Human Tracking Using Convolutional Neural Networks

Jialue Fan, *Student Member, IEEE*, Wei Xu, Ying Wu, *Senior Member, IEEE*, and Yihong Gong

***Abstract*— In this paper, we treat tracking as a learning problem of estimating the location and the scale of an object given its previous location, scale, as well as current and previous image frames. Given a set of examples, we train convolutional neural networks (CNNs) to perform the above estimation task. Different from other learning methods, the CNNs learn both spatial and temporal features jointly from image pairs of two adjacent frames. We introduce multiple path ways in CNN to better fuse local and global information. A creative shift-variant CNN architecture is designed so as to alleviate the drift problem when the distracting objects are similar to the target in cluttered environment. Furthermore, we employ CNNs to estimate the scale through the accurate localization of some key points. These techniques are object-independent so that the proposed method can be applied to track other types of object. The capability of the tracker of handling complex situations is demonstrated in many testing sequences.**

***Index Terms*— Convolutional neural networks, machine learning, visual tracking.**

I. Introduction

OBJECT tracking is a fundamental problem in computer vision. Traditional feature-based methods, such as those based on color [1] or motion blobs [2]–[4], perform tracking by maintaining a simple model of the target and adapting such a model over time. However, real situations in practice pose enormous challenges to these techniques, because: 1) over time, the object model can deviate from its original one, and 2) they do not have a discriminative model that distinguishes the object category of interest from others.

In recent years, learning-based approaches [5]–[7] and object-detection-based tracking methods [8], [9] have been used to overcome these limitations. In these methods, an online learning or a pretrained detector provide object candidates, and then an additional module generates trajectories by associating the candidate detections across frames. By adaptively updating the visual appearance or limiting the focus to detection candidates, these methods tend to adapt the tracker to the changes.

The major challenge of the traditional learning-based and/or tracking-by-detection methods is the false positive matches that lead to wrong association of the tracks. The reason is that those methods are based on applying an appearance model or object detector at all possible windows around the target, and the object detection in the current frame does not depend on knowing the position of the target in the previous frame. Therefore, in crowded scenarios, when the distracting objects are similar to the target, the object detector will generate similar high detection scores for both the target and the distracters, which would probably cause a drift problem. For example, if we want to track the head of a person in a crowd, it is very difficult if we use a head detector because the heads of other people in the crowd can also be good matches. In Fig. 1, suppose the target is at O in the previous frame. It is clear that the position A and A' will generate similar high detection scores by the object detector, as the detector does not know that the target is at O in the previous frame.

To alleviate this drift problem, we consider that the location and the appearance of an object in the previous frame should assist us to detect the object in the current frame. By using these additional pieces of information for detection, we effectively turn an object detector into a tracker that is capable of estimating the location and the scale of the target given its previous location, size, as well as current and previous image frames. In this paper, we use convolutional neural networks (CNNs) [10] as our base learner because they have been demonstrated to be able to extract local visual features (structures) and they are widely used in various visual recognition applications [11], [12]. Different from fully connected neural networks, CNNs force the extraction of local features by restricting the receptive fields of hidden units to be local, based on the fact that images have strong 2-D local structures.

The hurdle of applying conventional CNNs in tracking is that they have shift-invariant architectures (see Section III-C or [13]) which make them suitable for recognition or detection tasks but inappropriate for tracking tasks. In this paper, we design a CNN tracker with a shift-variant architecture. Such an architecture plays a key role so that it turns the CNN model from a detector into a tracker. The features (structures) are learned during offline training. We demonstrate the generality of the learned features by testing the CNN tracker on different scenarios and environments. Different from traditional learning methods which only extract local spatial structures [5], we extract both spatial and temporal structures (motion

Manuscript received February 5, 2010; revised April 15, 2010, June 24, 2010, and August 1, 2010; accepted August 2, 2010. Date of publication August 30, 2010; date of current version October 6, 2010. This work was supported in part by the National Science Foundation under Grant IIS-0347877 and Grant IIS-0916607, and in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant ARO W911NF-08-1-0504.

J. Fan and Y. Wu are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: jialue.fan@u.northwestern.edu; yingwu@eecs.northwestern.edu).

W. Xu was with NEC Laboratories America, Inc., Cupertino, CA 95014 USA. He is now with Facebook, Inc., Palo Alto, CA 94304 USA (e-mail: emailweixu@facebook.com).

Y. Gong is with NEC Laboratories America, Inc., Cupertino, CA 95014 USA (e-mail: ygongca@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2066286

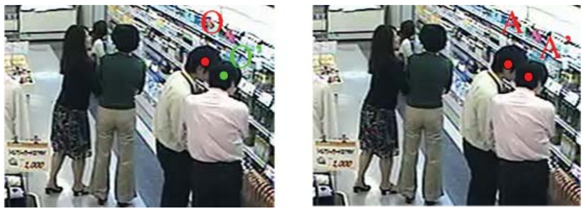


Fig. 1. Drift problem probably occurs when the distracting objects are similar to the target.

information) by considering the *image pair* of two consecutive frames rather than a single frame. The temporal structures provide a crude velocity signal to tracking, since the large signals in the temporal information (e.g., the frame difference) tend to occur near objects that are moving [14]. Furthermore, the CNN tracker not only extracts local features (e.g., from 7×7 patches), but also extracts global features from a wide scope of pixels, as the local features are not always reliable for matching because of view change or partial occlusion.

The contributions of this paper include the following: 1) a discriminative model extracts discriminant spatial and temporal features for specific object class tracking, where the features are learned from a parametric feature pool with rich degrees of freedom; 2) the shift-variant architecture of CNN is based on a multipath strategy which widely broadens the traditional use of CNN; and 3) the novel scale estimation method is based on the localization of key points which is object independent. Unlike the existing learning-based methods, the proposed model can largely alleviate the drift problem, and thus can perform robust tracking for a longer time.

The rest of this paper is organized as follows. Section II describes related work. Sections III and IV discuss the details of the CNN tracking algorithm. Section V shows promising comparative results and Section VI summarizes the conclusion.

II. Related Work

In this section, we review recent approaches related to this paper. Tracking-by-detection has been studied recently in [15]–[19]. As a matter of fact, tracking-by-detection methods can be viewed as a special kind of learning-based methods, since the object detector is learned offline. In [20], an assembly of body parts is used for detecting and tracking partially occluded people. In [8], refined limb detectors are taught to detect people's limbs in a wide variety of situations. A novel feedback connection from the object detector to pose estimation (visual odometry) is used for human tracking [21]. [22] proposes a two-stage approach which first builds an appearance model of individual people, and then tracks them by detecting those models in each frame. These methods are based on the analysis of the body parts or pose estimation. However, in many video surveillance applications, the subject may not have enough resolution, and therefore the detailed motion of the body parts is difficult to discover.

Learning-based methods have been in the focus of recent works [5]–[7], [23]–[25]. The support vector tracker [23] uses an offline-learned support vector machine as the classifier and embeds it into an optical flow-based tracker. In [6], the current frame is classified using a classifier learned in the previous frame. A variance ratio is used to measure feature discriminability and to select the best feature from a feature pool for tracking. In [5], an ensemble of weak classifiers is trained online, and pixels are labeled as either the target or the background. Since only spatial local structures are extracted, the confidence map degrades in crowded scenes, as there are many similar spatial patterns. In [26], the target is represented in a low-dimensional subspace which is updated adaptively using the images tracked in the previous frames. In [25], a cascade particle filter with feature pools of different life spans has been proposed, while the likelihood model is noisy and has many peaks without a Gaussian diffusion at the sampling stage. In [27], a semi-supervised approach is proposed where labeled examples come from the first frame only, and subsequent training examples are left unlabeled. In [28], a discriminative classifier is trained in an online manner based on multiple instance learning. All these approaches extract only the spatial structures but neglect the motion information. In this paper, we extract both spatial and temporal structures from two consecutive frames to model the likelihood and capture the motion information, which efficiently alleviates the drift problem as mentioned above.

A. CNNs

A popular CNN architecture that shows excellent performance for visual recognition is shown in Fig. 2, which is an instance of multistage Hubel–Wiesel architectures [10], [29]. Because the theory of CNNs is closely related to some deep knowledge about neural networks, which is beyond the scope of this paper, we briefly review the basic idea of CNNs here and refer the readers to [10] for more details. For visual recognition tasks, CNNs first extract and combine local features from the input image, and these features are then combined by the subsequent layers in order to obtain higher order features. Such high-order features are eventually encoded into a 1-D vector (target label), which is then categorized by a trainable classifier. It is worth noting that feature extraction is a nontrivial problem, because there are size, slant, and position variations for individual images. These cause variations in the position of distinctive features in the input objects. To eliminate these variations, CNNs combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights, and downsampling.

In Fig. 2, the input plane receives the image patches to be processed. Each unit in a layer receives inputs from a set of units located in a small neighborhood in the previous layer. With local receptive fields, neurons can extract primitive visual features such as oriented edges, endpoints, and corners.

Once a feature has been detected, its exact location becomes less important. Only its approximate position relative to other features is relevant. In CNN architecture, downsampling layers are introduced to reduce the resolution of the feature map as well as the sensitivity of the output to shifts and distortions. As shown in Fig. 2, this model includes alternating layers of convolutional feature detectors (C layers) and downsampling layers using a max or an average operation (S layers).

