中国科学院大学
University of Chinese Academy of Sciences

硕士学位论文

中国科学院大学学位论文LaTeX模板 $\pi\pi\pi$

作者姓名：莫晃锐

指导教师：刘青泉 研究员 中国科学院力学研究所

学位类别：理学硕士

学科专业：流体力学

培养单位：中国科学院力学研究所

2014 年 6 月

LaTeX Thesis Template

of

The University of Chinese Academy of Sciences $^{\pi\pi^{\pi}}$

A thesis submitted to the

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Natural Science

in Fluid Mechanics

By

Mo Huangrui

Supervisor: Professor Liu Qingquan

Institute of Mechanics, Chinese Academy of Sciences

June, 2014

中国科学院大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日　　期：

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院大学有关保存和使用学位论文的规定，即中国科学院大学有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

本文是中国科学院大学学位论文模板ucasthesis的使用说明文档。主要内容为介绍LaTeX文档类ucasthesis的用法，以及如何使用LaTeX快速高效地撰写学位论文。

关键词： 中国科学院大学，学位论文，LaTeX模板

Abstract

This paper is a help documentation for the LaTeX class ucasthesis, which is a thesis template for the University of Chinese Academy of Sciences. The main content is about how to use the ucasthesis, as well as how to write thesis efficiently by using LaTeX.

Keywords: University of Chinese Academy of Sciences (UCAS), Thesis, LaTeX Template

目 录

图形列表

表格列表

符号列表

字符

Symbol	Description	Unit
R	the gas constant	$m^2 \cdot s^{-2} \cdot K^{-1}$
C_v	specific heat capacity at constant volume	$m^2 \cdot s^{-2} \cdot K^{-1}$
C_p	specific heat capacity at constant pressure	$m^2 \cdot s^{-2} \cdot K^{-1}$
E	specific total energy	$m^2 \cdot s^{-2}$
e	specific internal energy	$m^2 \cdot s^{-2}$
h_T	specific total enthalpy	$m^2 \cdot s^{-2}$
h	specific enthalpy	$m^2 \cdot s^{-2}$
k	thermal conductivity	$kg \cdot m \cdot s^{-3} \cdot K^{-1}$
S_{ij}	deviatoric stress tensor	$kg \cdot m^{-1} \cdot s^{-2}$
τ_{ij}	viscous stress tensor	$kg \cdot m^{-1} \cdot s^{-2}$
δ_{ij}	Kronecker tensor	1
I_{ij}	identity tensor	1

算子

Symbol	Description
Δ	difference
∇	gradient operator
$\delta^{\pm}$	upwind-biased interpolation scheme

缩写

CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
EOS	Equation of State

JWL	Jones-Wilkins-Lee
WENO	Weighted Essentially Non-oscillatory
ZND	Zel'dovich-von Neumann-Doering

第1章　END-TO-END TEMPORAL FEATURE AGGREGATION FOR SIAMESE TRACKERS

1.1 Abstract

While siamese networks have demonstrated the significant improvement on object tracking performances, how to utilize the temporal information in siamese trackers has not been widely studied yet. In this paper, we introduce a novel siamese tracking architecture equipped with a temporal aggregation module, which improves the per-frame features by aggregating temporal information from adjacent frames. This temporal fusion strategy enables the siamese trackers to handle poor object appearance like motion blur, occlusion, *etc*. Furthermore, we incorporate the adversarial dropout module in the siamese network for computing discriminative target features in an end-to-end-fashion. Comprehensive experiments demonstrate that the proposed tracker performs favorably against state-of-the-art trackers.

1.2 Introduction

Visual object tracking is the task of estimating the state of an arbitrary target in each frame of a video sequence. Recently, siamese networks have demonstrated the significant improvement on object tracking performances. However, the learned generic representation may be less discriminative because of the deteriorated object appearances in videos (Fig. 3.1), such as motion blur, occlusion, *etc*. Researchers try different ways to improve the feature representation. For example, SA-Siam **?** separately trains two branches to keep the heterogeneity of semantic/appearance features. In DaSiamRPN **?**, a novel distractor-aware incremental learning module is designed, which can effectively transfer the general embedding to the current video domain and incrementally catch the target appearance variations during inference. SiamRPN++ **?** introduces a simple yet effective sampling strategy to drive the siamese tracker with more powerful deep architectures. These efforts have produced some impact and improved state-of-the-art accuracy. However, all above siamese algorithms perform tracking based on features cropped from only the current frame, which limits the power of siamese trackers.

图 1.1 A comparison of our method with SiamMask and SiamFCv2. The example frames are from the GOT-10k testing set. Our approach effectively handles poor object appearance compared to existing approaches.

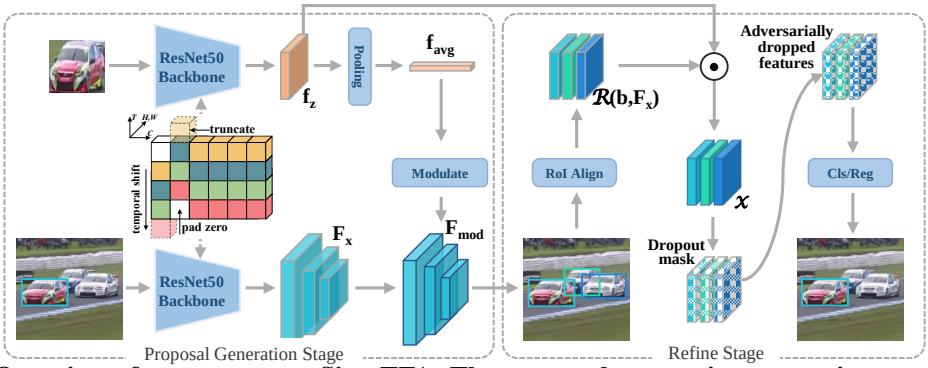

图 1.2 Overview of our two-stage SiamTFA. The proposal generation stage aims to generate proposals that are visually similar to the given template target. The refine stage aims to select the target from candidate proposals.

Actually, the video has rich information about the target and such temporal information is an important basis for video understanding and tracking. For example, in video object detection, FGFA **?** leverages temporal coherence on feature level. It improves the per-frame features by aggregation of nearby features along the motion paths, and thus improves the video recognition accuracy. In video object segmentation, STCNN **?** introduces a temporal coherence module, which focuses on capturing the dynamic appearance and motion cues to provide the guidance of object segmentation. In discriminative correlation filter-based object tracking, FlowTrack **?** focuses on making use of the rich flow information in consecutive frames to improve the feature representation and the tracking accuracy. However, how to utilize the temporal information in siamese trackers has not been widely studied yet.

In this paper, we aim to take full advantage of temporal information in siamese trackers. We introduce a novel siamese tracking architecture equipped with a temporal aggregation module, which improves the per-frame features by aggregating features

from adjacent frames. This temporal fusion strategy enables the siamese tracker to handle poor object appearance like motion blur, occlusion, *etc*. To achieve this, we shift the channels along the temporal dimension **?** in the backbone of the siamese network. Note that features of the same object are usually not spatially aligned across frames due to video motion **?**, so the temporal shift is only performed on the residual layers **?** to preserve the spatial feature learning capability of the siamese tracker. Different from other temporal fusion methods **??**, the proposed method is able to be trained end-to-end on large-scale datasets. Additionally, our temporal fusion method is easy to implement, without changing the siamese tracking architecture or using optical flow **?**.

To improve the robustness of target features, we further incorporate an adversarial dropout **?** module in the siamese tracking network. Specifically, we first predict adversarial dropout masks based on divergence maximum. Then, we aim to minimize the divergence between the randomly dropped features and the adversarially dropped features. This module has both the advantages of dropout and adversarial training: the dropout makes our siamese network randomly disconnects neural units during training to prevent the co-adaptation of target features and the adversarial training enforces our tracker to learn difficult cases.

1.3 The proposed Method

In this section, we will introduce the proposed siamese architecture-based tracking method, namely SiamTFA (Fig. 1.2), which is inspired by the great success of siamese trackers **??**. Specifically, SiamTFA takes an image pair as input, comprising a template image and a search image. The template image is the image patch cropped from the initial frame according to ground truth bounding box. The search image is one whole frame in the remaining of the video. Both inputs share the same feature extractor and parameters. Inspired by the success of the two stage detection paradigm **?**, our siamese tracker is also a two stage method. The first stage aims to generate proposals that are visually similar to the given template target. In this stage, we introduce a temporal aggregation module to enhance the temporal information (Sec. 1.3.1). The second stage aims to identify the target from candidate proposals. In this stage, we insert an

adversarial dropout module to learn more robust features (Sec. 1.3.2).

