UNIVERSITY OF SCIENCE
THE FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER VISION

NGUYEN ANH PHA

MULTIPLE PEDESTRIAN TRACKING WITH SIAMESE TRACKER

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

HO CHI MINH CITY, 2020

UNIVERSITY OF SCIENCE
THE FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER VISION

NGUYEN ANH PHA - 1612485

MULTIPLE PEDESTRIAN TRACKING WITH SIAMESE TRACKER

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THESIS ADVISOR:
DR. TRAN THAI SON

HO CHI MINH CITY, 2020

ACKNOWLEDGMENTS

First and foremost, I would like to show my great gratitude to the support of my lecturers, mentors, friends from The Faculty of Information Technology, University of Science.

I would like to deeply thank my advisor Dr. Tran Thai Son for giving me great guidance and support throughout my time doing this thesis. Dr. Tran Thai Son has been an awesome advisor who always encourages and motivates me to go beyond my limit to make better contributions to my research field. Dr. Tran Thai Son was also the one who raised the love in me for Computer Vision at the first time I took his course. Without Dr. Tran Thai Son, I would probably go in a completely different research direction.

I am also greatly thankful to my personal mentor - Dr. Thi Hue Tuan from University of New South Wales for giving me great advice for research directions. I have learned a lot from his evaluation. I have to stress that Dr. Thi Hue Tuan is more than an academic mentor to me; he is also a friend who supports me mentally in many aspects of life.

Nguyen Anh Pha

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In recent years, artificial intelligence is considered the world's utmost care domain. With the strong support of hardware devices, AI application is spreading within a variety of areas from economy, service, hospital, medicine, retail, just to name a few. Some of the most interesting issues to the retailer is the traffic counting problem and behavior analysis of in-store customers problem. Most entrepreneurs are in favor of amending the quality of their service by identifying as well as controlling the waiting time. By applying AI technology to analyze traffic, enterprises are expecting the method to enhance their service quality in a reasonably-priced and painless way, compared to the manual method of conducting traditional research. This dissertation concerns the tracking task, especially multiple pedestrian tracking, which is an important task to solve above key problems.

The proposed tracking system at the retailing sites is based on the combination of two stages. The first one relates to Siamese Tracker which is a promising method used for object tracking. This stage is based on the Siamese Network core. The latter stage is human detection that is simultaneously associated with the first one by a matching algorithm. The system is able to identify customers and track their target.

The whole system can be considered to break into three phases: initialization, tracking, and correction. At the initialization phase, the human detector detects the targets and generates its bounding boxes. In the tracking phase, the Siamese Tracker then will collect these bounding boxes and track itself by using its proposal generation framework. It works without any outside interference. And the correction phase is similar to the initialization phase but provides an associating strategy to combine the human detector prediction. Adhere to the given distance, it is possible to entitle the targets by the Hungarian assignment method. After matching, an update function is used to adapt the new human appearances with post-features. Finally, all pedestrians are group by IDs, it can be used to analyze its behaviors, count the lifespan of the tracks, also the number of customers,... which are very meaningful for customer

analytics in the retail industry.

My strategy outperforms state-of-the-art tracking algorithms on a busy town center street dataset [1] and our retail scene dataset, which is recorded at retail shopping sites and focuses on heavily occluded humans and unpredictable movement trajectories.

CHAPTER 1

INTRODUCTION

In this chapter, I briefly introduce the content of the thesis topic, the importance of the tracking problem, especially human tracking, in Computer Vision through applications. Furthermore, several types of approach and the Multiple Object Tracking methodology are also mentioned.

1.1 Overview

The development of the economy and the service sector, in particular, has sparked the imperative requirement for enhancing service quality among the companies and organizations. Applying high-tech devices or software are dominant to the enterprises with the aim of improving their service quality, reassuring the company's customer value proposition. Nowadays, the service firms, specifically the retailer, are struggling with the effectiveness measurement of sales, how much time do customers pay attention to their advertisements or products. An example, the waiting time, according to [21], plays a vital role by time in most services and is recommended to be paid more attention to improve the understanding of how customers perceive, budget, consume and value time. However, the number of researches on customer behavior in terms of waiting time is limited and scarce to Vietnam enterprises. This is because the manual implementation of traditional research on such a realm is costly and complicated. Hence, the application of AI to identify customer waiting time is in the hope of providing organizations with an appropriate assessment of their service quality, supporting the Sale-Marketing department with the data on their customer and consumer behavior and driving the decision-maker to the right selections. The whole process is believed to be reasonably-priced and painless, compared to traditional and manual methods. Using the surveillance camera system, the retailer can integrate an AI system to analyze behavior of customers at an affordable cost. This research work

studies on the scope of Deep Learning, structure and activity of Convolutional Neural Network and Siamese Network. They are the elemental knowledge to implement Siamese Tracking algorithm. The method will be principally used to accomplish the system.

1.2 The Visual Tracking Problem

Generic visual tracking aims to estimate the trajectory of an unknown visual target when only an initial state of the target (in a video frame) is available. Visual tracking is an open and attractive research field (see Fig. 1.1) with a broad extent of categories and applications; including self-driving cars, autonomous robots, surveillance, augmented reality, unmanned aerial vehicle (UAV) tracking, sports, surgery, biology, ocean exploration, to name a few.

