Supplementary for ROAM: Recurrently Optimizing Tracking Model

Tianyu Yang[1,2] Pengfei Xu[3] Runbo Hu[3] Hua Chai[3] Antoni B. Chan[2]
[1] Tencent AI Lab [2] City University of Hong Kong [3] Didi Chuxing

1. Gradient Flow

Note that our recurrent model optimization process involves a gradient update step (see Eq. 9), which may need to compute the second derivative when minimizing the meta loss. Specifically, we show the computation graph of the offline training process of our framework in Fig. 1, where the gradients flow backwards along the solid lines and ignores the dashed lines. For the initial frame (Fig. 1 left), we aim to learn a conducive tracking network $\boldsymbol{\theta}^{(0)}$ and learning rates $\boldsymbol{\lambda}^{(0)}$ that can fast adapt to different videos with one gradient step, which is correlated closely with the tracking model $\boldsymbol{\theta}^{(1)}$. Therefore, we back-propagate through both the gradient update step and the gradient of the update loss (i.e. computing the second derivative). For the subsequent frames, our goal is to learn the online neural optimizer $\mathcal{O}$, which is independent of the tracking model $\boldsymbol{\theta}^{(t)}$ that is being optimized. Hence, we choose to ignore the gradient paths along the update loss gradient $\nabla_{\boldsymbol{\theta}^{(t-1)}}\ell^{(t-1)}$, and the neural optimizer inputs $\mathcal{I}^{(t-1)}$, which are composed of historical context vectors computed from the tracking model. The effect of this simplification is to focus the training more on improving the neural optimizer to reduce the loss, rather than on secondary items, such as the LSTM input or loss gradient.

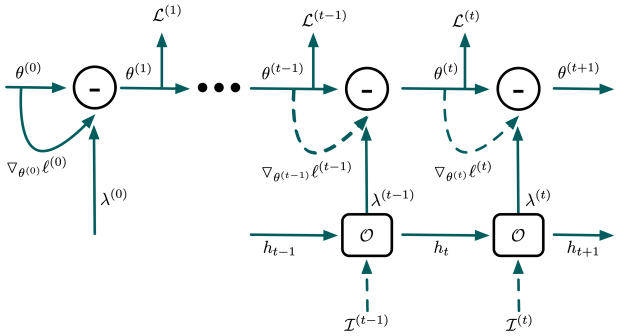

Figure 1. Computation graph of the recurrent neural optimizer.Dashed lines are ignored during backpropagation of the gradient.

2. Speed analysis

We further investigate the speed gain obtained by our recurrent neural optimizer and bounding box regression. To make a fair speed comparison, we use the same feature backbone (VGG16) and tracking model (two stacked convolutional layers) to build the tracker. We compare our ROAM and ROAM++ with trackers using traditional SGD for model updating under different number of gradient steps in Table 1. ROAM++ runs faster than ROAM because it does not need to compute multi-scale features. With similar run time, our ROAM achieves much better results compared with SGD-2, showing the effectiveness of our recurrent neural optimizer. Note that ROAM and SGD-2, which both uses two gradient steps, obtain similar speed demonstrating that our recurrent neural optimizer has negligible computational cost. Furthermore, using vanilla SGD requires more number of iterations to converge the model, thus leading to slower running speed.

	Speed (fps)	AUC
ROAM++	20.6	0.680
ROAM	13.4	0.681
SGD-2	13.7	0.593
SGD-4	12.5	0.597
SGD-8	10.6	0.621
SGD-16	8.0	0.645
SGD-32	5.5	0.644

Table 1. Speed comparison results on OTB2015 datasets. SGD-n means n gradient steps. SGD uses a learning rate of 7e-7. Note both ROAM and ROAM++ uses two gradient steps.

3. Update Loss and Meta Loss Comparison

We show the update loss $\ell^{(t)}$ and meta loss $\mathcal{L}^{(t)}$ comparison over time between ROAM and SGD after two gradient steps for all videos of OTB-2015 in Figure 2 and Figure 3 respectively. Our ROAM demonstrates better optimization ability on minimizing tracking loss.

4. Impact of Training Data.

We also train the variants of our ROAM++ and ROAM with ImageNet VID training data, which are denoted aas VID-ROAM++ and VID-ROAM, respectively. Table 2

shows the comparison results on VOT datasets. With only VID dataset for training, our ROAM++ still outperforms our preliminary tracker ROAM. Furthermore, both our VID-ROAM++ and VID-ROAM outperform the baseline MetaTracker [1] by a large margin.

	VOT-2016			VOT-2017		
	EAO	A	R	EAO	A	R
ROAM++	0.4414	0.5987	0.1743	0.3803	0.5433	0.1945
VID-ROAM++	0.4147	0.5886	0.1749	0.3330	0.5513	0.2366
ROAM	0.3842	0.5558	0.1833	0.3313	0.5048	0.2263
VID-ROAM	0.3568	0.5468	0.2091	0.2971	0.5098	0.2491
MetaTracker	0.317	0.519	-	-	-	-

Table 2. Comparison results on VOT-2016 and VOT-2017 datasets with different training datasets

5. More Ablations

We investigate the impact of removing scale smoothen and multi-scale search for ROAM tracker. We also present a variant of ROAM++ which does not resize tracking model to demonstrate the effectiveness of our resizable model. Table 3 shows the comparison results on OTB dataset. We can see that removing multi-scale search or filter resizing decrease the performance a lot while removing scale smooth has relative small affect.

