DiffusionTrack: Point Set Diffusion Model for Visual Object Tracking
Supplementary Materials

Fei Xie [†], Zhongdao Wang [‡], Chao Ma [†*]
[†] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[‡] Huawei Noah's Ark Lab
jaffe031@sjtu.edu.cn, wangzhongdao@huawei.com, chaoma@sjtu.edu.cn

1. Overview of Appendix

In this supplementary material, we first provide more implementation details in Sec. 2, including the model, training and inference details. Then we present more benchmark results in Sec. 3. Further, we give more discussions and the limitation of our tracking framework in Sec. 4. Finally, we provide more visualization results in Sec. 5.

2. Additional Implementation Details

In this section, we first provide more details about the diffusion model. Then, we depict the details of our proposed encoder. We also illustrate the multi-stage diffusion decoder used in the limitation subsection of the main text. Finally, we present more details about training, inference, and computation efficiency, which supplement our main text.

2.1. More about diffusion model

We provide a detailed review of the formulation of diffusion models, following the notion of [7, 11, 22]. Starting from a data distribution $\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0)$, we define a forward Markovian noising process q which produces data samples $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, ..., $\boldsymbol{z}_T$ by gradually adding Gaussian noise at each timestep t. In particular, the added noise is scheduled by the variance $\beta_t \in (0, 1)$:

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) := \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \tag{1}$$

$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}) \tag{2}$$

As noted by Ho *et al.* [11], we can directly sample data $\boldsymbol{z}_t$ at an arbitrary timestep t without the need of applying q repeatedly:

$$q(\boldsymbol{z}_t|\boldsymbol{z}_0) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0, (1-\bar{\alpha}_t)\boldsymbol{I}) \tag{3}$$

$$:= \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0 + \epsilon\sqrt{1-\bar{\alpha}_t}, \epsilon \in \mathcal{N}(0, \boldsymbol{I}) \tag{4}$$

where $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$ and $\alpha_t := 1 - \beta_t$. Then, we could use $\bar{\alpha}_t$ instead of β_t to define the noise schedule.

Based on Bayes' theorem, it is found that the posterior $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0)$ is a Gaussian distribution as well:

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_{t-1}; \tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0), \tilde{\beta}_t\mathbf{I}) \tag{5}$$

where

$$\tilde{\mu}_t(\boldsymbol{z}_t, \boldsymbol{z}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{z}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{z}_t \tag{6}$$

and

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{7}$$

are the mean and variance of this Gaussian distribution.

We could get a sample from $q(\boldsymbol{z}_0)$ by first sampling from $q(\boldsymbol{z}_T)$ and running the reversing steps $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ until $\boldsymbol{z}_0$. Besides, the distribution of $q(\boldsymbol{z}_T)$ is nearly an isotropic Gaussian distribution with a sufficiently large T and reasonable schedule of β_t ($\beta_t \rightarrow 0$), which making it trivial to sample $\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I})$. Moreover, since calculating $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ exactly should depend on the entire data distribution, we could approximate $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ using a neural network, which is optimized to predict a mean μ_θ and a diagonal covariance matrix Σ_θ:

$$p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) := \mathcal{N}(\boldsymbol{z}_{t-1}; \mu_\theta(\boldsymbol{z}_t, t), \Sigma_\theta(\boldsymbol{z}_t, t)) \tag{8}$$

Instead of directly parameterizing $\mu_\theta(\boldsymbol{z}_t, t)$, Ho *et al.* [11] found learning a network $f_\theta(\boldsymbol{z}_t, t)$ to predict the ϵ or $\boldsymbol{z}_0$ from Equation (4) worked best. We choose to predict $\boldsymbol{z}_0$ in this work.

2.2. Multi-stage DiffusionTrack

We introduce the model details of the limitation experiment in main text. As shown in Fig. 1, our six diffusion layers are not all connected to the last layer of encoder as that of

[*]Corresponding author.

Model	Hier.	Attn.	Layer	Channel	Param.	Speed
SwinT	✓	Win.	(2,2,18,2)	(96,128,256, 512)	88M	106FPS
ViT	✗	GL.	(0,0,12,0)	(0,0,768,0)	86M	95FPS
HViT	✓	GL.	(2,2,20,0)	(128,256,512,0)	64M	89FPS

Table 1. Architectural variants of three vision transformers: SwinT [18], ViT [8] and our HViT. Hier. denotes hierachical structure. Attn denotes the attention computation. Win. and GL. denote window attention and vanilla global attention.

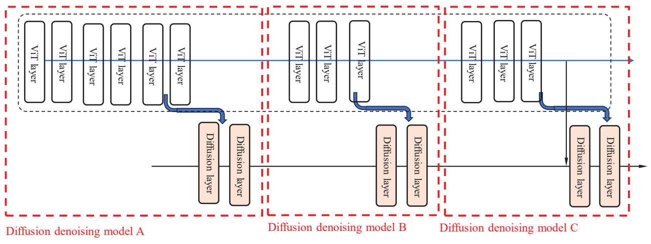

ViT layer
Diffusion layer
Diffusion denoising model A
Diffusion denoising model B
Diffusion denoising model C

Figure 1. The detailed pipeline of our multi-stage DiffusionTrack. We insert three diffusion based decoder into the shallow, mid and deep stages of the encoder. The output of each decoder can be sent into the next decoder for collaborative prediction.

