Deep Learning-based Multiple Pedestrians Detection-Tracking Framework

Xuan-Phung Huynh
Dept. of Computer Engineering
Sejong University
phunghx@sju.ac.kr

Yong-Guk Kim*
Dept. of Computer Engineering
Sejong University
*ykim@sejong.ac.kr

ABSTRACT
We propose a new Detection-Tracking (DT) framework whereby one can detect a pedestrian, or multiple ones, within a video image and then track them concurrently in a flexible manner. For the detection, a faster R-CNN will be used since it has a state-of-the-art detection accuracy as well as speed. For the tracking, we have developed a fast and reliable tracker, which mainly consists of Kernelized Correlation Filter (KCF) and Kalman filter and shows enhancing performance in the occlusion and human-crossing situations. After the faster R-CNN detects objects' regions and scores for that objects, our tracker estimates object's position based on kernel method and Kalman filter. We demonstrate that the proposed framework can detect and track multiple moving pedestrians concurrently for the walking crowd scene.

Author Keywords
Kernelized Correlation filter; Kalman filter; Convolutional Neural Network; R-CNN; Surveillance; Pedestrian

ACM Classification Keywords
Human-centered computing

INTRODUCTION

To track a moving object in a given video, it requires two important visual processes: object detection and tracking. For the first case, the most popular method has been Haar-feature based AdaBoost algorithm [19], which has been very effective in detecting the face among other objects. On the other hand, HOG-feature [4] has been often used in detecting pedestrians in the street scene. Since the recent success of deep learning network such as ImageNet [15] in recognizing many different (>1,000) objects with an impressively high rate, peoples somehow ask to have a system that detects many different objects within the images accumulated by portal sites. Among several initiatives, the Fast Region-based Convolutional Network (Fast R-CNN) [7] seems like a promising object detection system in terms of both accuracy and speed.

For the second case, human tracking has played an important role in the automatic surveillance system [21]. Currently, surveillance [1, 2, 6] is contributing vital roles not only in the academic research area but also in the market. Visual tracking is the process of estimating the trajectory of an object in the image plane as it moves around in the scene. According to [21], tracking objects can be complex due to its accuracy and speed as well as how it deals with occlusion and human-crossing cases. Numerous approaches for object tracking have been proposed. Among them, Kernelized Correlation Filter [11] has been a state-of-the-art tracker partly because of its high speed and relatively simple implementation. Recently, it has been reported that it is the third performing tracker among many in term of accuracy [14]. Yet, its performance in dealing occlusion requires some improvement for the robust tracker. We present a new tracking method whereby the reliability is greatly enhanced while the speed is also maintained, by combining a Kalman filter with the KCF. In particular, it is able to deal very well with occlusion and human-crossing tasks, which are critical requirements for the high-end surveillance case.

In general, an integrated system that can detect and track the moving objects is essential. Some previous works propose a framework to accomplish such task [1, 2, 9, 22]. In this paper, we present a new framework whereby the detection is greatly enhanced by Fast R-CNN while the tracking is elevated by our tracker that combines the Kernelized Correlation filter and Kalman filter.

The rest of the paper is organized as follows. In Section 2 and 3, we review the Selective Search for object localization and Fast R-CNN, respectively. In section 4, we describe our tracker in detail. In section 5, we present our framework that can detect and track the multiple pedestrians concurrently in a given video. In Section 6, we show experimental results. Finally, Section 7 summarizes this paper.

SELECTIVE SEARCH FOR OBJECT LOCALISATION

A selective search algorithm captures all scale, has a diverse set of strategies, and computes reasonably fast. A hierarchical grouping algorithm is the basic form of this method. First, the initial regions are created by [5]. Then a greedy algorithm groups iteratively the regions together. In [18], the authors calculate the similarities between all neighboring regions. The two most similar regions are grouped together, and the similarities of the resulting region

and its neighbors are computed. This process is repeated until the whole image becomes a single region. The general method is described in [18], including its source code.

Selective search utilizes a variety of color spaces with different invariance: *RGB*, grey-scale image, *Lab*, *rg* channels of normalized *RGB* plus intensity denoted as *rgI*, and the Hue channel *H* from *HSV*. To use in merging, [18] has defined four complementary, fast-to-compute similarity measures:

$s_{color}(r_i,rj)$ measures color similarity as follow:

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k) \quad (1)$$

Where a color histogram $C_i = \{c_i^1, \ldots, c_i^n\}$ for each region r_i with dimensionality $n = 25$ for each color channel.

