# SIAMESE RECURRENT ARCHITECTURE FOR VISUAL TRACKING

*Xiaqing Xu*[1,2], *Bingpeng Ma*[1,2], *Hong Chang*[1,*], *Xilin Chen*[1]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences(CAS), Institute of Computing Technology, CAS, Bejing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
{xiaqing.xu, bingpeng.ma, hong.chang, xilin.chen}@vipl.ict.ac.cn

## ABSTRACT

Treating visual tracking as a matching problem, siamese architecture has drawn increasing interest recently. In this paper, we propose a novel siamese recurrent architecture that can enhance the similarity matching by leveraging contextual information. Specifically, the multi-directional Recurrent Neural Network (RNN) is employed to memorize the long-range contextual dependencies of object parts and learn the self-structure information of the object. We test the proposed method on a challenging benchmark, and it gain promising results compared with the existing tracking algorithms.

***Index Terms*—** Siamese architecture, spatial recurrent neural network, contextual dependency, visual tracking

## 1. INTRODUCTION

Visual object tracking is a fundamental building block of computer vision applications. Despite significant progress has been made in the past several decades, real situations still pose enormous challenges such as drastic deformation, severe occlusions, cluttered background and fast motion.

Most existing tracking algorithms can be generally divided into generative and discriminative ones. Generative trackers utilize generative models to describe the target appearance and search for the candidates to minimize reconstruction errors. Some representative algorithms are sparse representation[1][2], online density estimation[3]. In contrast, discriminative trackers learn to discriminate foreground from background through a classifier. Various learning algorithms have been applied to visual tracking including multiple instance learning[4], online boosting[5], structured SVM[6], correlation filters[7], etc. Those algorithms usually adopt low-level hand-craft local features, which make them not robust to significant appearance changes.

Inspired by the great success of deep neural networks, many CNN-based trackers have been proposed[8][9][10]. Among them, [9][10] deploy the two-stream siamese convolutional architecture for visual tracking. They learn matching score functions through external annotated videos. During tracking, candidate patches in the current frame are compared with the initial one of the target in the first frame. The most similar patch is chosen by the learned matching function. However, the features learned from the siamese convolutional network are extracted from local regions. As a result, they are independent from the features of other regions.

In this paper, we propose a novel matching function by leveraging contextual information. RNNs have been adopted to model such spatial correlations successfully in various visual applications such as object detection[11], scene labelling[12] and text detection [13]. To gather the contextual information of the tracking object, we adopt multi-directional RNN to pass the spatially variations both horizontally and vertically across the object feature map. On one hand, the spatial RNNs can learn long-range contextual dependencies between parts of the object, which make the matching function have better discriminative ability to distinguish the current object from the similar ones in the background. On the other hand, the generated representation of the target is translation-invariant to some extent due to the spatial RNNs' recurrent proceed on local parts. The matching function can still recognize the object suffers from significant appearance changes in the current frame. Based on the advantages of spatial RNNs, we incorporated it into the siamese convolutional neural network to leverage local representations learned by the convolutional layers. Besides, a concatenation strategy to fuse the CNN and RNN feature maps is implemented to supply the siamese recurrent architecture with more information.

## 2. SIAMESE RECURRENT ARCHITECTURE

In this section, we first present the spatial RNNs for context information. Then we describe the whole tracking method, named Siamese Recurrent Tracker, SRT for abbreviation.

### 2.1. RNNs for Context Features

Our architecture for computing context features in SRT is shown in more details in Figure 1. The traditional RNN moves left-to-right along a sequence, updating its hidden

*: Corresponding Author

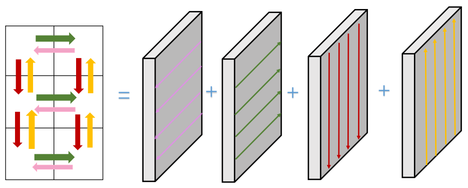
**Fig. 1**. Four-directional IRNN architecture.

state and producing an output. We extend this to two dimensions by placing a RNN along each row and each column of the image.

There are many possible forms of RNN that we could choose. In this paper, we explore RNNs composed of rectified linear units (ReLU)[14]. Le *at al.* have shown that these networks are easy to train and have a good capability of modeling long-range dependencies, if the recurrent weight matrix is initialized to the identity matrix. They name a ReLU RNN initialized this way as an "IRNN", and show that it performs almost as well as LSTM[15] for a real-world language modeling task. The IRNN is much faster than LSTM and has smaller parameter search space to avoid over-fitting.