DiffusionTracking in main text. Every two diffusion layers are assigned to different stages of encoder. For the shallow layer, we choose the 6^{th} encoder layer while 9^{th} and 12^{th} are chosen as the mid and deep layer. Thus, we construct three diffusion decoders, while each of them has two diffusion layers. On shallow, mid and deep decoders, the output target estimations of each decoder are sent to the next decoder. Thus, on the easy video cases, by applying early exit strategy, the forward inference of encoder can be suspend in early stage which greatly raise the tracking speed. Moreover, it also can exploit hierarchical features of encoder. We will do future research on achieving a better trade of efficiency and effectiveness and finding optimal encoder-decoder structure.

2.3. Training details

We introduce more training details which is a supplement to the main text. we train our model on the training splits of COCO [17], TrackingNet [21], LaSOT [9], and GOT-10k [12] datasets. For the video datasets (TrackingNet, LaSOT, and GOT-10k), we directly sample the image pairs from one video sequence to collect training samples. For COCO detection datasets, we apply some transformations to the original image to generate image pairs. Common data augmentation (such as translation and brightness jitter) is applied to enlarge the training set. The sizes of the search region patch and template patch are 256×256 and 256×256, respectively. The backbone parameters are initialized with Masked Image Modelling pre-training weights from DropTrack [25], and other parameters of our model are initialized with Xavier init [10]. We train the model with AdamW [19], setting the backbone's learning rate to 1e-5, other parameters' learning rate to 1e-4, and weight decay to 1e-4. We train the network on 4 NVIDIA A800 with 64 sample pairs on each GPU (batch size 112) for 300 epochs with 60,000 sample pairs per epoch.

2.4. Inference details

We introduce more inference details which is a supplement to the main text. The renewal strategy, voting strategy and early exit scheme are described in details.

Renewal strategy. During online inference, our diffusion decoder generates output layer by layer. Each layer has N target estimations G_N and its corresponding confidence scores C_N. Each layer refines the target estimation based on the previous estimations, except for the first layer which refines the initialized random estimations. Thus, we propose a renewal strategy inspired by [3] which make each two diffusion layers can collaborate with each other. To be more specific, for N predictions G^l_N in l^{th} diffusion layer, we only preserve the $N1$ predictions, whose corresponding confidence scores C_{N1} are larger than a pre-defined value 0.7. Then the $N1$ high-scoring predictions are concatenated with $N - N1$ estimations initialized from a random noise:

$$G^{l+1}_N = \textbf{Concat}(G^l_{N1}, \text{Random}(N - N1)). \tag{9}$$

The high-scoring part can prevent repeated predictions and contribute to the voting process. The random part is for searching the target again from the background. Then, the G^{l+1}_N estimations are used for the prediction of next diffusion layer.

Voting strategy. During online inference, each Diffusion Layer (DL) output independent target estimations G_N and its corresponding confidence scores C_N. Except for the initialized target estimations, we have total L groups of predictions for total L diffusion layers. For l^{th} diffusion layer, we have:

$$G^l_N, C^l_N = DL^l(z, x, t, G^N_{l-1}), \tag{10}$$

then, we collect the total L groups of predictions:

$$\mathcal{P}_L = \{G^l_N, C^l_N\}^L_{l=1}. \tag{11}$$

The total $L \times N$ groups of point set vote for the final prediction result. First, we remove target estimations with low-scoring values, which are most likely to locate at background. Thus, we have high-scoring target estimations. However, there are quantities of repeated predictions among them. Therefor, inspired by NMS processing [23] in detectors, we filter out repeated prediction results $\{b\}^n_{i=1}$ and count the number of repeated predictions num:

$$\{(b, \text{num})\}^n_{i=1} = \textbf{NMSwithCOUNT}\{G^l_N\}^L_{l=1}. \tag{12}$$

Model	Encoder	Decoder	Param.	Flops.	Speed
E1	ViT	DL×1	12.4M	28.1G	45FPS
E2	ViT	DL×2	25.1M	36.2G	42FPS
E3	ViT	DL×3	37.4M	41.8G	39FPS
E4	ViT	DL×6	74.8M	53.2G	30FPS

Table 2. Computation efficiency analysis on DiffusionTrack. Params. indicates the model parameters of decoler.

Method	OTB100 [26]	NFS [14]	UAV123 [20]
DiffusionTrack-b256 (1)	70.3	65.8	68.7
STARK [27]	68.5	66.2	68.2
TransT [4]	69.4	65.7	69.1
TrDiMP [24]	69.6	66.5	67.5
DiMP [2]	68.4	61.8	64.3
Ocean [29]	68.6	49.4	57.4
ATOM [6]	66.9	58.3	63.2
ECO [5]	69.1	52.2	53.5
RT-MDNet [13]	67.8	43.3	52.8
SiamFC[1]	57.5	37.7	46.8

Table 3. Comparison with state-of-the-art methods on additional benchmarks in AUC score: OTB100 [26], NFS [14] and UAV123 [20].

After NMS processing, The one with the largest number of overlapped predictions are chosen as the final prediction result. Thus, our voting strategy ensure that all target estimations participate in the visual tracking process.

Early exit scheme. As mentioned above, we have total total $L \times N$ groups of target estimations and corresponding scoring values. Each N predictions are generated layer by layer. Thus, we apply a simple threshold strategy to just whether the current predictions are confident enough to find the ground truth target. If the maximum confidence value of current predictions is larger than a pre-defined threshold value, we conclude that the target is find. Thus, we suspend the model inference process at current layer. It can facilitate the model inference speed as presented in the main text.

2.5. Computation efficiency analysis

We analyze the computation efficiency of DiffusionTrack. The model parameter and computation Flops are presentaed in Tab. 2.