$s_{texture}(r_i,rj)$ measures texture similarity by:

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k) \quad (2)$$

Where SIFT is the texture and $T_i = \{t_i^1, \ldots, t_i^n\}$ is a texture histogram for each region r_i with dimensionality n=10 for each color channel [18].

$s_{size}(r_i,rj)$ encourages small regions to merge early as follow:

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i)+size(r_j)}{size(im)} \quad (3)$$

Where *size(im)* is the size of the image in the pixels.

$s_{fill}(r_i,rj)$ measures how well region r_i and r_j fit into each other.

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) + size(r_j)}{size(im)} \quad (4)$$

Where BB_{ij} is the tight bounding box around r_i and r_j.

The final similarity combines four measurements:

$$s(r_i, r_j) = s_{color}(r_i, r_j) + s_{texture}(r_i, r_j) + s_{size}(r_i, r_j) + s_{fill}(r_i, r_j) \quad (5)$$

In addition, the method of [5] is the fastest in varying the complementary starting regions. The threshold parameter *k* in [5] is varied to get the corrected object regions.

FAST REGION-BASED CONVOLUTIONAL NETWORK

Fast region-based convolutional network (Fast R-CNN) fixes the drawbacks of R-CNN and Spatial pyramid pooling networks (SPPnet [10]) whereby their speed and accuracy are improving [7]. Fast R-CNN is implemented in Python as well as C++ (Caffe [12]) and is available under the open-source MIT License[1].

[1]https://github.com/rbgirshick/fast-rcnn

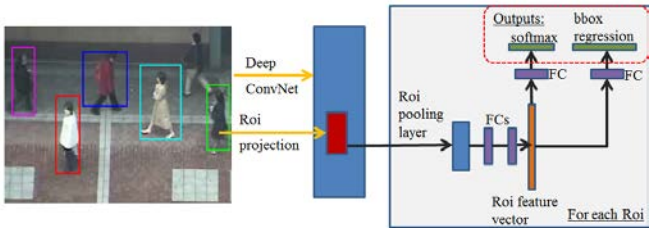


Fig. 1. Fast R-CNN architecture. A convolutional neural network processes input image and multiple regions of interest (RoIs). Feature map pools each RoI and maps to fully connected layers (FCs). The network has two outputs: classification and bounding-box regression.

Fig.1 shows the Fast R-CNN framework. An entire image and a set of object proposal are the inputs to the network. To create a feature map, several convolutional and max pooling layers process the whole image. Then, based on each object proposal, a region of interest (RoI) pooling layer extracts a fixed length feature. A sequence of fully connected (fc) layers process each feature vector to branch into two output layers: one that classifies the object based on softmax probability and another layer that outputs four real valued numbers for each object.

The RoI pooling layer

The RoI pooling layers utilizes max pooling to transfer the features inside any valid region of interest into a small feature map so that they are the forms of translation invariance. RoI is a rectangle window into a conv feature map.

Initializing from pre-trained networks

Fast R-CNN utilizes three pre-trained ImageNet [15] networks. First, a RoI pooling layer replaces the last max-pooling layer. Secondly, the two sibling layers, which conduct two outputs, replace the previous network's last fully connected layer and softmax. Thirdly, the network is changed to process two data inputs: a list of image and list of RoIs correspondingly.

Multi-task loss

A Fast R-CNN network has two output layers. A multi-task loss L on each labeled RoI is defined as follow

$$L(p, u, t^u, v) = L_{cls}(p, u) + [u \geq 1] L_{loc}(t^u, v) \quad (6)$$

Where $p = (p_0, \ldots, p_K)$, over K+1 categories is the probability distribution (per RoI). $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$ is the bounding-box regression offset for each of the K objects classes [8]. Each RoI is labeled with a ground-truth class u and a ground-truth bounding-box regression target v. In (6), $L_{cls}(p, u) = -\log p_u$ is log loss for true class u. L_{loc} is the loss of true bounding-box regression targets and predicted tuple for class u. The Iversion bracket indicator function $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and o otherwise. For instance, the loss of bounding-box regression is

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i) \quad (7)$$

in which

$$smooth_{L_1}(x) = \begin{cases} 0.5\,x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (8)$$

Fast R-CNN detection

The network processes as input an image and a list of *R* object proposals to score. For each test RoI *r*, a class posterior probability distribution and a set of predicted bounding-box are conducted. The confidence level to r for each class uses the estimated probability $\Pr(class = k|r) = p_k$. Based on R-CNN, Fast R-CNN performs non-maximum suppression independently for each class.

