main goal of correlation-filters based method is training a good correlation filter. With the use of this filter, a good response map can be obtained after calculating the correlation: a sharp peak can be obtained on the tracking target and the value rapidly decays in the surrounding area.

The speed of correlation-filters based method can reaches hundreds of frames per second because of the using of FFT

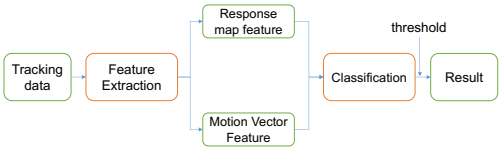


Figure 4. Verifier

for calculating the correlation, which can be expressed as:

$$G = F \odot H^* \tag{1}$$

where F is the two-dimensional Fourier transform of the image and H^* is the complex conjugate of the filter. The result G can be converted to the spatial domain, which is known as response map. The overall process is shown in Figure 3.

However, there are some situations where the tracking effect of KCF is poor. KCF cannot handle the situation where the object moves fast, considering that the object may run out of the candidate boxes, causing the match to fail. At the same time, KCF cannot handle the collision and overlap situations. When two people move facing each other, KCF has a certain probability of losing the target and tracking another.

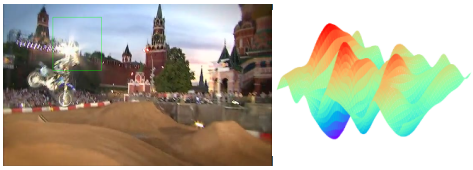
3.3 Verifier

As is mentioned before, the KCF is robust for the lighting changes, motion blur and background clutter, however, it can perform poorly because of the targets rapid movement, multi-scale variation, rigid deformation or occluding. So it is hard for KCF to handle drift caused by continuous tracking. In order to ensure the tracking performance, the verifier needs to verify the tracking quality of the edge tracker. When the accuracy of the tracker has decreased significantly, the verifier should immediately initiate a validation request to the validator. Therefore, the validation time can be defined as the time when the tracker accuracy has a large deviation.

Periodic Validation: The implementation of periodic validation is simple because the validation period is fixed. The disadvantage is that it can be difficult to select the proper interval: if the period is too large, the performance may decrease significantly due to the effect of drift effect and if the period is too small, frequent call for validator will increase the burden on the server and increase the overall computational overhead.

Dynamic Validation: It is necessary to design a dynamic validation strategy to optimize the tracking performance.As shown in the Figure 4, the verifier consists of two components: feature extraction and classification. Feature extraction is to extract features from current frame which can evaluate the tracker's performance. We take motion vector and KCF's response map into consideration. The motion vector is extracted from video coding which describes the target motion and the response map represents the confidence of the tracker. Classification utilizes these features to evaluate the tracking performance. We regress the precision of the tracker and then set a threshold to judge whether we need validate the tracker with Server Validator.

(a) Response map when KCF tracking successfully.



(b) Response map when KCF tracking failed.

Figure 5. The response map of KCF.

3.3.1 Feature Extraction

We extract features from motion vector which represents the motion of object and features from response map of KCF which represents the confidence of tracker performance.

Motion vectors are proposed in video coding to optimize video compression size. It is designed to exploit the motion information of corresponding image macroblock to reduce the bit rate of the video. For a data set, the cluster is surrounded by some data points with low local density and these low density points are far away from points with high density. Therefore we use density peak clustering to process the motion vectors, the number of clusters and other statistical features, such as the maximum, average amplitude in the cluster will be used.

According to the theory of KCF, when the algorithm tracks correctly, the response map should be a Gaussian distribution with obvious peak. When the algorithm fails, especially when the fail is caused by lost or obstructing of the target, there will not be obvious peak in the response map, as shown in Figure 5. So we evaluate the current tracking performance by calculating the difference between the corresponding peak value and the surrounding area value as confidence. It is obvious that confidence shows the difference between the maximum value and average value in response map. Greater difference leads to higher confidence, while less difference leads to smaller confidence.