The ill-posed definition of the visual tracking (i.e., model-free tracking, on-the-fly learning, single-camera, 2D information) is more challenging in complicated real-world scenarios which may include arbitrary classes of target appearance and their motion model (e.g., human, drone, animal, vehicle), different imaging characteristics (e.g., static/moving camera, smooth/abrupt movement, camera resolution), and changes in environmental conditions (e.g., illumination variation, background clutter, crowded scenes). Although traditional visual tracking methods utilize various frameworks – like discriminative correlation filters (DCF), silhouette tracking, Kernel tracking, point tracking, and so forth – these methods cannot provide satisfactory results in unconstrained environments. The main reasons are the target representation by handcrafted features (such as the histogram of oriented gradients (HOG) [7] and Color-Names (CN)) [30] and inflexible target modeling.

Inspired by deep learning (DL) breakthroughs [14, 26, 29, 8] in ImageNet large scale visual recognition competition (ILSVRC) [25] and also visual object tracking (VOT)

[1] Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]

Figure 1.1: An overview of visual target tracking. [1]

challenge [13, 12], DL-based methods have attracted considerable interest in visual tracking community to provide robust visual trackers. Although convolutional neural networks (CNNs) have been dominant networks initially, the broad range of architectures such as Siamese neural networks (SNNs), recurrent neural networks (RNNs), auto-encoders (AEs), generative adversarial networks (GANs), and custom neural networks are currently investigated. Fig. 1.2 presents a brief history of the development of deep visual trackers in recent years. The state-of-the-art DL-based vi-

sual trackers have distinct characteristics such as exploitation of deep architecture, backbone network, learning procedure, training datasets, network objective, network output, types of exploited deep features, CPU/GPU implementation, programming language and framework, speed, and so forth. Besides, several visual tracking benchmark datasets have been proposed in the past few years for practical training and evaluating of visual tracking methods. Despite various properties, some of these benchmark datasets have common video sequences. Thus, a comparative study of DL-based methods, their benchmark datasets, and evaluation metrics are provided in this chapter to facilitate developing advanced methods by the visual tracking community.

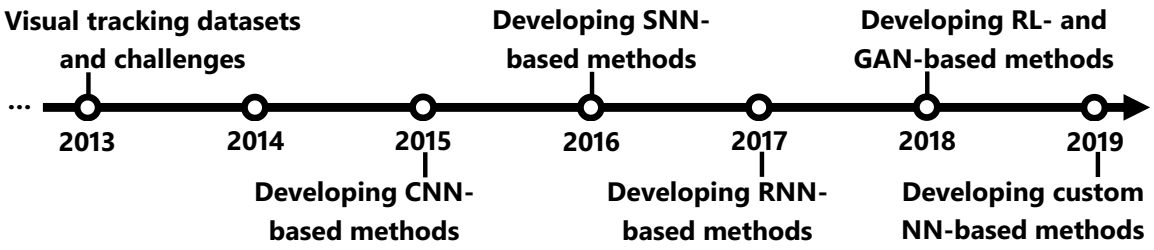

Figure 1.2: Timeline of deep visual tracking methods. [2]

The main points of this chapter are summarized as follows:

- State-of-the-art DL-based visual tracking methods are categorized based on their architecture (i.e., CNN, SNN, RNN, GAN, and custom networks), network exploitation (i.e., off-the-shelf deep features and deep features for visual track-

[2]Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]
2015: Exploring and studying deep features to exploit the traditional methods.
2016: Offline training/fine-tuning of DNNs for visual tracking purpose, Employing Siamese network for real-time tracking, Integrating DNNs into traditional frameworks.
2017: Incorporating temporal and contextual information, Investigating various offline training on large-scale image/video datasets.
2018: Studying different learning and search strategies, Designing more sophisticated architectures for visual tracking task.
2019: Investigating deep detection and segmentation approaches for visual tracking, Taking advantages of deeper backbone networks.

ing), network training for visual tracking (i.e., only offline training, only online training, both offline and online training), network objective (i.e., regression-based, classification-based, and both classification and regression-based), and exploitation of correlation filter advantages (i.e., DCF framework and utilizing correlation filter/layer/functions). Such a study covering all of these aspects in detailed categorization of visual tracking methods has not been previously presented.

- The main motivations and contributions of the DL-based methods to tackle the visual tracking problems are summarized. This classification provides a proper insight in designing accurate and robust DL-based visual tracking methods.

Also, the following observations are made:

- The SNN-based methods are the most attractive deep architectures due to their satisfactory balance between performance and efficiency for visual tracking. Moreover, the visual tracking methods recently attempt to exploit the advantages of RL and GAN methods to refine their decision making and alleviate the lack of training data. Based on these advantages, the recent visual tracking methods aim to design custom neural networks for visual tracking purposes.

- The offline end-to-end learning of deep features appropriately adapts the pre-trained features for visual tracking. Although the online training of DNN increases the computational complexity such that most of these methods are not suitable for real-time applications, it considerably helps visual trackers to adapt with significant appearance variation, prevent from visual distractors, and improve the accuracy and robustness of visual tracking methods. Hence, exploiting both offline and online training procedures provides more robust visual trackers.