3. Additional Benchmark Results

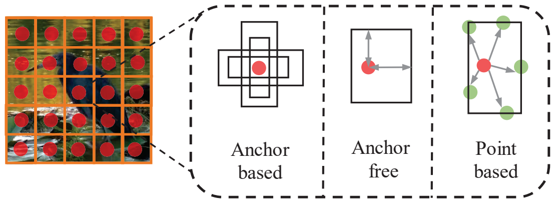
In this section, we present more experimental results on additional VOT benchmarks which is a supplement to the main text. It includes UAV123 [20], OTB100 [26], VOT2020 [16] and NFS [14] datasets. As shown in Tab 3 and Tab. 4, our DiffusionTrack exhibits promising results on these VOT benchmarks.

	UPDT [30]	DiMP [24]	TransT [4]	TrDiMP [27]	Stark [27]	DiffusionTrack-b256
EAO ↑	27.8	27.4	29.3	30.0	30.3	31.4

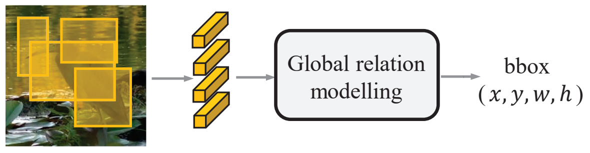
Table 4. Comparison on the VOT2020 [15] test set.

Model	Encoder	Decoder	Representation	AO
C1	ViT	Diff	Bounding box	73.8
C2	ViT	Diff	Point set (2)	74.4
C3	ViT	Diff	Point set (3)	74.2

Table 5. Ablation on point set and bounding box representation. AO is evaluated on GOT10k [12] benchmark. Other settings are kept as same.



(a) Traditional per-pixel manner prediction



(b2 Our global interaction manner

Figure 2. Comparisons among anchor-based or anchor-free prediction head and our diffusion-based head. DiffusionTrack gets rid of the pre-defined priors on all grids of image (a). The prediction network in DiffusionTrack are guided by point groups (b)

4. Additional Discussions

In this section, we present more discussions to analyze our proposed DiffusionTrack framework, including our decoder design and employing diffusion model to other trackers.

4.1. On point set representation

As shown in Tab. 5, the point set representation settings whose point numbers are two and three, achieves better AO results than axis-aligned bounding box representation. It shows the effectiveness of our proposed point set presentation on diffusion models.

4.2. On point set number

In main text, the point number n = 2 has almost the same performance (74.4 vs.74.2) as n = 3, and further increases

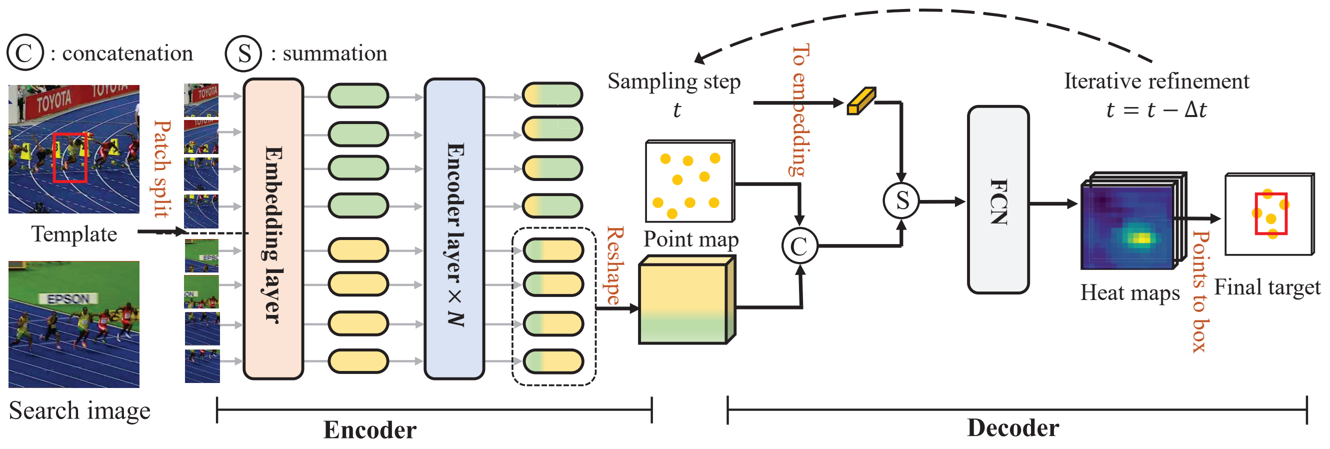


Figure 3. Employing diffusion-style training and inference to a representation anchor-free vision transformer based tracker.

in the number of points (n = 3, 5, 7) leads to degraded performance. We infer that the degeneration may be caused by a lack of effective supervision for an excessive amount of points during training. We choose to set point set number as two for efficiency. Setting point set number as three also can achieve sota results on multiple tracking benchmarks. In Fig.5 and Fig. 6, we visualize both the tracking results of n = 2 and n = 3 settings. We can see the point sets are gradually focused on the target and filter out the background clutters progressively.

4.3. On less diffusion layers.

In main text, we find that less diffusion layers can have a significant faster speed but lower performance. It may attribute to the prediction with refinement-style which more layers to refine can ease the training burden. We will investigate on simplified diffusion layer design to achieve the same results using less layers or iterations.

4.4. On Diffusion decoder and other head designs

We illustrate the difference between our diffusion decoder and previous popular anchor-based or anchor free prediction head designs. DiffusionTrack gets rid of the pre-defined priors on all grids of image. The prediction network in DiffusionTrack are guided by point-set groups. As shown in Fig. 2, in contrast to the dominating per-pixel prediction manner which uses pre-defined anchor box on all grids of image feature, e.g. RPN based or anchor-free based, DiffusionTrack generates object proposals from the whole image and model the instance-level relationship globally to enhance the discriminative power.