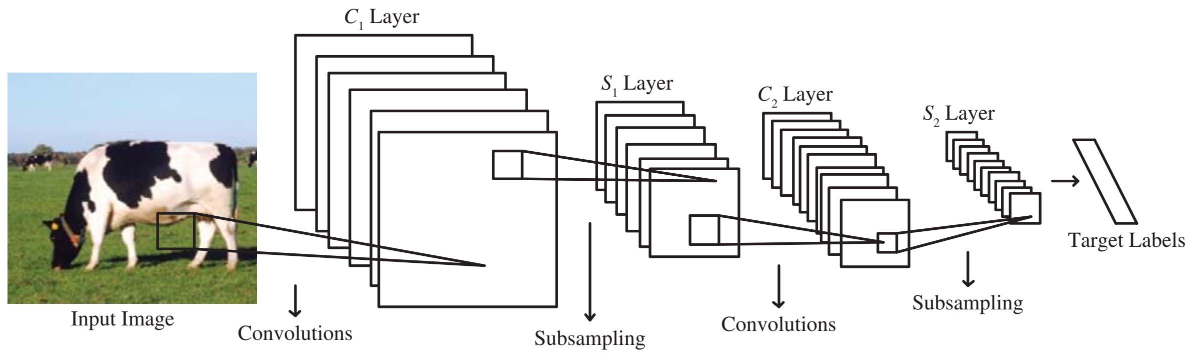


Fig. 2. Architecture of CNN for visual recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.

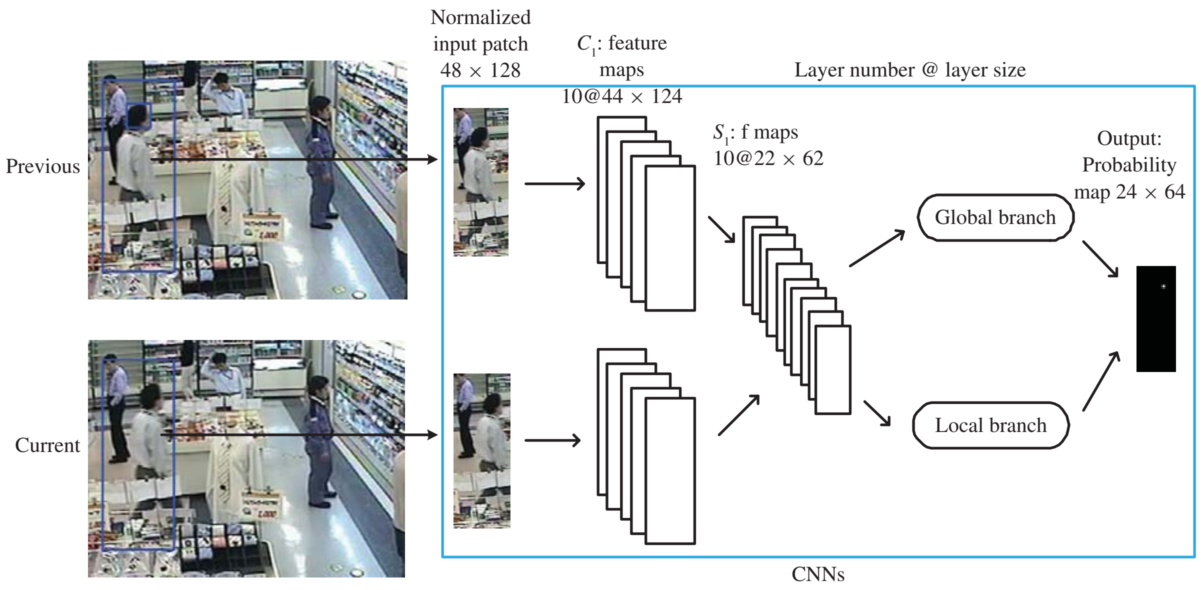

Normalized input patch 48 × 128
C_1: feature maps 10@44 × 124
Layer number @ layer size
S_1: f maps 10@22 × 62
Output: Probability map 24 × 64
Previous
Current
Global branch
Local branch
CNNs

Fig. 3. Architecture of CNN tracking.

For recognition or detection tasks, the primitive feature detectors that are useful on one part of the image are likely to be useful across the entire image. Units in a layer are organized in planes within which all the units share the same set of weights. By sharing the same set of weights, CNNs have the shift-invariant property so that they are able to achieve excellent performance in various recognition or detection tasks.

In [14], Nowlan and Platt describe a CNN hand tracker to track the position of the hand across a sequence of video frames. Although the temporal features are extracted from the difference of the consecutive frames, they use a shift-invariant architecture so that the CNNs operate as an object detector. So the CNN hand tracker does not include mechanisms to handle the drift problem in Fig. 1. In this paper, we effectively turn a CNN object detector into a tracker by constructing a shift-variant architecture of the CNN.

III. CNN Tracking

As mentioned above, CNNs can extract local structures and feature vectors from the input image. This motivates us to apply the neural networks for tracking problems. The basic idea is that, given two corresponding image patches (from the previous and the current frame), where in the first patch the target is located at the center, we want to find the target location in the second image patch on the basis of spatial and temporal structures of these two patches. The spatial structures indicate the appearances, while the temporal structures capture the motion information. We do not rely on background subtraction because it is not reliable and may not be available for dynamic background. The architecture of CNN tracking is shown in Fig. 3.

For the tracking problem, assume that the target position (e.g., the center of the human head) at time $t-1$ is known (the initialization can be done manually or by human detection), the goal is to find the target position at time t. Denote the target position at time $t-1$ by $\mathbf{x}_{t-1} = (x_{t-1}, y_{t-1}, s_{t-1})$, where (x_{t-1}, y_{t-1}) is the position of the human head, s_{t-1} is the scale. The associated bounding box (e.g., the blue square in Fig. 3) is denoted by $R(\mathbf{x}_{t-1})$. We extract an image patch (the blue rectangle in Fig. 3) which includes the surrounding of the target at time $t-1$. Denote this image patch by $N_{t-1}(\mathbf{x}_{t-1})$. The ratio[1] of the size of $N_{t-1}(\mathbf{x}_{t-1})$ to that of $R(\mathbf{x}_{t-1})$ is

[1]e.g., the area ratio of the blue rectangle to the blue square in Fig. 3.

fixed (we expect that the whole human body can be included in $N_{t-1}(\mathbf{x}_{t-1})$ for most cases, so we empirically set the ratio to 3×8 in our framework). We extract the image patch $N_t(\mathbf{x}_{t-1})$ at time t which has the same position as $N_{t-1}(\mathbf{x}_{t-1})$. $N_{t-1}(\mathbf{x}_{t-1})$ and $N_t(\mathbf{x}_{t-1})$ are normalized to image patches of fixed size $w \times h$ which are the inputs of CNN. After normalization, the target object is always at the same position O in the $w \times h$ input patch at time $t-1$ (Fig. 6). Our CNN is expected to detect an object at time t which corresponds to the target located at position O. The output of CNN is the probability map of size $(w/2) \times (h/2)$ which shows the target position at time t. The peak of the map indicates where the target is. The reason why the size of the probability map is one-half of the input image is that the downsampling in the S1 layer decreases the resolution of the input image. Therefore, the accuracy of the target position is about 2 pixels. However, as we will explain in Section V-D, our method can tolerate small measurement deviations such that the error will not be accumulated.

Regarding the architecture of the network, the inputs of CNN are the image pair at time $t-1$ and t, which means that we can extract both spatial and temporal structures from the target neighborhood. The weights of CNN are learned during the offline training procedure. In contrast to the confidence map in [5], the probability map we use here has two important advantages: 1) only around the target center the probability of the unit is high, so the probability map reflects an accurate localization property, and 2) we can measure how well the probability map describes the true situation, i.e., the noisier the probability map, the more complex is the environment and, therefore, the less confident we are in using the map to estimate the target position.

A. Detail Description

The detail description of our CNN is as follows. As CNNs contain a large number of parameters in the hierarchical architecture, it would be quite lengthy if we use general constants to denote the parameters. Hence, we denote parameters by specific numbers for clarity.

1) Each normalized patch of the image pair is split into five input feature maps, i.e., R/G/B channels and additional two channels D_x and D_y, which are the horizontal and vertical gradients of gray intensities. Throughout the experiment in this paper, the size of the input image patch is 48×128.
2) Layer C_1 is a convolutional layer with 10 feature maps. Each unit in each feature map is connected to a 5×5 neighborhood of the input. The size of the feature maps is 44×124. We denote $C_1(k, i, j)$ the value at position (i, j) in the kth feature map of layer C_1.
3) Layer S_1 is a downsampling layer with 10 feature maps using max operation. Such an operation introduces some local translation invariance to the model. We denote $S_1(k, i, j)$ the value at position (i, j) in the kth feature map of layer S_1. Then we have

$$S_1(k,i,j) = \max\{C_1(k,2i,2j), C_1(k,2i+1,2j), C_1(k,2i,2j+1), C_1(k,2i+1,2j+1)\}. \quad (1)$$

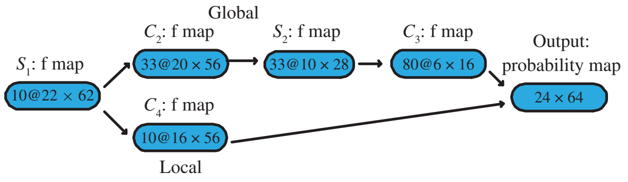


Fig. 4. Global branch and local branch.