- Leveraging deeper and wider backbone networks improves the discriminative power of distinguishing the target from its background.

- The best visual tracking methods use both regression and classification objective functions not only to estimate the best target proposal but also to find the tightest BB for target localization.

- The exploitation of different features enhances the robustness of the target model. For instance, most of the DCF-based methods fuse the deep off-the-shelf features and hand-crafted features (e.g., HOG and CN) for this reason. Also, the exploitation of complementary features such as temporal or contextual information has led to more discriminative and robust features for target representation.

- The most challenging attributes for DL-based visual tracking methods are occlusion, out-of-view, and fast motion. Moreover, visual distractors with similar semantics may result in drifting problem.

The rest of this chapter is as follows. Section 3 introduces the taxonomy of deep visual tracking methods. The Multiple Object Tracking (MOT) methodology - the main topic of this thesis is briefly described in Section 4.

1.3 Taxonomy of Deep Visual Tracking Methods

In this section, three major components of: target representation/information, training process, and learning procedure are described. Then, the proposed comprehensive taxonomy of DL-based methods is presented.

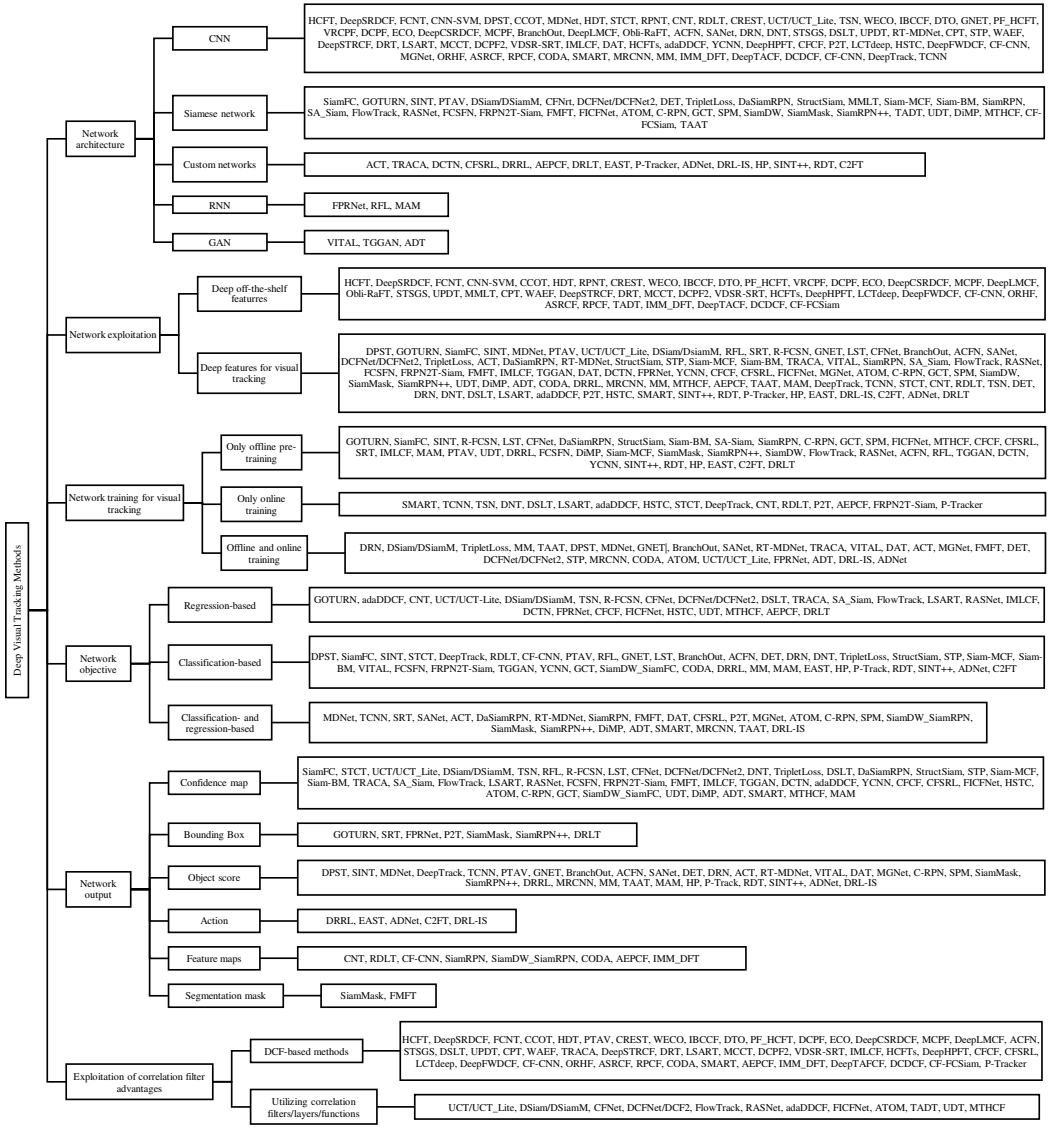

Figure 1.3: Taxonomy of DL-based visual tracking methods. [3]

One of the primary motivations of DL-based methods is improving a target representation by utilizing/fusing deep hierarchical features, exploiting contextual information or motion information, and selecting more discriminative and robust deep features. Also, the DL-based methods aim to train DNNs for visual tracking systems, effectively. Their general motivations can be classified into either employing different network training (e.g., network pre-training, online training, or both) or handling some training problems (e.g., lacking training data, over-fitting on training data, and computational complexity). Unsupervised training is another recent scheme to use abundant unlabeled samples, which can be performed by clustering these samples according to contextual information, mapping training data to a manifold space, or exploiting consistency-based objective function. At last, the main motivations of DL-based trackers according to their learning procedures are classified to online update schemes, aspect ratio estimation, scale estimation, search strategies, and a providing long-term memory.

