Probabilistic Regression for Visual Tracking

Supplementary Material

Martin Danelljan Luc Van Gool Radu Timofte
Computer Vision Lab, D-ITET, ETH Zürich, Switzerland

In this supplementary material of [6], we first derive the employed loss from the KL divergence in Section 1. We then provide details about the target center regression module in Section 2. Lastly, we report more detailed results in Section 3 and provide additional visualizations of the predicted distributions in Section 4.

1. Derivation of KL Divergence Loss

Here, we derive the loss in Eq. (8) in the paper from the KL divergence between the predicted distribution $p(y|x_i, \theta)$ and the ground-truth density $p(y|y_i)$. Starting from the definition of the KL divergence and inserting our formulation (Eq. (6) in the paper), we achieve,

$$\begin{aligned}
\mathrm{KL}(p(\cdot|y_i), p(\cdot|x_i,\theta)) &= \int p(y|y_i) \log \frac{p(y|y_i)}{p(y|x_i,\theta)} \mathrm{d}y \\
&= \int p(y|y_i) \log \frac{p(y|y_i)}{e^{s_\theta(y,x_i)}/Z_\theta(x_i)} \mathrm{d}y \\
&= \int p(y|y_i) \Big(\log p(y|y_i) - \log e^{s_\theta(y,x_i)} + \log Z_\theta(x_i) \Big) \mathrm{d}y \\
&= \int p(y|y_i) \log p(y|y_i) \mathrm{d}y + \log Z_\theta(x_i) - \int p(y|y_i) s_\theta(y,x) \mathrm{d}y \\
&\sim \log \left(\int e^{s_\theta(y,x_i)} \mathrm{d}y \right) - \int s_\theta(y,x_i) p(y|y_i) \mathrm{d}y \,. \qquad (1)
\end{aligned}$$

In the last row, we have discarded the first term (the negative entropy of $p(y|y_i)$) and substituted the definition of the partition function $Z_\theta(x_i)$ (Eq. (6) in the paper).

2. Target Center Regression Module

In this section, we give a detailed description and derivation of the optimization module employed for target center regression in our PrDiMP tracker. The goal of the optimizer module is to predict the weights w_θ of the target center regression component,

$$s_\theta^{\mathrm{tc}}(y^{\mathrm{tc}}, x) = (w_\theta * \phi_\theta(x))(y^{\mathrm{tc}}) \,. \qquad (2)$$

Here, x is an input image, $y^{\mathrm{tc}} \in \mathbb{R}^2$ is an image coordinate and ϕ_θ is the backbone feature extractor (see Section 4.2 in the paper). The weights w_θ are learned from a set of training (support) images $\{z_j\}_{j=1}^n$ and corresponding target bounding box annotations $\{\tilde{y}_j^{\mathrm{bb}}\}_{j=1}^n$. As in the baseline DiMP [2], these images are sampled from an interval within each sequence during training. During tracking, the images z_j are obtain by performing augmentations on the first frame, and by gradual update of the memory.

In DiMP, the optimizer module is derived by applying the steepest descent algorithm to a least-squares objective. In our case however, the employed loss function does not admit a least-squares formulation. In this work, we therefore replace the Gauss-Newton approximation with the quadratic Newton approximation in order to compute the step-length necessary for the steepest descent algorithm. We derive closed form solutions of all operations, ensuring simple integration of the optimizer module as a series of deep neural network layers.

Similarly to our offline training objective, we let the optimizer module minimize the KL-divergence based learning loss (Eq. (8) in the paper). We also add an L^2 regularization term to benefit generalization to unseen frames. The loss is thus formulated as follows,

$$L(w_\theta) = \sum_{j=1}^{n} \gamma_j L_{\text{CE}}(\tilde{z}_j * w_\theta; p_j) + \frac{\lambda}{2}\|w_\theta\|^2 \,. \tag{3}$$

The non-negative scalars λ and γ_j control the impact of the regularization term and sample z_j respectively. We also make the following definitions for convenience,

$$\tilde{z}_j = \phi_\theta(z_j) \qquad \text{Extracted image features.} \tag{4a}$$
$$p_j^{(k)} = p(y^{\text{tc},(k)}|y_j^{\text{tc}}) \qquad \text{Label density value at location } k. \tag{4b}$$
$$s_j^{(k)} = (\tilde{z}_j * w_\theta)(y^{\text{tc},(k)}) \qquad \text{Convolution output scores at location } k. \tag{4c}$$
$$\hat{p}_j^{(k)} = \frac{\exp(s_j^{(k)})}{\sum_{l=1}^{K}\exp(s_j^{(l)})} \qquad \text{SoftMax of the convolution output scores.} \tag{4d}$$