Trackers	AUC
ROAM	0.681
ROAM++	0.680
ROAM w/o scalesmooth	0.667
ROAM w/o multiscale	0.590
ROAM++ w/o filter resizing	0.553

Table 3. Performance comparison of different variants on OTB dataset

6. Bounding Box Parametrization

We follow the way of bounding box parameterization in [2] when computing smooth L1 loss in (7). Let (x, y, w, h) be the center coordinate, width and height of the ground truth box, and (x^*, y^*, w^*, h^*) be the predicted box. The parametrization of the bounding boxes are:

$$B = (t_x, t_y, t_w, t_h) = (\frac{x-x_a}{a_w}, \frac{y-y_a}{a_h}, \log\frac{w}{a_h}, \log\frac{h}{a_h}),$$
$$B^* = (t_x^*, t_y^*, t_w^*, t_h^*) = (\frac{x^*-x_a}{a_w}, \frac{y^*-y_a}{a_h}, \log\frac{w^*}{a_w}, \log\frac{h^*}{a_h})$$

where the anchor size (a_w, a_h) is defined in (6) and this anchor is used on each spatial location on the feature map. (x_a, y_a) is the center of each anchor. $\mathcal{R}(F; \boldsymbol{\theta}_{reg})$ (in Eq.7) corresponds to B^*, which is the parametrized predicted bbox, and B is the parameterized ground truth bbox. After getting $\mathcal{R}(F; \boldsymbol{\theta}_{reg})$, we compute the predicted object box (x^*, y^*, w^*, h^*).

References

[1] Eunbyung Park and Alexander C. Berg. Meta-Tracker: Fast and Robust Online Adaptation for Visual Object Trackers. In *ECCV*, 2018. 2