1.3.1 Temporal aggregation module

The proposal generation stage consists of 3 components: (1) feature extractor, (2) temporal aggregation module, and (3) feature modulation module. The feature extractor generates the search features and the template feature for the search image and the template image, respectively. The temporal aggregation module is integrated into the feature extractor to utilize the temporal information. The feature modulation module merge the search features and the template feature to recognize the candidate targets.

Feature extractor To deal with the scale change of the target, we use Res50-FPN **?** as our feature extractor. **F**eature **P**yramid **N**etwork (FPN) exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost. Our siamese FPN takes a template image and a search image as input. For the search image, the FPN outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. We denote the output for the search image as $F_x = \{f_x^i\}_{i=1:5}$, and note that they have strides of $\{4, 8, 16, 32, 64\}$ pixels with respect to the input search image. For the template image, we use the last stage of the FPN output as the template feature with a spatial size of 7×7.

Temporal aggregation module Most popular siamese trackers **??** use the still image to make prediction. This limits the ability of these siamese trackers. On one hand, tracking on single frame generates unstable results and fails when appearance is poor (Fig. 3.1); on the other hand, temporal adjacent frames can provide more information about the target. So we aim to improve the per-frame features by aggregating features of adjacent frames. Specifically, we insert a temporal aggregation module into the last stage of the feature extractor. To model temporal information, the images in one batch are several adjacent frames in the same video and are sorted by time, so we can regard the batch dimension as the time dimension. Assume the feature map at the last stage of the feature extractor is $f \in \mathbb{R}^{T\times C\times H\times W}$. For each time $t \leq T$, we first split feature $f^t \in \mathbb{R}^{C\times H\times W}$ into 3 parts along the channel dimension: $f^t_{1:K} \in \mathbb{R}^{K\times H\times W}$, $f^t_{(K+1):2K} \in \mathbb{R}^{K\times H\times W}$, and $f^t_{(2K+1):C} \in \mathbb{R}^{(C-2K)\times H\times W}$. Then we shifts the channels along

the temporal dimension following **?**:

$$f_{agg}^{t} = \mathcal{C}(f_{1:K}^{t-1}, f_{(K+1):2K}^{t+1}, f_{(2K+1):C}^{t}), \tag{1.1}$$

where $\mathcal{C}(\cdot)$ is the concatenation operation. According to **?**, the shift operation is only performed at the residual layer to preserve the spatial feature learning capability of the siamese tracker. Note that the aggregated feature f_{agg}^{t} has the same shape with f^{t}, so we can insert this module into the backbone directly, without the need to change other part of the network. What's more, this operation only needs to do data movement, so it is computationally free and can be trained end-to-end.

Feature modulation module After getting the template feature f_z and the search feature pyramid $F_x = \{f_x^i\}_{i=1:5}$, they are modulated to generate target-specific features. Specifically, The modulation vector f_{avg} is generated from f_z using global average pooling, which carries the target-specific appearance information. The modulated feature pyramid $F_{mod} = \{f_{mod}^i\}_{i=1:5}$ is generated as follows:

$$f_{mod}^{i} = \mathcal{M}(f_{avg}, f_x^i), \tag{1.2}$$

where $\mathcal{M}(\cdot)$ is the depth-wise correlation **?**. Each modulated feature map is fed into two sibling fully-connected layers—a box-regression layer with channel dimension $4k$, and a box classification layer with channel dimension $2k$, where k is the number of maximum possible proposals for each location. The object/background criterion and bounding box regression are defined with respect to a set of anchors. Following **?**, we assign anchors with the same scale to each of the different pyramid levels. For detail information of the anchor setting, please refer to **?**. We use the top-N ranked proposal regions for the refine stage.

1.3.2 Adversarial dropout module

The refine stage aims to select the target from candidate proposals. Features of these candidate proposals are cropped from the search feature pyramid F_x using RoIAlign **?**, and then fused with the target feature f_z:

$$\mathcal{X} = \mathcal{R}(b, F_x) \odot f_z, \tag{1.3}$$

where $\mathcal{R}$ represents the RoIAlign, $\odot$ represents the element-wise multiplication, b represents an RoI in candidate proposals and $\mathcal{X}$ represents the fused feature of b.

Adversarial dropout After the feature fusion, we use adversarial dropout **??** to increase the discriminative ability of $\mathcal{X}$. We first predict the adversarial dropout mask based on divergence maximum. The mask is applied to $\mathcal{X}$ to get the adversarially dropped features. Then, we aim to minimize the divergence between the randomly dropped features and the adversarially dropped features. Specifically, let h^{cls} and h^{reg} denote the classification layer and the regression layer in stage 2, respectively. The adversarial dropout mask is calculated as follows according to **?**:

$$\begin{aligned} \mathbf{m}^{adv} &= \arg\max_{\mathbf{m}} D[h^{cls}(\mathcal{X} \odot \mathbf{m}^s), h^{cls}(\mathcal{X} \odot \mathbf{m})] \\ where\ &||\mathbf{m}^s - \mathbf{m}|| \leq \delta_e L, \end{aligned} \tag{1.4}$$

where L represents the dimension of $\mathbf{m} \in \mathbb{R}^L$, $\mathbf{m}^s$ represents the random mask and $\mathbf{m}^{adv}$ represents the adversarial mask. δ_e is a hyper parameter to control the perturbation magnitude with respect to $\mathbf{m}^s$ **?**. $D[p, p'] \geq 0$ measures the divergence between two distributions p and p'.

To calculate $\mathbf{m}^{adv}$, **?** optimizes a 0/1 knapsack problem with appropriate relaxations in the process. Please refer to **?** for detail information. After generating $\mathbf{m}^{adv}$, we then aim to minimize the divergence between two predicted distribution regarding to $\mathcal{X}$: one with a random dropout mask $\mathbf{m}^s$ and another with an adversarial dropout mask $\mathbf{m}^{adv}$ **?**.

$$\mathcal{L}_{adv} = \mathbb{E}[D_{KL}[h^{cls}(\mathcal{X} \odot \mathbf{m}^s)||h^{cls}(\mathcal{X} \odot \mathbf{m}^{adv}))]], \tag{1.5}$$

where D_{KL} is the Kullback-Leibler divergence.

Finally, for each RoI, the classification layer produces softmax probability estimates over two classes (foreground or background) and the regression layer outputs four real-valued numbers for the foreground class. These four values encode the refined bounding-box position for the RoI. The loss of SiamTFA is:

$$\mathcal{L} = \mathcal{L}_{cls}^{stage1} + \mathcal{L}_{cls}^{stage2} + \mathcal{L}_{reg}^{stage1} + \mathcal{L}_{reg}^{stage2} + \lambda \mathcal{L}_{adv}, \tag{1.6}$$

where λ is a hyper-parameter to balance the adversarial loss and the classification/regression loss. $\mathcal{L}_{cls}^{\cdot}$ is the cross entropy loss and $\mathcal{L}_{reg}^{\cdot}$ is the standard smooth $L1$ loss for

regression. During testing, the RoI with the top classification score is selected as the predicted target.

1.4 Experiments

In this section, we first present the implementation details. Then we evaluate out method on GOT-10K **?** testing set and the UAV20L **?** dataset.

1.4.1 Implementation details

The proposed network is trained on the training set of GOT-10k **?** and the backbone is pretrained on ImageNet. We apply stochastic gradient descent with momentum of 0.9 and set the weight decay to 0.0005. The learning rate is decreased from 10^{-2} to 10^{-4}. The batch size is set to 2 and the network is trained for 90000 iterations. Our tracker is implemented in Python, using PyTorch.

1.4.2 Evaluation on GOT-10k dataset

In this subsection, we evaluate our method on GOT-10k **?** dataset. GOT-10k is a recent large-scale high-diversity dataset consisting of over 10,000 video sequences with targets annotated by axis-aligned bounding boxes. The GOT-10k testing set includes 180 sequences with 84 different object classes and 32 motion patterns. As performance measure, we use the average overlap (AO) scores and success rate (SR) as proposed in **?**. The AO denotes the average of overlaps between all groundtruth and estimated bounding boxes, while the SR measures the percentage of successfully tracked frames where the overlaps exceed 0.5/0.75.

Ablation Studies From Table 3.1 (the 1^{st} and 2^{nd} row), we see that the AO perfor-

表 1.1 Performance of our algorithm with different components on GOT-10k test set.

Temporal aggregation	Adversarial dropout	AO	$SR_{0.50}$	$SR_{0.75}$
		0.542	0.607	0.456
✓		0.561	0.645	0.480
✓	✓	0.577	0.662	0.509

表 1.2 Comparing the results of our approach against other approaches over the GOT-10k test set.