Note that we use superscript $k \in \{1, \dots, K\}$ to denote spatial grid location $y^{\text{tc},(k)} \in \mathbb{R}^2$. In the following, the quantities in (4b)-(4d) are either seen as vectors in $\mathbb{R}^K$ or 2D-maps $\mathbb{R}^{H\times W}$ (with $K = HW$), as made clear from the context.

In (3), the per-sample loss L_{CE} is the grid approximation (Eq. (9) in the paper) of the original KL-divergence objective. Without loss of generality, we may assume $A = 1$, obtaining

$$L_{\text{CE}}(s;p) = \log\left(\sum_{k=1}^{K} e^{s^{(k)}}\right) - \sum_{k=1}^{K} s^{(k)}p^{(k)} = \log\left(\mathbf{1}^{\text{T}}e^{s}\right) - p^{\text{T}}s \,. \tag{5}$$

Here, $\mathbf{1}^{\text{T}} = [1, \dots, 1]$ denotes a vector of ones. Note that the grid approximation thus corresponds to the SoftMax-Cross Entropy loss, commonly employed for classification.

To derive the optimization module, we adopt the steepest descent formulation [2], but employ the Newton approximation discussed above. This results in the following optimization strategy,

$$w_\theta^{(i+1)} = w_\theta^{(i)} - \alpha^{(i)}\nabla L(w_\theta^{(i)}) \,, \qquad \alpha^{(i)} = \frac{\nabla L(w_\theta^{(i)})^{\text{T}}\nabla L(w_\theta^{(i)})}{\nabla L(w_\theta^{(i)})^{\text{T}}H(w_\theta^{(i)})\nabla L(w_\theta^{(i)})} \,. \tag{6}$$

Here, $\nabla L(w_\theta^{(i)})$ and $H(w_\theta^{(i)})$ is the gradient and Hessian of L (3), evaluated at the current estimate $w_\theta^{(i)}$ of the weights. To implement (6), we efficiently compute these quantities by deriving closed-form expressions of both.

First, we can first easily compute the gradient and hessian of (5) w.r.t. s as,

$$\nabla_s L_{\text{CE}}(s;p) = \hat{p} - p \tag{7a}$$
$$\frac{\partial^2}{\partial s^2}L_{\text{CE}}(s;p) = \text{diag}(\hat{p}) - \hat{p}\hat{p}^{\text{T}} \tag{7b}$$

Here, $\hat{p}$ is the SoftMax of s as defined in (4d). By applying (7a) together with the chain rule, we can compute the gradient of (3) as,

$$\nabla L(w_\theta) = \sum_{j=1}^{n}\gamma_j\left[\frac{\partial s_j}{\partial w_\theta}\right]^{\text{T}}\nabla_s L_{\text{CE}}(s_j;p_j) + \lambda w_\theta = \sum_{j=1}^{n}\gamma_j\left[\tilde{z}_j *\right]^{\text{T}}(\hat{p}_j - p_j) + \lambda w_\theta \,. \tag{8}$$

We denote the transpose of the linear convolution operator $w \mapsto \tilde{z} * w$ as $[\tilde{z}*]^{\text{T}}$, which corresponds to the transpose of the Jacobian $\frac{\partial s_j}{\partial w_\theta} = [\tilde{z}*]$. By another differentiation w.r.t. w_θ, using (7b), the chain rule and the linearity of s_j in w_θ, we obtain the Hessian of (3) as

$$H(w_\theta) = \frac{\partial^2}{\partial w_\theta^2}L(w_\theta) = \sum_{j=1}^{n}\gamma_j\left[\frac{\partial s_j}{\partial w_\theta}\right]^{\text{T}}\left[\frac{\partial}{\partial^2 s}L_{\text{CE}}(s_j;p_j)\right]\frac{\partial s_j}{\partial w_\theta} + \lambda I = \sum_{j=1}^{n}\gamma_j\left[\tilde{z}_j *\right]^{\text{T}}\left(\text{diag}(\hat{p}_j) - \hat{p}_j\hat{p}_j^{\text{T}}\right)\left[\tilde{z}_j *\right] + \lambda I \,. \tag{9}$$