4) We split the following procedure into two branches. The global branch aims to enlarge the *receptive field* [10] so that each unit in the probability map is affected by a large area, i.e., global structures are obtained, the local branch aims to discover more details about local structures.

The global and local branches are shown in Fig. 4. For the global branch, layer C_2 is a convolutional layer with 33 feature maps. Each unit in each feature map is connected to several 3×7 neighborhood at identical locations in a subset of S_1 maps. Table I shows the set of S_1 feature maps combined by each C_2 feature map. We refer the readers to [10] for the reason for not using a complete connection.

For the λth column in Table I, we denote the marked row indices by $\eta_{\lambda,0}, \eta_{\lambda,1}, \ldots, \eta_{\lambda,p-1}$. For instance, if $\lambda = 23$, we have $p = 6$, $\eta_{23,0} = 1$, $\eta_{23,1} = 3$, $\eta_{23,2} = 4$, $\eta_{23,3} = 6$, $\eta_{23,4} = 8$, $\eta_{23,5} = 9$. Then the size of the convolution kernel of the λth feature map of C_2 layer is $p \times 3 \times 7$. We denote this kernel by K_λ, which is trainable. Denote $C_2(k, i_0, j_0)$ the value at position (i_0, j_0) in the kth feature map of layer C_2. Then we have

$$\begin{aligned} C_2(\lambda, i_0, j_0) &= \sum_{r=0}^{p-1}\sum_{i=0}^{2}\sum_{j=0}^{6}\{S_1(\eta_{\lambda,r}, i+i_0, j+j_0) \\ &\quad \times K_\lambda(p-1-r, 2-i, 6-j)\}. \end{aligned} \quad (2)$$

This is a 3-D convolution operation. For example, for the 0th column, $p = 3$, $\eta_{0,0} = 0$, $\eta_{0,1} = 1$, $\eta_{0,2} = 2$. We have

$$\begin{aligned} C_2(0, i_0, j_0) &= \sum_{r=0}^{2}\sum_{i=0}^{2}\sum_{j=0}^{6}\{S_1(\eta_{0,r}, i+i_0, j+j_0) \\ &\quad \times K_0(2-r, 2-i, 6-j)\}. \end{aligned} \quad (3)$$

Finally, the size of the receptive field of the global branch is 28×68, so the features can capture *global structures* from a wide range in the image patch. Since the downsampling impairs the resolution in the global branch, we use the local branch to compensate the accuracy. The local branch uses only the convolution operation without downsampling in order to focus on the details of *local structures*.

From Fig. 3, the 0th–4th feature maps of layer S_1 are from the previous patch and the 5th–9th feature maps of layer S_1 are from the current patch. From Table I, for 0–10 out of C_2 feature maps, the connection to C_2 are from the previous patch only, because the entries of 5th–9th rows and 0–10th columns are zero. For 11–21 out of C_2 feature maps, the connection to C_2 are from the current patch only. Therefore, for 0–21 out of C_2 feature maps, the connection to C_2 are from only one patch, so it is supposed to capture spatial features. For the

TABLE I
COLUMNS INDICATING WHICH FEATURE MAP IN S_1 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C_2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	X			X	X	X		X	X	X	X												X		X	X		X			X	X	X
1	X	X			X	X	X		X	X	X													X		X	X	X	X			X	X
2	X	X	X			X	X	X		X	X												X		X		X	X	X	X			X
3		X	X	X		X	X	X	X		X												X	X		X			X	X	X		X
4			X	X	X		X	X	X	X	X													X	X		X			X	X	X	X
5												X			X	X	X		X	X	X	X	X		X	X		X	X	X			X
6												X	X			X	X	X		X	X	X		X		X	X		X	X	X		X
7												X	X	X			X	X	X		X	X	X		X		X			X	X	X	X
8													X	X	X		X	X	X	X		X	X	X		X		X			X	X	X
9														X	X	X		X	X	X	X	X		X	X		X	X	X			X	X

remaining C_2 feature maps, the connection to C_2 are from two patches, thus it is supposed to capture motion related features. Similarly, layer S_2 is a downsampling layer and layer C_3 is a convolutional layer with 80 feature maps. We follow a similar notation as in (1), and have

$$S_2(k, i, j) = \max\{C_2(k, 2i, 2j), C_2(k, 2i+1, 2j), \\ C_2(k, 2i, 2j+1), C_2(k, 2i+1, 2j+1)\}. \quad (4)$$

The connection between S_2 and C_3 is randomly chosen, i.e., we randomly choose 10 feature maps from S_2 to connect to C_3 in order to reduce the number of the trainable parameters. Based on this hierarchical structure, as well as the nonlinear operations, the features are extracted from a parametric feature pool with rich degrees of freedom.

From layer C_3 to the output probability map, there is a four-times upsampling (from 6×16 to 24×64). This is different from the conventional CNN in which two-times upsampling is adopted. We will show in Section III-C that the use of four-times upsampling together with the local branch makes the CNN architecture shift-variant so that such an architecture can be used for tracking.

For layer C_4, it is a convolutional layer with 10 feature maps. Each unit in each feature map is connected to a 7×7 neighborhood of layer S_1. There is a translation transform from layer C_4 to the output. For the position (i_0, j_0) in the output map, this translation transform ensures that the center of its local receptive field in the normalized input patch is at $(2i_0, 2j_0)$, thus the local spatial structures are correctly extracted.

The convolution filter of the probability map is a linear function followed by a sigmoid transformation (see [10] for details). We follow the above notations, and denote $Op(i_0, j_0)$ the value at position (i_0, j_0) in the output probability map. Then we have

$$Op(i_0, j_0) = \zeta\left(\sum_{r=0}^{79} C_3\left(r, \lfloor\frac{i_0}{4}\rfloor, \lfloor\frac{j_0}{4}\rfloor\right) K_3(r) \right. \\ \left. + \sum_{r=0}^{9} C_4(r, i_0 - 4, j_0 - 4) K_4(r) + K_b\right) \quad (5)$$

where $K_3(\cdot)$, $K_4(\cdot)$, and K_b are trainable weights, ζ is the sigmoid function, and $\lfloor\ \rfloor$ is the floor function.[2]

[2]For negative arguments, for example, $C_4(0, -1, 0) = 0$.

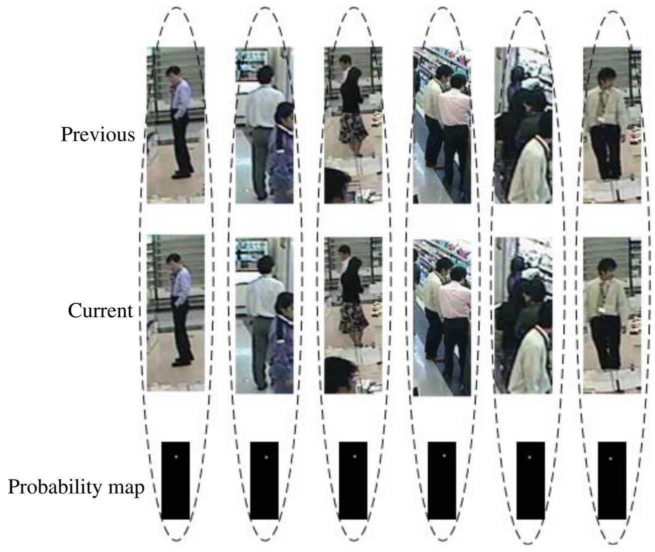


Fig. 5. Six training samples from the offline training set.

B. Training Procedure

The offline training set includes around 20 000 samples. The dataset was collected by NEC Laboratories. For the training samples, we manually annotate the bounding boxes of human heads in some video sequences (e.g., the blue square in Fig. 3). Given one bounding box centered at $\mathbf{x}_{t-1}$, we extract $N_{t-1}(\mathbf{x}_{t-1})$ and $N_t(\mathbf{x}_{t-1})$ as the CNN input. The probability map is generated by a Gaussian function and the peak of the probability map is at $\mathbf{x}_t$. Hence, one training sample consists of the image patch $N_{t-1}(\mathbf{x}_{t-1})$ in the previous frame, the patch $N_t(\mathbf{x}_{t-1})$ in the current frame, and the target detection probability map in the current frame. The examples of training samples are shown in Fig. 5. We collect those samples randomly with different human views in some surveillance videos.