Model	Encoder	Decoder	Rep.	AO
D1	ViT	Diffusion layer	Point set	74.1
D2	ViT	Conv. head	Bounding box	63.2

Table 6. Ablation on apply diffusion-style training and inference on other tracking pipelines. AO is evaluated on GOT10k [12] benchmark. Other settings are kept as same.

4.5. Employing diffusion-style training to other tracking pipeline

In this sub-section, we present our exploratory experiment to apply diffusion-style training and inference to a representative transformer based tracker. We select OStrack [28] as our base tracker, which is also an encoder-decoder structure. OStrack adopts ViT [8] as encoder and a fully convolutional network consisting of four stacked Conv-BN-Relu layers. A center-based prediction which is borrowed from CenterNet [31], is adopted to predict the target. We concatenate a heat map and target-aware search features and send it to the prediction head, which is shown in Fig. 3. The diffusion step index is also embedded into a feature vector to add into the concatenated feature map. Then, we follow the training and inference of diffusion model, which are presented in the main text, to formulate a new diffusion based tracking model. As shown in Tab. 6, traditional tracking pipeline with simple Conv. head cannot effectively model the diffusion process, which has 9.9% AO performance drop than our proposed DiffusionTrack with diffusion based decoder. The element-wise summation of step index directly to the feature map may not effectively embeds the diffusion step information. The heat map also cannot effectively model the process from random state to a high certainty. Therefore, it fully validates the effectiveness of our diffusion layer

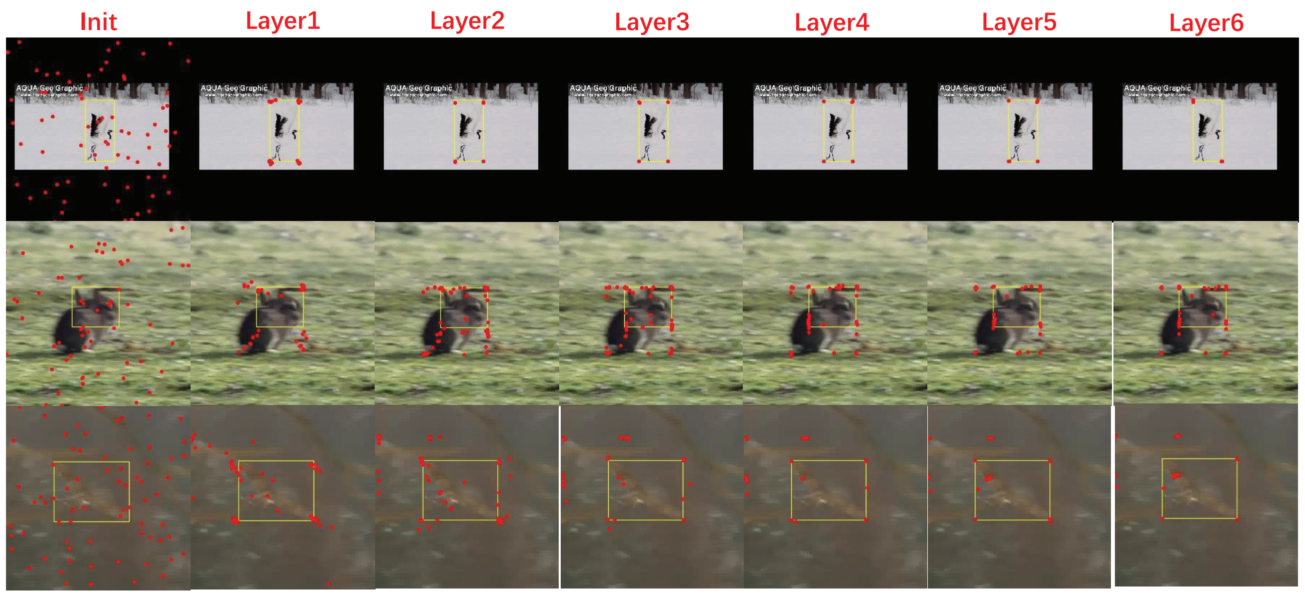


Figure 4. Visualization results of initial target estimations and refined results from six stacked diffusion layers. The number of points is set to be 2. Yellow bounding box is the final predict result.

based decoder.

5. Additional Visualization Results

In this section, we present more visualization results which is a supplement to the main text. In Fig. 4, we give more examples on visualization results of initial target estimations and refined results from six stacked diffusion layers. Furthermore, in Fig. 5 and Fig. 6, we give the visualization results of the case whose the number of points is set to 3. We all can see the point sets are gradually focused on the target and filter out the background clutters progressively. In the final, we present the attribute analysis of DiffusionTrack on LaSOT [9] benchmark in Fig 7 and Fig 8.

