Supplementary file for Robust Visual Tracking Using Oblique Random Forests

Le Zhang
ADSC
Singapore 138632
zhang.le@adsc.com.sg

Jagannadan Varadarajan
ADSC
Singapore 138632
vjagan@adsc.com.sg

Ponnuthurai Nagaratnam Suganthan
NTU
Singapore 639798
epnsugan@ntu.edu.sg

Pierre Moulin
UIUC
IL 61820-5711 USA
moulin@ifp.uiuc.edu

Narendra Ahuja
UIUC
IL 61820-5711 USA
n-ahuja@illinois.edu

In this document we first provide the derivation for incremental learning of PSVM parameters, which is used to update every decision node in our online Obli-RaF. Next, we provide a detailed analysis of various parameters of the proposed tracker, and demonstrate the merits of our Obli-RaF. Finally, we provide more detailed results of our approach on the OTB-51, OTB-100 datasets and VOT2016 under various challenging scenarios.

1. Online update of PSVM parameters

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in I\!R^{N \times M}$ be an $N \times M$ matrix obtained by stacking N samples in X. The non-italicised $\top$ indicates a transpose operation. Let $\mathbf{Y} \in \{-1, +1\}^N$ be a vector obtained by stacking the labels of samples in X. Here, we drop the time index t in X^t, since the following formulation applies to any t. We define a diagonal matrix $\mathbf{D}$, whose diagonal entries $D_{i,i} = 1$ if $\mathbf{x}_i$ belongs to the positive class and -1 otherwise. The parameters of PSVM are obtained by solving the following problem:

$$\begin{aligned} &\min_{(\mathbf{w},b,\boldsymbol{\xi})} \frac{1}{2}||\boldsymbol{\xi}||^2 + \nu * \frac{1}{2}(\mathbf{w}^\top \mathbf{w} + b^2) \\ &\text{s.t. } \mathbf{D}(\mathbf{X}\mathbf{w} - b\mathbf{e}) + \boldsymbol{\xi} = \mathbf{e}. \end{aligned} \tag{1}$$

where $\boldsymbol{\xi}$ is the error vector and ν is the regularization parameter, $\mathbf{w}$ and b are the coefficient of the hyperplane, and $\mathbf{e}$ is a vector of all ones. The parameters of PSVM $\{\mathbf{w}, b\}$, can be computed in closed form using the following formulation:

$$[\mathbf{w}; b]^\top = (\nu I + \mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{D}\mathbf{e};\ \mathbf{H} = [\mathbf{X}, -\mathbf{e}] \tag{2}$$

Let $\mathbf{D}\mathbf{e} = \mathbf{Y}$ and suppose at time t, we have the solution $\boldsymbol{\beta}_t = [\mathbf{w_t}, b_t]^\top$. The parameters $\boldsymbol{\beta}_{t+1}$ corresponding to time $t+1$ can be calculated using $\boldsymbol{\beta}_t$ and newly available data without directly solving Eq. (2). The key step in updating the parameters is the calculation of $(\nu I + \mathbf{H}^\top \mathbf{H})^{-1}$. Suppose the features corresponding to data at time $t+1$ is $\mathbf{H}_{t+1}$, then the problem becomes

$$\underset{\boldsymbol{\beta}_{t+1}}{\text{minimize}} \left\| \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} \boldsymbol{\beta}_{t+1} - \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix} \right\|^2 + \nu ||\boldsymbol{\beta}_{t+1}||^2, \tag{3}$$

It is straightforward to observe that

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \left(\begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} + \nu I \right)^{-1} \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix} \\ &= \mathbf{S}_{t+1}^{-1} \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix}, \end{aligned} \tag{4}$$

where

$$\mathbf{S}_{t+1} = \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} + \nu I, \tag{5}$$

Then,

$$\mathbf{S}_{t+1} = \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix} + \nu I = \mathbf{S}_t + \mathbf{H}_{t+1}^\top \mathbf{H}_{t+1}, \tag{6}$$

And,

$$\begin{aligned} \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix} &= \mathbf{H}_t^\top \mathbf{Y}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}, \\ &= \mathbf{S}_t \mathbf{S}_t^{-1} \mathbf{H}_t^\top \mathbf{Y}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}, \\ &= \mathbf{S}_t \boldsymbol{\beta}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}, \\ &= (\mathbf{S}_{t+1} - \mathbf{H}_{t+1}^\top \mathbf{H}_{t+1}) \boldsymbol{\beta}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}, \\ &= \mathbf{S}_{t+1} \boldsymbol{\beta}_t - \mathbf{H}_{t+1}^\top \mathbf{H}_{t+1} \boldsymbol{\beta}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}. \end{aligned} \tag{7}$$

From Eqn. (4),

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \mathbf{S}_{t+1}^{-1} \begin{bmatrix} \mathbf{H}_t \\ \mathbf{H}_{t+1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{Y}_t \\ \mathbf{Y}_{t+1} \end{bmatrix} \\ &= \mathbf{S}_{t+1}^{-1} (\mathbf{S}_{t+1} \boldsymbol{\beta}_t - \mathbf{H}_{t+1}^\top \mathbf{H}_{t+1} \boldsymbol{\beta}_t + \mathbf{H}_{t+1}^\top \mathbf{Y}_{t+1}) \\ &= \boldsymbol{\beta}_t + \mathbf{S}_{t+1}^{-1} \mathbf{H}_{t+1}^\top (\mathbf{Y}_{t+1} - \mathbf{H}_{t+1} \boldsymbol{\beta}_t). \end{aligned} \tag{8}$$

From matrix inversion lemma, we have

$$\begin{aligned} \mathbf{S}_{t+1}^{-1} &= (\mathbf{S}_t + \mathbf{H}_{t+1}^\top \mathbf{H}_{t+1})^{-1} \\ &= \mathbf{S}_t^{-1} - \mathbf{S}_t^{-1} \mathbf{H}_{t+1}^\top (I + \mathbf{H}_{t+1} T_t^{-1} \mathbf{H}_{t+1}^\top)^{-1} \mathbf{H}_{t+1} \mathbf{S}_t^{-1}, \end{aligned} \tag{9}$$