Here, I denotes the identity matrix. We further obtain a simple expression of the denominator of the step-length in (6) by evaluating the product,

$$\begin{aligned} g^{\mathrm{T}} H(w_\theta) g &= \sum_{j=1}^{n} \gamma_j (\tilde{z}_j * g)^{\mathrm{T}} (\operatorname{diag}(\hat{p}_j) - \hat{p}_j \hat{p}_j^{\mathrm{T}})(\tilde{z}_j * g) + \lambda g^{\mathrm{T}} g \\ &= \sum_{j=1}^{n} \gamma_j v_j^{\mathrm{T}} \big(\hat{p}_j \cdot (v_j - \hat{p}_j^{\mathrm{T}} v_j)\big) + \lambda g^{\mathrm{T}} g \,, \qquad v_j = \tilde{z}_j * g \,. \end{aligned} \tag{10}$$

Here, $\cdot$ denotes element-wise multiplication. We summarize the full optimization module in Algorithm 1.

Algorithm 1 Target Model Weight Prediction ψ_θ for Center Regression.

```
Input: Training (support) images $\{z_j\}_{j=1}^{n}$
Input: Corresponding bounding box annotations $\{\tilde{y}_j^{\text{bb}}\}_{j=1}^{n}$
1: $\tilde{z}_j = \phi_\theta(z_j), \quad j = 1, \dots, n$                              Extract features
2: $w_\theta^{(0)} \leftarrow \psi_\theta^{\text{init}}\left(\{(z_j, \tilde{y}_j^{\text{bb}})\}_{j=1}^{n}\right)$           Initialize weights [2]
3: for $i = 0, \dots, N_{\text{iter}} - 1$ do                                  Optimizer module loop
4:     $s_j \leftarrow \tilde{z}_j * w_\theta^{(i)}$                                          Using (4a)
5:     $\hat{p}_j \leftarrow \text{SoftMax}(s_j)$                                        Using (4d)
6:     $g \leftarrow \nabla L(w_\theta^{(i)})$                                            Using (8)
7:     $\alpha^{(i)} = \frac{g^{\mathrm{T}} g}{g^{\mathrm{T}} H(w_\theta^{(i)}) g}$                    Using (10) for the denominator
8:     $w_\theta^{(i+1)} \leftarrow w_\theta^{(i)} - \alpha^{(i)} g$                               Update weights
9: end for
```

3. Detailed Results

We provide more detailed results from the state-of-the-art comparison performed in the paper (in Section 5.3).

3.1. LaSOT

In addition to the success plot shown in the paper, we here provide the normalized precision plot over the LaSOT [8] test set, containing 280 videos. The normalized precision score NPr_D is computed as the percentage of frames where the normalized distance (relative to the target size) between the predicted and ground-truth target center location is less than a threshold D. NPr_D is plotted over a range of thresholds $D \in [0, 0.5]$. The trackers are ranked using the area under this curve, which is shown in the legend (see [8] for details). The normalized precision plot is shown in Figure 1. For the ResNet-18 and 50 versions of our PrDiMP, we report the average result over 5 runs. We compare with state-of-the-art trackers DiMP [2], ATOM [5], SiamRPN++ [12], MDNet [14], VITAL [15], and SiamFC [1]. Our approach outperforms previous state-of-the-art by a large margin. Compared to the ResNet-50 based SiamRPN++ [12] and DiMP-50 [2], our PrDiMP50 version achieves absolute gains of 11.9% and 3.8% respectively. As demonstrated by the plot, this gain is obtained by an improvement in both accuracy (small thresholds) and robustness (large thresholds).

3.2. GOT10k

The success plot over the 180 videos in the GOT10k [11] test set is shown in Figure 2a. It displays the overlap precision OP_T as a function of the IoU threshold T. Overlap precision OP_T itself is defined as the percentage of frames with a ground-truth IoU overlap larger than a threshold T. The final area-under-the-curve (AUC) score is shown in the legend. We compare our approach with trackers with available results: DiMP [2], ATOM [5], SiamFCv2 (CFNet) [16], SiamFC [1], GOTRUN [10], CCOT [7], ECO [4], and CF2 (HCF) [13]. Our PrDiMP versions outperform previous methods, in particular for large overlap thresholds. This demonstrates the superior accuracy of our probabilistic bounding box regression formulation.