Concretely, we suppose the feature map of object $\mathbf{x}$ is represented by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}_{i=1:N}$ denotes the vertex set and $\mathcal{E} = \{e_{i,j}\}$ is the edge set($e_{i,j}$ denotes an edge from $v_i$ to $v_j$). The structure of the spatial RNNs follows the same topology as $\mathcal{G}$. The forward propagation sequence can be seen as traversing $\mathcal{G}$ from the start point. Each vertex relies on its predecessor. For vertex $v_i$, the hidden output $h^{(v_i)}$ is expressed as a non-linear function (ReLU here) over its local input $\mathbf{x}^{(v_i)}$ and the hidden representation of its predecessor. In detail, the forward operation of the spatial RNNs is calculated by the following equations:

$$\begin{aligned}\mathbf{h}^{(v_i)} &= f(U\mathbf{x}^{(v_i)} + W\mathbf{h}^{p(v_i)} + b)\\ \mathbf{o}^{(v_i)} &= g(V\mathbf{h}^{(v_i)} + c)\end{aligned} \tag{1}$$

where $\mathbf{x}^{(v_i)}$, $\mathbf{h}^{(v_i)}$ and $\mathbf{o}^{(v_i)}$ are the representations of input, hidden and output layers located at $v_i$, respectively, $p(v_i)$ is the direct predecessor of vertex $v_i$. $U$ is the weight matrix between the input and hidden layers. $W$ is the recurrent matrix between the previous hidden unit and the current hidden units. $W$ is initialized to the identity matrix in our IRNN. While $V$ is the output matrix connecting the hidden and output layers, $b$ and $c$ are the bias vectors and $f(\cdot)$ and $g(\cdot)$ are the element-wise non-linear activation functions. Since the local information is progressively stored into the hidden layers by applying Equation 1, the contextual information (the summarization of the preceding information) is encoded into $\mathbf{h}^{(v_i)}$.

The derivatives are computed in the backward pass, and each vertex is processed in the reverse order of forward propagation sequence. We use $\mathbf{s}(v_i)$ to represent the direct successor for vertex $v_i$ in $\mathcal{G}$. It can be inferred from Equation 1 that the errors backpropagated to the hidden layer ($d\mathbf{h}^{(v_i)}$) have two sources: direct errors from its output $\mathbf{o}^{(v_i)}$ ($\frac{\partial \mathbf{o}^{(v_i)}}{\partial \mathbf{h}^{(v_i)}}$) and indirect errors propagated from its successor ($\frac{\partial \mathbf{o}^{s(v_i)}}{\partial \mathbf{h}^{s(v_i)}} \frac{\partial \mathbf{h}^{s(v_i)}}{\partial \mathbf{h}^{(v_i)}}$). The derivatives at $v_i$ can then be computed by the following equations:

$$\begin{aligned}\Delta V^{v_i} &= g^{'}(\mathbf{o}^{(v_i)})(\mathbf{h}^{(v_i)})^T\\ d\mathbf{h}^{(v_i)} &= V^T g^{'}(\mathbf{o}^{(v_i)}) + W^T(d\mathbf{h}^{s(v_i)} \circ f^{'}(\mathbf{h}^{s(v_i)}))\\ \Delta W^{(v_i)} &= \mathbf{h}^{s(v_i)} \circ f^{'}(h^{s(v_i)})(h^{(v_i)})^T\\ \Delta U^{(v_i)} &= d\mathbf{h}^{(v_i)} \circ f^{'}(\mathbf{h}^{(v_i)})(\mathbf{x}^{(v_i)})^T\end{aligned} \tag{2}$$

where $\circ$ is the Hardmard product, $g^{'} = \frac{\partial L}{\partial \mathbf{o}(\cdot)} \frac{\partial \mathbf{o}(\cdot)}{\partial g}$ is the derivative of loss function with respect to the output function $g$, $f^{'}(\cdot) = \frac{\partial h}{\partial f}$. The indirect errors propagated to $d\mathbf{h}^{(v_i)}$ in the second term of Equation 2 enables our spatial RNNs to propagate contextual information.