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NeurIPS*, 2015. 2

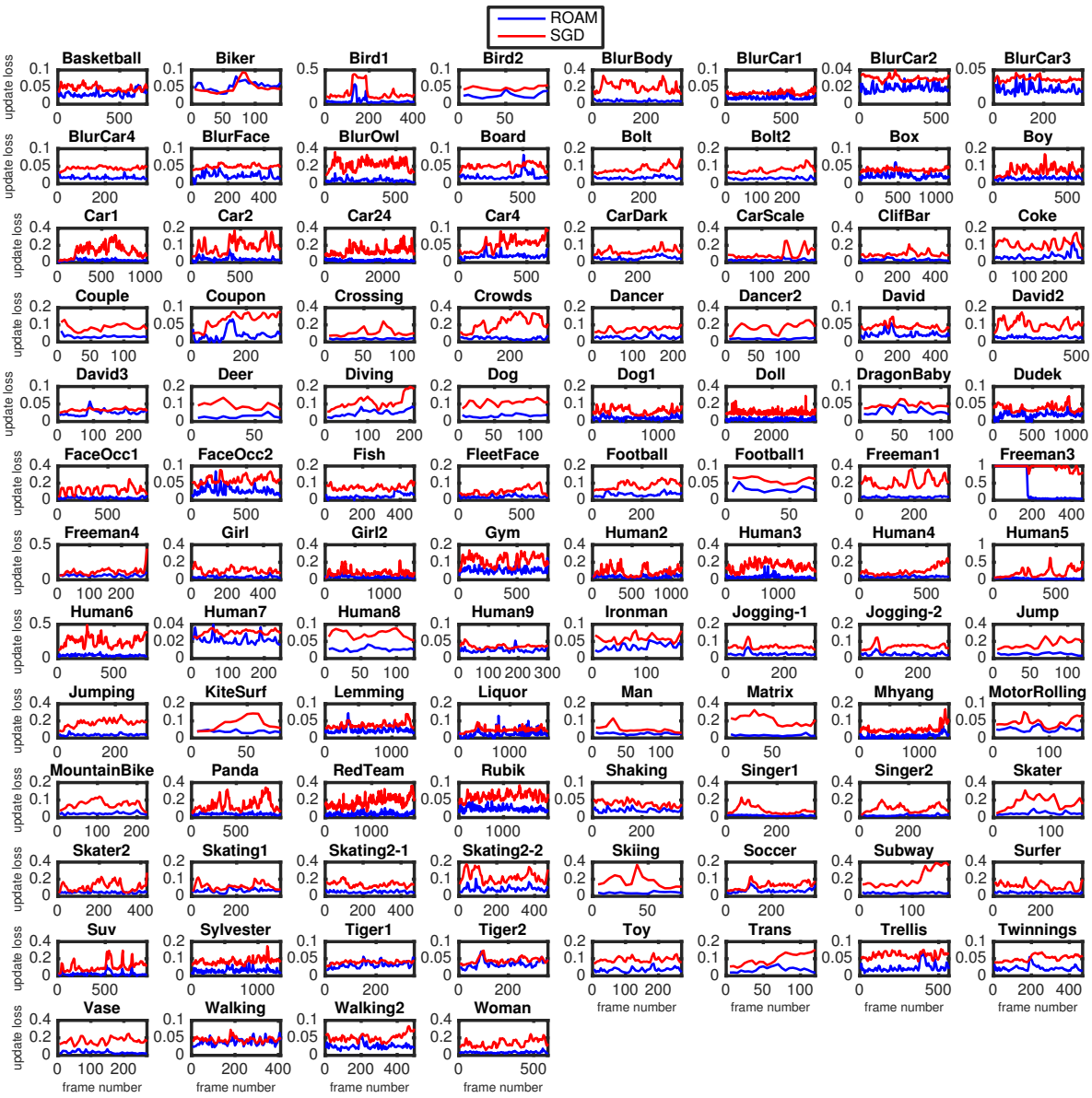

Figure 2. Update loss comparison between ROAM and SGD.

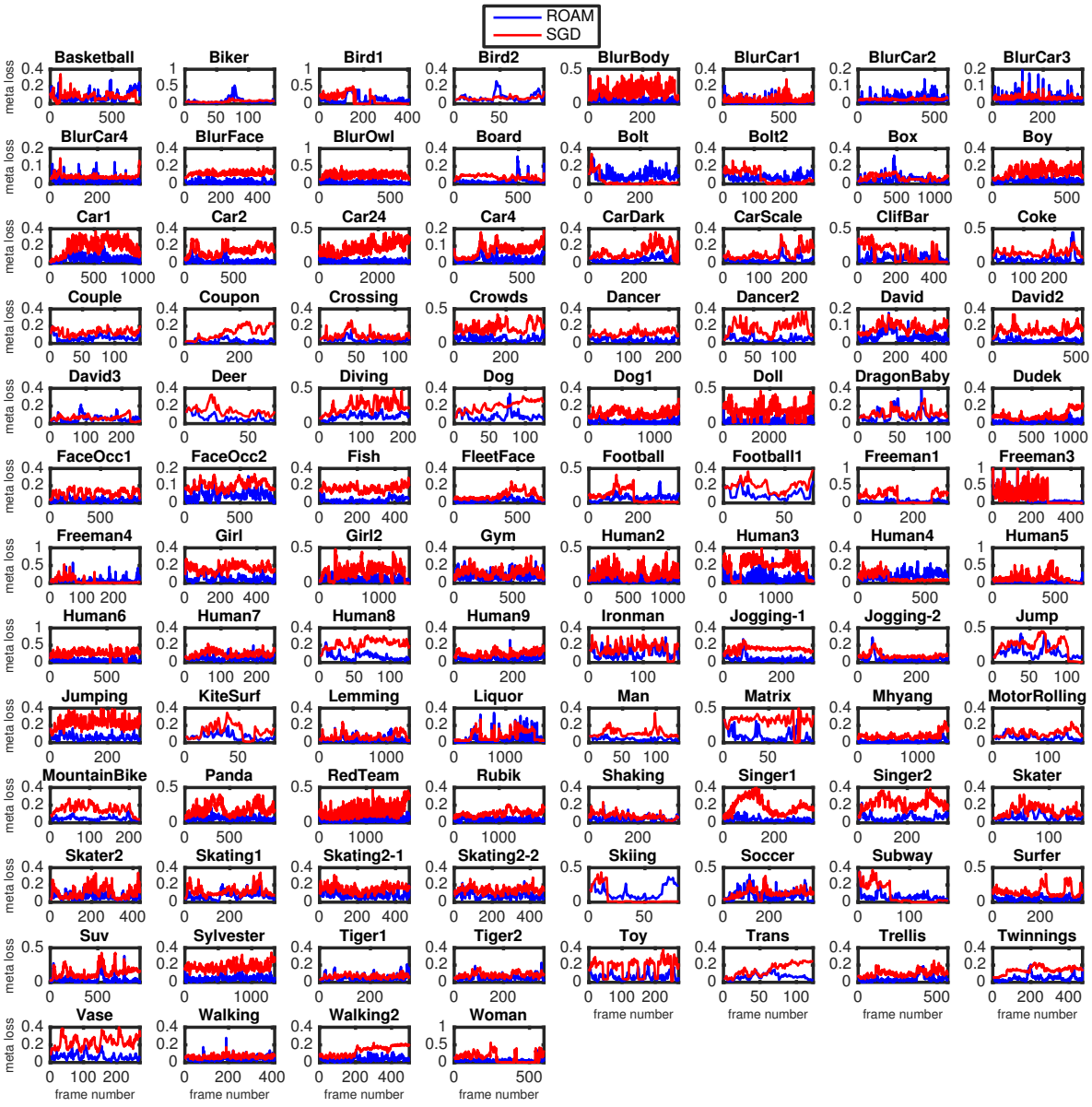

Figure 3. Meta loss comparison between ROAM and SGD.