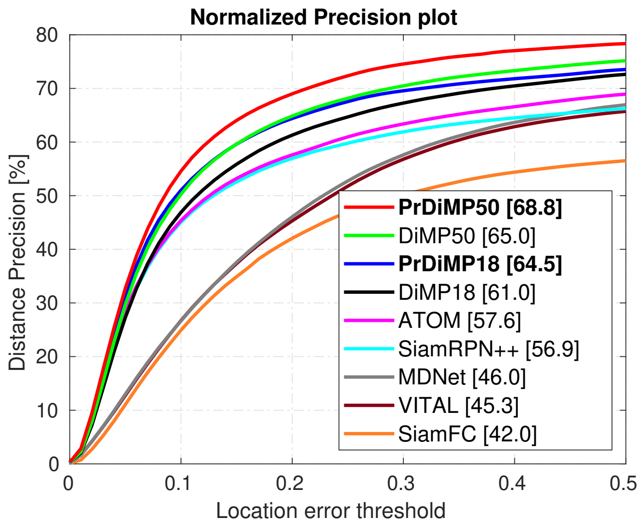
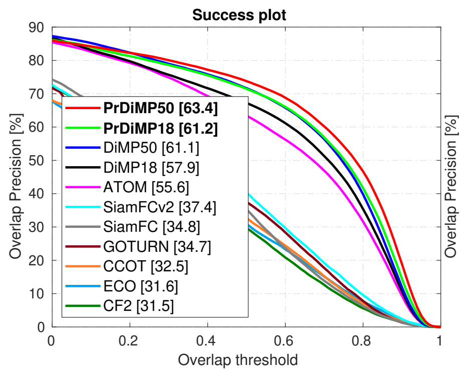
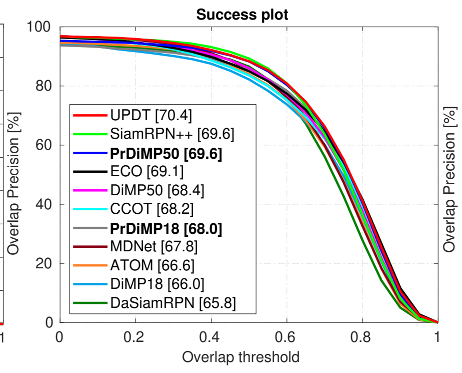


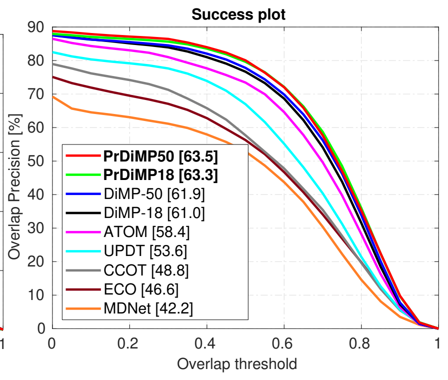
Figure 1. Normalized precision plot on the LaSOT dataset [8]. The average normalized precision is shown in the legend. Our approach outperforms previous trackers by a large margin.



(a) GOT10k [11]



(b) OTB-100 [17]



(c) NFS [9]

Figure 2. Success plots on the GOT10k [11] (a), OTB-100 [17] (b), and NFS [9] (c) datasets, showing the percentage of frames with a ground-truth IoU overlap larger than a threshold. The area-under-the-curve (AUC) metric for each tracker is shown in the legend. Our approach outperforms previous approaches by a significant margin on the challenging GOT10k and NFS datasets, while performing similarly to the top trackers on the largely saturated OTB-100 dataset.

3.3. OTB-100

We provide the success plot over the 100 videos in the OTB dataset [17] in Figure 2b. We compare with state-of-the-art trackers UPDT [3], SiamRPN++ [12], ECO [4], DiMP [2], CCOT [7], MDNet [14], ATOM [5], and DaSiamRPN [18]. Despite the highly saturated nature of this dataset, our tracker performs among the top methods, providing a significant gain over the baseline DiMP.

3.4. NFS

Figure 2c show the success plot over the 30 FPS version of the challenging NFS dataset [9]. We compare with top trackers with available results: DiMP [2], ATOM [5], UPDT [3], CCOT [7], ECO [4], and MDNet [14]. In this case, both our ResNet-18 and 50 versions achieve similar results, significantly outperforming the previous state-of-the-art DiMP-50.