KERNELIZED CORRELATION FILTER WITH KALMAN FILTER: KK TRACKER

Kernelized Correlation Filter

KCF trains a linear classifier using both a base sample, i.e. a positive example, and several virtual samples, which serve as the negative examples, obtained by translating it. Here, a cyclic shift operator is utilized in modeling this translation. Because of the cyclic property [11], the signal *x* becomes identical after nth shifts, and the full set of shifted signals is

$$\{P^u x \mid u = 0, \dots, n-1\} \quad (9)$$

with *P* is the permutation matrix [11] and *n* is the number of translation step.

Fast Kernel Regression

The kernel matrix *K(n x n)* stores the dot-products between all pairs of samples

$$K_{ij} = \kappa(x_i, x_j) = \varphi^T(x_i)\varphi(x_j) \quad (10)$$

with high-dimensional space $\varphi(.)$.

The following kernels satisfy the condition to claim that *K* is circulant [11]:

- *Radial Basic Function kernels -e.g., Gaussian.*
- *Dot-product kernels -e.g., linear, polynomial.*
- *Additive kernels -e.g., intersection, χ^2 and Hellinger kernels.*
- *Exponentiated additive kernels.*

Therefore, the kernelized version of Ridge Regression is possible to diagonalize

$$\hat{\alpha}^* = \frac{\hat{y}^*}{\hat{k}^{xx} + \lambda} \quad (11)$$

where $K = C(k^{xx})$ is kernel matrix.

Learning phase utilizes Eq. (11) to update the model for next frame.

Fast Detection

Several candidate patches, *z*, that can be modeled by cyclic shifts are evaluated by the regression function *f(z)*. To compute efficiently, detection phase diagonalizes regression function

$$\hat{f}(z) = \hat{k}^{xz} \odot \hat{\alpha} \quad (12)$$

where k^{xz} is a kernel correlation of *x* and *z*.

During detecting phase, Eq. (12) predicts where is the center of target within the given frame with a learned coefficient α.

Kalman Filter

By assuming the system noise has Gaussian distribution, Kalman filter utilizes the linear dynamical systems to resolve the linear optimal filtering problem. More specifically, x_n is a discrete time system with state at time *n*. In the next time step *n + 1*, the state is

$$x_{n+1} = F_{n+1,n}x_n + w_{n+1} \quad (13)$$

where $F_{n+1,n}$ is the transition matrix from state x_n to x_{n+1}, and w_{n+1} is white Gaussian noise with zero mean and covariance matrix Q_{n+1}.

While the measurement vector z_{n+1} is given by

$$z_{n+1} = H_{n+1}x_{n+1} + v_{n+1} \quad (14)$$

where H_{n+1} is the measurement matrix and v_{n+1} is white Gaussian noise with zero mean and covariance matrix R_{n+1}, that is independent of noise w_{n+1}. The system gets measurement value for each step then estimates the correct state based on the minimum mean-square error of (14). The solution is a recursive procedure [3, 13, 16].

Our tracker: KK Tracker

We have recently developed a new tracker that improves the KCF tracker by correcting target's position with Kalman filter. It is known that KCF often makes failed prediction for the case of occlusion and human crossing simply because the object is disappear. Moreover, performance of the KCF tracker is often deteriorated for rotation, illumination variation, motion blur and etc. We thought that Kalman filter can improve these limitations. In fact, after the KCF estimates target's position based on the prediction by Kalman filter and the estimated value is given to the updating step of Kalman filter. During the KCF learning phase, our tracker uses the correct state to update the kernel model.

First, the fast detection would acquire the peak value $f_{max}(k)$ at *(x(k), y(k))* according to formula (12) in the current frame. Then, Kalman filter uses this position to adjust the system state to correct the position. After several frames of progress, when Kalman filter is stable, it can correct the position of the target in the current frame. Then, during the KCF's learning phase, it is able to extract the target's feature successfully for the forthcoming frames.