To summarize, let $\boldsymbol{\varphi}_t = \mathbf{S}_t^{-1}$, the on-line update of the weight can be achieved by the following operation:

$$\begin{aligned} \boldsymbol{\varphi}_{t+1} &= \boldsymbol{\varphi}_t - \boldsymbol{\varphi}_t \mathbf{H}_{t+1}^\top (I + \mathbf{H}_{t+1} \boldsymbol{\varphi}_t \mathbf{H}_{t+1}^\top)^{-1} \mathbf{H}_{t+1} \boldsymbol{\varphi}_t \\ \boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t + \boldsymbol{\varphi}_{t+1} \mathbf{H}_{t+1}^\top (Y_{t+1} - \mathbf{H}_{t+1} \boldsymbol{\beta}_t). \end{aligned} \tag{10}$$

2. Complexity Analysis

Suppose we have N data samples in total and at each node $M_\mathrm{s} \ll M$ features are selected to search for the decision hyperplane. Without loss of generality, we assume each tree grows to its largest extent with a maximum depth Γ_depth, where the root node is at level 0 and the leaf nodes is at level Γ_depth. In order to simplify the analysis, we further assume that each tree is balanced, *i.e.*, each decision node will divide the data into two branches with equal size. In this case, $2^{\Gamma_\mathrm{depth}} = N$.

In a binary tree, there are 2^i nodes at any depth i with each node associated with $\frac{N}{2^i}$ samples. The orthogonal split algorithm firstly ranks each feature and finds a threshold to optimize the impurity criterion. Regardless of the computational time for calculating the impurity value, the computation complexity to find a "best-split" in an orthogonal decision tree takes time on the order of $M_\mathrm{s} \frac{N}{2^i} \log \frac{N}{2^i}$, where $\frac{N}{2^i} \log \frac{N}{2^i}$ is the cost of sorting each feature. From this, it is straightforward to show that the total complexity of a (balanced) orthogonal decision tree is:

$$C_\mathrm{Orth} = \sum_{i=0}^{L} 2^i (M_\mathrm{s} \frac{N}{2^i} \log \frac{N}{2^i}); \tag{11}$$

The overall complexity of an orthogonal decision tree, which is denoted by C_Orth, is around $\mathcal{O}(M_\mathrm{s} N \log^2 N)$. With complexity of PSVM being of order M_s^3, the overall complexity of an oblique decision tree is given by:

$$C_\mathrm{Obli} = (2N - 1) * M_\mathrm{s}^3, \tag{12}$$

which is about $\mathcal{O}(N * M_\mathrm{s}^3)$. But typically, only a few features are sampled at each internal node, resulting in $M_\mathrm{s} \ll \log N$ for the cases with a large number of training samples (such as visual tracking where data samples accumulate). Thus, we get $C_\mathrm{Obli} \ll C_\mathrm{Orth}$.

2.1. Obli-RaF improves ConNets

Our Obli-RaF+ConvNet tracker can be regarded as a combination of the simple Obli-RaF tracker and FCNT tracker [1], as mentioned in the main paper. These two models work collaboratively to improve each other. This allows high performance to be achieved since it becomes unlikely for compatible classifiers trained on independent features to agree on an incorrect label. Table 1 support our hypothesis by providing detailed results on the precision value of our method and the FCNT model when the threshold for the error is set to be 20 pixels, as suggested in [2].

Dataset	Ours	FCNT	Dataset	Ours	FCNT	Dataset	Ours	FCNT	Dataset	Ours	FCNT
carDark	100.00	100.00	doll	96.82	98.68	tiger2	73.97	80.27	girl2	9.33	7.20
car4	100.00	100.00	girl	99.80	91.20	biker	50.70	50.70	gym	72.88	94.78
david	99.79	93.84	walking2	100.00	100.00	bird1	36.52	34.31	Human2	64.18	65.69
david2	100.00	100.00	walking	100.00	100.00	bird2	92.93	94.95	Human3	79.27	98.00
sylvester	98.44	99.26	fleetface	74.68	75.53	blurBody	70.36	63.17	Human4	50.67	81.56
trellis	99.82	98.59	freeman1	100.00	100.00	blurCar1	94.47	93.13	Human5	51.47	49.37
fish	100.00	100.00	freeman3	100.00	91.74	blurCar2	86.32	92.82	Human6	29.17	30.68
mhyang	99.93	99.93	freeman4	99.65	99.65	blurCar3	98.04	98.88	Human7	98.80	78.40
soccer	52.81	43.62	david3	94.05	93.65	blurCar4	92.63	86.58	Human8	98.44	89.84
matrix	80.00	2.00	jumping	100.00	99.68	blurFace	99.59	99.59	Human9	83.28	85.90
ironman	64.46	6.63	carScale	82.54	73.81	blurOwl	87.64	94.77	jump	8.20	9.84
deer	98.59	100.00	skiing	100.00	100.00	board	85.67	31.95	kiteSurf	100.00	91.67
skating1	100.00	65.50	dog1	100.00	100.00	bolt2	89.42	96.25	man	100.00	100.00
shaking	98.36	92.05	suv	92.91	93.54	box	78.90	88.46	panda	100.00	100.00
singer1	99.15	92.88	motorRolling	92.07	90.85	car1	76.47	18.43	redTeam	100.00	100.00
singer2	87.98	80.87	mountainBike	99.56	100.00	car2	100.00	100.00	rubik	89.98	89.63
coke	89.35	90.03	lemming	85.85	43.26	car24	99.84	93.69	skater	98.75	95.00
bolt	87.71	96.57	liquor	68.58	87.02	clifBar	75.85	56.14	skater2	94.48	82.53
boy	100.00	100.00	woman	91.96	91.96	coupon	100.00	39.45	skating2-1	25.16	27.70
dudek	93.45	91.18	FaceOcc1	53.14	54.15	crowds	100.00	12.68	skating2-2	36.36	34.46
crossing	100.00	100.00	FaceOcc2	84.36	85.47	dancer	98.67	98.67	surfer	99.73	100.00
couple	100.00	100.00	basketball	86.21	85.79	dancer2	100.00	99.33	toy	97.79	93.73
football1	100.00	56.76	football	96.96	96.69	diving	11.63	12.09	trans	38.71	47.58
jogging-1	97.39	97.39	subway	100.00	99.43	dog	100.00	98.43	twinnings	100.00	100.00
jogging-2	96.09	95.44	tiger1	71.92	71.92	dragonBaby	92.04	86.73	vase	79.70	81.18

Table 1. Effect of combining Obli-RaF with FCNT.Each entry in the table stands for the mean avrage precision on each dataset.