4. Visualization of Predicted Distributions

In this section, we provide additional visualizations of the predicted probability distributions for target center and bounding box regression branches in our tracker. We visualize the output as in Figure 4 in the paper. Several example frames for two challenging sequences are shown in Figure 3. As during standard tracking, the predicted distribution $p(y^{\text{tc}}|x,\theta)$ for the target center regression (second column) is computed by applying the fully convolutional center regression branch on the search region centered at the previous target location.

To visualize the probability distribution $p(y^{\mathrm{bb}}|x,\theta)$ predicted by the bounding box regression branch, we evaluate the density in a grid. Note that the bounding box $y^{\mathrm{bb}} \in \mathbb{R}^4$ is 4-dimensional, and we therefore cannot visualize the full distribution as a 2-dimensional heatmap. We therefore plot two *slices* of this density, showing the variation in the bounding box location and size as follows. Our bounding box is parametrized as $y^{\mathrm{bb}} = (c_x/w_0, c_y/h_0, \log w, \log h)$, where (c_x, c_y) is the center position, (w,h) is the size (width and height), and (w_0, h_0) is a constant reference size. The latter is set to the current estimate of the target size. The distribution of the bounding box center (third column in Figure 3) is obtained by predicting the density value $p(y^{\mathrm{bb}}|x,\theta) \sim \exp(s_\theta^{\mathrm{bb}}(y^{\mathrm{bb}},x))$ in a dense grid of center coordinates (c_x, c_y), while keeping the size (w,h) constant at the current target estimate. To visualize the distribution over the bounding box size (fourth column), we conversely evaluate $p(y^{\mathrm{bb}}|x,\theta) \sim \exp(s_\theta^{\mathrm{bb}}(y^{\mathrm{bb}},x))$ in a dense grid of log-size coordinates $(\log w, \log h)$, while keeping the box centered at the current target estimate (c_x, c_y).

In Figure 3, the target center density is visualized in the range $\pm 2.5\sqrt{wh}$ relative to the previous target location. The bounding box center density is plotted within the range $c_x \pm w$ and $c_y \pm h$. The distribution over the bounding box size is plotted from $1/3$ to 3 times the estimated target size (w,h), *i.e.* $\pm \log 3$. Outputs for two challenging sequences are visualized in Figure 3. The left part shows a cat and its mirror image. Due to their similarity and proximity, it is in many frames difficult to predict the exact bounding box of the cat. In these cases, the predicted distribution captures this uncertainty. For example, in the fourth row, two clear modes are predicted, which correspond to aligning the box with the real cat or with the right edge of the reflection. Moreover, the last row shows a failure case, where the box briefly expands. However, note that the size probability distribution (right column) is highly uncertain. This information could thus be used to indicate low reliability of the estimated bounding box size.

The right part of Figure 3 depicts a challenging sequence with multiple distractors. The target center regression, which has a wider view of the scene, captures the presence of distractors in uncertain cases. In the last row, the tracker briefly fails by jumping to a distractor object. However, there is a strong secondary mode in the target center regression distribution that indicates the true target. Thus, the estimated distribution accurately captures the uncertainty in this ambiguous case. Moreover, in row 4-7, the target object is small with many nearby similar-looking objects. This makes bounding box regression extremely hard, even for a human. Our network can predict flexible distributions reflecting meaningful uncertainties for both bounding box position and size, when encountered with these difficulties.