In the following, DL-based visual tracking methods are comprehensively categorized based on six main aspects of network architecture, network exploitation, network training for visual tracking purposes, network objective, network output, and exploitation of correlation filter advantages. The proposed taxonomy of DL-based visual tracking methods is shown in Fig. 1.3. Furthermore, other important details including the pre-trained networks, backbone networks, exploited layers, type of deep features, the fusion of hand-crafted and deep features, training datasets, tracking output, tracking speed, hardware implementation details, programming language, and DL framework will be presented in this section. In this section, not only the state-of-the-art DL-based visual tracking methods are categorized, but also the main motivations and contributions of those methods are classified that can provide helpful perspectives to identify future directions.

[3]Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]

1.3.1 Network Architecture

Although CNNs have been used in extensive DL-based methods, other architectures also have been mainly developed to improve the efficiency and robustness of visual trackers in recent years. According to the extent of techniques based on various deep structures, the taxonomy consists of the CNN-based, SNN-based, GAN-based, RNN-based, and custom network-based methods.

1.3.1.1 Convolutional Neural Network (CNN)

Motivated by CNN breakthroughs in computer vision tasks and some attractive advantages such as parameter sharing, sparse interactions, and dominant representations, a wide range of methods utilize CNNs for visual tracking. The CNN-based visual trackers are mainly classified according to the following motivations.

- **Robust target representation:** Providing a powerful target representation is the main advantage of employing CNNs for visual tracking. To achieve the goal of learning generic representations for target modeling and constructing a more robust target models, the main contributions of methods are classified into:

  i) offline training of CNNs on large-scale datasets for visual tracking,

  ii) designing specific deep convolutional networks instead of employing pre-trained models,

  iii) constructing multiple target models to capture variety of target appearances,

  iv) incorporating spatial and temporal information to improve model generalization,

  v) fusion of different deep features to exploit complementary spatial and semantic information,

  vi) learning different target models such as relative model or part-based models to handle partial occlusion and deformation, and

vii) utilizing two-stream network to prevent from over-fitting and learn rotation information.

- **Balancing training data:** According to the definition of visual tracking, there is just one positive sample in the first frame that increases the risk of over-fitting. Although the background information arbitrary can be considered as negative ones in each frame, target sampling based on imperfect target estimations may also lead to noisy/unreliable training samples. These problems dramatically affect the performance of visual tracking methods. To alleviate them, CNN-based methods propose:

  i) domain adaption mechanism (i.e., transferring learned knowledge from source domain to target domain with insufficient samples),

  ii) various update mechanisms (e.g., periodic, stochastic, short-term, and long-term updates),

  iii) convolutional Fisher discriminative analysis (FDA) for positive and negative sample mining,

  iv) multiple-branches CNN for online ensemble learning, and

  v) efficient sampling strategies to increase the number of training samples.

- **Computational complexity problem:** Despite significant progress of CNNs in terms of target estimation accuracy, the CNN-based methods still suffer from high computational complexity. To reduce this limitation, CNN-based visual tracking methods exploit different solutions namely:

  i) disassembling a CNN into several shrunken networks,

  ii) compressing or pruning training sample space or feature selection,

  iii) feature computation via RoIAlign operation (i.e., feature approximation via bilinear interpolation) or oblique random forest for better data capturing,

  iv) corrective domain adaption method,

v) lightweight structure,

vi) efficient optimization processes,

vii) exploiting advantages of correlation filters for efficient computations,

viii) particle sampling strategy, and

ix) utilizing attentional mechanism.

1.3.1.2 Siamese Neural Network (SNN)

To learn similarity knowledge and achieve real-time speed, SNNs are widely employed for visual tracking purposes in the past few years. Given the pairs of target and search regions, these twin networks compute the same function to produce a similarity map. The common aim of SNN-based methods is to overcome the limitations of pre-trained deep CNNs and take full advantage of end-to-end learning for real-time applications.

- **Discriminative target representation:** The ability of visual tracker to construct a robust target model majorly relies on target representation. For achieving more discriminative deep features and improving target modeling, SNN-based methods propose:

  i) learning distractor-aware or target-aware features,

  ii) fusing deep multi-level features or combining confidence maps,

  iii) utilizing different loss functions in Siamese formulation to train more effective filters,

  iv) leveraging different types of deep features such as context information or temporal features/models,

  v) full exploring of low-level spatial features,

  vi) considering angle estimation of target to prevent from salient background objects,

vii) utilizing multi-stage regression to refine target representation, and

vii) using deeper and wider deep network as the backbone to increase receptive field of neurons which is equivalent to capturing the structure of the target.