Method	AO	$SR_{0.50}$	$SR_{0.75}$
Ours	**0.577**[1]	**0.662**[1]	**0.509**[1]
SiamMask	0.459	0.560	0.205
SiamFCv2	0.374	0.404	0.144
SiamFC	0.348	0.353	0.098
GOTURN	0.347	0.375	0.124
CCOT	0.325	0.328	0.107
ECO	0.316	0.309	0.111
CF2	0.315	0.297	0.088
MDNet	0.299	0.303	0.099

mance increases by 1.9% by adding the temporal aggregation module. This is because the temporal aggregation module improves the per-frame features by aggregating temporal information from adjacent frames. From Table 3.1 (the 2^{nd} and 3^{rd} row), we see that with the adversarial dropout module, the AO increases by 1.6%. This is because the adversarial dropout module improves the discrimination power of our siamese tracking network.

Overall Performance We compare our proposed method with 8 trackers, including state-of-the-arts. The performances of the evaluated trackers is shown in Table 3.2. Compared to other listed approaches, our approach achieves a superior AO of 0.577. Compared with SiamMask, our tracker aims to make full use of the temporal information. As a result, our tracker outperforms SiamMask by 11.8% in terms of AO, which highlights the importance of the proposed temporal aggregation module.

1.4.3 Evaluation on UAV20L dataset

In this subsection, we evaluate our tracker on the UAV20L **?** long term tracking dataset. It contains 20 HD video sequences captured from a low-altitude aerial perspective with average sequence length of 2934 frames. In this experiment, all trackers are compared using two measures: precision and success. Precision is measured as

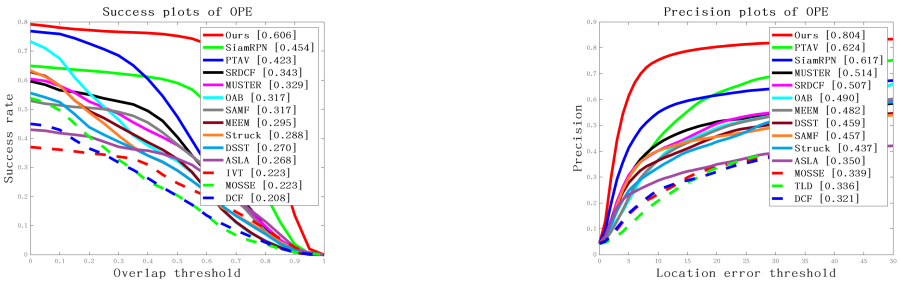

图 1.3 Success and precision plots on UAV20L dataset.

the distance between the centers of the predicted bounding box and the corresponding ground truth bounding box. Success is measured as the intersection over union of pixels in predicted bounding box and those in ground truth bounding box. In Fig. 3.2, we can find that the proposed algorithm achieves better tracking performance compared with some representative trackers. In the success plot, our tracker obtains an AUC score of 0.606. In the precision plot, the proposed algorithm obtains a score of 0.804. It shows that our tracker surpass other state-of-the-art algorithms, such as SiamRPN **?** and PTAV **?**. This demonstrates the effectiveness of our tracker in long-term tracking scenario.

1.5 Conclusion

In this paper, we introduce a novel siamese architecture for visual object tracking. Specifically, our proposed algorithm contains two main modules, *i.e.* temporal aggregation module and adversarial dropout module. The temporal aggregation module improves the per-frame features by aggregating features of adjacent frames. The adversarial dropout module improves the discrimination power of the siamese tracking network. Extensive experimental results show that the proposed algorithm performs favorably against the state-of-the-art algorithms.

第2章　Manipulating Template Pixels for Model Adaptation of Siamese Visual Tracking

2.1　Abstract

In this letter, we show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image in Siamese networks. For a target that is not included in the offline training set, a slight modification of the template image pixels will improve the prediction result of the offline trained Siamese network. The popular adversarial example generation methods can be used to perform template pixel manipulation for model adaptation. Different from current template update methods, which aim to combine the target features from previous frames, we focus on the initial adaptation using target ground-truth in the first frame. Our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of its base tracker. To our knowledge, this work is the first attempt to directly manipulating template pixels for model adaptation in Siamese-based trackers. Extensive experiments on recent benchmarks demonstrate that our method achieves better performance than some other state-of-the-art trackers. Our code is available at https://github.com/lizhenbang56/MTP.

2.2　Introduction

Object tracking refers to the task of sequentially locating a specified moving object in a video, given only its initial state. Recently, Siamese networks **??** have demonstrated a significant improvement in object tracking performances. Siamese trackers formulate the visual object tracking problem as learning cross-correlation similarities between a target template and a search region. Tracking is then performed by finding the target object from the search image region by computing the highest visual similarity. Despite its recent success, the learned similarity measure of the Siamese network is not necessarily reliable for objects that are not included in the offline training set, leading to poor generalization **?**. Several recent works aim to adapt the model to the current target appearance. For example, TADT **?** identifies the importance of each convolutional

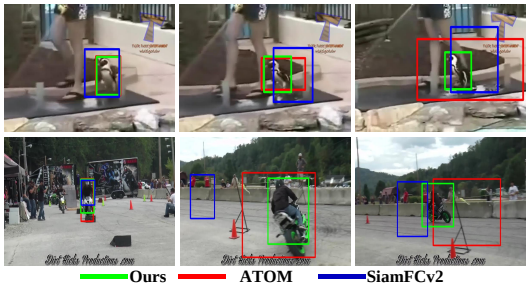

图 2.1 A comparison of our method with the state-of-the-art trackers ATOM ? and SiamFCv2 ? in challenging situations. The example frames are from the GOT-10k ? testing set.

filter according to the back-propagated gradients and selects the target-aware features based on activations for representing the targets. However, the feature extractor of TADT is pre-trained on ImageNet **?**, not on large-scale visual tracking datasets. This limits the representation ability of its features on the object tracking task. GradNet **?** exploits the discriminative information in gradients and updates the template in the Siamese network through feed-forward and backward operations. However, the extra sub-network increases the computational cost and is prone to overfitting. UpdateNet **?** learns to combine the target features from previous frames. However, it does not use ground truth information to adaptively adjust the template features of the first frame.

In this work, we show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image in Siamese networks. Given an object tracker, our algorithm modifies template pixels in only a few gradient-descent iterations using the target ground-truth in the first frame. For a target that is not included in the offline training set, we believe that a slight modification of the template image pixels can improve the prediction result of the offline trained Siamese network. We use the adversarial example generation method to achieve this, because it is commonly used to slightly modify the input image, and thereby impose an impact on the prediction result of the network. We depart from the purpose of adversarial sample generating in that the latter is aimed to make the prediction of the network worse, while we hope the prediction of the Siamese network is better. The proposed model adaptation method can be integrated with varieties of Siamese trackers like SiamFC++ **?**. Note that the parameters of the Siamese network remain intact to preserve the generative ability of offline-trained embedding space. We perform comprehensive experiments on

4 tracking benchmarks: VOT2018**?**, TrackingNet**?**, GOT-10k**?**, and OTB2015**?**. Our approach achieves state-of-the-art results while running at over 80 FPS (see Fig. 2.1).

2.3 Proposed Algorithm

In this section, we present a new model adaptation approach for Siamese trackers via directly manipulating template pixels. We start by revisiting the tracking process of popular template matching-based trackers, which is closely related to the proposed approach. Following **?**, we formulate object tracking as a confidence-based regression problem, which learns a function $s_\theta : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$, and predicts a scalar confidence score $s_\theta(y, x) \in \mathbb{R}$ given an output-input pair (y, x). The final estimate $f(x) = y^*$ is predicted as follows:

$$f(x) = \arg\max_{y \in \mathcal{Y}} s_\theta(y, x), \tag{2.1}$$

where x is an input image. y usually represents the center 2D image coordinate of the target object. Currently, there are two prevalent template matching-based paradigms: discriminative correlation filter (DCF) approaches and Siamese tracking methods.

In DCF-based methods, a circular correlation filter w_θ is trained during tracking to predict a target confidence score:

$$s_\theta(y, x) = (w_\theta * \phi(x))(y), \tag{2.2}$$

where $\phi(x)$ is the features extracted from the search image x.

In contrast to DCF, Siamese trackers exploit a two-stream architecture. One stream extracts the target' s features $\phi_\theta(z)$ based on the template image z which is cropped from the first frame according to the ground truth bounding box. The other stream receives as input a large search image x and outputs the search features $\phi_\theta(x)$. The two outputs are cross-correlated to predict a target confidence score:

$$s_\theta(y, x) = (\phi_\theta(z) * \phi_\theta(x))(y). \tag{2.3}$$

DCF-based trackers and Siamese trackers both have the advantage of utilizing large-scale visual tracking datasets to train the feature extractor $\phi(\cdot)$ or the embedding network $\phi_\theta(\cdot)$ by their own right. This way the representation ability of features on the object tracking task can be enhanced.

In contrast to Siamese trackers, DCF learns the filter w_θ from example patches of the target appearance to discriminate it from the background. Despite the improved tracking efficiency using the circular correlation operation, its boundary effect and sophisticated optimization prevent itself from making a good trade-off between computational speed and tracking performance. Siamese trackers do better in this aspect, although the learned similarity measure in the cross-correlation is not necessarily reliable for objects that are not included in the offline training set, leading to poor generalization.