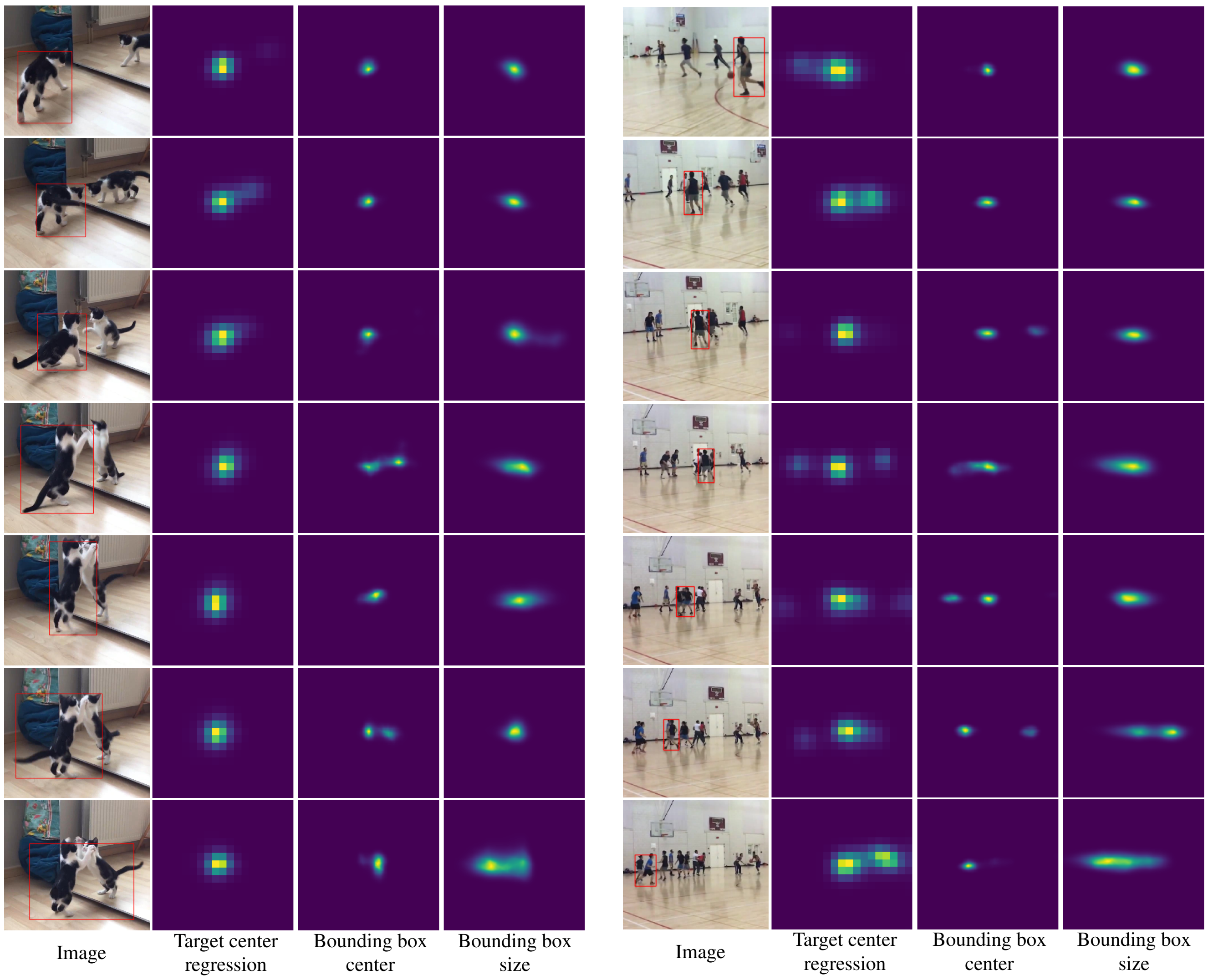


Figure 3. Visualization of the probability densities $p(y^{\text{tc}}|x,\theta)$ and $p(y^{\text{bb}}|x,\theta)$ predicted by the target center and bounding box regression branch respectively. We illustrate the output for example frames in two highly challenging sequences, where capturing uncertainty is important. Refer to the text for details.

References

[1] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV workshop*, 2016. 3

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *IEEE/CVF International Conference on Computer Vision, ICCV, Seoul, South Korea*, pages 6181–6190, 2019. 1, 2, 3, 4

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *15th European Conference on Computer Vision ECCV, Munich, Germany*, pages 493–509, 2018. 4

[4] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA*, pages 6931–6939, 2017. 3, 4

[5] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: accurate tracking by overlap maximization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Long Beach, CA, USA*, pages 4660–4669, 2019. 3, 4

[6] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *CVPR*, 2020. 1

[7] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *14th European Conference on Computer Vision ECCV, Amsterdam, The Netherlands*, pages 472–488, 2016. 3, 4

[8] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. *CoRR*, abs/1809.07845, 2018. 3, 4

[9] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017. 4

[10] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 3

[11] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 3, 4

[12] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 3, 4

[13] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 3

[14] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 3, 4

[15] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W. H. Lau, and Ming-Hsuan Yang. VITAL: visual tracking via adversarial learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Salt Lake City, UT, USA*, pages 8990–8999, 2018. 3

[16] Jack Valmadre, Luca Bertinetto, João F Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 3

[17] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 4

[18] Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 4