3.3.2 Classification

The classification algorithm aims to compute the characteristics and decide whether the current tracking result needs to be validated according to the tracker's output accuracy. The algorithm used in our work is a combination of regression method and threshold judgment. The regression method is responsible for predicting the accuracy of the tracker algorithm. Samples below the threshold are judged to need validation. We use two kinds of regression algorithms: Linear Regression algorithm (LR) and RNN regression method.

In LR, we input the features extracted before as $X = x_1, x_2, .., x_6$, and build the linear model between IoU(Intersection of Union, predicted bounding box and the ground truth) and the features,

$$h_{iou} = \sum w_i^T x_i + b \tag{2}$$

$$y_{iou} = \frac{area(GroundtTruth) \cap area(Predicted)}{area(GroundtTruth) \cup area(Predicted)} \tag{3}$$

The model predicts IoU between predicted boundingbox and the groundtruth.

In RNN, we use LSTM (Long-Short Term Memory) to predict the IoU. The features we use are number of clusters, maximum value, average value from motion vectors and average fluction, peak value from response map. And the LSTM model output the IoU between predicted bounding box with the groundtruth.

Finally, we carefully select the threshold to control the union performance of edge tracker and server validator.

3.4 Server Validator

In this section, we will introduce the server validator including the base validator and cross-camera validator. In previous section, the frames need to be validated will be sent to server and the server should return the best matching result for the edge tracker.

We adopt a two-stage validator which is re-identification-by-detection to promote the tracking performance in this paper. The adaptor contains four sub-module: detection module, re-identification module, template update module and cross-camera tracking module. Detection module proposes candidate bounding box and filters the reduntancy with the metric of disatance. Re-identification module extracts the appearance feature of the candidate bounding box, and select the nearest one with the tamplate as the validator's result. Template update module updates the template with the selected bounding box.

Detection Module: The detection module adopts RPN(Region Proposal Network) which is widely used in object detection task. The RPN outputs a serious of candidate bounding boxes and their probability of containing object.

For every point in the feature map, RPN will produce 9 bounding box according to the 9 anchors(three resolutions together with three ratios). The RPN is trained end-to-end by back-propagation(BP) and stochastic gradient descent(SGD).

Re-Identificaiton Module: The re-identification algorithm uses triplet hard loss method. We set a threshold ε to determine whether there is the target in the image. If the minimum value of the distance between the object and the target in the set of candidate bounding boxes is greater than ε, the target is not in the frame, otherwise the target exists.

We obtain the threshold using logistic regression. After the experiment, we choose $\varepsilon = 0.7245$ and the classification accuracy is 85%.

Template Update Module: We update template feature together with the current frame and previous frame by linear interpolation.

$$template = (1-\alpha) \times template + \alpha \times new\_template \tag{4}$$

where α is the hyper-parameter for updating, $new\_template$ is the feature of the object in current frame.

Cross-camera Tracking Module: If the target object's trace crosses multi-camera, the task of single camera track-

ing will fail and the continuous tracking of the target need the coordination of multiple cameras.

From video data, we can recover the position relationship matrix between cameras. After getting transition matrix, we need to re-identify the target person from target camera got from the transition matrix. Each time the verifier fails, the server will execute the validator who will generate a feature description of the target. After a series of verifications, a series of feature descriptions from different perspectives will be obtained. Then we integrate the features and find the most similar object.

4 Experiment

4.1 Experimental Setup

Hardware: Our TVV system contains three components, edge tracker, verifier, and server validator. Edge tracker and verifier run in Raspberry Pi 2 Model B with ARM Cortex-A7 900MHz CPU, 1GB RAM. Server validator runs on device with NVIDIA GeForce GTX 1080 Ti GPU, Intel(R) Core(TM) i7-6800K CPU, 16GB RAM.

Dataset and Metrics: The dataset we use in this paper is DukeMTMC[13] which is a large-scale tracking dataset recorded on the Duke University campus with 2.8k identities. The dataset was recorded by 8 cameras and the duration of each is 1 hour and 25 minutes. We evaluate our experiment on this dataset and evaluate the tracking results in one-pass evaluation (OPE) using distance precision rate (DPR) and overlap success rate (OSR) as shown Figure 8 compared with other methods.