- **Adapting target appearance variation:** Using only offline training of the first generation of SNN-based methods caused a poor generalization of these methods to adapt to target appearance variations. To solve it, recent SNN-based methods have proposed:

  i) online update strategies,

  ii) background suppression,

  iii) formulating tracking task as a one-shot local detection task, and

  iv) giving higher weights to important feature channels or score maps.

  Alternatively, the DaSiamRPN [37] and MMLT [16] use a local-to-global search region strategy and memory exploitation to handle critical challenges such as full occlusion and out-of-view and enhance local search strategy, respectively.

- **Balancing training data:** As a same problem for the CNN-based methods, some efforts by SNN-based methods have been performed to address imbalance distribution of training samples. The main contributions of the SNN-based methods are:

  i) exploiting multi-stage Siamese framework to stimulate hard negative sampling,

  ii) adopting sampling heuristic such as fixed foreground-to-background ratio or sampling strategies such as random sampling or flow-guided sampling, and

  iii) taking advantages of correlation filter/layer into Siamese framework.

1.3.2 Network Training

The state-of-the-art DL-based visual tracking methods mostly exploit end-to-end learning with train/re-train a DNN by applying gradient-based optimization algorithms. However, these methods have differences according to their offline network training, online fine-tuning, computational complexity, dealing with lack of training data, addressing overfitting problem, and exploiting unlabeled samples by unsupervised training. In this survey, DL-based methods are categorized into only offline pre-training, only online training, and both offline and online training for visual tracking purposes. The training details of these methods are shown in Table 1.1 to Table 1.3.

1.3.2.1 Only Offline Training

Most of the DL-based visual tracking methods only pre-train their networks to provide a generic target representation and reduce the high risk of over-fitting due to imbalanced training data and fixed assumptions. To adjust the learned filter weights for visual tracking task, the specialized networks are trained on large-scale data to not only exploit better representation but also achieve acceptable tracking speed by preventing from training during visual tracking (see Table 1.1).

Table 1.1: Only offline training for visual tracking. The abbreviations are denoted as: confidence map (CM), saliency map (SM), bounding box (BB), object score (OS), feature maps (FM), segmentation mask (SGM), rotated bounding box (RBB), action (AC), deep appearance features (DAF), deep motion features (DMF). [4]