In this letter, we aim to design a new Siamese tracking method that has the ability to make full use of the specific information of the current video for model adaptation like the DCF-based trackers, although the object in it is not included in the offline training set. This is achieved by utilizing the annotation information in the first frame to perform model adaptation.

Note that Equ. (2.2) and Equ. (2.3) bear some similarities with each other, and the main difference lies in the kernel term for correlation: the kernel of DCF is the online-learned w_θ, and the kernel of the Siamese network is $\phi_\theta(z)$. In order for the Siamese network to have the model adaptation ability, we need to adapt $\phi_\theta(z)$ using the first frame ground truth annotation of the current video. There are two design choices to adapt $\phi_\theta(z)$: changing $\phi_\theta(\cdot)$ or changing z. However, changing $\phi_\theta(\cdot)$ may cause too much tedious meta-learning settings to ensure the generative ability of offline-trained embedding space **??**. In contrast, our solution is in a simple way to perform model adaptation of Siamese trackers by changing z, i.e., modifying template pixels in only a few gradient-descent iterations using the target ground-truth at the first frame. Compared with current model adaptation methods of Siamese trackers, the proposed method has the following advantages. First, we never modify parameters of the Siamese network to preserve the representation ability of offline-trained embedding space. Second, different from current template update methods **??**, which aim to combine the target features from previous frames, we focus on the initial adaptation using target ground-truth at the first frame. Finally, our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of the base tracker. In the next subsection, we will show how the popular adversarial example generation methods can be used to perform

template pixel manipulation for model adaptation.

2.3.1 Manipulating Template Pixels for Model Adaptation

At first glance, there may be a contradiction between the model adaptation task and the adversarial example generation task, because these two tasks have different purposes. An adversarial example **?** is a sample of input data which has been modified very slightly in a way that is intended to perform an attack on machine learning systems, which means to cause a machine learning model to make wrong predictions. However, the purpose of model adaptation in our work is to make full use of the annotation information in the first frame to improve the tracking performance for the current video. We will point out in the following that there are some similarities between these two tasks and we can utilize adversarial example generation methods to perform the model adaptation task.

Before introducing the proposed method, we first revisit the popular adversarial example generation methods. One of the simplest methods to generate adversarial images I^{adv} works by linearizing loss function in L_∞ neighbourhood of a clean image and finds exact maximum of linearized function using following closed-form equation **?**:

$$I^{adv} = I + \epsilon \operatorname{sign}\big(\nabla_I L(I, y_{true})\big), \tag{2.4}$$

where I is the input image, and the values of the pixels are integer numbers in the range $[0, 255]$. y_{true} is the true label for the image I. $L(I, y)$ is the cost function of the neural network for the attack purpose, given image I and label y. ϵ is a hyper-parameter to be chosen. A straightforward way to extend the above method is applying it multiple times with small step size, and clipping pixel values of intermediate results after each step to ensure that they are in an ϵ-neighbourhood of the original image. This leads to the Basic Iterative Method (BIM) introduced in **?**:

$$\begin{aligned} I_0^{adv} &= I, \\ I_{N+1}^{adv} &= Clip_{I,\epsilon}\{I_N^{adv} + \alpha \operatorname{sign}(\nabla_I L(I_N^{adv}, y_{true}))\}, \end{aligned} \tag{2.5}$$

where $Clip_{I,\epsilon}\{I'\}$ is the function which performs per-pixel clipping of the image I', so that the result will be in L_∞ ϵ-neighbourhood of the source image I. The BIM can be

表 2.1 State-of-the-art comparison on the popular tracking benchmark OTB2015 with the running speed. FPS: frame per second. Our speed is tested on an NVIDIA RTX 2080Ti GPU.

Trackers	ECO	MDNet	SiamRPN++	ATOM	SiamFC++_G	Ours
Success	70.0	67.8	69.6	66.9	68.3	69.7
FPS	8	1	35	30	90	82

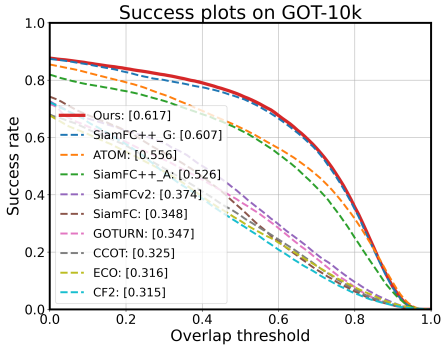

图 2.2 Comparing the results of our approach against other approaches over the GOT-10k test set. The trackers are ranked by their average overlap scores.

easily made into an attacker for a specific desired target class, called the Iterative Target Class Method **?**:

$$\begin{aligned} I_0^{adv} &= I, \\ I_{N+1}^{adv} &= Clip_{I,\epsilon}\{I_N^{adv} - \alpha \ \text{sign}(\nabla_I L(I_N^{adv}, y_{target}))\}. \end{aligned} \tag{2.6}$$

Equ. (2.6) shows that manipulating pixels of the input image in only a few gradient-descent iterations is able to change the network's prediction to the target class y_{target}. Note that our purpose is manipulating pixels of the template image in the first frame, so that the prediction is closer to the ground truth bounding box. So we can perform model adaptation for Siamese networks using Equ. (2.6) with some modifications:

$$\begin{aligned} z_0 &= z, \\ z_{N+1} &= Clip_{z,\epsilon}\{z_N - \alpha \ \text{sign}(\nabla_z L(z_N, y_{bb}))\}, \end{aligned} \tag{2.7}$$

where z is the template image in the first frame, and y_{bb} is the label for Siamese trackers generated from the ground truth bounding box. In the next subsection, we will introduce the overall tracking process equipped with the proposed model adaptation module.

表 2.2 state-of-the-art comparison on the TrackingNet test dataset in terms of precision, normalized precision and success.

Method	Prec.	Norm. Prec.	Succ.
Ours	70.6	81.7	74.9
SiamFC++_G	70.5	80.0	75.4
SiamFC++_A	64.6	75.8	71.2
ATOM	64.8	77.1	70.3
SiamRPN++	69.4	80.0	73.3
MDNet	56.5	70.5	60.6
ECO	49.2	61.8	55.4
SiamFC	51.8	65.2	55.9

2.3.2 Tracking Framework

The proposed model adaptation method is integrated with the SiamFC++ tracker **?** in a plug-and-play manner. SiamFC++ is based on SiamFC **?** and progressively refined according to several guidelines proposed in **?**. The input of the original SiamFC++ network is composed of the template image z_0 cropped from the first frame and search image x_i cropped from the i-th frame. However, we expect to perform template manipulating such that, after N steps of pixel update on the input pair (z_0, x_0) to obtain z', the tracker performs well on (z', x_i). To achieve this, we start by cropping the initial template image $z_0 \in \mathbb{R}^{3\times128\times128}$ and the initial search image $x_0 \in \mathbb{R}^{3\times289\times289}$ from the first frame using the ground truth bounding box. Then, z_0 and x_0 are sent to the SiamFC++ network to obtain the tracking prediction of the first frame. The tracking loss in SiamFC++ **?** is calculated as follows:

$$L = L_{\text{cls}} + L_{\text{quality}} + L_{\text{reg}}, \tag{2.8}$$

where L_{cls} is the focal loss **?**. L_{quality} is the binary cross entropy (BCE) loss for quality assessment. L_{reg} is the IoU loss **?** for bounding box regression. The gradient with respect to the template z_0 is used to generate z_1 according to Equ. (2.7). The updated template z' is obtained by applying a very small number of iterations of Equ. (2.7). Note that the template image is only updated with the first frame of given sequences and is kept fixed during the whole tracking process to ensure stability. The subsequent

tracking procedure remains the same as SiamFC++.

2.4 Experiments

In this section, we first present the implementation details. Then we compare our method with the state-of-the-art trackers on four tracking datasets: OTB2015 **?**, VOT2018 **?**, GOT-10k **?** and TrackingNet **?**. Specifically, OTB2015 **?** includes 100 sequences, which are labeled with different attributes for in-depth analysis of tracking performance. VOT2018 **?** uniquely applies a reset-based methodology and specially selects 60 sequences of various tracking scenarios for evaluation. GOT-10k **?** and TrackingNet **?** are two recent large-scale high-diversity datasets. They both cover diverse object classes and scenes in the train and test splits. For GOT-10k, there is no overlap in object classes between the train and test splits, promoting the importance of generalization to unseen object classes.

2.4.1 Implementation Details

We use SiamFC++ **?** as our base tracker, and the backbone Siamese network adopts GoogLeNet **?**. We do not perform any changes except for the model adaptation component. The parameter α in Equ. (2.7) is set to 0.05. Our method is implemented in Python with PyTorch. The proposed tracker runs at over 80 FPS on an NVIDIA RTX 2080Ti GPU.