The model is trained offline using standard stochastic gradient descent by minimizing the difference between the probability map outputted by the CNN and the target probability map. Once the model is trained, it is fixed during tracking. We do not use adaptive models since they are susceptible to the drift problem. Unlike adaptive models, our model has less chance to drift to unrelated types of object. Although fixed, the model is capable for dramatic view changes, since we have collected enough training samples including various cases of human

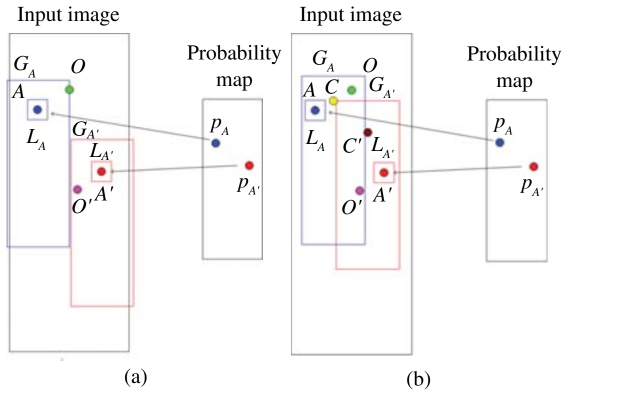


Fig. 6. Receptive field of the shift-invariant and the shift-variant architectures. (a) Shift-invariant. (b) Shift-variant.

motion. Extensive experiments on real test video sequences show the effectiveness of this model.

C. From Shift-Invariant to Shift-Variant

The difference between shift-invariant and shift-variant architecture is shown in Fig. 6. In this section, we first point out that the conventional CNNs with a shift-invariant architecture are not proper for tracking, and the use of two-times upsampling from layer C_3 to the output makes the CNN model shift-invariant. Then we illustrate how the use of four-times upsampling breaks such shift-invariant property, and explain why it can alleviate the drift problem significantly during tracking.

Denote the target position at time $t-1$ and time t by O and A, respectively. Consider there is another person moving from O' to A' at the same time. The global and local receptive field of A is denoted by G_A and L_A, respectively. Denote the *center* of the receptive field G_A by C_{G_A}. The probability p_A in the probability map is the sigmoid function of the summation of G_A and L_A responses, which is monotonically increasing.

The conventional architecture of CNNs for detection is the so-called space displacement neural network (SDNN) [13], [14]. In this architecture, a single detector is replicated over the input, and the output is a detection score map which is shift-invariant. This means that the position A and A' will generate similar high detection scores. The high detection score at A' is very harmful because it may cause confusion with A (the readers can refer to Fig. 1 for the intuition).

If we use two-times upsampling from layer C_3 to the output (similar to the architecture in [14]), our network will be a special instance of SDNN, which operates as an object detector. As shown in Fig. 6(a), C_{G_A} is at A. It means that the relative locations of the global receptive field (C_{G_A}) and the object position (A) are the same. Hence the obtained detection map is shift-invariant.

The use of four-times upsampling has a nice property. The center of the global receptive field of one output unit is at the midpoint of the previous location and the current location of the object (Property 3.1, the proof is shown in Appendix). Having this property, the four-times upsampling breaks the shift-invariant property of SDNN. In the shift-variant CNN architecture, C_{G_A} depends on O (it is at the midpoint of O

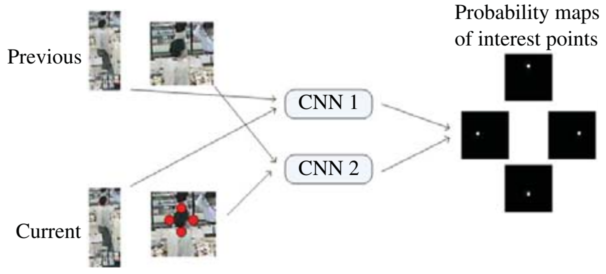


Fig. 7. Architecture of CNNs to detect key points.

and A, see Fig. 6(b)), so the CNN output depends on the object's previous location. This is different from the shift-invariant architecture where C_{G_A} is independent of O (it is at A). Hence, the proposed model is different from conventional CNN models which operate as object detectors.

In this shift-variant architecture, we claim that the tracker has less chance to drift to A' by showing that p_A is larger than $p_{A'}$. As p_A is the sigmoid function of the summation of G_A and L_A responses, it is large only if both G_A and L_A respond large. For A and A', the response of the local branch L_A and $L_{A'}$ are similar, as they operate as local object detectors. For the response of the global branch G_A and $G_{A'}$, G_A responds large, as it extracts the similar pattern at O (at time $t-1$) and A (at time t) (this is learned during training). But $G_{A'}$ responds small, because the global patterns at O (at time $t-1$) and A' (at time t) are different.[3] Therefore, p_A is larger than $p_{A'}$, and the tracker does not drift to A'. From this point of view, the drift problem is alleviated. It is clear that the tracker does not drift to other locations, because the responses of the global and local receptive fields are both small.

Property 3.1: Assume the target is at O at time $t-1$. At time t, if we want to check the hypothesis at A, then the global receptive field is located at C, which is the midpoint of O and A

$$\mathbf{x}_C = \frac{\mathbf{x}_O + \mathbf{x}_A}{2}. \tag{6}$$

IV. Handling the Scale Change

The scale is usually handled by testing different scales to minimize the matching costs. However, these types of methods are unstable when view changes or partial occlusion occur. In [30] a solution to scale integrated in mean-shift framework is presented. In our method, we calculate the scale by detecting the key points of the target (see Fig. 7).

In our tracking method, the target position $\mathbf{x}_t$ is bounded by a rectangle. Therefore, we choose the centers of four sides of the rectangle as the key points (red circles in Fig. 7 for illustration). This choice of key points is general, so the method can be extended to any object class tracking unless the shape of objects is concave. The input images of the CNN are the same as our CNN tracking method, and we have a finer

[3]Recall that the size of global receptive field is 28×68, so global structures are much more distinctive and informative than local ones. Hence, for O and A', although the local features might be similar, the global structures are usually different.

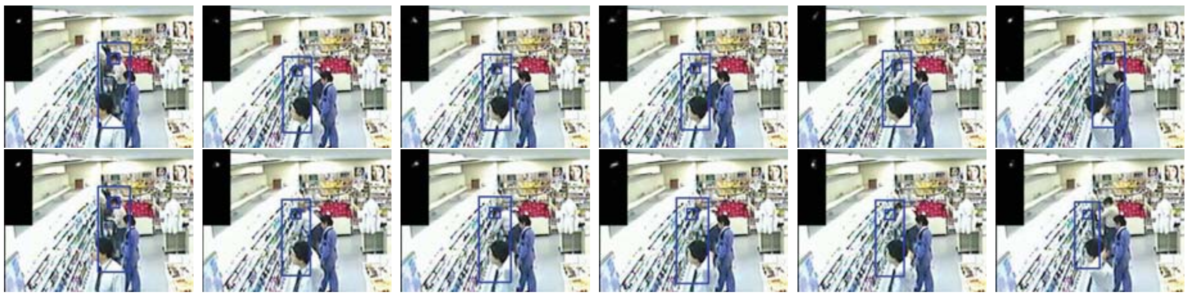


Fig. 8. Tracking a human with pose changes: (top) CNN tracker using both temporal and spatial features; (bottom) CNN tracker using only spatial features.

scale version of the input images as the other two inputs. The reason for using finer versions is that a good scale estimation requires the accurate localization of key points.

The two image pairs are the inputs of the two CNNs, respectively. For the output, there are four probability maps corresponding to the four key points. The architecture of each CNN is similar to the architecture in Fig. 3, but the main difference is that the output is four probability maps. Since the four probability maps share the same architecture in the first several steps, they are jointly trained, not independently. This makes sense since the positions of the four key points are correlated.

After offline training, such a CNN architecture for detecting key points can be applied for online tracking. Once the key points are determined, the scale of $\mathbf{x}_t$, i.e., s_t, is determined since the key points are located at the target boundary.

Denote the locations of the four key points at time $t-1$ by $\mathbf{x}_{t-1}^i (i = 1, \ldots, 4)$. Denote the locations of the four key points at time t by $\mathbf{x}_t^i (i = 1, \ldots, 4)$, with the probability score $p^i (i = 1, \ldots, 4)$, respectively. For $1 \le i < j \le 4$, $\|\mathbf{x}_t^i - \mathbf{x}_t^j\| / \|\mathbf{x}_{t-1}^i - \mathbf{x}_{t-1}^j\|$ gives an estimation of s_t. Each estimation is weighted by $p^i p^j$, meaning that the key point with a high probability score has a larger weight. Considering all pairs of i, j, we have the scale estimation in (7)

$$s_t = \frac{1}{\sum\limits_{1 \le i < j \le 4} p^i p^j} \sum_{1 \le i < j \le 4} p^i p^j \frac{\|\mathbf{x}_t^i - \mathbf{x}_t^j\|}{\|\mathbf{x}_{t-1}^i - \mathbf{x}_{t-1}^j\|}. \tag{7}$$

As the training samples include all different views, the scale estimation is robust to view changes. This estimation, together with the original CNN estimator, provides a robust tracking result.

V. Experimental Results

A. Setup and Comparison Baseline

We test the proposed CNN tracker for a variety of challenging video sequences. The experiments are divided into two parts designed to evaluate the performances of several components of the algorithm and the robustness under different circumstances. No testing sequences shown here have been trained. The test sequences shown in Figs. 8, 9, 13, and 18 are from NEC Laboratories dataset but different from the training videos. The test sequences shown in Figs. 14–17 are completely different from the training ones. Therefore, we can show that the features selected by CNN are widely applicable through the experiments. All the public video sequences and the comparisons are available on the author's website.

Our algorithm is implemented in C++ and tested on an Intel 3.6-GHz desktop. Without code optimization, the program runs at 10–15 frames/s on average (640×480).