References

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

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 3

[3] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *CVPR*, 2023. 2

[4] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021. 3

[5] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017. 3

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, 2019. 3

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NIPS*, 2021. 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 4

[9] 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. In *CVPR*, 2019. 2, 5, 9

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *ICAIS*. ICAIS, 2010. 2

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NIPS*, 2020. 1

[12] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*. 2, 3, 4

[13] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time MDNet. In *ECCV*, 2018. 3

[14] 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*, pages 1125–1134, 2017. 3

[15] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, Luka Čehovin Zajc, Alan Lukežič, Ondrej Drbohlav, et al. The eighth visual object tracking vot2020 challenge results. In *ECCV*. Springer, 2020. 3

[16] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Martin Danelljan, Alan Lukezic, Ondrej Drbohlav, Linbo He, Yushan Zhang, Song Yan, Jinyu Yang, Gustavo Fernandez, and et al. The eighth visual object tracking vot2020 challenge results, 2020. 3

[17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 2

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv*, 2017. 2

[20] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016. 3

[21] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 2

[22] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 1

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[24] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, 2021. 3

[25] Qiangqiang Wu, Tianyu Yang, Ziquan Liu, Baoyuan Wu, Ying Shan, and Antoni B. Chan. Dropmae: Masked autoencoders with spatial-attention dropout for tracking tasks. In *CVPR*, 2023. 2

[26] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 3

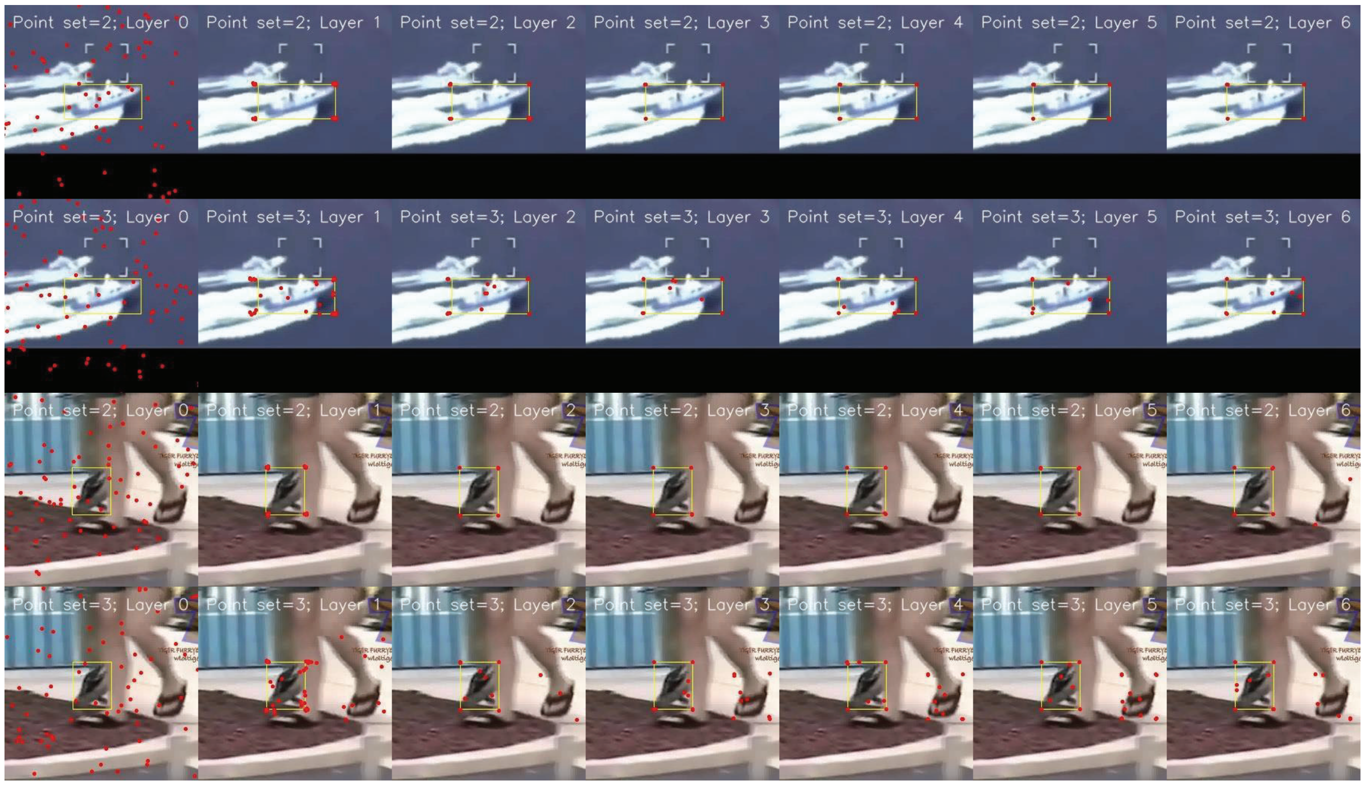
[27] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *ICCV*, 2021. 3

[28] Botao Ye, Hong Chang, Bingpeng Ma, and Shiguang Shan. Joint feature learning and relation modeling for tracking: A one-stream framework. *arXiv*, 2022. 4