2.4.2 State-of-the-art Comparison

OTB2015 We use success rate to evaluate the performance of trackers on the OTB2015 dataset. Success rate relies on the intersection over union (IOU) of the predicted bounding box and the ground truth bounding box. We compare our method with various tracking algorithms, including ECO **?**, MDNet **?**, SiamRPN++ **?**, ATOM **?** and SiamFC++_GoogLeNet **?**. The iteration number for template update is set to 16. The results are shown in Table 2.1. Our tracker outperforms the online tracker ATOM by 2.8% in terms of the success score, which demonstrates the powerful model adaptation capability of our method.

VOT2018 We compare our method with RCO **?**, UPDT **?**, SiamRPN **?**, MFT **?**,

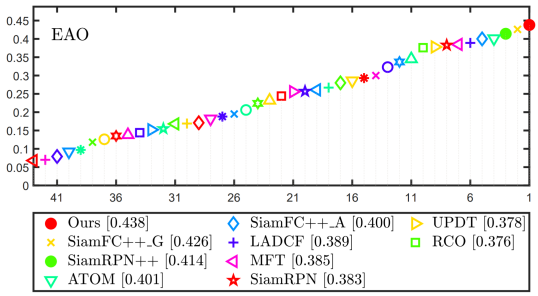

图 2.3 Expected average overlap graph with trackers ranked from right to left. The right-most tracker is the top-performing according to the VOT2018 expected average overlap values.

LADCF **?**, ATOM **?**, SiamRPN++ **?**, SiamFC++_AlexNet **?** and SiamFC++_GoogLeNet **?** on VOT2018. The trackers are compared using the robustness and accuracy measures. Robustness indicates the number of tracking failures, while accuracy denotes the average overlap between tracker prediction and the ground-truth box. Both of the measures are combined into a single expected average overlap (EAO) score. The iteration number for template update is set to 2. As shown in Fig. 2.3, the performances of all the listed trackers are not as good as our algorithm.

GOT-10k We use the average overlap (AO) score as performance measure following **?**. We compare our method with CF2 **?**, ECO **?**, CCOT **?**, GOTURN **?**, SiamFC **?**, SiamFCv2 **?**, ATOM **?**, SiamFC++_AlexNet **?** and SiamFC++_GoogLeNet **?** on this dataset. The iteration number for template update is set to 2. In Fig. 2.2, we can find that the proposed algorithm achieves better tracking performance compared with the listed state-of-the-art trackers.

TrackingNet We compare our method with SiamFC **?**, ECO **?**, MDNet **?**, SiamRPN++ **?**, ATOM **?**, SiamFC++_AlexNet **?** and SiamFC++_GoogLeNet **?**. The iteration number for template update is set to 32. Table 2.2 shows that our tracker performs best in terms of precision and the normalized precision while maintaining a very competitive success value.

2.4.3 Ablation Study

Robustness to Noisy Initial Frames To study how robust the approach is if the first frame is noisy compared to the next frames in the sequence, we add three kinds of noise to the first frame by: changing the image brightness, applying Gaussian blur, and using non-accurate ground truth bounding box annotation. We denote $\gamma_1 \in [0.5, 1.5]$

表 2.3 Performance on OTB2015 using noisy initial frames.

Coefficients of Noise			Success Score	
γ_1	γ_2	γ_3	SiamFC++_G	Ours
0.57	1.6	0.93	67.3	69.5
0.88	0.22	0.86	65.7	68.0
0.88	0.43	0.77	64.1	64.4
0.92	0.98	0.82	65.6	66.6
0.98	0.83	0.84	66.5	67.9
1.0	0.81	0.79	65.3	66.0
1.1	1.9	0.90	67.6	68.8
1.2	0.70	0.79	65.7	66.6
1.5	0.19	0.79	64.8	66.3

表 2.4 11 attributes comparison on OTB2015 in term of success score.

	BC	DEF	FM	IV	IPR	LR	MB	OCC	OPR	OV	SV
SiamFC++_G	62.5	63.9	68.0	69.2	69.5	**72.5**	66.7	60.9	66.9	55.8	68.4
Ours	**66.5**	**66.8**	**68.7**	**71.3**	**69.6**	69.8	**67.3**	**62.5**	**68.4**	**57.1**	**69.8**

as the coefficient of brightness variation, $\gamma_2 \in [0, 2]$ the blur radius, and $\gamma_3 \in [0.75, 1]$ the IoU between the non-accurate first frame bounding box annotation and the real bounding box annotation, respectively. We run both the baseline tracker and our tracker 10 times on OTB2015 with randomly sampled γ_1, γ_2 and γ_3 (see Table 2.3). Compared with SiamFC++_GoogLeNet **?**, our tracker always performs better under different noise levels, which shows the robustness of our tracker to noisy initial frames.

Attribute-based Analysis For further analyses on the tracking performance, we also demonstrate the advantages of our algorithm through the attribute-based comparison on sequences of the OTB-2015 dataset (see Table 2.4). In OTB2015, each sequence is annotated with 11 different attributes, namely: background clutters (BC), deformation (DEF), fast motion (FM), illumination variation (IV), in-plane rotation (IPR), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). Compared with SiamFC++_GoogLeNet **?**, our tracker achieves better performance on 10 out of 11 attributes, which demonstrates its robustness in challenging tracking scenarios such as illumination variation and motion blur.

2.5 Conclusion

In this letter, we propose a novel model adaptation method for Siamese trackers, achieving accurate tracking with high speed. We show that the challenging model adaptation task in visual object tracking can be handled by simply manipulating pixels of the template image using the target ground-truth in the first frame. Our model adaptation method is pluggable, in the sense that it does not alter the overall architecture of the base tracker. Numerous experimental results on four object tracking benchmarks demonstrate the effectiveness of the proposed model adaptation method.

第3章 GLOBALLY SPATIAL-TEMPORAL PERCEPTION: A LONG-TERM TRACKING SYSTEM

3.1 Abstract

Although siamese trackers have achieved superior performance, these kinds of approaches tend to favour the local search mechanism and are thus prone to accumulating inaccuracies of predicted positions, leading to tracking drift over time, especially in long-term tracking scenario. To solve these problems, we propose a siamese tracker in the spirit of the faster RCNN's two-stage detection paradigm. This new tracker is dedicated to reducing cumulative inaccuracies and improving robustness based on a global perception mechanism, which allows the target to be retrieved in time spatially over the whole image plane. Since the very deep network can be enabled for feature learning in this two-stage tracking framework, the power of discrimination is guaranteed. What's more, we also add a CNN-based trajectory prediction module exploiting the target's temporal motion information to mitigate the interference of distractors. These two spatial and temporal modules exploit both the high-level appearance information and complementary trajectory information to improve the tracking robustness. Comprehensive experiments demonstrate that the proposed Globally Spatial-Temporal Perception-based tracking system performs favorably against state-of-the-art trackers.

3.2 Introduction

Object tracking **???** is a challenging problem in the field of computer vision, which aims to establish the positional relationship of the object to be tracked in a continuous video sequence. The popular siamese trackers **???** are typically based on the local search mechanism: searching the target within a small neighborhood centered on the target position of the previous frame to determine its current position. This mechanism works well if the target only has a small displacement between two adjacent frames. It also brings benefits in another aspect, which is to avoid interference from the distractors in the background.

However, the local search mechanism bears some shortcomings. First, it could

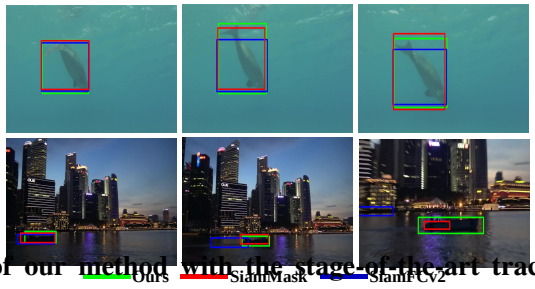

图 3.1 A comparison of our method with the state-of-the-art trackers SiamMask ? and SiamFCv2 ? in challenging situations. The example frames are from the GOT-10k ? testing set.

cause irreversible cumulative errors if the predictions of the target positions in the previous frames drift away due to challenging illumination change, motion blur, *etc.*, because the search area generated in the current frame may not cover the target leading to a complete failure in subsequent frames. Second, it is difficult for the local mechanism-based trackers to meet the needs of long-term tracking **??**. Under the long-term scenario, the target frequently re-enters and re-exits the screen. Since the tracker cannot set the correct search area when the target leaves and re-enters the screen, it often fails to retrieve the target due to the wrong search area without the target covered.