The CNN tracker is compared with the ensemble tracker [5], which is a learning-based method. According to [5], we implement the ensemble tracker with an 11-D feature vector per pixel that consists of an 8-bin local histogram of oriented gradients calculated on a 5×5 window as well as the pixel R, G, and B values.

We also compare the CNN tracker with support vector tracker [23], where an offline-learned support vector machine is applied as the classifier for tracking. The classifier is trained on a set of 20 000 images of human heads and non-heads. The training images are manually selected and reduced to the size of 24×24 pixels. A homogeneous quadratic polynomial kernel is used to perform the learning phase.

As [5] and [23] do not contain the component of estimating scale, for the video sequences with large scale changes we implemented two reference methods for comparison. One is the mean-shift tracker [31] in the enhanced YCbCr space with 1040 bins. The other method is an adaptive model [32] with a clustering procedure. We refer to this method as the clustering method.

In many situations, only some parts of the whole human body are included in the video sequences (e.g., Fig. 17). In such scenarios, we still use 48×128 image patches as our CNN input, and set the values to zero for those pixels out of the image boundary. So we do not need to train extra models, which indicates that our CNN model is very flexible to handle various complex situations.

B. Impact of Using Temporal Features

In this experiment, we show the impact of using temporal features. In conventional learning methods, only spatial features are used. In our method, we also extract temporal features. To illustrate the benefits of temporal features, we implemented another CNN with only one image (the current image) as input. Such CNNs have a similar architecture as the proposed method, the only difference is the number of feature maps in each layer.

Fig. 8 shows the results of tracking a human with dramatic pose changes. In this situation, the subject stoops down to

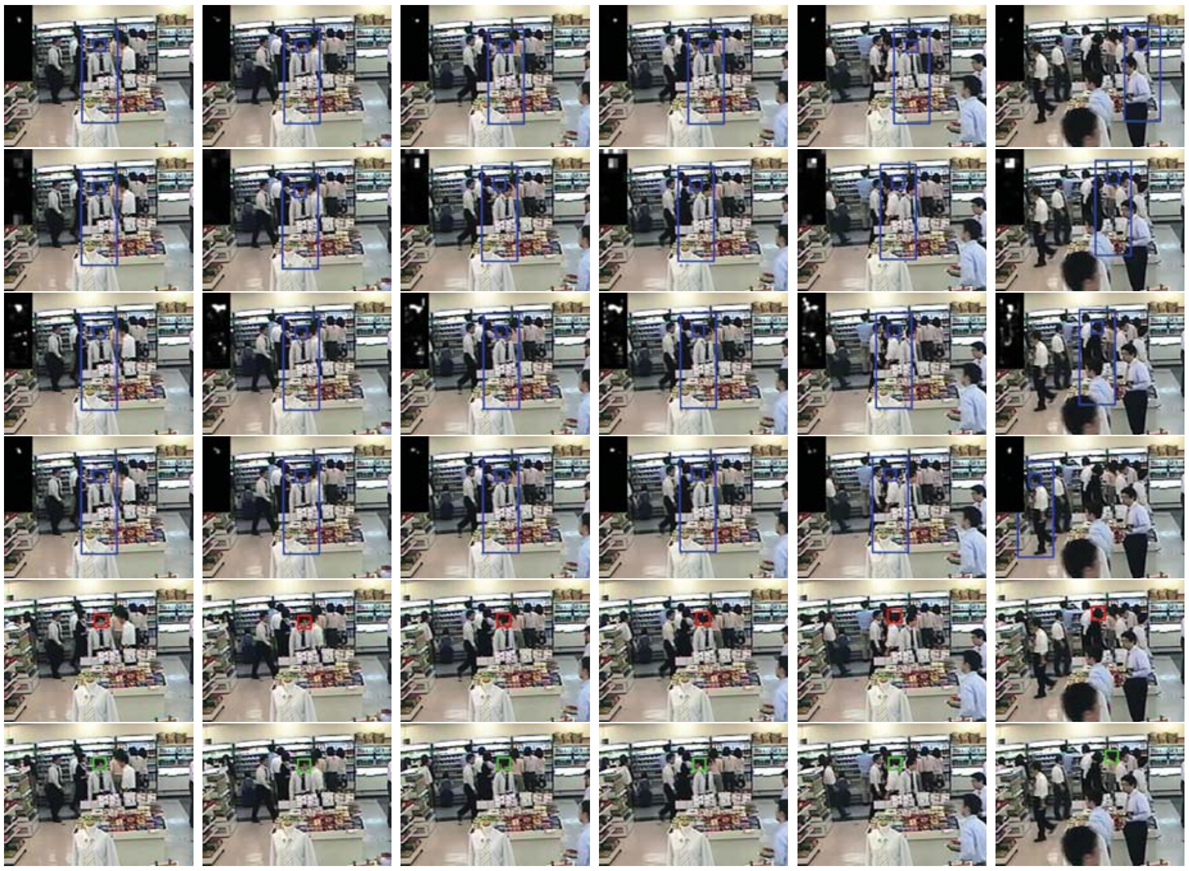


Fig. 9. Comparison between different architectures [shopping]: (first row) the proposed CNN tracker; (second row) the CNN tracker using only global branch; (third row) CNN tracker using only local branch; (fourth row) CNN tracker with shift-invariant architecture; (fifth row) ensemble tracker; (sixth row) support vector tracker.

see the items on the shelf, and then she stands up. When the subject moves upward, the CNN tracker using both temporal and spatial features correctly follows her head, while the CNN tracker with only spatial features does not. This is because the background information dominates $N_{t-1}(\mathbf{x}_{t-1})$ when she stoops down, i.e., spatial features extract much information from the background, not from the target. Therefore, when she moves upward, the CNN tracker with only spatial features tends to follow the background, and thus loses track. Generally, any tracker with adaptive appearance models faces the same problem in such a situation. On the contrary, the CNN tracker using both temporal and spatial features can capture the motion information when the subject stands up so that the drift does not occur.

From the experiment, we show that using both temporal and spatial features is better than using spatial features only. One possible explanation is based on the fact that temporal features can encode the difference between two adjacent images. In some cases, when the background dominates the input image patch (like in Fig. 8), the temporal features contain more useful information than the spatial features.

Note that this comparison is to investigate the feature selection component of CNN (the useful features are learned during the training procedure). In this experiment, neither of the CNN architectures is a detector, since we apply the shift-variant structure for both CNNs. The comparison of the CNN tracker and the CNN detector is illustrated in the next subsection.

C. Impact of Shift-Variant Architecture

The comparison between different CNN architectures is shown in Fig. 9. The CNN tracker with only global branch or local branch is shown in the second and third row, respectively. We implemented another CNN tracker with shift-invariant property (the result is shown in the fourth row). Note that the architecture of the CNN with shift-invariant property is very similar to the method in [14]. We find that only the proposed CNN can avoid the drift problem in this situation. In fact, the CNN with only local branch can be treated as a shift-invariant architecture, as the receptive fields share the same weights. When several people are very close to the target in $N_{t-1}(\mathbf{x}_t)$, the shift-invariant architecture cannot differentiate between those people, because such an architecture only plays a role like human detectors. The probability map of the proposed CNN tracker is the combination of the probability maps of the CNN trackers in the second and third row (Note this is not a linear combination, because the sigmoid function is used). From Fig. 9, the peak in the probability maps of the global branch and that of the local branch are different, but their combination gives a good performance after all.

We also show the performance of the ensemble tracker and the support vector tracker in this scenario. As the ensemble

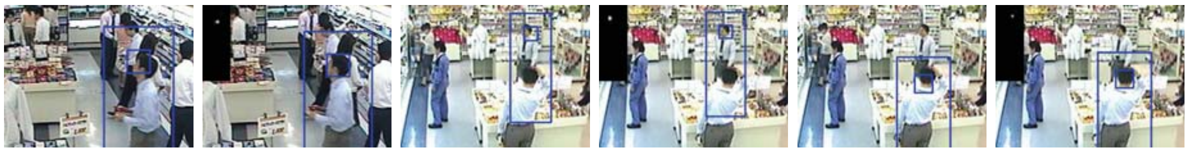


Fig. 10. Recovery from tracking errors in the previous frame (three pairs).

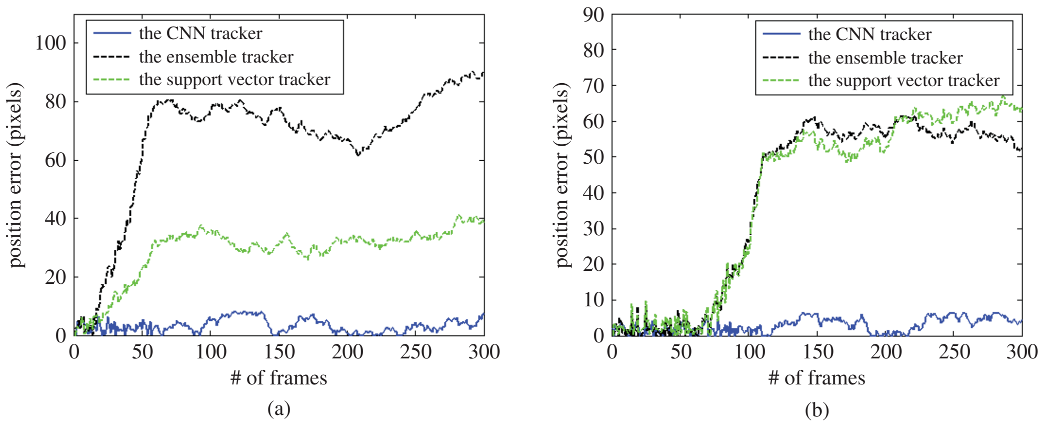


Fig. 11. Quantitative comparison of position errors between the CNN tracker, the ensemble tracker, and the support vector tracker. (a) Shopping. (b) Occlusion.