As shown in Figure 1, we decompose the graph into four directions: right, left, down, up. Let $\mathcal{G}^{\mathcal{U}} = \{\mathcal{G}^1, \mathcal{G}^2, \mathcal{G}^3, \mathcal{G}^4\}$ represent the composed graphs, where $\mathcal{G}^1$, $\mathcal{G}^2$, $\mathcal{G}^3$ and $\mathcal{G}^4$ represent the four directed graph. The aggregation of the different hidden layers yields the output layer $\mathbf{o}$. These operations can be mathematically expressed as follows:

$$\begin{aligned}\mathbf{h}_d^{(v_i)} &= f(U_d\mathbf{x}^{(v_i)} + W\mathbf{h}^{p(v_i)} + b_d)\\ \mathbf{o}^{(v_i)} &= g(\sum_{\mathcal{G}_d \in \mathcal{G}^{\mathcal{U}}} V_d\mathbf{h}_d^{(v_i)} + c)\end{aligned} \tag{3}$$

where $U_d, W_d, V_d, b_d$ and $b_m$ are the matrix parameters and bias term for $\mathcal{G}_d$, $c$ is the bias term for the final output. After the 4-directional IRNNs, each cell on the output layer is the combination of local and global, and can be seen as the global summary of the object feature map.

## 2.2. Proposed Tracking Algorithm

In this section, we illustrate the proposed tracking algorithm in details.

**Network Architecture** The architecture of the proposed network is depicted in Figure 2. We adopt a siamese architecture composed of two branches. The two branches process the inputs separately while share the same network structure and parameters. For each branch, the network first processes the entire image by the VGGNet[16] for a few layers, then the region pooling layers[17] after conv4_3 and conv5_3 layers output fixed-size representations for the current object region. The 4-directional IRNN is implemented after the second region pooling layer to compute the context features that describe the current object both globally and locally. We concatenate the outputs from the RNN and former region pooling layers as the intermediate representations and then feed it to the loss function.

We improve the VGGNet network with several aspects. Firstly, the two max-pooling layers after conv3_3 and conv4_3 are removed to preserve more detailed information for precise

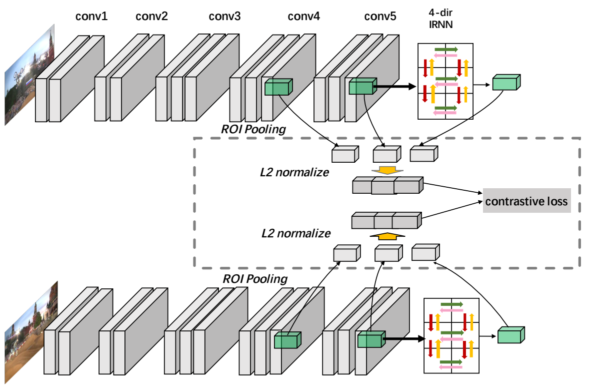

**Fig. 2**. The proposed siamese recurrent architecture.

localization. Max-pooling layer maintains only the strongest of the activations from a local neighborhood, as a result, adopting too many max-pooling layers will lose most of the detailed information. Secondly, the outputs from the RNN and former region pooling layers are concatenated as the intermediate representations and then fed to the loss function. The fused representations combine both contextual and local features, which can supply more information for the siamese architecture. Finally, the $l_2$ normalization layers are adopted after the output before the concatenation to balance the output scales from different layers.

**Loss Function** The two branches in the siamese network are connected with a single loss layer. For the input pair images, we expect the network to generate very close representations for the positive pairs while distinct ones for the negative pairs. As a result, the margin contrastive loss[18] is a good choice here. The core idea of the contrastive loss is to "attract" the similar inputs towards each other and "split" the dissimilar inputs. Let the $l_2$ normalized latent features from the pairs $(I_p, I_q)$ be $x_p$ and $x_q$ respectively, and $y_{pq} \in \{0, 1\}$ indicates whether $I_p$ and $I_q$ are the same object or not. The loss function can be designed as:

$$\mathcal{L}(x_p, x_q, y_{pq}) = \frac{1}{2} y_{pq} D^2 + \frac{1}{2}(1 - y_{pq})\max(0, \epsilon - D^2) \quad (4)$$

where $D = ||x_p - x_q||_2$, and $\epsilon$ is the minimum distance margin that pairs contain different objects should satisfy.

**Online Tracking** After completing the learning of the matching function through network training, we can deploy it for online tracking. The annotated object location in the first frame is put into the first branch of our siamese recurrent network as it is the only reliable data during our tracking period. The candidate boxes in the subsequent frames are passed into another branch of the siamese architecture and the candidate box which matches the original target best is picked:

$$\hat{I}_t = \arg_{I_{j,t}} \max m(I_{t=0}, I_{j,t}) \quad (5)$$

where $I_{j,t}$ are all the candidate boxes at frame $t$, $m$ is the learned matching function, $m(I_p, I_q) = x_p^T x_q$.

**Bounding Box Regression** Provided that the bounding box prediction is accurate enough, [11] [8] showed that a refinement step can significantly improve the quality of localization, and find a more tight bounding box enclosing the target. We adopt their bounding regression technique and refine the predicted bounding box at every frame.