Inspired by faster RCNN's two stage detection paradigm **?**, we propose a siamese tracker based on the global perception mechanism. During the tracking process, our tracker is always able to perceive the target over the entire image. Therefore, even if the tracker makes a mistake due to the challenging target appearance variations, the target can still be retrieved in time once its appearance returns to normal. Especially under the long-term out-of-view disappearance scenario, where the tracker cannot find the target in the full image when the target leaves the screen, our tracker can continue to work when the target re-enters the screen from any position.

Besides the above globally spatial perception mechanism, we also propose a temporal motion model to mitigate the interference of distractors. It is known that the siamese-based tracking framework is sometimes plagued by distractors because it is difficult for the end-to-end trained siamese matching network to distinguish well between objects that look very similar. Different from the simply designed hand-crafted strategies **??**, our proposed CNN-based motion model is end-to-end trained and can predict the target current position distribution using its historical trajectory information and

current target appearance information. Specifically, the motion patterns are automatically learned from the trajectory dataset in the training phase instead of the hand-crafted features or rules **?**; in the testing phase, we can use the position information of any number of historical frames instead of just the previous frame for prediction.

3.3 THE PROPOSED ALGORITHM

Our method explores the key idea that the task of visual object tracking can be tackled by splitting it into first extracting candidate targets, followed by eliminating distractors using the motion model. By adopting this paradigm, we can achieve better performance by designing a more accurate tracking component and a more robust motion model, especially in the long-term scenario.

Specifically, we propose the Globally Spatial-Temporal Perception tracking system, which means: (1) We use an entire image instead of a small image patch as the input to the tracker to provide the global spatial information for it. (2) In order to better perceive the global spatial information, we propose a two-stage tracking component, which is able to detect candidate targets that are visually similar to the ground truth target. (3) To perceive the temporal information, we propose a motion model, which is able to exclude the distractors by predicting the location distribution to obtain the final tracking result.

3.3.1 Tracking component

We build our tracking component based on the popular object detection architecture — faster RCNN **?**. Although the use of object detection modules and siamese-based feature extractor for object tracking is not first proposed in this paper, our tracker has advantages over relevant tracking algorithms **????**. Compared with SiamRPN **?**, whose input is always image patches with target located at the center, the input of our tracking component is the entire image, which means the target may appear at any spatial location. This prevents the network from learning bias to the center location, thereby breaking the spatial invariance restriction **?** of the network architecture. As a result, the very deep networks such as ResNet50 **?** can be used as the feature extractor of the tracking component. For detailed explain about spatial invariance restriction and center bias, please refer to **?**. Compared with SiamRPN **?** and SiamMask **?**, we use a two

stage architecture. The second stage (RoI head **?**) can distinguish the foreground and background more effectively. Compared with SiamMask **?** and ATOM **?**, our tracker can search for the target in the whole image, which allows the target to be retrieved in time spatially after the tracking module makes a mistake. Compared with Siam R-CNN **?**, whose backbone and RPN are frozen, our tracker generate the target-specific features before RPN, and the whole network can be trained end to end. In object tracking, a target may be foreground in one video and background in another. Therefore, the candidates generated by our RPN can better meet the requirements of general object tracking.

To describe our tracking component, we we first briefly revisit faster RCNN. It consists of a feature extractor followed by two detection stages: the RPN head and the RoI head. The feature extractor ϕ_1 is a variant of ResNet50. The first stage uses a region proposal network (RPN) to slide on the last feature map of the backbone layers and predict whether there is an object or not and also predict the bounding box of those objects. The second stage (*i.e.*, RoI head) is run for each region proposed by the RPN by performing RoI Align **?** to extract deep features from this proposed region. Each RoI (Region of Interest) is classified as a specific category using a classification layer and the bounding box is refined using a regression layer. Please refer to **?** for more detailed information.

For the task of object tracking, there are two inputs to the feature extractor ϕ_1: the template image z and the search image x. According to the design of the siamese architecture, the two inputs share the same network parameters to extract features. After generating the template feature $f_z = \phi_1(z)$ and search feature $f_x = \phi_1(x)$, we crop the object feature $f_{obj} \in \mathbb{R}^{1024\times7\times7}$ by the RoI Align operation **?** from the template features according to the ground truth of the target:

$$f_{obj} = \mathcal{R}(b_{obj}, f_z), \tag{3.1}$$

where $\mathcal{R}$ represents the RoI Align operator and b_{obj} is the ground truth bounding box of the target. Next, the search feature and the object feature are merged via the depth-wise cross-correlation **?**:

$$f_{corr} = f_{obj} * f_x, \tag{3.2}$$

where f_{corr} is the fusion feature and $*$ is the depth-wise cross-correlation operator. Then f_{corr} is sent to the RPN head to generate candidates $B = \{b^i_{roi}\}^{i=1:N}$. At a second stage, the RoI Align operation is performed on the fusion feature f_{corr}, generating a small feature map with a channel dimension of 2048 and a fixed spatial extent of 7×7 for every RoI:

$$f^i_{roi} = \mathcal{R}(b^i_{roi}, f_{corr}), \tag{3.3}$$

where b^i_{roi} is the bounding box of candidate region i. Finally, each RoI is classified as foreground/background. During testing, we select top K ranked RoI as the candidate targets, which will be post-processed in the motion model.

3.3.2 Motion model

After detecting object regions that are visually similar to the given first-frame template object with our tracking component, we use a motion model to eliminate distractors and obtain the final tracking result. The motion model works in an end-to-end manner by learning the target position distribution using historical trajectory information and appearance information of the current frame. It then rescores the candidate targets according to the position distribution, which is a 2D heatmap measuring the likelihood that the target is located at each spatial location.

Let $H^k_t = \{h_{t-i}\}_{i=1:k}$ denote the historical trajectory information, where t is the index of the current frame, k is the length of history, and $h_j = \{x_j, y_j\}$ is a two-dimensional coordinate representing the position of the target in frame j. Inspired by the pose estimation task, we present h_j as a two-dimensional heatmap $m_j \in \mathbb{R}^{h \times w}$ with a 2D Gaussian centered on the target position (x_j, y_j). To model the temporal information, The generated k heatmaps are concatenated according to the time order to obtain the trajectory tensor $\mathcal{M} \in \mathbb{R}^{k \times h \times w}$ with channel dimension k:

$$\mathcal{M}^k_t = \mathcal{C}(m_{t-k}, m_{t-k+1}, ..., m_{t-1}), \tag{3.4}$$

where $\mathcal{C}(\cdot)$ is the concatenation operation.

Our motion model not only utilizes the historical trajectory information for prediction, but also considers the appearance information of the current frame. To achieve this, the RGB image of the current frame $\mathcal{I} \in \mathbb{R}^{3 \times h \times w}$ and the trajectory tensor are

concatenated to obtain the enhanced trajectory tensor $\mathcal{N} \in \mathbb{R}^{(3+k)\times h\times w}$ with channel dimension $(3 + k)$:

$$\mathcal{N}_t^k = \mathcal{C}(\mathcal{I}_t, \mathcal{M}_t^k). \tag{3.5}$$

Assuming ϕ_2 is the motion model, which is a CNN network, the output of ϕ_2 is calculated as follows:

$$\mathcal{O}_t^k = \phi_2(\mathcal{N}_t^k), \tag{3.6}$$

where $\mathcal{O}_t^k \in \mathbb{R}^{h\times w}$ is a 2D heatmap reflecting the position distribution of the target in the frame t.

The network of our motion model ϕ_2 is the same as the pose estimation network HRNet **?**. A brief description is provided here. The first stage of HRNet is a high-resolution subnetwork. Then high-to-low resolution subnetworks are added one by one to form more stages. The multi-resolution subnetworks are connected in parallel. Repeated multi-scale fusions are conducted by exchanging the information across the parallel multi-resolution subnetworks over and over through the whole process. We estimate the position distribution over the high resolution representations output by HRNet. Please refer to **?** for details of the network structure.

3.4 EXPERIMENTS

3.4.1 Implementation details

Our tracking component is trained on the GOT-10k **?** training set. It contains more than 10000 video segments of real-world moving objects and over 1.5 million manually labeled bounding boxes, which covers 563 classes of real-world moving objects and 87 classes of motion patterns. We perform multi-scale training: the target size varies from 64×64 to 256×256. The image size remains the same during the tracking progress. The tracking component is trained with stochastic gradient descent (SGD). We use a weight decay of 10^{-4} and momentum of 0.9. We train the tracking component for 27k iterations. The learning rate is decreased from 10^{-2} to 10^{-4}.

The motion model is also trained on the GOT-10k training set. The Adam optimizer **?** is adopted for training. The base learning rate is set as 10^{-3}, and is dropped to 10^{-4}

表 3.1 Performance of our algorithm with different components on GOT-10k test set.

ROI head	Motion model	AO	$SR_{0.50}$	$SR_{0.75}$
		0.410	0.486	0.162
✓		0.521	0.595	0.440
✓	✓	0.560	0.645	0.457

and 10^{-5} at the 170th and 200th epochs, respectively. The training process is terminated within 210 epochs.