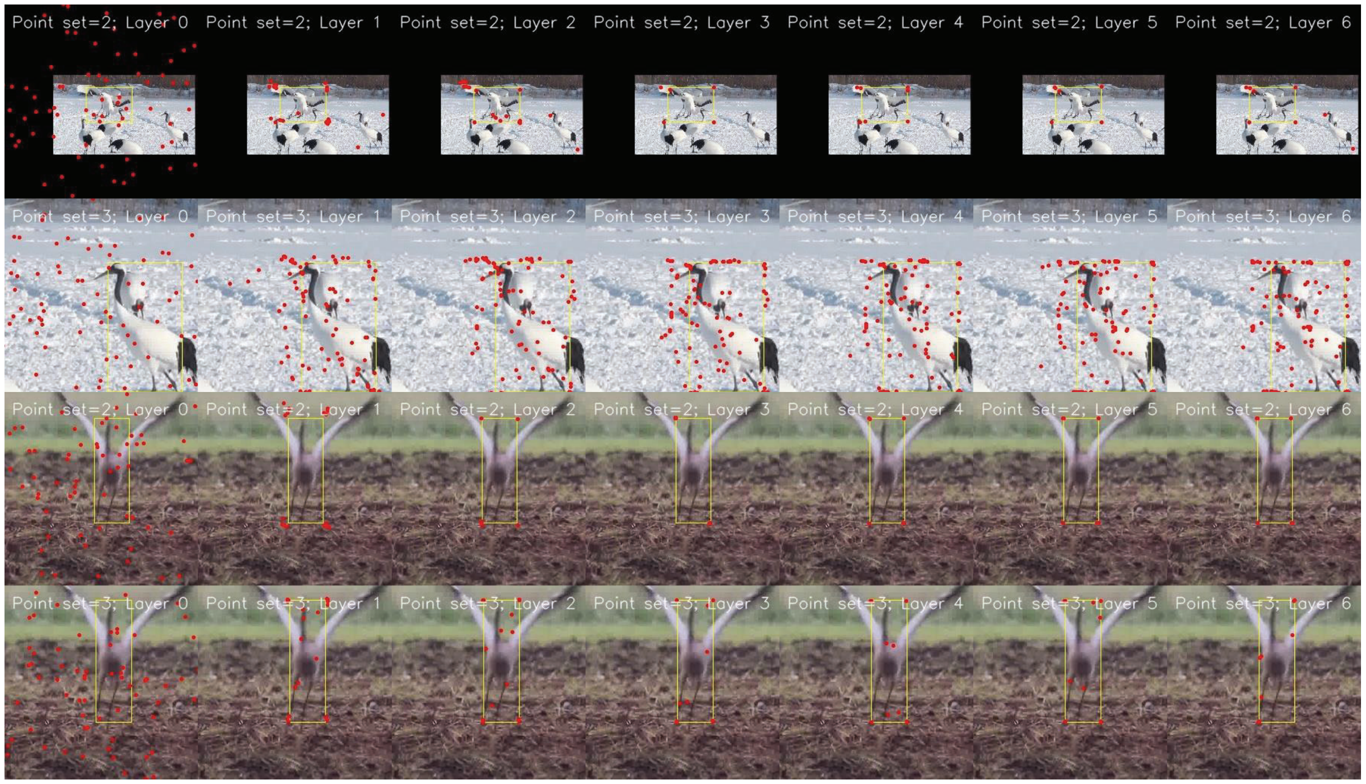
[29] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *ECCV*, 2020. 3

[30] Zhipeng Zhang, Yihao Liu, Xiao Wang, Bing Li, and Weiming Hu. Learn to match: Automatic matching network design for visual tracking. In *ICCV*, 2021. 3

[31] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv*, 2019. 4

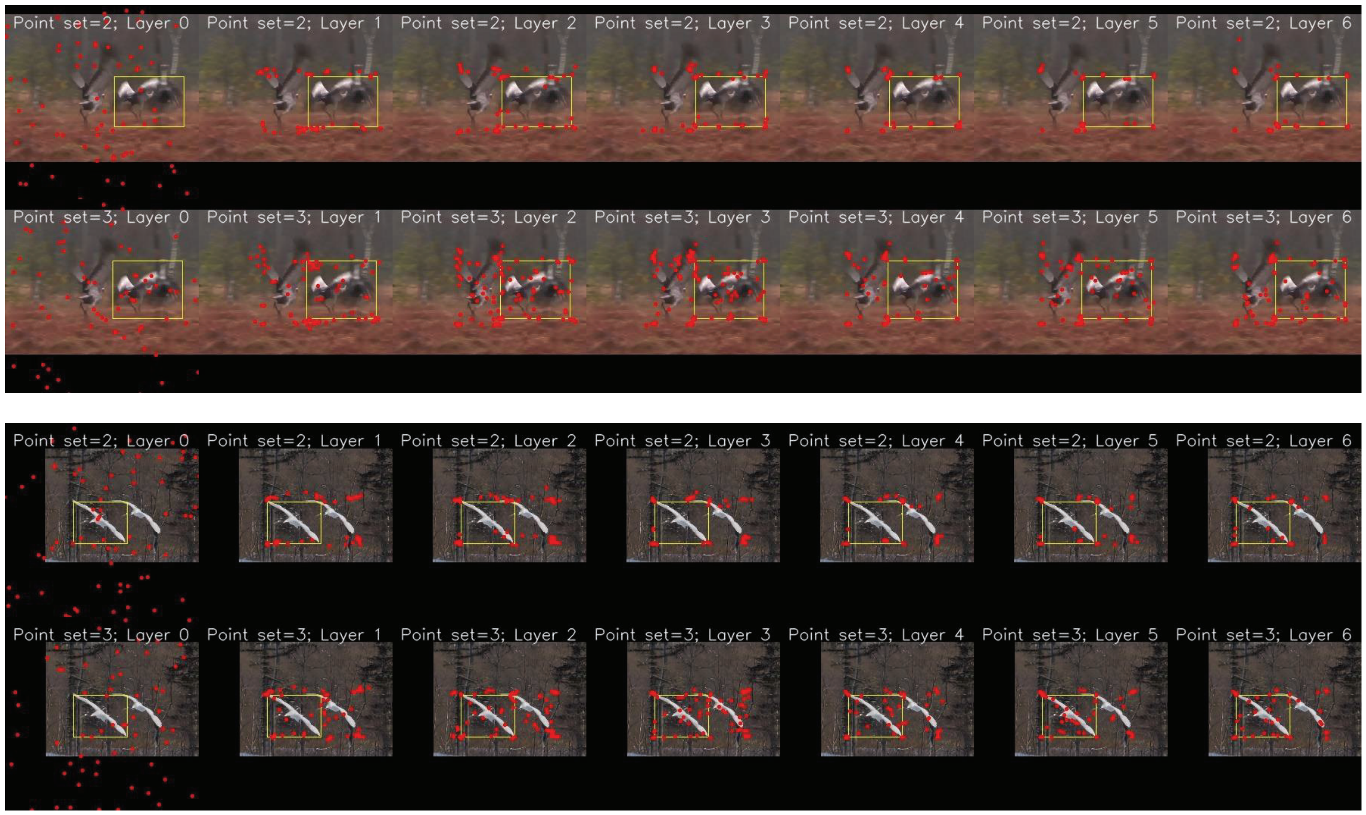


(a)