Secondly, in order to resolve the occlusion as well as human crossing cases, we propose a novel approach. When the peak value is less than threshold *T*, it is assumed that this is the occlusion case. In other words, KCF would predict the incorrect position in that frame, and we could not use it as a measurement value of Kalman filter. During such case, our tracker will use the target's position based on the

prediction step by Kalman filter, and only this step will run. Fig. 2 and algorithm 1 describe it in detail.

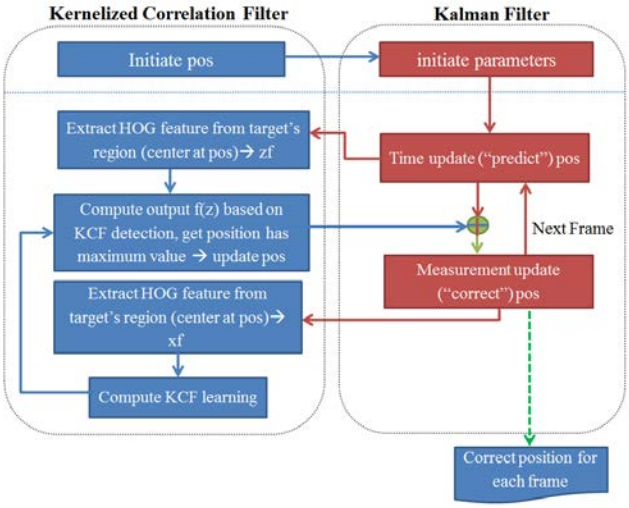


Fig. 2. Flowchart of our tracker, which basically combines KCF and Kalman filter for our purpose

Algorithm 1. KK Tracker

1. Initialization

   pos ← target's center (first frame)

   $\hat{x}_0 = pos$

   $P_0 = E[(x_0 - E[x_0])(x_0 - E[x_0])^T]$

2. Kalman prediction

   $pos = \hat{x}_n^-$

3. KCF detection

   zf ← HOG target's feature (center at pos)

   $f_{peak}, pos' \leftarrow \max(\hat{k}^{xz} \odot \hat{\alpha}$

   If $f_{peak} < T$ then

      Goto step 5

   Else

      $z_n = pos'$

      Goto step 4

   End if

4. Kalman estimation

   $pos = \hat{x}_n$

5. KCF learning

   $xf \leftarrow HOG\ target'sfeature\ (center\ at\ pos)$

   $\hat{\alpha}^* = \frac{\hat{y}^*}{\hat{k}^{xx} + \lambda}$

Goto step 2 for new frame.

THE PROPOSED FRAMEWORK

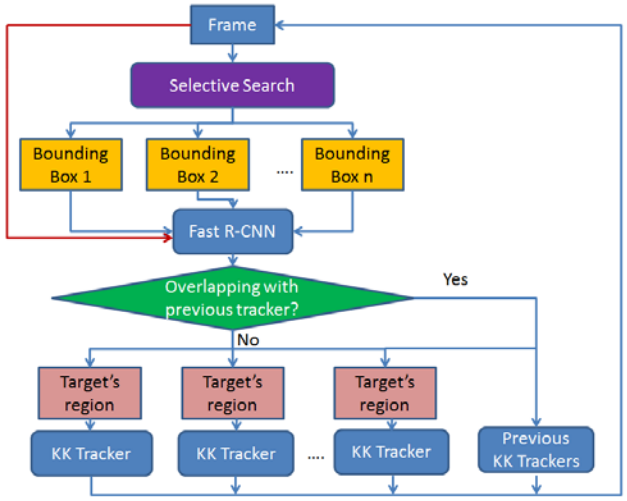


Fig. 3. Flowchart of our framework, which basically combines detection and tracking in a complete manner

We propose a completed framework by combining Fast R-CNN for detecting multiple objects and KK tracker for tracking them concurrently. First, the Selective Search part localizes objects' location in the frame. With diversity of strategies, we get lots of bounding boxes after this step. Secondly, a Fast R-CNN takes a frame as input and list of bounding boxes, i.e., object proposals. After the Fast R-CNN process, we have the targets' region, which will be the inputs for the tracking phase. Since each region needs one tracker, we implement a tracking system with multiple KK trackers. Each KK tracker will predict the target's region in the next frame. With new frame, which is not the first frame, the Fast R-CNN continues detecting objects' region that can be the object, which already has been tracking in the previous frame. To resolve such case, we propose a novel approach. When the new detected region overlaps less than the threshold, it is assumed that a KK Tracker has tracked this and the system will not create new tracker. The process will continue as shown in Fig. 3.