3. Sensitivity analysis

The free parameters involved in our model are: i) depth of the tree Γ_{depth}, ii) maximum number of trees in the forest K, iii) number of features sampled for a node M_{s}, and iv) regularization parameter ν. We demonstrate the result with the simple Obli-RaF tracker. Similiar conclusion can be applied to the ConvNets based Obli-RaF tracker.

Tree depth. We find that in general Obli-RaF results in shallow decision trees. Therefore, setting Γ_{depth} to 3 is mostly sufficient. However, we set $\Gamma_{\text{depth}} = 100$ which does not impact the performance.

Number of features. The number of features M_{s} used in each node controls the diversity of each node in Obli-RaF. A small value for M_{s} results in less-correlated decision trees whereas large values reduce the diversity of the tree ensembles as most of the trees tend to agree on unseen test data. From our experiments, setting $M_{\text{s}} = \sqrt{M}$ gives the best performance.

Number of trees. Typically, increase in the ensemble size K provides more generalization ability of Obli-RaF. However, setting K to be too large increases the computational complexity. We experimented with a range of values for K as shown in Fig. 1 and found that $K = 100$ gives satisfactory results.

Effect of ν. The regularization parameter ν in Eqn.1 allows us to adjust the trade-off between complexity and training accuracy of PSVM. While it may be beneficial to tune this parameter for each node, we simply set it to 100. Generally speaking, a large value of ν will favor a hyperplane with less complexity while a small value results in over fitting. We experimented by varying ν between 10 and 1000 and the results are summarized in Fig. 2.

4. Advantages of Oblique Random forest

Obli-RaF learns shallow trees. Our tree induction method results in shallow trees and thus, more efficient. On average, the

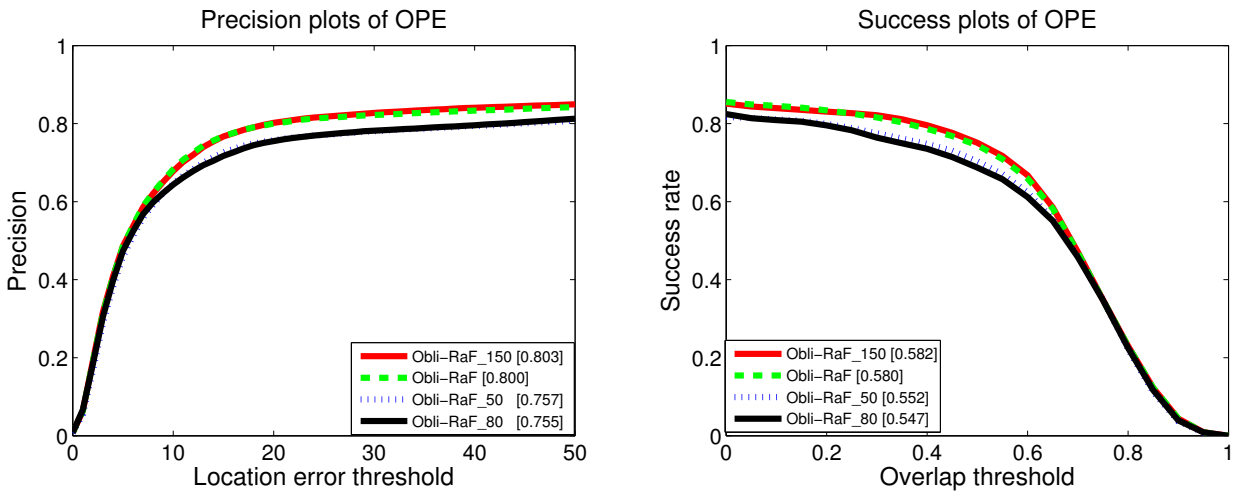


Figure 1. Average success plot and precision plot of Obli-RaF tracker with different values for K. Obli-RaF indicates setting $K = 100$. Obli-RaF_50, Obli-RaF_80 and Obli-RaF_150 stand for $K = 50, 80$ and 150 respectively.

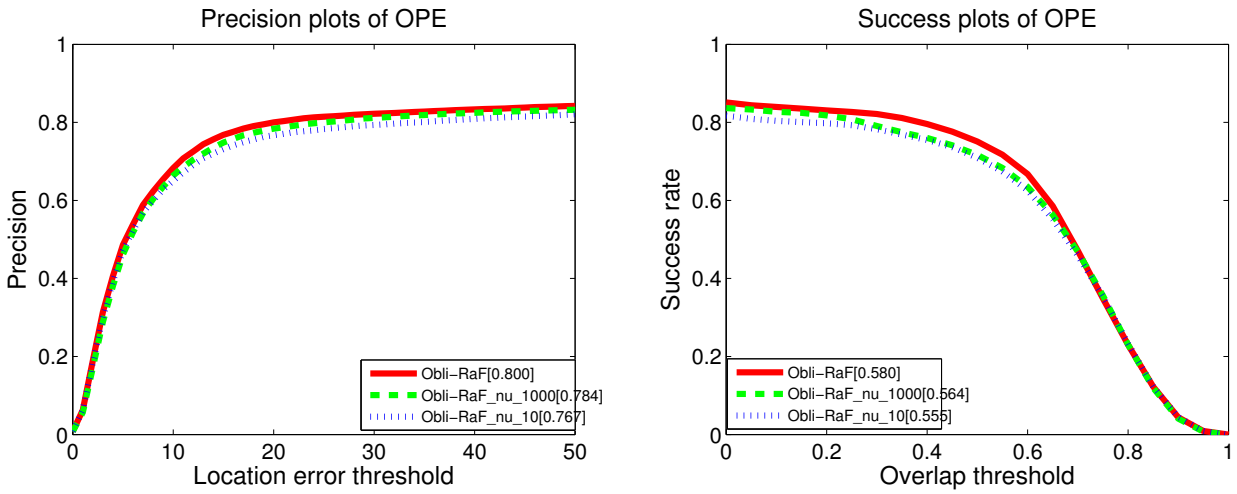


Figure 2. Average success plot and precision plot of RaF tracker with different ν parameter. Obli-RaF indicates setting $\nu = 100$. Obli-RaF_nu_10 and Obli-RaF_nu_1000 denotes $\nu = 10$ and 1000 respectively.