3.4.2 Evaluation on GOT-10k Dataset

In this subsection, we evaluate our method on GOT-10k **?** dataset. The evaluation metric of GOT-10k includes average overlap (AO) and success rate (SR). The AO denotes the average of overlaps between all groundtruth and estimated bounding boxes, while the SR measures the percentage of successfully tracked frames where the overlaps exceed 0.5/0.75.

Ablation Studies From Table 3.1 (the 1^{st} and 2^{nd} row), we see that the AO performance increases by 11.1% by adding the RoI head. The RPN stage rapidly filters out most background samples, and the RoI head adopts a fixed foreground-to-background ratio to maintain a manageable balance between foreground and background. From Table 3.1 (the 2^{nd} and 3^{rd} row), we see that with the motion model, the AO, the $\text{SR}_{0.50}$ and the $\text{SR}_{0.75}$ increases by 3.9%, 5.0% and 1.7%, respectively. This is because the proposed motion model can effectively predict the position distribution of the target, effectively avoiding the adverse effects of distractors.

Overall Performance We compare our proposed method with 8 trackers including state-of-the-arts, on GOT-10k testing set. Compared to the listed approaches, our approach achieves a superior AO of 0.560 (Table 3.2). Compared with SiamMask **?**, our two stage tracker is designed based on the global perception mechanism to reducing cumulative inaccuracies. The motion model suppresses distractors and improves the tracking robustness. As a result, our tracker outperforms SiamMask by relative 22.00%

表 3.2 Comparing the results of our approach against other approaches over the GOT-10k test set. The trackers are ranked by their average overlap (AO) scores.

Method	AO	$SR_{0.50}$	$SR_{0.75}$
Ours	**0.560**[1]	**0.645**[1]	**0.457**[1]
SiamMask	0.459	0.560	0.205
SiamFCv2	0.374	0.404	0.144
SiamFC	0.348	0.353	0.098
GOTURN	0.347	0.375	0.124
CCOT	0.325	0.328	0.107
ECO	0.316	0.309	0.111
CF2	0.315	0.297	0.088
MDNet	0.299	0.303	0.099

表 3.3 Performance on subsets with different attributes collected from GOT-10k validation set.

Att.	SiamFC		SiamMask		Ours	
	AO	$SR_{0.5}$	AO	$SR_{0.5}$	AO	$SR_{0.5}$
FM	0.472	0.538	0.526	0.608	0.639	0.715
OC	0.411	0.447	0.494	0.559	0.585	0.659
CU	0.505	0.545	0.595	0.701	0.738	0.837
LO	0.557	0.655	0.643	0.779	0.721	0.807

in terms of AO, which highlights the importance of the proposed tracker and the motion model.

Performance Analysis by Attributes Every video in GOT-10k training/validation dataset is annotated with multiple attributes including: visible ratios, motion speed, video length and cut by image. To analyze the performance of trackers in various scenarios, we compare our tracker with two state-of-the-art trackers (*i.e.*, SiamFC **?** and SiamMask **?**) on GOT-10k validation set. We collect four subset from the validation set according to the attribute annotations: FM (fast motion) subset, OC (occlusion) subset, CU (cut by image) subset, and LO (long video) subset. The FM subset includes videos

in which the target motion speed is fast. The OC subset include videos in which the target is occluded frequently. The CU subset include videos in which the target is cut by the image boundary frequently. The LO subset includes the longest 40 videos in the validation set. Table 3.3 shows the different performance characteristics of the tracking algorithms. On the FM subset, our method outperforms SiamMask **?** with a relative gain of 21.48% in terms of AO. In terms of AO, our algorithm outperforms SiamMask by relative 18.42% and 24.03% on OC and CU subset, respectively. On the LO subset, the relative improvement of the AO score is 12.13% compared with SiamMask. This result shows that the global perception mechanism allows our tracker to reduce the cumulative error when tracking the long videos.

3.4.3 Evaluation on UAV20L Dataset

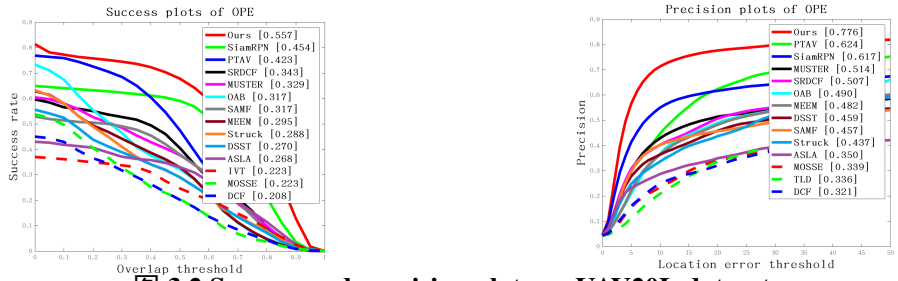

图 3.2 Success and precision plots on UAV20L dataset.

UAV20L **?** is an aerial video dataset captured from a low-altitude aerial perspective. Designed for long-term tracking, the UAV20L database has 20 videos with an average length of 2934 frames. Following the evaluation method of OTB50 **?**, we use precision and success to evaluate the performance of trackers on the UAV20L dataset. Precision refers to the distance from the center point of the predicted bounding box to the center point of the ground truth bounding box. Success refers to the intersection over union (IOU) of the predicted bounding box and the ground truth bounding box. In Fig. 3.2, the performance comparison of different trackers is visualized by precision plot and success plot.

The proposed method is compared against 13 recent trackers. Fig. 3.2 clearly shows that our algorithm outperforms all other trackers in terms of success and precision scores. Specifically, in the success plot, our tracker obtains a AUC score of 0.557. Compared with the state-of-art method PTAV **?** and SiamRPN **?**, the proposed tracker outperforms

these trackers by relative 31.7% and 22.7%. In the precision plot, the proposed algorithm obtains a score of 0.776. Compared with SiamRPN **?** and PTAV **?**, the proposed tracker outperforms these trackers by relative 25.8% and 24.4%.

3.5 CONCLUSION

In this paper, we propose a novel tracking architecture including the tracking component and the data-driven motion model. The global perception mechanism allows the tracking component to reduce the cumulative error during the tracking process. The tracking component uses a very deep network for two-stage tracking, which makes the tracker more discriminative. The motion model is trained end-to-end and is capable of learning the motion patterns of targets from large-scale trajectory datasets. Through the collaborative work of the tracking component and the motion model, the proposed method performs favorably against state-of-the-art trackers.

附录 A　中国科学院大学学位论文撰写要求

学位论文是研究生科研工作成果的集中体现，是评判学位申请者学术水平、授予其学位的主要依据，是科研领域重要的文献资料。根据《科学技术报告、学位论文和学术论文的编写格式》（GB/T 7713-1987）、《学位论文编写规则》（GB/T 7713.1-2006）和《文后参考文献著录规则》（GB7714—87）等国家有关标准，结合中国科学院大学（以下简称“国科大”）的实际情况，特制订本规定。

A.1　论文无附录者无需附录部分

A.2　测试公式编号 $\Lambda, \lambda, \theta, \bar{\Lambda}, \sqrt{S_{NN}}$

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}) = 0 \\ \frac{\partial (\rho \boldsymbol{V})}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}\boldsymbol{V}) = \nabla \cdot \boldsymbol{\sigma} \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E \boldsymbol{V}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\boldsymbol{\sigma} \cdot \boldsymbol{V}) \end{cases} \quad \ldots \text{ (A.1)}$$

$$\frac{\partial}{\partial t} \int_{\Omega} u \, \mathrm{d}\Omega + \int_{S} \mathbf{n} \cdot (u \boldsymbol{V}) \, \mathrm{d}S = \dot{\phi} \quad \ldots \text{ (A.2)}$$

$$\mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, \mathrm{d}t, \ \mathscr{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, \mathrm{d}t$$

$$\mathcal{F}\big(f(x + x_0)\big) = \mathcal{F}\big(f(x)\big) e^{2\pi i \xi x_0}, \ \mathscr{F}\big(f(x + x_0)\big) = \mathscr{F}\big(f(x)\big) e^{2\pi i \xi x_0}$$

mathtext: $A, F, L, 2, 3, 5, \sigma$, mathnormal: $A, F, L, 2, 3, 5, \sigma$, mathrm: $\mathrm{A, F, L, 2, 3, 5, \sigma}$.

mathbf: $\mathbf{A, F, L, 2, 3, 5, \sigma}$, mathit: $A, F, L, 2, 3, 5, \sigma$, mathsf: $\mathsf{A, F, L, 2, 3, 5, \sigma}$.