Method	Backbone network	Offline training dataset(s)	Exploited features	PC (CPU, RAM, Nvidia GPU)	Language	Framework	Speed (fps)	Tracking output
GOTURN	AlexNet	ILSVRC-DET, ALOV	DAF	N/A, GTX Titan X GPU	C/C++	Caffe	166	BB
SiamFC	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel I7-4790K 4.00GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	58	CM
SINT	AlexNet, VGG-16	ImageNet, ALOV	DAF	N/A	Matlab	Caffe	N/A	OS
R-FCSN	AlexNet	ImageNet, ILSVRC-VID	DAF	N/A, GTX Titan X GPU	Matlab	MatConvNet	50.25	CM
LST	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel Xeon 3.50GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	24	CM
CFNet	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel I7 4.00GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	75	CM
DaSiamRPN	AlexNet	ILSVRC, YTBB, Augmented ILSVRC-DET, Augmented MSCOCO-DET	DAF	Intel I7 CPU, 48GB RAM, GTX Titan X GPU	Python	PyTorch	160	CM
StructSiam	AlexNet	ILSVRC-VID, ALOV	DAF	Intel I7-4790 3.60GHz, GTX 1080 GPU	Python	TensorFlow	45	CM
Siam-BM	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel Xeon 2.60GHz CPU, Tesla P100 GPU	Python	TensorFlow	48	CM
SA-Siam	AlexNet	ImageNet, TC128, ILSVRC-VID	DAF	Intel Xeon 2.40GHz CPU, GTX Titan X GPU	Python	TensorFlow	50	CM
SiamRPN	AlexNet	ILSVRC, YTBB	DAF	Intel I7 CPU, 12GB RAM, GTX 1060 GPU	Python	PyTorch	160	FM
C-RPN	AlexNet	ImageNet, ILSVRC-VID, YTBB	DAF	N/A, GTX 1080 GPU	Matlab	MatConvNet	36	CM
GCT	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel Xeon 3.00GHz CPU, 256GB RAM, GTX 1080Ti GPU	Python	TensorFlow	49.8	CM
SPM	AlexNet, SiameseRPN, RelationNet	ImageNet, ILSVRC-VID, YTBB, ILSVRC-DET, MSCOCO, CityPerson, WiderFace	DAF	N/A, P100 GPU	N/A	N/A	120	OS
FICFNet	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel I7 4.00GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	28	CM
MTHCF	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel 6700 3.40GHz CPU, GTX Titan GPU	Matlab	MatConvNet	33	CM
HP	AlexNet	ImageNet, ILSVRC-VID	DAF	N/A	Python	Keras	69	CM
EAST	AlexNet	ImageNet, ILSVRC-VID	DAF	Intel I7 4.00GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	23.2	AC
CFCF	VGG-M	ImageNet, ILSVRC-VID, VOT2015	HOG, DAF	Intel Xeon 3.00GHz CPU, Tesla K40 GPU	Matlab	MatConvNet	1.7	CM
CFSRL	VGG-M	ILSVRC-VID	DAF	Intel Xeon 2.40GHz CPU, 32GB RAM, GTX Titan X GPU	Matlab, Python	PyTorch	N/A	CM
C2FT	VGG-M	ImageNet, N/A	DAF	Intel Xeon 2.60GHz CPU, GTX 1080Ti GPU	N/A	N/A	N/A	AC
SRT	VGG-16	ImageNet, ALOV, SOT	DAF	N/A	N/A	N/A	N/A	BB
IMLCF	VGG-16	ImageNet, ILSVRC-VID	DAF	Intel 1.40GHz CPU, GTX 1080Ti GPU	Matlab	MatConvNet	N/A	CM
SINT++	VGG-16	ImageNet, OTB2013, OTB2015, VOT2014	DAF	Intel I7-6700K CPU, 32GB RAM, GTX 1080 GPU	Python	Caffe, Keras	N/A	AC
MAM	VGG-16, Faster-RCNN	ImageNet, PASCAL VOC 2007, OTB100, TC128	DAF	3.40GHz CPU, Titan GPU	Matlab	Caffe	3	SM
PTAV	VGGNet	ALOV	HOG, DAF	N/A, GTX Titan Z GPU	C/C++	Caffe	27	CM
UDT	VGGNet	ILSVRC	DAF	Intel I7-4790K 4.00GHz, GTX 1080Ti GPU	Matlab	MatConvNet	55	CM
DRRL	VGGNet	ImageNet, VOT2016	DAF	N/A, GTX 1060 GPU	Python	TensorFlow	6.3	OS
FCSFN	VGG-19	ImageNet, ALOV	DAF	N/A	N/A	N/A	N/A	CM
DiMP	ResNet-18, ResNet-50	ImageNet, TrackingNet, LaSOT, GOT10k, MSCOCO	DAF	N/A, GTX 1080 GPU	Python	PyTorch	43 57	OS
Siam-MCF	ResNet-50	ImageNet, ILSVRC-VID	DAF	Intel Xeon E5 CPU, GTX 1080Ti GPU	Python	TensorFlow	20	CM
SiamMask	ResNet-50	ImageNet, MSCOCO, ILSVRC-VID, YouTube-VOS	DAF	N/A, RTX 2080 GPU	Python	PyTorch	55 60	SGM, RBB
SiamRPN++	ResNet-50	ImageNet, MSCOCO, ILSVRC-DET, ILSVRC-VID, YTBB	DAF	N/A, Titan Xp Pascal GPU	Python	PyTorch	35	OS, BB
SiamDW	ResNet, ResNeXt, Inception	ImageNet, ILSVRC-VID, YTBB	DAF	Intel Xeon 2.40GHz CPU, GTX 1080 GPU	Python	PyTorch	13 93	CM, FM
FlowTrack	FlowNet	Flying chairs, Middlebur, KITTI, Sintel, ILSVRC-VID	DAF, DMF	Intel I7-6700 CPU, 48GB RAM, GTX Titan X GPU	Matlab	MatConvNet	12	CM
RASNet	Attention networks	ILSVRC-DET	DAF	Intel Xeon 2.20GHz CPU, Titan Xp Pascal GPU	Matlab	MatConvNet	83	CM
ACFN	Attentional correlation filter network	OTB2013, OTB2015, VOT2014, VOT2015	HOG, Color, DAF	Intel I7-6900K 3.20GHz CPU, 32GB RAM, GTX 1070 GPU	Matlab, Python	MatConvNet, TensorFlow	15	OS
RFL	Convolutional LSTM	ILSVRC-VID	DAF	Intel I7-6700 3.40GHz CPU, GTX 1080 GPU	Python	TensorFlow	15	CM
DRLT	YOLO	ImageNet, PASCAL VOC	DAF	N/A, GTX 1080 GPU	Python	TensorFlow	45	BB
TGGAN	-	ALOV, VOT2015	DAF	N/A, GTX Titan X GPU	Python	Keras	3.1	CM
DCTN	-	TC128, NUS-PRO, MOT2015	DAF, DMF	N/A	N/A	N/A	27	CM
YCNN	-	ImageNet, ALOV300++	DAF	N/A, Tesla K40c GPU	Python	TensorFlow	45	CM
RDT	-	VOT2015	DAF	Intel I7-4790K 4.00GHz, 24GB RAM, GTX Titan X GPU	Python	TensorFlow	43	CM, OS

[4] Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]

Table 1.2: Only online training for visual tracking. The abbreviations are denoted as: confidence map (CM), bounding box (BB), object score (OS), deep appearance features (DAF), deep motion features (DMF). [5]