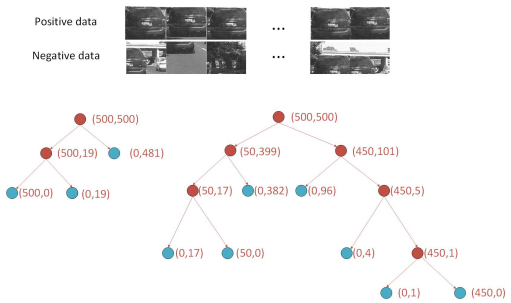


Figure 3. Visualization of different decision trees. (left) proposed oblique decision tree; (right) conventional decision tree. Leaf nodes are denoted in blue. The first and second number in the bracket denote the number of samples from the first and second class within that node respectively. Oblique decision trees result in shallow trees.

proposed simple Obli-RaF tracker runs 3 times faster than the orthogonal one. In order to show this, we crop 500 positive training samples and 500 negative samples from the first frame of the "car4" video sequence and train an oblique and an orthogonal decision tree. The learned trees are shown in Fig. 3. As we can see, the oblique decision tree (on the left)

classifies the samples correctly with a tree depth of three by using only two clustering operations, whereas the orthogonal decision tree (on the right) needs 5 levels and several orthogonal splits to achieve the same result.

Fig. 3 shows histograms of the tree depth of oblique and orthogonal RaF when applied on the OTB50 benchmark videos. From this, we clearly see that the average depth of an oblique trees is 1.51 whereas the average depth of orthogonal decision trees is 8.86. The maximum tree depth obtained for oblique and orthogonal decision trees are 7.16 and 22.86 respectively. We also calculated the ratio of tree depth obtained from orthogonal and oblique RaF for each frame in the benchmark and obtained an average depth ratio of 5.89, which denotes that on average the oblique decision tree is around 5.89 times shallower than the orthogonal one. From these results, we can conclude that our PSVM based Obli-RaF can better capture the geometric structure of the data samples and yield shallower trees.

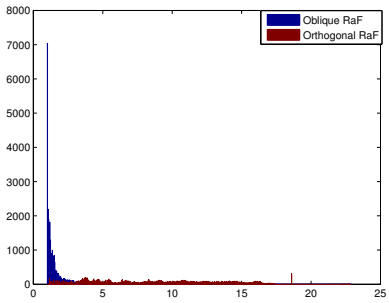
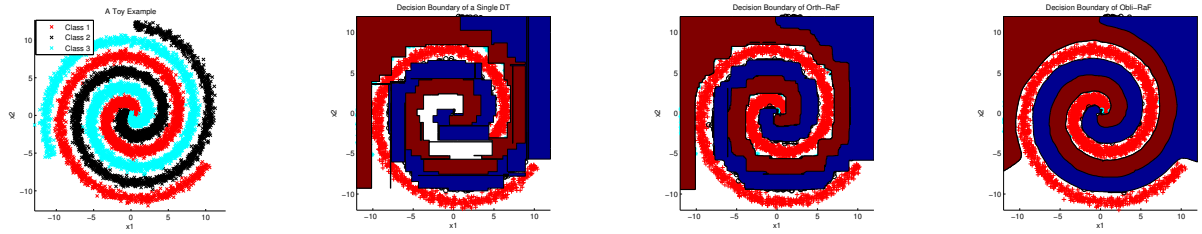


Figure 4. The histogram of the average tree depth of the forest.

Obli-RaF learns smoother decision boundaries. Here, we choose a toy example to demonstrate the the advantages of our method compared to conventional random forest. To this end, a three class artificial spirals dataset was generated as shown in the first part Fig 5(a). Fig. 5(b) demonstrates the hierarchical partitions from a single axis-parallel decision tree. We can clearly see that the spiral decision boundary is approximated by many "stage-like" decision hyperplanes. The decision boundary can be somehow smoothed by averaging many decision trees, which is demonstrated in Fig 5(c). The decision boundary in this case is obtained by a conventional orthogonal random forest (Orth-RaF). Fig 5(d) shows the decision boundary from the proposed Obli-RaF, which fits the data much better than the former two methods resulting in a "smoother" decision boundary.



(a) a (b) b (c) c (d) d

Figure 5. Decision boundary of the spirals dataset learned by different methods. (a) Ground truth, decision boundary from a (b) single decision tree, (c) Orthogonal RaF, (d) Oblique RaF.

5. Detailed results

In this section we provide detailed results of from our collaborative tracker (Obli-RaF tracker with ConvNets) under each challenging scenarios within OTB-51 and OTB-100 datasets. We present the OPE, SRE and TRE results for both OTB-51 and OTB-100 in Fig. 6, 7, 8, 9, 10, 11 respectively. The precision curves indicate that our method works remarkably well under many challenging scenarios. For instance, our method outperforms state-of-the-art methods under illumination variations, out-of-plane rotation, out-of-view, background cluttering and low resolution.

We also evaluate the proposed hybrid tracker on VOT2016 which consists of 62 datasets and 74 trackers . The AR ranking of all the methods are summarized in Table 2. Note we adopt the same parameter setting as done in OTB without any further tuning.