4.2 System Configure

Verifier: Verifier evaluates KCF's performance of the current frame and determine whether it needs the validator to validate. We propose two mechanisms for dynamic verification and compare them with periodic validation(period is 5,10,15 frames respectively). The result is shown in Table 1.

$$precision = \frac{Count(frame\_validate \cap frame\_need)}{Count(frame\_validate)} \quad (5)$$

$$recall = \frac{Count(frame\_validate \cap frame\_need)}{Count(frame\_need)} \quad (6)$$

$$Fscore = \frac{2 * precision * recall}{precision + recall} \quad (7)$$

In Figure 6, as the interval period continues to increase, the accuracy of recognition increases but the recall rate decreases significantly. The periodically corrected F-value fluctuates and stabilizes around 0.73. The periodic correction ignores the spatio-temporal information of the target motion and cannot dynamically adjust the time interval, so the average efficiency is low. From the Table 1, we can find the dynamic mechanisms both LR and RNN, perform better than the periodic mechanisms with higher precision. LR performs a bit worse than RNN, but the computational complexity of LR is much lower than that of RNN that LR can consume much fewer resources and perform faster operations on edge devices than RNN.

Validator: We obtain the threshold using logistic regression. The rule for preparing the training samples is: If the re-identification algorithm identifies correctly, we generate two sets of samples: the minimum distance is 1 and the the second-smallest distance is 0. If the re-identification algorithm fails, we generate two sets of samples: the minimum distance is 0, and the second-smallest distance is 1.

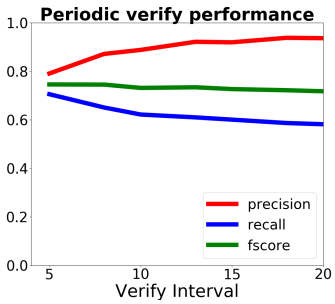


Figure 6. Periodically verify performance evaluation.

Table 1. Performance of different verifier

10-Periodic	0.6218	0.8882	0.7315
LR-Dynamic	0.8523	0.8014	0.8261
RNN-Dynamic	0.8499	0.9039	0.8761

After the experiment, we choose $\varepsilon = 0.7245$ and the classification accuracy is 85%. As is shown in Figure 7, validator refines the tracking performance, when KCF begin to miss the target.

4.3 Evaluation

Overall performance: Following the protocol in [5], we report the result in one-pass evaluation(OPE) using distance rate(DPR) and overlap success rate(OSR) as shown in Figure 8. We compare our TVV with simple KCF mode and detection mode(Det). The simple KCF mode means tracking object only with KCF, while detection mode means that every frame need to be detected and search the closest candidate bounding box in the detected result. Overall, It shows that TVV outperforms Det and KCF in both rates. In addition, we present quantitative comparison of DPR at 20 pixels, OSR at 0.5. TVV performs can our tracker achieves a DPR of 73% and an OSR of 81% the better performance with Det, while the TVV runs at real-time while Det can't. Compared with KCF, TVV refine the tracking result, when KCF begins to miss the goal, which is a restart step for KCF. TVV even has a better performance than Det, as Det just rely on re-identify by detection.

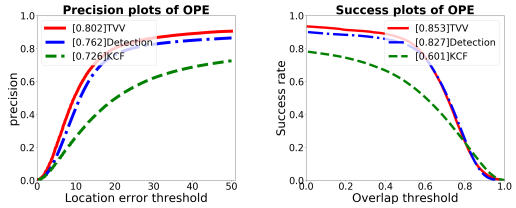
Runtime Analysis: We compare the speed of TVV with Det and KCF. In TVV, about 80% frames run on the edge with KCF, and the others run on the server. As the speed of KCF and Det are 243 fps and 6.1 fps, the speed of TVV is 45.75 fps which enables the real-time tracking.

Cross-Camera Performance: The main task of cross-camera object tracking is re-identify the target object in target cameras selected by transition matrix of Markov model. The top-1 precision of mean-based feature integration to re-identify target are 52.2% for mean value and 54.0% for median value, and top-5 are 72.6% and 73.5% respectively. The top-1 and top-5 for hierarchical-based feature integration are 32.6% and 54.5% respectively. Then, we can use

Table 2. Comparisons of the tracking methods on DukeMTMC in distance precision rate (DPR) at a threshold of 20 pixels and overlap success rate (OSR) at an overlap threshold of 0.5.