EXPERIMENTS

Implementation Details

The present framework is implemented using Matlab library. The pipeline for the system is illustrated in Fig.3. For Fast R-CNN, we utilize the very deep VGG16 model from [17]. In addition, some heuristics are used for Kalman filter in KK tracker. For instance, the sampling rate, acceleration magnitude, process noise, and measurement noise are given as 1 frame/s, 0.2, 0.5, and 1.0, respectively. The system is implemented using Matlab on Ubuntu OS running on a computer having an i7 CPU with 64 GB RAM and GPU NVIDIA Geforce TITAN x GDDR5 12GB.

Evaluation

The performance of a Fast R-CNN in term of its speed is demonstrated in [7]. Since the authors assume frame and bounding boxes are the inputs of network, their experiment result is fast enough to implement. Here, since we propose and add a Selective Search method in extracting bounding boxes, the speed of our system is slightly slow than that of the original one.

For the tracking part, when testing on standard databases[2] [6, 20], our KK Tracker outperforms the standard KCF, MOSSE and MIL trackers, respectively. Fig.4. depicts tracker's performance for the full datasets and Fig.5. shows the performance for occlusion as well as human-crossing cases. In addition, our tracker is fast as standard KCF so that it can run in the real time basis.

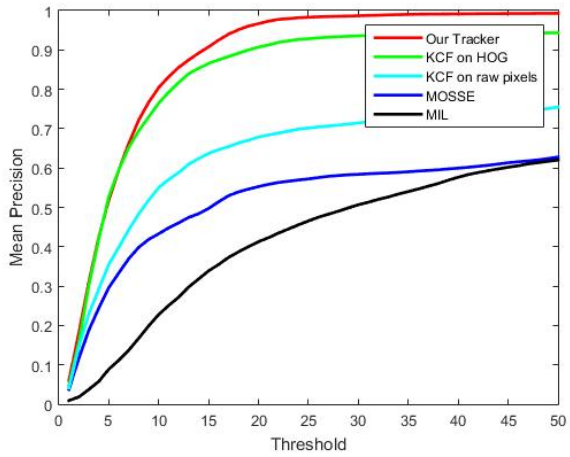
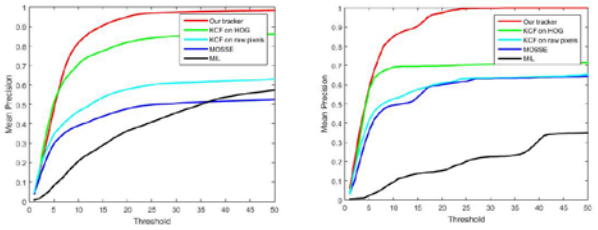


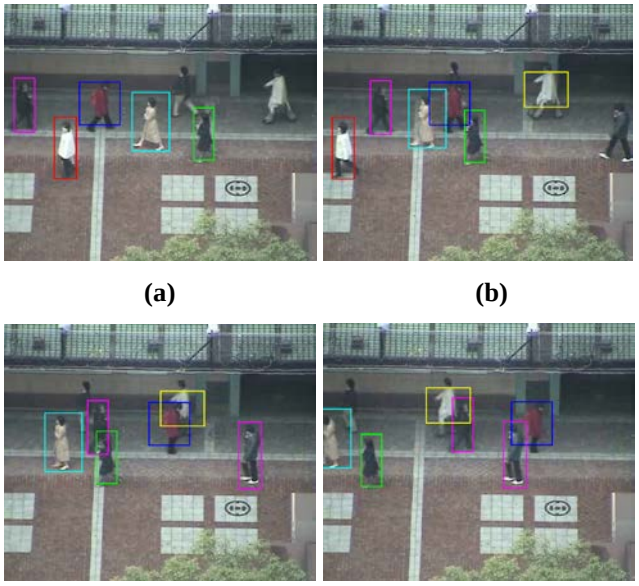
Fig. 4. Comparing the trackers' performance for full datasets



(a) **(b)**

Fig. 5. Comparing the trackers' performance for occlusion (a) and human-crossing (b) cases

Finally, we demonstrate our framework on the video of walking crowd in the subway. Fig.6. illustrates the results from some frames of it. The performance of tracker is excellent with the human-crossing cases occurred on the crowd. Initially, detection is not yet good for the first frame in which two objects are missed. However, as the process of detection-tracking continues, an object, wearing white cloth, is detected and tracked in the next frames as shown in Fig.6.b and Fig.6.c, respectively, even though it misses one dark object, partly because of the low quality of the image.