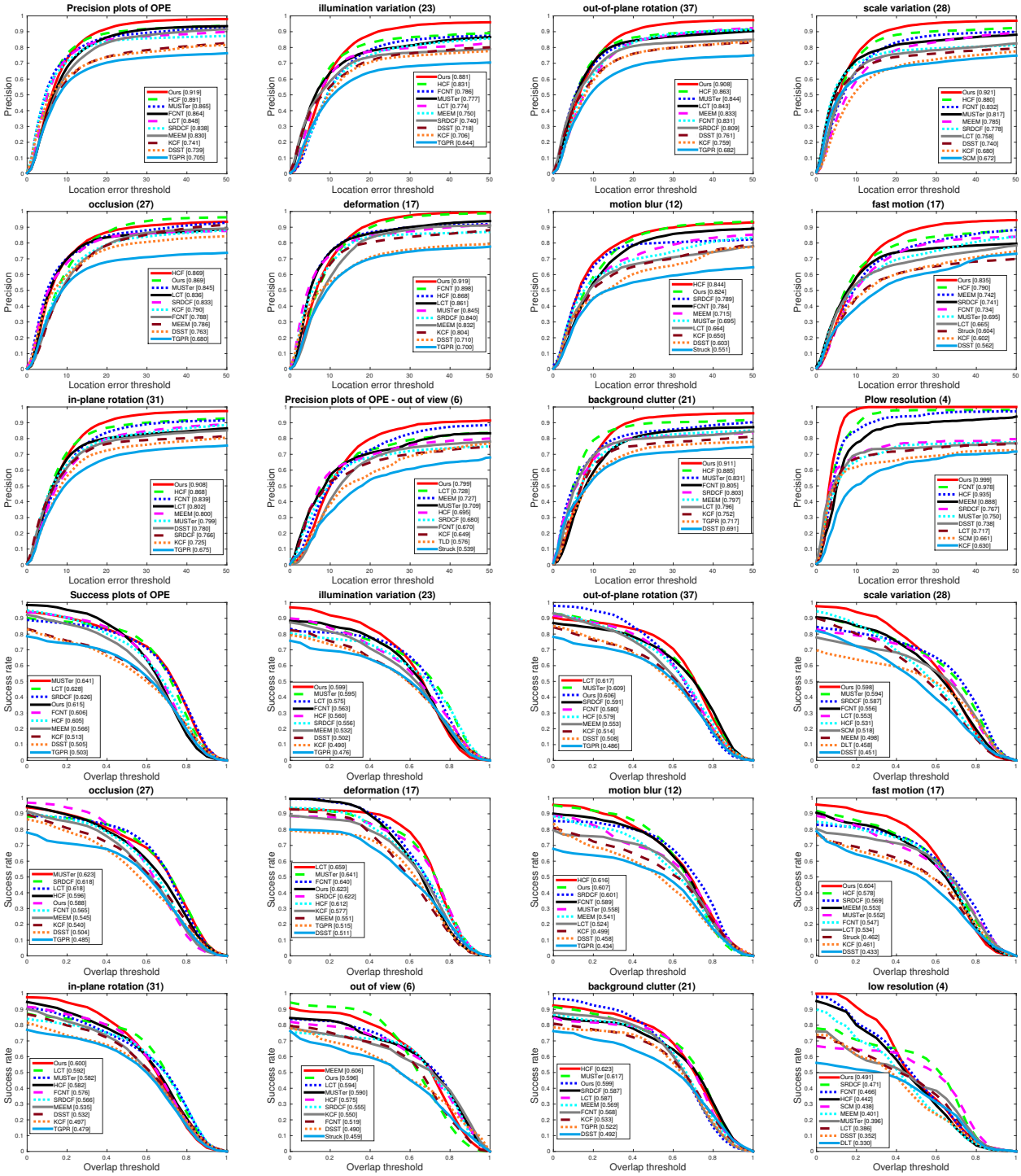


Figure 6. Detailed Results of OPE on OTB-51 benchmark.

References

[1] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision*, pages 3119–3127, 2015. 3

[2] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013. 3

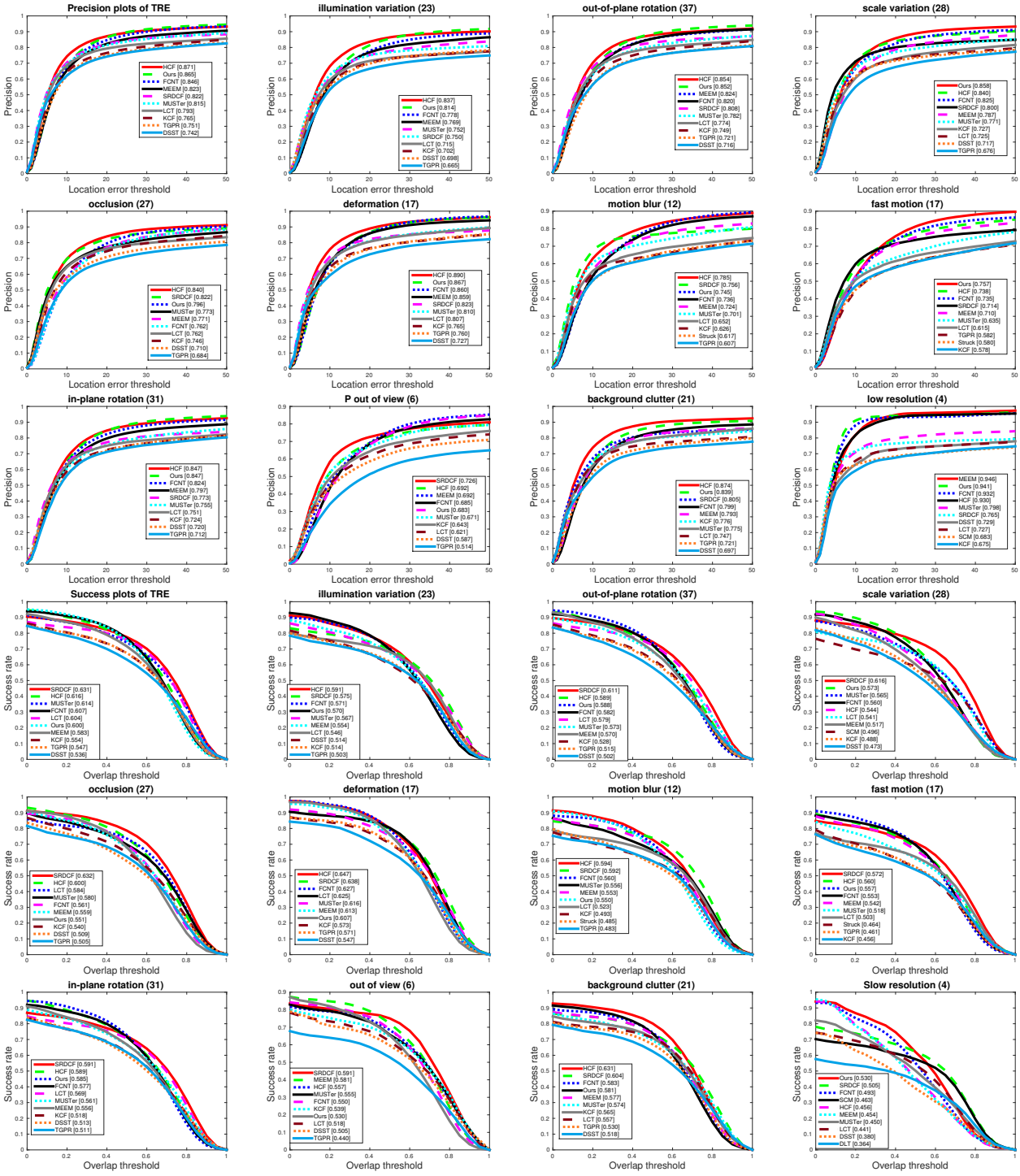


Figure 7. Detailed Results of TRE on OTB-51 benchmark.