TABLE II
POSITION ERROR OF DIFFERENT METHODS (UNIT: PIXEL)

Sequence	[5]	[23]	Proposed method
shopping	66.74	28.81	3.27
occlusion	39.07	39.67	2.88

TABLE III
POSITION ERROR OF DIFFERENT METHODS (UNIT: PIXEL)

Sequence	[31]	[32]	Proposed method
girl	24.79	–	8.85
indoor	29.93	11.28	7.85

tracker trains a classifier to distinguish the target from the background, the tracker drifts to the people nearby because the visual appearance of the two heads are nearly the same, as shown in the fifth row of Fig. 9.

D. Drift Correction

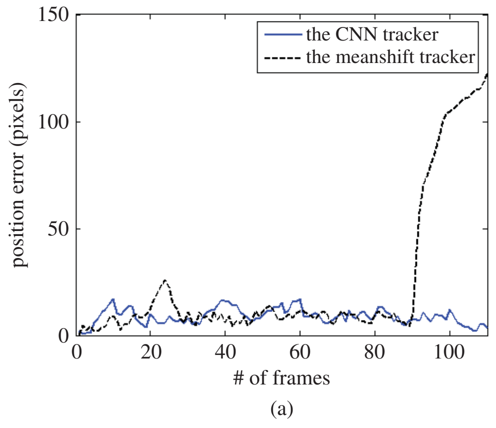
CNNs have the ability of object-class detection by extracting spatial features. For the purpose of drift correction, we generate some small random shifts in the training procedure such that the target (i.e., the human head) is not exactly at the center position in the previous frame. Therefore, during online tracking, our CNN tracker can recover tracking errors even if tracking is inaccurate at the previous frame. In Fig. 10, we introduce some shift in the previous frame so that the target is not well tracked. As we can see, the CNN tracker can recover such errors at the current frame by the mechanism of human detection.

E. Quantitative Experiments

For a quantitative evaluation, we manually labeled the ground truth of the sequence shopping, occlusion, girl, and indoor, respectively. The evaluation criteria of tracking error are based on the relative position errors between the center of the tracking result and that of the ground truth. We put the video sequences online to show that our CNN tracker can perform tracking for a long duration.

As shown in Fig. 11, the position errors of the results in the CNN tracker are much smaller than those of the ensemble tracker and the support vector tracker. It demonstrates the advantages of the CNN tracker. Note that the ensemble tracker and the support vector tracker drift when two people move very close to each other in these examples. The reason is that the reference trackers behave like object detectors, which means drift may often occur in crowd scenarios. On the contrary, the CNN tracker does not lose track, as we effectively turn the object detector into a tracker by the shift-variant CNN architecture. In fact, the authors in [23] have mentioned the limitation that the support vector tracker cannot handle partial occlusions and may switch from one subject to another in case of two nearby subjects. The corresponding statistical results are summarized in Table II.

Since the sequences girl and indoor contain large scale changes, we compare the CNN tracker with the mean-shift tracker and the clustering method for these sequences. As shown in Fig. 12, the position errors of the results in the CNN tracker are much smaller than that of the reference methods, which indicate that the proposed method is more accurate and stable. As the reference methods only contain online learning mechanisms, they may not be appropriate to handle dramatic or rapid appearance changes. In contrast, the proposed method has less chance to drift to unrelated objects because of the pretrained discriminative model. For the girl sequence, we did not show the curves of the clustering method, because the tracker drifts to the background very early at around the 15th frame. The corresponding statistical results are summarized in Table III.



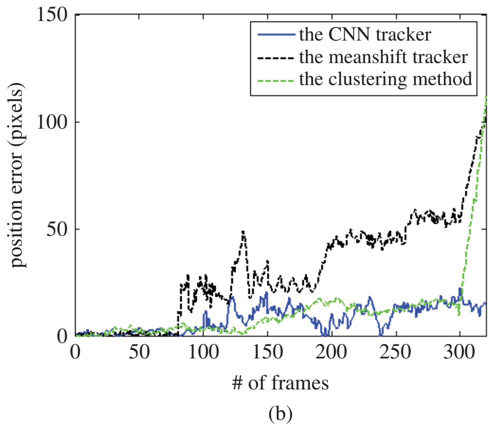


Fig. 12. Quantitative comparison of position errors among the CNN tracker, the mean-shift tracker, and the clustering method. (a) Girl. (b) Indoor.

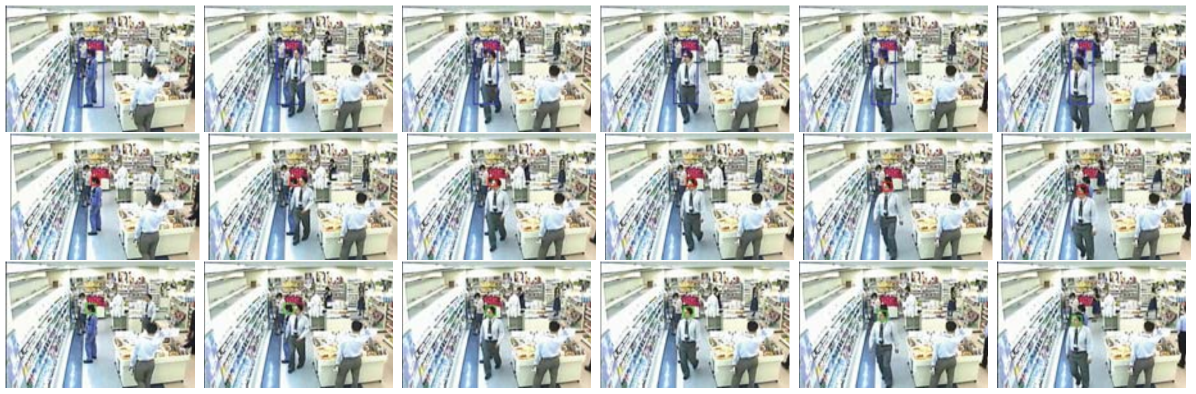


Fig. 13. Tracking with partial occlusion [occlusion] (first row) the CNN tracker, (second row) the ensemble tracker, (third row) the support vector tracker.

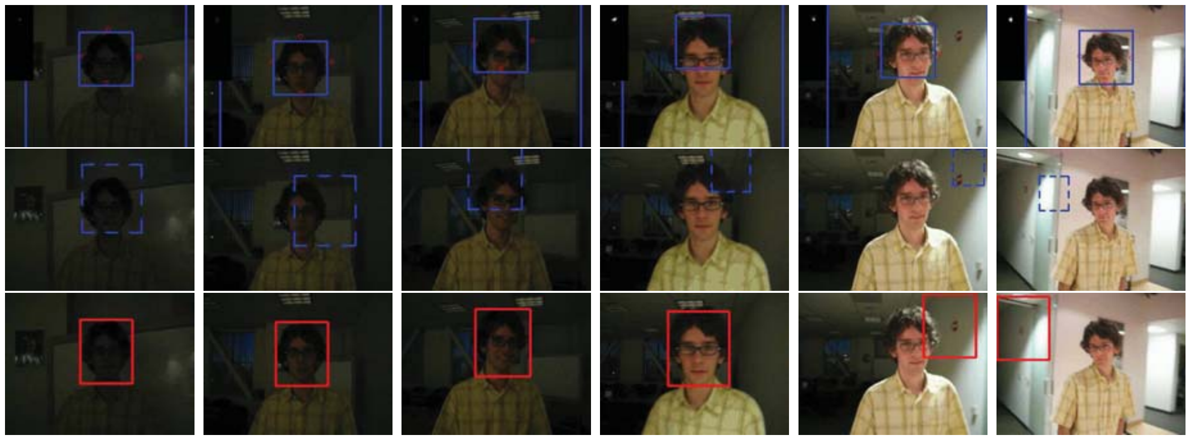


Fig. 14. Tracking with illumination changes [indoor] (first row) the CNN tracker, (second row) the mean-shift tracker, (third row) the clustering method.

F. Tracking with Partial Occlusion

Fig. 13 shows tracking one person with partial occlusion. Such a task is generally difficult, as the learning-based methods adapt to other image regions when the target is partially occluded by the distracters. For the ensemble tracker, only spatial features are extracted, so the tracker is simply distracted because the head of the distracter also gives a good matching. On the contrary, the CNN tracker extracts temporal features so that the motion information is captured. In addition, we observe that, although the person is severely occluded, the two key points (the centers of the top and the left sides of the head) provide good probability scores. Therefore, the scale of the person is accurately estimated according to (7). Hence, the ambiguity is largely reduced, as the scales of the distracter and the target are inconsistent.