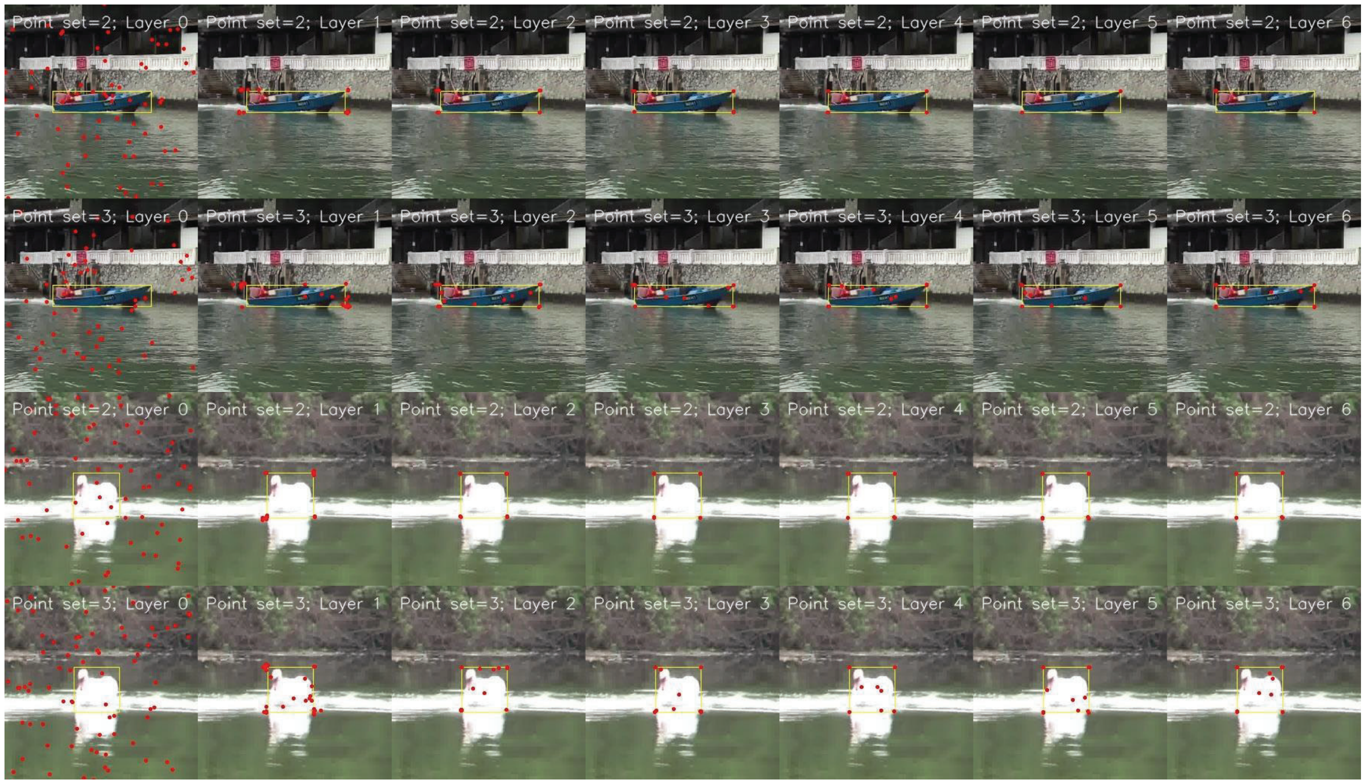


(b)

Figure 5. Visualization results of initial target estimations and refined results from six stacked diffusion layers. The number of points is set to be 2 and 3. Yellow bounding box is the final predict result.

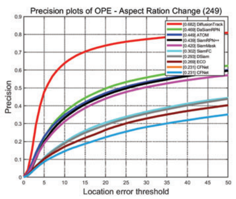


(a)

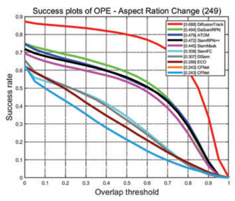


(b)

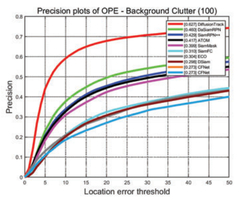
Figure 6. Visualization results of initial target estimations and refined results from six stacked diffusion layers. The number of points is set to be 2 and 3. Yellow bounding box is the final predict result.