We train four Ridge regressors for the center coordinates, width and height of the box based on the first frame. The features from conv4_3 are adopted for training as the output of ConvNet's lower layer reserves more detailed information. The bounding box regressors are trained only in the first frame and not updated during tracking to avoid the risk of contaminating the regressors with noisy data. For each frame, the regressors take the representation of the picked candidate bounding box as the input and produce a refined box.

## 3. EXPERIMENTS

This section presents the experimental validation of the proposed method.

### 3.1. Implement Details

**Candidate Sampling** The candidate boxes are generated by the radius sampling strategy[6]. We adopt 10 radial and 10 angular divisions. The search radius is set to be the longest axis of the initial box in the first frame. At each sample positions, five scaled versions of the initial box are generated.

**Network Training** We choose 50 videos from the ALOV dataset[19] and the deform SOT[20] dataset for training and validation. We choose two datasets for training because they includes more types of variations which occurs in visual tracking. For ALOV dataset, we exclude the 12 videos that are also in OTB benchmark [21]. For deform SOT dataset, 4 videos are excluded for the same reason. We generate multiple pairs for every two frames in the same video. For the first frame, we use the groundtruth bounding box in all the pairs as the reference sample. For the second frame, multiple bounding boxes are sampled. The pair is considered to be positive if the the sampled box has a intersection-over-union overlap(IOU) larger than 0.7 with the corresponding groundtruth box. The pair is considered to be negative if its IOU is smaller than 0.5. We have sampled 50, 000 pairs of frames for training and each pair of frames has 128 pairs of boxes. For validation, we have gathered 2000 pairs of frames. To make sure that the validation set is independent from the training set, we generate the validation pairs from videos that are not included in the training pairs.

Instead of training the siamese recurrent architecture from scratch, we load the pre-trained VGGNet parameters and finetune the convolutional layers to avoid overfitting. For the 4-directional IRNN, we share the input matrix $U$ for different directions, which is more efficient for implementation. We use the gaussian initialization for $U$ with 0.002 std, while

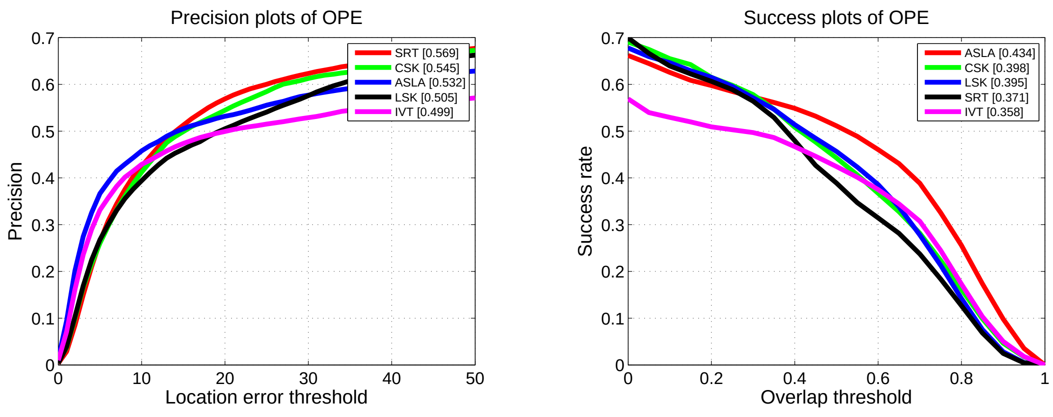

**Fig. 3**. Precision and success plot comparison on OTB.

**Table 1**. Average precision plot on five challenging attributes.

| | DEF | MB | SV | FM | OPR |
|---|---|---|---|---|---|
| ASLA | 0.445 | 0.272 | 0.552 | 0.253 | 0.518 |
| CSK | 0.476 | 0.342 | 0.503 | 0.381 | 0.540 |
| LSK | 0.437 | 0.254 | 0.461 | 0.357 | 0.503 |
| IVT | 0.409 | 0.222 | 0.494 | 0.220 | 0.464 |
| SRT(ours) | **0.597** | **0.544** | **0.558** | **0.544** | **0.546** |

“Xavier” initialization for $V$. The initial learning rate is set to 0.0005. The learning rate is decreased by 2 after each epoch.

## 3.2. Evaluation on OTB

To evaluate the tracking performance, we use the OTB benchmark [21]. OTB is a popular tracking benchmark containing 50 fully annotated video. 51 tracking sequences are defined with bounding box annotations. The sequences are further tagged with 11 attributes according to different challenging factors, such as fast motion, motion blur and deformation.