(a) **(b)**

(c) **(d)**

Fig.6. Our framework's performance on a video. (a): detection results on the first frame. (b): tracking task and detection for the new object. (c) and (d): it continues tracking and detecting objects.

CONCLUSION AND FUTURE WORK

In this paper, we present a new framework that combines the Fast R-CNN, which is based on deep learning technique, and our KK tracker. Using the Selective Search, we acquire the regions of interest that is one of the inputs to Fast R-CNN to detect object's region. Our tracker, which combines Kernelized Correlation filter and Kalman filter, estimates the target's location in the next frame. It is demonstrated that our framework can detect and track multiple moving pedestrians concurrently using a walking crowd scene. As far as we know, this kind of integration of detection and tracking is new, especially for the multiple objects tracking. We plan to improve the performance of our framework, in particular for the detection part.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A2006969) and by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2015-R0134-15-1032) supervised by the IITP Institute for

[2] Dataset is available at https://goo.gl/2bPKi7

Information and Communication Technology Promotion).

REFERENCES

1. Cohen, I., and Medioni, G. Detecting and tracking moving objects for video surveillance. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*. (1999), vol. 2, IEEE.
2. Cucchiara, R., Grana, C., Neri, G., Piccardi, M., and Prati, A. The sakbot system for moving object detection and tracking. In *Video-Based Surveillance Systems*. Springer, 2002, pp. 145–157.
3. Cuevas, E. V., Zaldivar, D., Rojas, R., et al. Kalman filter for vision tracking.
4. Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 1, IEEE, pp. 886–893.
5. Felzenszwalb, P. F., and Huttenlocher, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2 (2004), 167–181.
6. Ferryman, J., and Ellis, A. Pets2010: Dataset and challenge. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on* (2010), IEEE, pp. 143–150.
7. Girshick, R. Fast r-cnn. *arXiv preprint arXiv:1504.08083* (2015).
8. Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (2014), IEEE, pp. 580–587.
9. Haritaoglu, I., Harwood, D., and Davis, L. S. W 4 s: A real-time system for detecting and tracking people in 2 1/2d. *In Computer Vision?ECCV'98*. Springer, 1998, pp. 877–892.
10. He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*. Springer, 2014, pp. 346–361.
11. Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. High-speed tracking with kernelized correlation filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37, 3 (2015), 583–596.
12. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia* (2014), ACM, pp. 675–678.
13. Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering 82*, 1 (1960), 35–45.
14. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J.,Cˇ ehovin, L.,Nebehay, G., Vojˊırˇ, T., Fernandez, G., Lukeˇzicˇ, A., Dimitriev, A., et al. The visual object tracking vot2014 challenge results. In *Computer Vision-ECCV 2014 Workshops (2014)*, Springer, pp. 191–217.
15. Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.
16. Mohinder, S. G., and Angus, P. A. Kalman filtering: theory and practice using matlab. *John Wileys and Sons* (2001).
17. Simonyan, K., and Zisserman, A. Very deep convolutional networks for large- scale image recognition. arXiv preprint *arXiv:1409.1556* (2014).
18. Uijlings, J. R., van de Sande, K. E., Gevers, T., and Smeulders, A. W. Selective search for object recognition. *International journal of computer vision 104*, 2 (2013), 154–171.
19. Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), vol. 1, IEEE, pp. I–511.
20. Wu, Y., Lim, J., and Yang, M.-H. Online object tracking: A benchmark. In *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on* (2013), IEEE, pp. 2411–2418.
21. Yilmaz, A., Javed, O., and Shah, M. Object tracking: A survey. *Acm computing surveys* (CSUR) 38, 4 (2006), 13.
22. Zhang, R., and Ding, J. Object tracking and detecting based on adaptive background subtraction. *Procedia Engineering 29* (2012), 1351–1355.