G. Illumination Changes

As shown in Fig. 14, the video sequence is very challenging due to dramatic illumination changes. The mean-shift tracker and the clustering method drift to the background and are unable to recover. However, the CNN tracker is able to track the object and correct the drift. This further verifies the fact

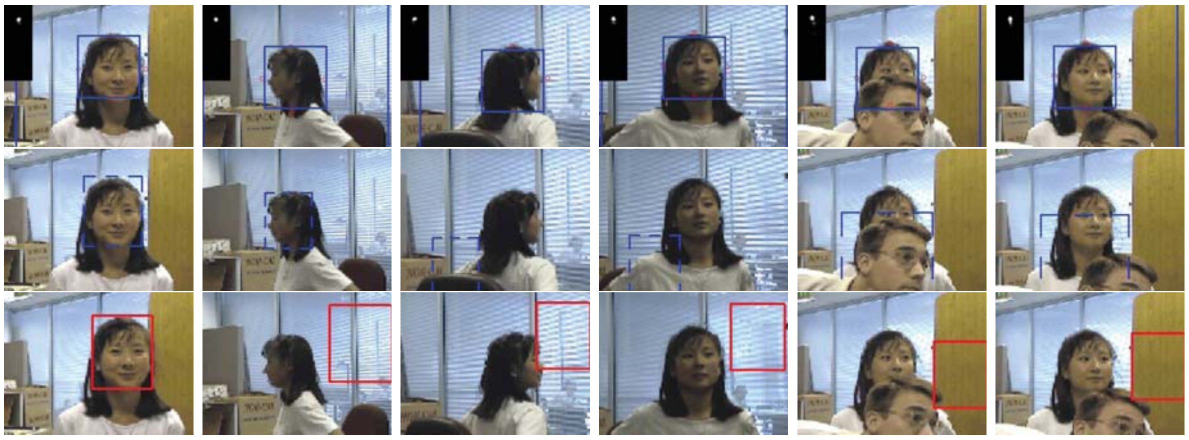


Fig. 15. Tracking with scale and view changes [girl]: (first row) the CNN tracker; (second row) the mean-shift tracker; (third row) the clustering method.

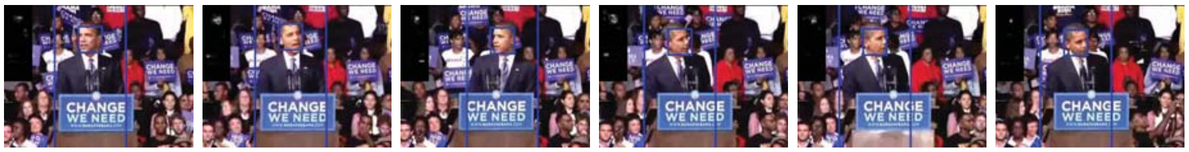


Fig. 16. Example of handling view change.

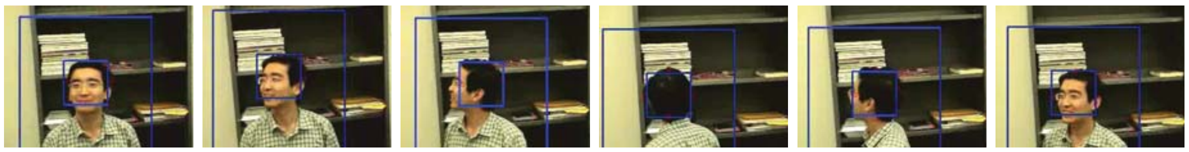


Fig. 17. Another example of handling view change.

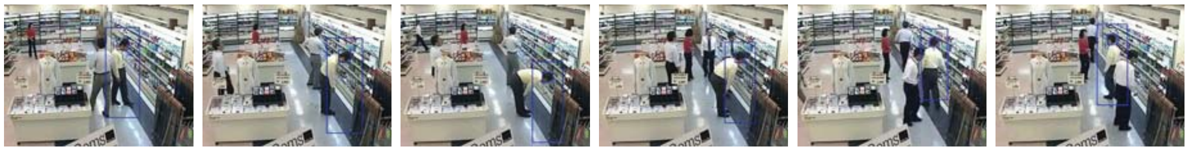


Fig. 18. Example of handling pose change.

(Section V-D) that our model has less chance to drift to unrelated types of object. Note that the maximum of the probability map is very small in the first few frames, since we do not have training samples in such dark environments, which shows that our method is proper for various environments.

H. Scale and View Changes

We show the tracking performance when the target undergoes large scale and view changes in Fig. 15. For the mean-shift tracker and the clustering method, the tracker drifts to the background when the view change occurs. While for the proposed CNN tracker, it discovers four key points (illustrated by red circles in figures) which give an accurate boundary of the human head, as we collect different views and scales of human in training samples.

I. More Experiments

Figs. 16 and 17 show the results of tracking a head presenting rotation and view changes. The appearances of different views of the head are significantly different, which makes the tracking difficult. Our experiment shows that the CNN tracker can successfully track the head, since the CNN model learns the head with different views during offline training. Therefore, the CNN tracker can handle the case of dramatic view changes. Fig. 18 illustrates the result of tracking a person undergoing severe pose changes. Although the CNN tracker extracts the spatial features from the background, it successfully captures the motion information so that the tracker does not drift when the person gets up.

J. Discussion

As demonstrated in a large number of challenging sequences, there are two primary scenarios when the CNN tracker performs much more stable than the reference methods. 1) The target undergoes drastic view change or pose change. For traditional learning-based tracking methods, the appearance models adapt to other image regions rather than the target of interests since the appearance of the target changes dramatically. On the contrary, the CNN tracker is for a specific object class, so it is unlikely for the tracker to drift to unrelated types of object. 2) There are some false-positive matches that lead to wrong association of the tracks. This is a challenging problem for all learning based methods. However, we successfully alleviate this problem by in a unified shift-variant CNN model.

Since the features are learned from a parametric feature pool in neural networks, they have richer degrees of freedom than the features used in [5]. This data-driven feature extraction approach naturally fits the object class tracking problem, as the best features to track different object classes may be different.

In our experiment, we treat humans as our subject for tracking. We did not perform the experiment on other object classes (e.g., cars), because we have not collected those training samples (this may be our future work). Now we explain how the proposed CNN can be extended to general object class tracking from several perspectives.

1) Note that the features are learned during the training procedure. This training framework (the CNN model) is very general and not designed for a certain object class. So the proper features for other object class tracking can be learned in a similar CNN model.
2) The relative position of the object and the input image patch can be chosen arbitrarily. As shown in Figs. 3 and 7, the relative positions are different in these cases. Therefore, if we wish to use the CNN model to track a car, the input image patches include the whole car, and the peak of the probability map can be located at the center of the input image patch for training. If we wish to track a pedestrian, one way is to train a new CNN model (like the situation of cars). An alternative way is to use the proposed CNN model, and the pedestrian position can be estimated by the bounding box (blue rectangle in Fig. 3). In fact, we do not distinguish between head tracking and pedestrian tracking very much. On one hand, the objective of pedestrian tracking is to obtain the position of a moving person. This can be achieved by accurately locating the human heads. On the other hand, in complex or crowded scenarios, the whole human bodies may not be available due to occlusion, therefore locating the head position becomes the most critical cue to determine the pedestrian position.
3) In [5], the input image patch is a square equally spaced around the head, which is different from the proposed method, as the input here is the whole human body. We emphasize that the improvement over [5] is mainly because the proposed method can largely alleviate the drift problem when the distracting objects are similar to the target (by using learned temporal and spatial features in a novel CNN model), while the approach in [5] does not contain such mechanisms.

In the proposed model, we estimate the location of the human head on the basis of a large image patch which includes the whole human body. The motivation is similar to the basic principle of contextual flow [33]. The target is rarely isolated and independent of the environment, but related to its spatial context that is induced by the pixels in its vicinity. For example, the silhouette of the shoulder and the torso provides useful information for locating the head position. To take advantage of such contextual information for robust tracking, the CNNs learn the object and its surrounding context. Hence, we may need 20 000 training samples in order to learn the head and all the possible backgrounds that can be found.

An interesting problem is whether we can use fewer training samples in our model to achieve comparable tracking results. We conducted the experiment with a CNN tracker trained using 1000 samples (randomly chosen from 20 000 training samples). In general, the tracking performance does not degrade (if there are no distractions). However, for some challenging cases, the tracker succeeded (e.g., in occlusion), but drifted in others (e.g., shopping) due to the cluttered background. This implies that training samples should be selected more carefully when we use fewer samples. The problem of how to select representative training samples is nontrivial and may be our future work.

VI. Conclusion

In this paper, we proposed a novel learning method for tracking based on CNNs. Considering human tracking as a special case of object class tracking, spatial and temporal structures have been learned during offline training. The shift-variant architecture extended the use of conventional CNNs, and combined global features and local features in a natural way. The tracking of key points was used to solve the scale problem, which was formed in a general way that can be extended to an arbitrary object class tracking problem.

The main limitation is that the CNN model is not designed to handle full and long-term occlusions by the distracter of the same object class. This limitation commonly exists for all the learning-based methods.

Appendix

The proof of Property 3.1 is as follows.