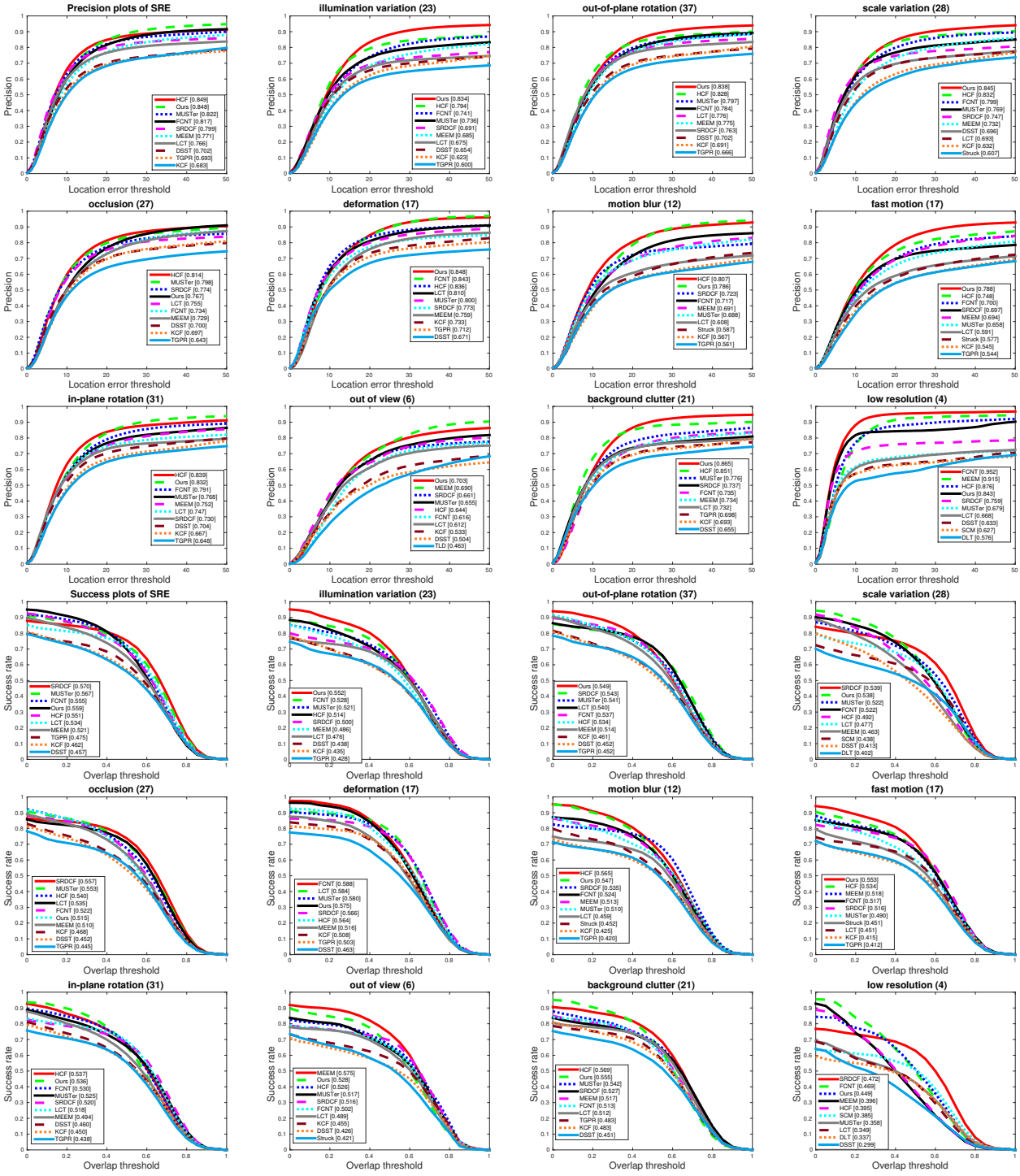


Figure 8. Detailed Results of SRE on OTB-51 benchmark.