Method	Backbone network	Exploited features	Strategy to alleviate the over-fitting problem	PC (CPU, RAM, Nvidia GPU)	Language	Framework	Speed (fps)	Tracking output
SMART	ZFNet	DAF	Set the learning rates in conv1-conv3 to zero	Intel 3.10GHz CPU, 256 GB RAM, GTX Titan X GPU	Matlab	Caffe	27	CM
TCNN	VGG-M	DAF	Only update fully-connected layers	Intel I7-5820K 3.30GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	1.5	OS
TSN	VGG-16	DAF	Coarse-to-fine framework	N/A, GTX 980Ti GPU	Matlab	MatConvNet	1	CM
DNT	VGG-16	DAF	Set uniform weight decay in the objective functions	3.40GHz CPU, GTX Titan GPU	Matlab	Caffe	5	CM
DSLT	VGG-16	DAF	Use seven last frames for model update	Intel I7 4.00GHz CPU, GTX Titan X GPU	Matlab	Caffe	5.7	CM
LSART	VGG-16	DAF	Two-stream training network to learn network parameters	Intel 4.00GHz CPU, 32GB RAM, GTX Titan X GPU	Matlab	Caffe	1	CM
adaDDCF	VGG-16	DAF	Regularization item for training of each layer	3.40GHz CPU, Tesla K40 GPU	Matlab	MatConvNet	9	CM
HSTC	VGG-16	DAF	Dropout layer and convolutional with the mask layer	Intel Xeon 2.10GHz CPU, GTX 1080 GPU	Matlab	Caffe	2.1	CM
P-Track	VGG-16	DAF	Learning policy for update and re-initialization	N/A, Tesla K40 GPU	N/A	N/A	10	CM
STCT	Custom	DAF	Sequential training method	3.40GHz CPU, GTX Titan GPU	Matlab	Caffe	2.5	CM
DeepTrack	Custom	DAF	Temporal sampling mechanism for the batch generation in SGD algorithm	Quad-core CPU, GTX 980 GPU	Matlab	N/A	2.5	OS
CNT	Custom	DAF	Incremental update scheme	Intel I7-3770 3.40GHz CPU, GPU	Matlab	N/A	5	BB
RDLT	Custom	DAF	Build relationship between the stable factor and iteration number	Intel I7 2.20GHz CPU	Matlab	N/A	5	CM
P2T	Custom	DAF	Generate large scale of part pairs in each mini-batch	Intel I7-4790 3.60GHz CPU, 32GB RAM, GTX 980 GPU	Matlab	Caffe	2	BB
AEPCF	Custom	DAF	Select a proper learning rate	Intel I7 3.40GHz, 32GB RAM, GPU	N/A	N/A	4.15	CM
FRPN2T-Siam	Custom	DAF	Only update fully-connected layers	N/A	Matlab	Caffe	N/A	CM

[5] Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]

Table 1.3: Both offline and online training for visual tracking. The abbreviations are denoted as: confidence map (CM), bounding box (BB), object score (OS), voting map (VM), action (AC), deep appearance features (DAF), deep motion features (DMF), compressed deep appearance features (CDAF). [6]