Generally, denote the size of the normalized input patch of CNNs by $w \times h$, then the size of the probability map is $(w/2) \times (h/2)$. In our CNN architecture, $w = 48$, $h = 128$ (Fig. 3). Since there is four-times upsampling from C_3 to the output probability map, every pixel in a 4×4 block in the output probability map shares the same global receptive field. Since the size of the output probability map is one-half of the input patch, the $w \times h$ normalized input patch can be divided into several adjacent blocks with the same size 8×8. Every pixel in one certain 8×8 block shares the same global receptive field. We denote the scaling factor $m = 8$. The number of the adjacent blocks is equal to the size of the C_3 feature map, denoted by $(x_m+1) \times (y_m+1)$ (in Fig. 4, $x_m = 5$, $y_m = 15$).

Denote the size of the global receptive field by $w_r \times h_r$. For (i, j)th pixel in the output probability map, its corresponding global receptive field in the normalized input patch is $[(1/2)m\lfloor i/4 \rfloor, (1/2)m\lfloor i/4 \rfloor + w_r - 1] \times$

$[(1/2)m\lfloor j/4 \rfloor, (1/2)m\lfloor j/4 \rfloor + h_r - 1]$. Let $i_m = (w/2) - 1$, we have $(1/2)m\lfloor i_m/4 \rfloor + w_r - 1 = w - 1$. As $\lfloor i_m/4 \rfloor = x_m$, it is clear that

$$\frac{1}{2} m x_m + w_r = w. \tag{8}$$

Similarly, we have

$$\frac{1}{2} m y_m + h_r = h. \tag{9}$$

The width of C_3 feature map is $(1/8)m(x_m + 1)$. From (8), it can be represented by $1/4(w - (w_r - (m/2)))$. On the other hand, the width of C_3 feature map is $(1/8)w$ due to four-times upsampling in the last stage. So we have $(1/4)(w - (w_r - (m/2))) = (1/8)w$

$$2w_r = m + w. \tag{10}$$

Let us prove $\mathbf{x}_C = (\mathbf{x}_O + \mathbf{x}_A)/2$ in Property 3.1. Recall that the $w \times h$ normalized input patch can be divided into $(x_m+1) \times (y_m+1)$ adjacent blocks with the same size $m \times m$, and the pixels in each block share the same receptive field. Suppose A is located in one $m \times m$ block with the index (x, y), $x \in \{0, 1, ..., x_m\}$, $y \in \{0, 1, ..., y_m\}$. And we assume A is at the center of that $m \times m$ block (if A is not at the center, then the equation approximately holds). Then we have $\mathbf{x}_A = (mx + ((m-1)/2), my + ((m-1)/2))$. As C is the center of the (x, y)th receptive field, and the (x, y)th receptive field is $[(1/2)mx, (1/2)mx + w_r - 1] \times [(1/2)my, (1/2)my + h_r - 1]$, we have $\mathbf{x}_C = ((m/2)x + ((w_r - 1)/2), (m/2)y + ((w_r - 1)/2))$. We also have $\mathbf{x}_O = (((w-1)/2), ((w-1)/2))$.

It is easy to see $\mathbf{x}_C = (\mathbf{x}_O + \mathbf{x}_A)/2$ from (10). Then we finish the proof. In our CNN architecture, from (8) and (9), $w_r = 28$, $h_r = 68$.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments. The main part of this paper was done when J. Fan was a summer intern at NEC Laboratories America, Inc., Cupertino, CA.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer, "The variable bandwidth mean shift and data-driven scale selection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Vancouver, Canada, Jul. 2001, pp. 438–445.

[2] J. Fan, M. Yang, and Y. Wu, "A bi-subspace model for robust visual tracking," in *Proc. IEEE Int. Conf. Image Process.*, San Diego, CA, Oct. 2008, pp. 2660–2663.

[3] A. D. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Kauai, HI, Dec. 2001, pp. 415–422.

[4] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1208–1221, Sep. 2004.

[5] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[6] R. T. Collins and Y. Liu, "On-line selection of discriminative tracking features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Madison, WI, Jun. 2003, pp. 346–352.

[7] J. Ho, K. Lee, M.-H. Yang, and D. J. Kriegman, " Visual tracking using learned linear subspaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Washington D.C., Jun. 2004, pp. 782–789.

[8] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, Jun. 2008, pp. 1–8.

[9] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *Proc. Eur. Conf. Comput. Vis.*, Marseille, France, Oct. 2008, pp. 788–801.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[11] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *Proc. Eur. Conf. Comput. Vis.*, Marseille, France, Oct. 2008, pp. 69–82.

[12] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[13] O. Matan, C. Burges, Y. LeCun, and J. Denker, "Multi-digit recognition using a space displancement neural networks," in *Proc. Neural Inform. Process. Syst.*, Denver, CO, Nov. 1992, pp. 488–495.

[14] S. Nowlan and J. Platt, "A convolutional neural network hand tracker," in *Proc. Neural Inform. Process. Syst.*, Denver, CO, Nov. 1994, pp. 901–908.

[15] B. Leibe, K. Schindler, and L. Van Gool, "Coupled detection and trajectory estimation for multi-object tracking," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[16] K. Okuma, A. Taleghani, O. Freitas, J. Little, and D. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Proc. Eur. Conf. Comput. Vis.*, Prague, Czech Republic, May 2004, pp. 28–39.

[17] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco, "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.

[18] O. Tuzel, F. Porikli, and P. Meer, "Learning on lie groups for invariant detection and tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, Jun. 2008, pp. 1–8.

[19] Y. Wu and T. Yu, "A field model for human detection and tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 753–765, May 2006.

[20] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors," *Int. J. Comput. Vis.*, vol. 75, no. 2, pp. 247–266, Nov. 2007.

[21] A. Ess, B. Leibe, K. Schindler, and L. V. Gool, "A mobile vision system for robust multi-person tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, Jun. 2008, pp. 1–8.

[22] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 65–81, Jan. 2007.

[23] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.

[24] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, New York, Jun. 2006, pp. 260–267.

[25] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1728–1740, Oct. 2008.

[26] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, May 2008.

[27] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. Eur. Conf. Comput. Vis.*, Marseille, France, Oct. 2008, pp. 234–247.

[28] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, Jun. 2009, pp. 983–990.

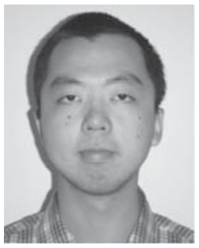
[29] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962.

[30] R. T. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Madison, WI, Jun. 2003, pp. 234–240.

[31] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Hilton Head, SC, Jun. 2000, pp. 142–149.

[32] M. Yang, Z. Fan, J. Fan, and Y. Wu, "Tracking non-stationary visual appearances by data-driven adaptation," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1633–1644, Jul. 2009.

[33] Y. Wu and J. Fan, "Contextual flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, Jun. 2009, pp. 33–40.



Jialue Fan (S'08) received the B.E. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree in the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL.

He completed internships at NEC Laboratories America, Inc., Cupertino, CA, Siemens Corporate Research Center, Princeton, NJ, and Adobe Systems Inc., San Jose, CA, in 2008, 2009, and 2010, respectively. His current research interests include computer vision, image/video processing, and machine learning.

Mr. Fan was a recipient of the Walter P. Murphy Fellowship at Northwestern University in 2007.

Wei Xu received the B.S. degree from Tsinghua University, Beijing, China, in 1998, and the M.S. degree from Carnegie Mellon University (CMU), Pittsburgh, PA, in 2000.

He was a Research Assistant at the Language Technology Institute, CMU, from 1998 to 2001. He was with NEC Laboratories America, Inc., Cupertino, CA, in 2001, working on intelligent video analysis. He has been a Research Scientist at Facebook Inc., Palo Alto, CA, since November 2009. His current research interests include computer vision, image and video understanding, machine learning, and data mining.

Ying Wu (SM'06) received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1994, the M.S. degree from Tsinghua University, Beijing, China, in 1997, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, in 2001.

He was a Research Assistant at the Beckman Institute for Advanced Science and Technology at UIUC from 1997 to 2001. He was a Research Intern with Microsoft Research, Redmond, WA, from 1999 to 2000. He has been with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, since 2001. His current research interests include computer vision, image and video analysis, pattern recognition, machine learning, multimedia data mining, and human-computer interaction.

Dr. Wu was a recipient of the Robert T. Chien Award at UIUC in 2001 and the NSF CAREER award in 2003. He serves as an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING, SPIE *Journal of Electronic Imaging*, and IAPR *Journal of Machine Vision and Applications*.

Yihong Gong received the B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively.

He was with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he worked as an Assistant Professor for 4 years. He worked at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, as a Project Scientist, from 1996 to 1998, where he served as a Principal Investigator for both the Informedia Digital Video Library project and the Experience-On-Demand project funded by NSF, DARPA, NASA, and other government agencies. He has been with NEC Laboratories America, Inc., Cupertino, CA, since 1999, working as a Senior Research Staff Member, Department Head, and Site Manager. He established the multimedia analysis group at NEC for the company and initiated many research projects in the areas of video content analysis and summarization, information extraction, and event detection. He has published more than 100 technical papers. He is the author or co-author of two monographs, and has contributed chapters to two multimedia handbooks. His current research interests include video content analysis, summarization, and machine learning.