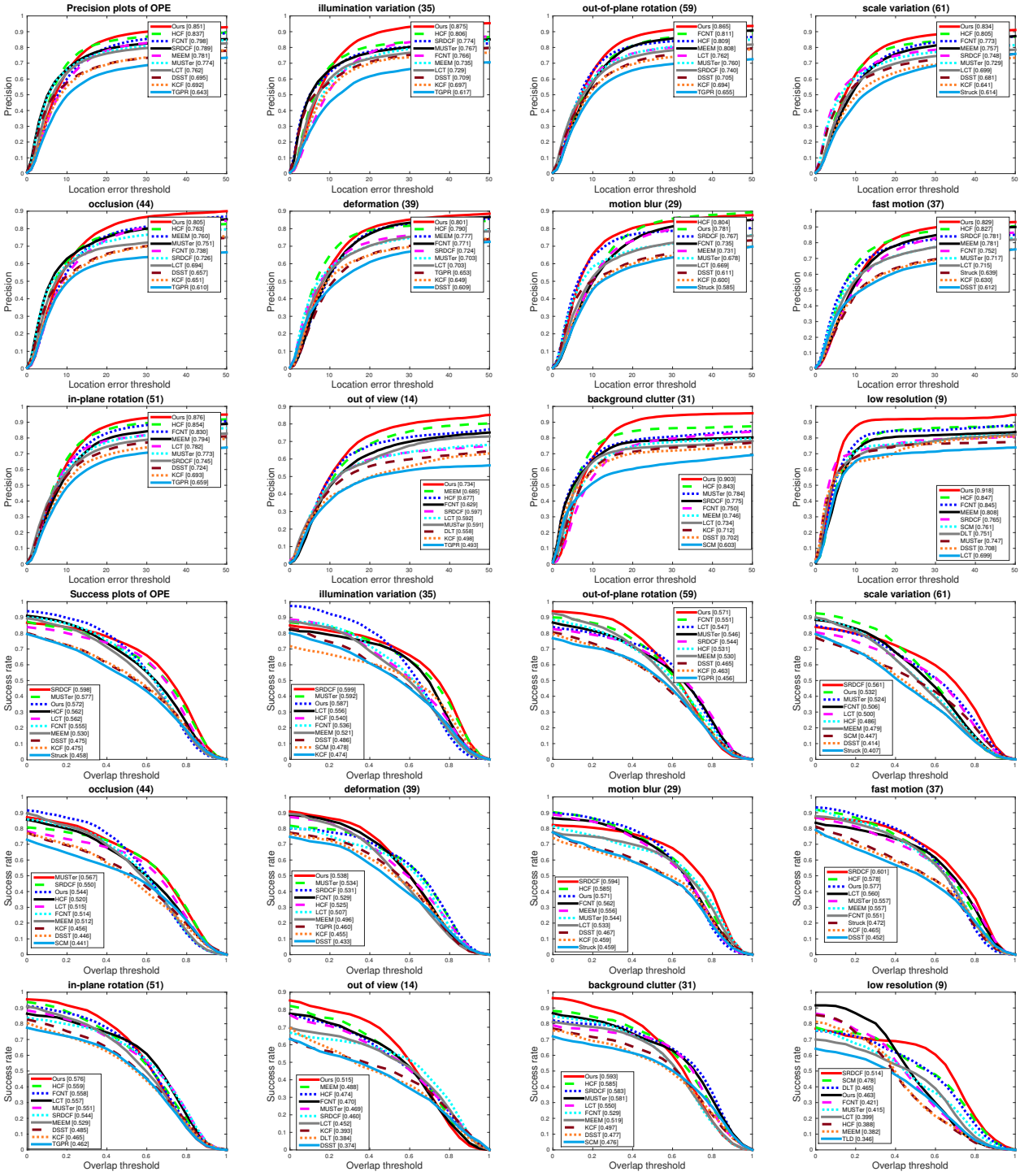


Figure 9. Detailed Results of OPE on OTB-100 benchmark.

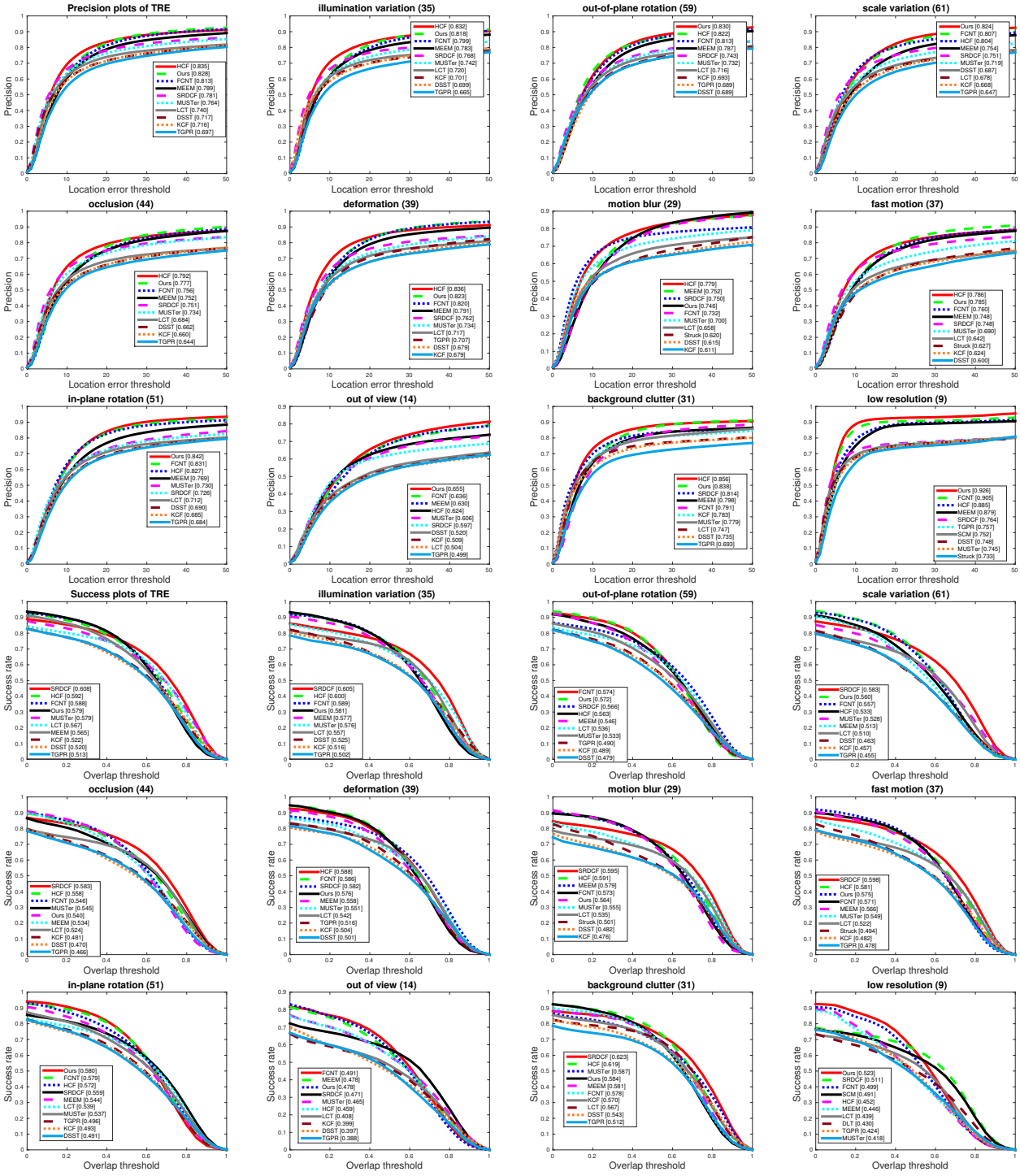


Figure 10. Detailed Results of TRE on OTB-100 benchmark.