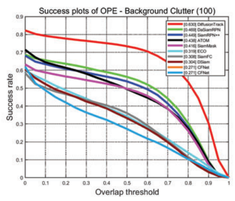
(a) subfig figure title



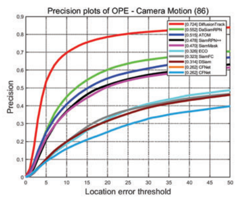
(b) subfig figure title



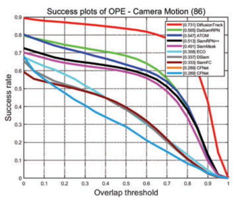
(c) subfig figure title



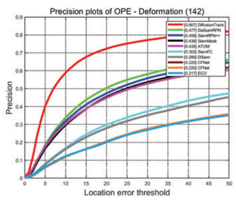
(d) subfig figure title



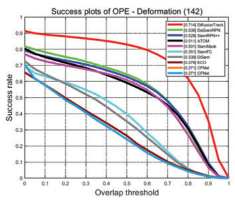
(e) subfig figure title



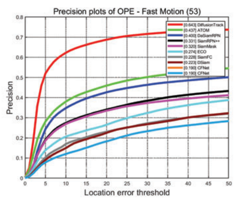
(f) subfig figure title



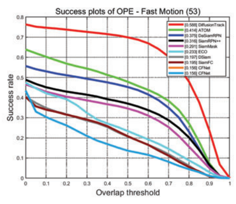
(g) subfig figure title



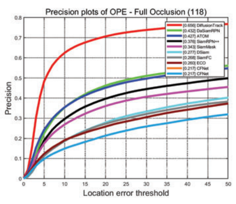
(h) subfig figure title



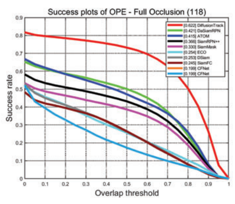
(i) subfig figure title



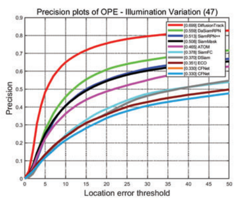
(j) subfig figure title



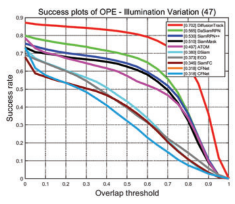
(k) subfig figure title



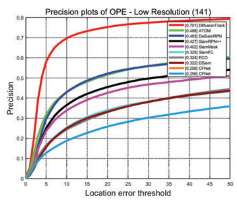
(l) subfig figure title



(m) subfig figure title

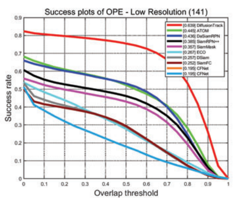


(n) subfig figure title

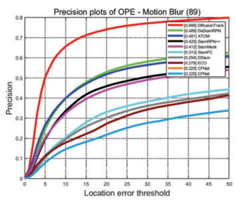


(o) subfig figure title

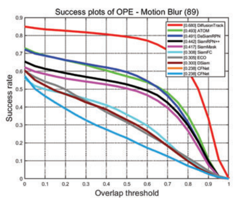
Figure 7. Attribute analysis of DiffusioTrack-b256 on LaSOT [9] benchmark-part A.



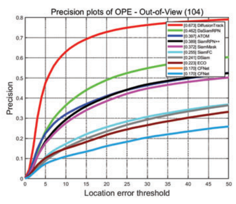
(a) subfig figure title



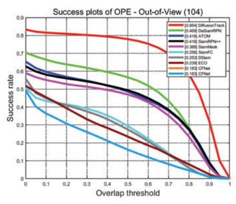
(b) subfig figure title



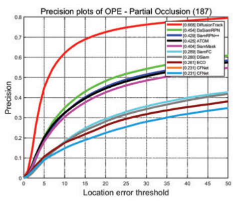
(c) subfig figure title



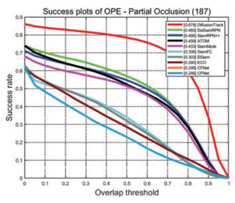
(d) subfig figure title



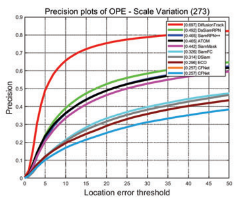
(e) subfig figure title



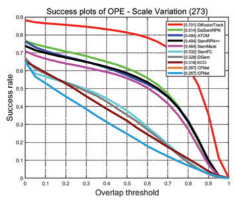
(f) subfig figure title



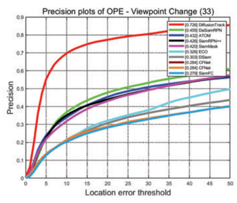
(g) subfig figure title



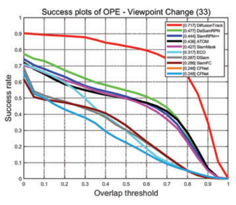
(h) subfig figure title



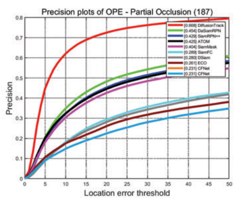
(i) subfig figure title



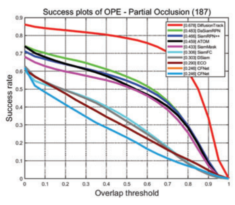
(j) subfig figure title



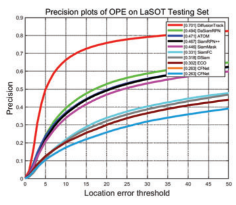
(k) subfig figure title



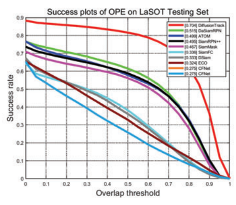
(l) subfig figure title



(m) subfig figure title



(n) subfig figure title



(o) subfig figure title

Figure 8. Attribute analysis of DiffusioTrack-b256 on LaSOT [9] benchmark-part B.