	TVV	Det	KCF
DPR(%)	0.802	0.762	0.49
OSR	0.853	0.827	0.601

Figure 7. The frames verifier. Bounding boxes in red and green mark the results of KCF and validator, respectively.



(a) Precision plots of OPE (b) Success plots of OPE

Figure 8. Comparison on DukeMTMC distance precision rate (DPR) and overlap success rate (OSR).

the re-identification to achieve cross-camera tracking. Further more, we can use more sophisticated re-identification algorithms to do cross-camera person re-identify.

5 Conclusion and Future work

This paper proposes a real-time, high-precision video tracking system solution. The main contributions are summarized as follows.

An edge computing framework is proposed. By assigning different tasks to edge and server, the bandwidth starvation caused by transmitting a large number of original video blocks can be avoided.

A visual tracking algorithm framework is presented. The framework consists of three main components, namely as tracker, verifier, and validator. The tracker is responsible for providing real-time tracking results, suitable for smooth, slow motion, and performs well most of the time. The verifier aims to evaluate the tracker's tracking results. If the accuracy is lower than the system requirements, a validation request is sent to the validator. The validator will return the results of high precision to tracker. Through the collaboration between the three components, the framework enjoys both the high speed of the tracker and the accuracy of the validator.

At the end, we have deployed TVV in a real-world digital twin system Pavator[7][6] for real-time people tracking in an ultra-high voltage converter station.

Although a real-time high-precision tracking algorithm framework is proposed in this paper, we need to handle multi-target tracking problem in real world whose environment is much more complicated than the experimental dataset. And the server is also a resource-restricted device. Therefore, there is still huge space for improvement.

6 Acknowledgments

This work was supported by the State Grid of China Science and Technology Fund No.52110417000G.

7 References

[1] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *computer*, 50(10):58–67, 2017.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[3] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann. Dynamic urban surveillance video stream processing using fog computing. In *2016 IEEE second international conference on multimedia big data (BigMM)*, pages 105–112. IEEE, 2016.

[4] J. Dick, C. Phillips, S. H. Mortazavi, and E. de Lara. High speed object tracking using edge computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 26. ACM, 2017.

[5] H. Fan and H. Ling. Parallel tracking and verifying. 2018.

[6] J. Guo, Y. He, and X. Zheng. Pangu: Towards a software-defined architecture for multi-function wireless sensor networks. In *Parallel and Distributed Systems (ICPADS), 2017 IEEE 23rd International Conference on*, pages 730–737. IEEE, 2017.

[7] Y. He, J. Guo, and X. Zheng. From surveillance to digital twin: Challenges and recent advances of signal processing for industrial iot.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[10] C. Jiang, Y. He, X. Zheng, and Y. Liu. Orientation-aware rfid tracking with centimeter-level accuracy. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 290–301. IEEE Press, 2018.

[11] N. McLaughlin, J. Martinez del Rincon, and P. Miller. Recurrent convolutional network for video-based person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[13] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35, 2016.

[14] M. Schneider, J. Rambach, and D. Stricker. Augmented reality based on edge computing using the example of remote live support. In *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pages 1277–1282. IEEE, 2017.

[15] D. Simonnet, M. Lewandowski, S. A. Velastin, J. Orwell, and E. Turkbeyler. Re-identification of pedestrians in crowds using dynamic time warping. In *European Conference on Computer Vision*, pages 423–432. Springer, 2012.

[16] H. Sun, X. Liang, and W. Shi. Vu: video usefulness and its application in large-scale video surveillance systems: an early experience. In *Proceedings of the Workshop on Smart Internet of Things*, page 6. ACM, 2017.

[17] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015.

[18] Z. Zhou, Y. Huang, W. Wang, L. Wang, and T. Tan. See the forest for the trees: Joint spatial and temporal recurrent neural networks for video-based person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6776–6785, 2017.