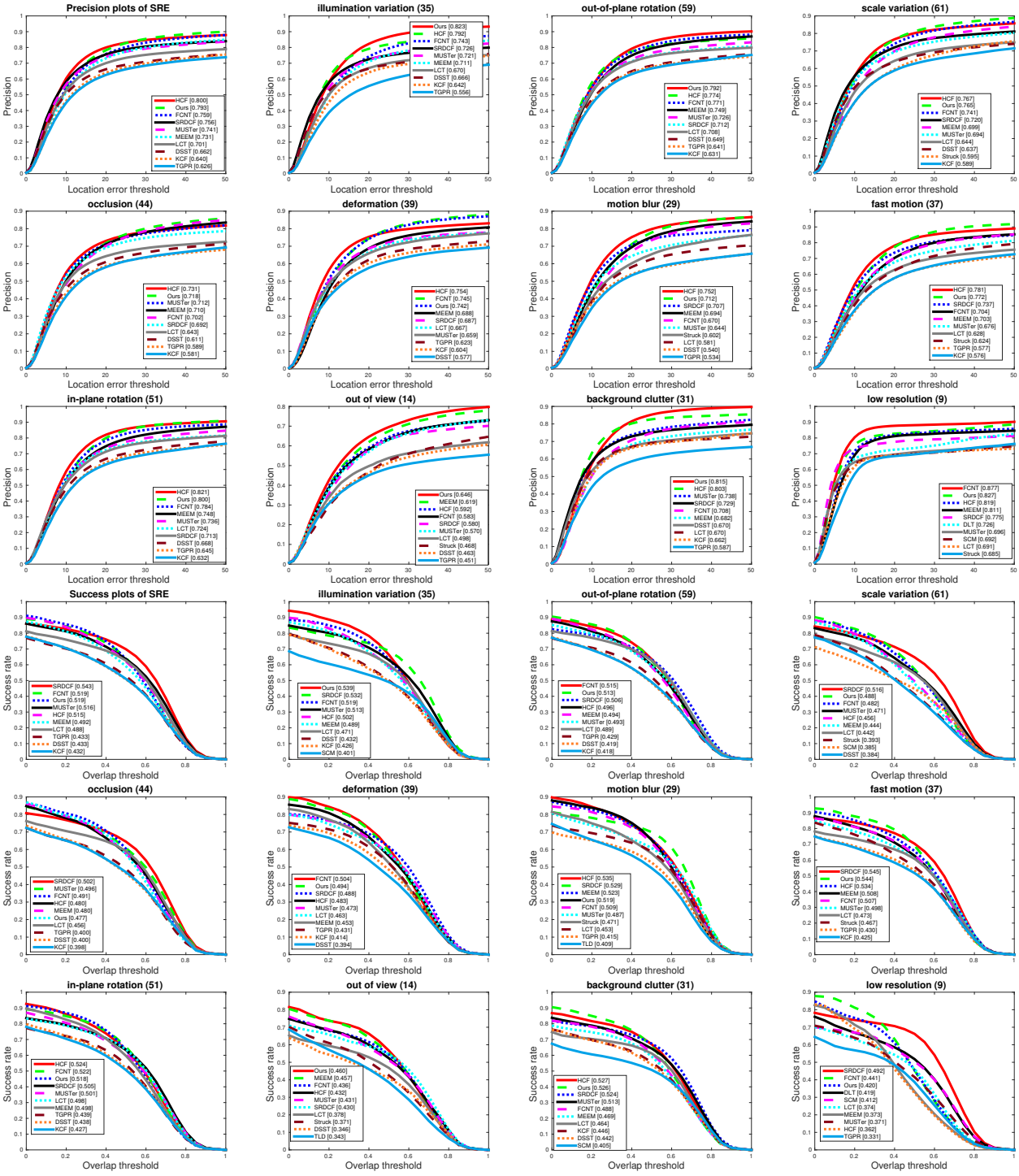


Figure 11. Detailed Results of SRE on OTB-100 benchmark.

Table 2. Overall AR ranking on VOt2016.

Method	Accuracy	Robustness
ACT	19.75	26.79
ANT	14.11	25.21
ART_DSST	13.89	27.25
ASMS	12.18	26.61
BDF	29.35	34.98
BST	33.11	20.09
CCCT	21.05	19.18
CCOT	8.11	10.00
CDTT	21.75	22.72
CMT	31.93	56.77
CTF	13.96	48.09
ColorKCF	11.72	22.68
DAT	15.84	21.35
DDC	5.84	14.86
DFST	15.54	30.44
DFT	18.47	41.53
DNT	10.09	15.56
DPT	12.61	21.60
DPTG	13.95	25.33
DSST2014	11.26	27.30
DeepSRDCF	10.47	16.02
EBT	17.39	7.96
FCF	3.96	16.58
FCT	29.09	34.23
FRT	31.07	48.54
FoT	29.21	35.96
GCF	8.25	19.72
GGTv2	7.77	21.89
HCF	12.11	14.02
HMMTxD	8.39	23.44
HT	26.67	35.96
IVT	24.74	42.95
KCF2014	13.30	22.39
KCF_SMXPC	6.84	17.53
LGT	25.33	29.89
LT_FLO	21.40	51.51

Method	Accuracy	Robustness
LoFT_Lite	40.68	51.33
MAD	12.84	18.65
MDNet_N	5.58	13.02
MIL	23.47	33.93
MLDF	13.44	8.07
MWCF	13.49	21.26 6
MatFlow	26.37	32.14
Matrioska	22.68	45.09
NSAMF	10.77	15.25
OEST	8.98	25.72
Ours	26.23	40.60
PKLTF	25.23	31.70
RFD_CF2	12.58	12.42
SAMF2014	9.51	19.82
SCT4	15.35	21.60
SHCT	5.39	15.74
SMPR	18.51	32.51
SODLT	10.46	21.42
SRBT	11.79	14.70
SRDCF	7.39	17.49
SSAT	3.61	12.40
SSKCF	7.98	15.65
STAPLEp	4.28	14.84
STC	31.61	40.72
STRUCK2014	21.35	38.72
SWCF	13.96	24.81
SiamAN	10.35	18.81
SiamRN	3.81	19.11
Staple	6.07	16.33
TCNN	5.49	12.72
TGPR	16.65	28.72
TricTRACK	19.09	25.70
deepMKCF	6.91	16.49
ncc	15.82	56.82
sKCF	14.28	34.81