mathtt: $\mathtt{A, F, L, 2, 3, 5, \sigma}$, mathfrak: $\mathfrak{A, F, L, 2, 3, 5, \sigma}$, mathbb: $\mathbb{A, F, L, 2, 3, 5, \sigma}$.

mathcal: $\mathcal{A, F, L, \in, \ni, \bigtriangledown, \sigma}$, mathscr: $\mathscr{A, F, L, , , , \sigma}$, boldsymbol: $\boldsymbol{A, F, L, 2, 3, 5, \sigma}$.

vector: $\boldsymbol{\sigma, T, a, F, n}$, unitvector: $\boldsymbol{\sigma}, \mathbf{T, a, F, n}$

matrix: $\boldsymbol{\sigma}, \mathbf{T, a, F, n}$, unitmatrix: $\boldsymbol{\sigma}, \mathbf{T, a, F, n}$

tensor: $\boldsymbol{\sigma}, \mathbf{T, a, F, n}$, unittensor: $\boldsymbol{\sigma}, \mathbf{T, a, F, n}$

A.3 测试生僻字

霜蟾盥薇曜灵霜颸妙鬘虚霏淩澌菀枯菡萏沉寥窅冥毰毸濩落雩雩便嬛岧峣瀺灂姽婳愔嫕飒纚棽俪緸冤莩甲摛藻卮言倥侗椒觞期颐夜阑彬蔚倥偬澄廓簪缨陟遐迤逦缥缃鹣鲽憯懔闺闼璀错媕婀噌吰澒洞阛闠覼缕玓瓑逡巡諓諓琭琭瀌瀌踽踽叆叇氤氲瓠犀流眄蹀躞赟嬛茕頔璎珞螓首蘅皋惏悷缱绻昶皴皱颟顸愀然菡萏卑陬纯懿犇麤掱喤塨塈墎墪墐墒墒墓墔墕墖墘墖墚墛坠墝增墠墡墢墣墤墥墦墧墨墩墪樽墬墭堕墯墰墱墲坟墴墵垯墷墸墹墺墙墼墽垦墿壀壁壂壃壄壅壆坛壈壉壊垱壌壍埙壏壐壑壒压壔壕壖壗垒圹垆壛壜壝垄壠壡坜壣壤壥壦壧壨坝壪圭婔媹媺媻媼媽媾嫀嫃嫄嫅嫆嫇嫈嫋嫌嫍嫎嫏嫐嫑娇嫓嫔嫕嫖嫗嫘嫙嫚嫛嫜嫝嫞嫟嫠嫡嫢嫣嫤嫥嫦嫧嫨嫩嫪嫫嫬嫭嫮嫯嫰嫱嫲嫳嫴嫵嫶嫷嫸嫹嫺娴嫼嫽嫾婳妫嬁嬂嬃嬄嬅嬆嬇娆嬉嬊娇嬍嬎嬏嬐嬑嬒嬓嬔嬕嬖嬗嬘嫱嬚嬛嬜嬞嬟嬠嫒嬢嬣嬥嬦嬧嬨嬩嫔嬫嬬奶嬬嬮嬯婴嬱嬲嬳嬴嬵嬶嬷婶嬹嬺嬻嬼嬽嬾嬿孀孁孂娘孄孅孆孇孆孈孉孊娈孋孊孍孎孏嫫婿媚嵭嵮嵯嵰嵱嵲嵳嵴嵵嵶嵷嵸嵹嵺嵻嵼嵽嵾嵿嶀嶁嶂嶃崭嶅嶆岖嶈嶉嶊嶋嶌嶍嶎嶏嶐嶑嶒嶓嵚嶕嶖嶘嶙嶚嶛嶜嶝嶞嶟峤嶡峣嶣嶤嶥嶦峄峃嶩嶪嶫嶬嶭崄嶯嶰嶱嶲岛嶴嶵嶶嶷嶸嶹岭嶻屿岳帋巀巁巂巃巄巅巆巇巈巉巊岿巌巍巎巏巐巑峦巓巅巕岩巗巘巙巚帠帡帢帣帤帨帩帪帬帯帰帱帲帴帵帷帹帺帻帼帽帾帿幁幂帏幄幅幆幇幈幉幊幋幍幎幏幐微幒幓幖幙幚幛幜幝幞帜幠幡幢币幤幥幦幧幨幩幪幭幮幯幰庁庂広庅庇庈庉庋庌庍庎庘庒庖庘庛庝庠庡庢庣庤庥庨庩庪庫庬庮庯庰庱庲庳庴庵庹庺庻庼庽庿廀厕廃厩廅廆廇廈廋廌廍庼廏廐廑廒廔廕廖廗廘廙廛廜廞庑廤廥廦廧廨廭廮廯廰痈廲巡廸廹廻廼廽廿弁弅弆弇弉弖弙弚弜弝弞弡弢弣弤弨弩弪弫弬弭弮弰弲弪弸弻弽弾弿彁彂彃彄彅彇彉彋彍彎彐彑彔彗彚彛彜彝彞彟彠彣彤彦彧彨彩彪彫彬彭彮彯彳彴彵彶彷彸役彺彻彽彾佛徂徃徆徇徉后徍徎徏徐徒従徔徕徖徙徚徛徜徝从徟徕御徢徣徤徥徦徧徨复循徫旁徭微徯徰徱徲徳徵徶徹德徸彻徺徻徼忀忁忂惗愔忇忈忉忊忋忎忓忔忕忚忛応忝忞忟忪挣挦挧挨挩挪挫挬挭挮挰挱挲挳挴挵挶挷挸挹挺挻挼挿捀捁捂捃捄捅捆捇捈捊捋捌捍捎捏捐捑捒捓捔捕捖捗捘捙捚捛捜捝捞损捠捡换捣捤捥捦捧捨捩捪捫捬捭据捯捰捱捲捳捴捵捶捷捸捹捺捻捼捽捾捿掀掁掂掃掄掅掆掇授掉掊掋掍掎掏掐掑排掓掔掕掖掘掗掙掚挂掜掝掞掟掠采探掣掤掦措掫掬掭掮掯掰掱掲掳掴掵掶掸掹掺掻掼掽掾掿拣揁揂揃揅揄揆揇揈揉揊揋揌揍揎揑揓揔揕揖揗揘揙揜揝揞揟揠揢揤揥揦揧揨揫捂揯揰揱揲揳援揵揶揷揸揻揼揾揿搀搁搂搃搄搅搇搈搉搊搋搌搎搏搐搑搒搓搕搖搗搘搙搚搛搝搞搟搠搡搢搣搤捶搧搨搩搪搫搬搮搯搰搱搲搳搴搵搷搸搹搻搼搽榨搿摂摃摄摅摆摇摈摊摋摌摍摎摏摐摑摒摓摔摕摖摗摙摚摛掼摝摞摠摡斫斩斮斱斲斳斴斵斶斸旪旫旮旯晒晓晔晕晖晗晘晙晛晜晞晟晠晡晰晣晤晥晦晧晪晫晬晭晰晱晲晳晴晵晷晸晹晻晼晽晾晿暀暁暂暃暄暅暆暇晕晖暊暋暌暍暎暏暐暑暒暓暔暕暖暗旸暙暚暛暜暝暞暟暠暡暣暤暥暦暧暨暩暪暬暭暮暯暰昵暲暳暴暵

参考文献

作者简历及攻读学位期间发表的学术论文与研究成果

本科生无需此部分。

作者简历

casthesis作者

吴凌云，福建省屏南县人，中国科学院数学与系统科学研究院博士研究生。

ucasthesis作者

莫晃锐，湖南省湘潭县人，中国科学院力学研究所硕士研究生。

已发表(或正式接受)的学术论文:

1. ucasthesis: A LaTeX Thesis Template for the University of Chinese Academy of Sciences, 2014.

申请或已获得的专利:

(无专利时此项不必列出)

参加的研究项目及获奖情况:

可以随意添加新的条目或是结构。

致　谢

感激casthesis作者吴凌云学长，gbt7714-bibtex-style开发者zepinglee，和ctex众多开发者们。若没有他们的辛勤付出和非凡工作，LaTeX菜鸟的我是无法完成此国科大学位论文LaTeX模板ucasthesis的。在LaTeX中的一点一滴的成长源于开源社区的众多优秀资料和教程，在此对所有LaTeX社区的贡献者表示感谢！

ucasthesis国科大学位论文LaTeX模板的最终成型离不开以霍明虹老师和丁云云老师为代表的国科大学位办公室老师们制定的官方指导文件和众多ucasthesis用户的热心测试和耐心反馈，在此对他们的认真付出表示感谢。特别对国科大的赵永明同学的众多有效反馈意见和建议表示感谢，对国科大本科部的陆晴老师和本科部学位办的丁云云老师的细致审核和建议表示感谢。谢谢大家的共同努力和支持，让ucasthesis为国科大学子使用LaTeX撰写学位论文提供便利和高效这一目标成为可能。