We adopt the evaluation metrics provided by OTB: success plot and precision plot. Both metrics measure the percentage of successfully tracked frames. For the success plot, a frame is considered to be successfully tracked if the IOU of the predicted bounding box and the groundtruth box is larger than the threshold. For the precision plot, a frame is declared to be successful if the distance between centers of the predicted box and the groundtruth box is under a certain threshold. Trackers are ranked based on the precision at threshold 20(Prec @20) and Area Under Curve (AUC) score for the success plot.

The comparison results are shown in Figure 3. Our SRT outperforms the popular trackers such as CSK [22], ASLA[1], IVT[23] and LSK[24] by a considerable margin on precision plot, while a bit lower on the success plot. We attribute the improvement of the precision plot to the enhanced matching function. The matching function can successfully recognize target object even the object suffers from significant appearance changes in the current frame. However, since the annotated bounding boxes in the deform SOT dataset are extremely tight and often only enclose the main body of no-rigid object, the bounding boxes generated by our method is often smaller than OTB’s annotations, which result to the slightly inferior performance on success plot.

To facilitate better analysis on the tracker performance, we further evaluate all the tracker on 5 challenging attribute subsets: severe appearance changes(DEF), motion blur(MB), drastic scale changes(SV), fast motion(FM) and out of plane rotation(OPR). Table 1 demonstrate the results of the different methods. Our SRT performs robustly while these trackers only adopt local features drift easily. These results show that compared to other trackers who only use local representations, the proposed siamese recurrent architecture has enhanced the matching ability through leveraging the contextual information. The generated representation of the target is more translation-invariant to some extents.

## 4. CONCLUSION

In this paper, we propose a novel siamese recurrent architecture for visual tracking. A four-directional IRNN is adopted to model the contextual information of object. The proposed tracker is trained and evaluated on the challenging tracking benchmark. Experiment results verify that the proposed method can enhance the performance of matching function especially challenging situations.

## 5. ACKNOWLEDGEMENT

This work was supported in part by National Basic Research Program of China (973 Program): 2015CB351802, and Natural Science Foundation of China (NSFC): 61390515 and 61572465.

## 6. REFERENCES

[1] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Computer Vision and Pattern Recognition*, 2012, pp. 1822–1829.

[2] Xue Mei and Haibin Ling, "Robust visual tracking using $l1$ minimization," in *International Conference on Computer Vision*, 2009, pp. 1436–1443.

[3] Bohyung Han, Dorin Comaniciu, Ying Zhu, and Larry S Davis, "Sequential kernel density approximation and its application to real-time visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1186–1197, 2008.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[5] Amir Saffari, Martin Godec, Thomas Pock, Christian Leistner, and Horst Bischof, "Online multi-class lpboost," in *Computer Vision and Pattern Recognition*, 2010, pp. 3570–3577.

[6] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr, "Struck: Structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[7] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[8] Hyeonseob Nam and Bohyung Han, "Learning multidomain convolutional neural networks for visual tracking," in *Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[9] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders, "Siamese instance search for tracking," in *Computer Vision and Pattern Recognition*, 2016, pp. 1420–1429.

[10] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision*, 2016, pp. 850–865.

[11] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Computer Vision and Pattern Recognition*, 2016, pp. 2874–2883.

[12] Bing Shuai, Zhen Zuo, Bing Wang, and Gang Wang, "Dag-recurrent neural networks for scene labeling," in *Computer Vision and Pattern Recognition*, 2016, pp. 3620–3629.

[13] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao, "Detecting text in natural image with connectionist text proposal network," in *European Conference on Computer Vision*. Springer, 2016, pp. 56–72.

[14] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[15] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] Ross Girshick, "Fast r-cnn," in *International Conference on Computer Vision*, 2015, pp. 1440–1448.

[18] Sumit Chopra, Raia Hadsell, and Yann LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 539–546.

[19] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[20] Dawei Du, Honggang Qi, Wenbo Li, Longyin Wen, Qingming Huang, and Siwei Lyu, "Online deformable object tracking based on structure-aware hyper-graph," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3572–3584, 2016.

[21] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[22] Jo Henriques, o F, Rui Caseiro, Pedro Martins, and Jorge Batista, *Exploiting the Circulant Structure of Tracking-by-Detection with Kernels*, Springer Berlin Heidelberg, 2012.

[23] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[24] Baiyang Liu, Junzhou Huang, Lin Yang, and Casimir Kulikowsk, "Robust tracking using local sparse appearance model and k-selection," in *Computer Vision and Pattern Recognition*, 2011, pp. 1313–1320.