Method	Backbone network	Offline training(s)	Online network training	Exploited features	PC (CPU, RAM, Nvidia GPU)	Language	Framework	Speed (fps)	Tracking output
DRN	AlexNet	ImageNet	Yes	DAF	N/A, K20 GPU	Matlab	Caffe	1.3	CM
DSiam/DSiamM	AlexNet, VGG-19	ImageNet, ILSVRC-VID	Yes	DAF	N/A, GTX Titan X GPU	Matlab	MatConvNet	45	CM
TripletLoss	AlexNet	ImageNet, ILSVRC-VID, ILSVRC	Dependent	DAF	Intel I7-6700 3.40GHz CPU, GTX 1080Ti GPU	Matlab	MatConvNet	55 86	CM
MM	AlexNet	ImageNet, OTB2015, ILSVRC	Yes	DAF	Intel I7-6700 4.00GHz CPU, 16GB RAM, GTX 1060 GPU	Matlab	MatConvNet	1.2	OS
TAAT	AlexNet, VGGNet, ResNet	ImageNet, ALOV, ILSVRC-VID	Yes	DAF	Intel Xeon 1.60GHz CPU, 16GB RAM, GTX Titan X GPU	Matlab	Caffe	15	BB
DPST	VGG-M	ImageNet, ILSVRC-VID, ALOV	Only on the first frame	DAF	Intel I7 3.60GHz CPU, GTX 1080Ti GPU	Matlab	MatConvNet	1	OS
MDNet	VGG-M	ImageNet, OTB2015, ILSVRC-VID	Yes	DAF	Intel Xeon 2.20GHz CPU, Tesla K20m GPU	Matlab	MatConvNet	1	OS
GNet	VGG-M	ImageNet, VOT	Yes	DAF	Intel Xeon 2.66GHz CPU, Tesla K40 GPU	Matlab	MatConvNet	1	OS
BranchOut	VGG-M	ImageNet, OTB2015, ILSVRC	Yes	DAF	N/A	Matlab	MatConvNet	N/A	OS
SANet	VGG-M	ImageNet, OTB2015, ILSVRC	Yes	DAF	Intel I7 3.70GHz CPU, GTX Titan Z GPU	Matlab	MatConvNet	1	OS
RT-MDNet	VGG-M	ImageNet, ILSVRC-VID	Yes	DAF	Intel I7-6850K 3.60GHz, Titan Xp Pascal GPU	Python	PyTorch	46	OS
TRACA	VGG-M	ImageNet, PASCAL VOC	Yes	CDAF	Intel I7-2700K 3.50GHz CPU, 16GB RAM, GTX 1080 GPU	Matlab	MatConvNet	101.3	CM
VITAL	VGG-M	ImageNet, OTB2015, ILSVRC	Yes	DAF	Intel I7 3.60GHz CPU, Tesla K40c GPU	Matlab	MatConvNet	1.5	OS
DAT	VGG-M	ImageNet, OTB2015, ILSVRC	Yes	DAF	Intel I7-3.40GHz CPU, GTX 1080 GPU	Python	PyTorch	1	TP
ACT	VGG-M	ImageNet-Video, ILSVRC	Yes	DAF	3.40GHz CPU, 32GB RAM, GTX Titan GPU	Python	PyTorch	30	OS
MGNet	VGG-M	ImageNet, OTB2015, ILSVRC	Yes	DAF, DMF	Intel I7-5930K 3.50GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	2	OS
DRL-IS	VGG-M	ImageNet, VOT2013 2015	Yes	DAF	Intel I7 3.40GHz CPU, 24GB RAM, GTX 1080Ti GPU	Python	PyTorch	10.2	AC
ADNet	VGG-M	ImageNet, VOT2013 2015, ALOV	Yes	DAF	Intel I7-4790K, 32GB RAM, GTX Titan X GPU	Matlab	MatConvNet	15	AC, OS
FMFT	VGG-16	ImageNet	Yes	DAF	Intel Xeon 3.50GHz CPU, GTX Titan X GPU	Matlab	MatConvNet	N/A	CM
DET	VGG-16	ImageNet, ALOV, VOT2014, VOT2015	Yes	DAF	Intel I7-4790 3.60GHz CPU, GTX Titan X GPU	Python	Keras	3.4	OS
DCFNet/DCFNet2	VGGNet	ImageNet, TC128, UAV123, NUS-PRO	Yes	DAF	Intel Xeon 2.40GHz CPU, GTX 1080 GPU	Matlab	MatConvNet	65	CM
STP	VGGNet	ImageNet	Yes	Votes	N/A, GTX Titan X GPU	Python	PyTorch	4	VM
MRCNN	VGGNet	ImageNet, VOT2015	Yes	DAF	Intel I7 3.50GHz CPU, GTX 1080 GPU	Matlab	MatConvNet	1.2	CM
CODA	VGG-19, SSD	ImageNet, VOT2013, VOT2014, VOT2015	Yes	DAF	Intel I7-4770K CPU, 32GB RAM, GTX 1070 GPU	Matlab	Caffe	34.8	CM
ATOM	ResNet-18, IoU-Nets	ImageNet, MSCOCO, LaSOT, TrackingNet	Yes	DAF	N/A, GTX 1080 GPU	Python	PyTorch	30	CM
UCT/UCT-Lite	ResNet101	ImageNet, TC128, UAV123	Only on the first frame	DAF	Intel I7-6700 CPU, 48GB RAM, GTX Titan X GPU	Matlab	Caffe	41	CM
FPRNet	ResNet-101, FlowNet	ImageNet, ILSVRC, SceneFlow	Yes	DAF, DMF	N/A	Matlab	Caffe	N/A	BB
ADT	-	ImageNet, ALOV300++, UAV123, NUS-PRO	Only on the first frame	DAF	Intel 2.40GHz CPU, GTX TITAN X GPU	Python	TensorFlow	7	CM

[6] Source: Deep Learning for Visual Tracking: A Comprehensive Survey [22]

1.3.2.2 Only Online Training

To discriminate unseen targets which may consider as the target in evaluation videos, some DL-based visual tracking methods use online training of whole or a part of DNNs to adapt network parameters according to the large variety of target appearance. Because of the time-consuming process of offline training on large-scale training data and insufficient discrimination of pre-trained models for representing tracking particular targets, the methods shown in Table 1.2 use directly training of DNNs and inference process alternatively online. However, these methods usually exploit some strategies to prevent over-fitting problem and divergence.

1.3.2.3 Both Offline and Online Training

To exploit the maximum capacity of DNNs for visual tracking, the methods shown in Table 1.3 use both offline and online training. The offline and online learned features are known as shared and domain-specific representations, which majorly can discriminate the target from foreground information or intra-class distractors, respectively. Because visual tracking is a hard and challenging problem, the DL-based visual trackers attempt to employ feature transferability and online domain adaption simultaneously.

1.3.3 Network Objective

After the training and inference stages, DL-based visual trackers localize the given target based on network objective function. Hence, the DL-based visual tracking methods are categorized into classification-based, regression-based, or both classification and regression-based methods as follows. This categorization is based on the objective functions of DNNs that have been used in visual tracking methods (see Fig. 1.3); hence, this sub-section does not include the methods that exploit deep off-the-shelf features because these methods do not design and train the networks and usually

employ DNNs for feature extraction.

1.3.3.1 Classification-based Objective Function

Motivated by other computer vision tasks such as image detection, classification-based visual tracking methods employ object proposal methods to produce hundreds of candidate/proposal BBs extracted from the search region. These methods aim to select the high score proposal by classifying the proposals to the target and background classes. This two-class (or binary) classification involves visual targets from various class and moving patterns, and also individual sequences, including challenging scenarios. Due to the main attention of these methods on inter-class classification, tracking a visual target in the presence of the same labeled targets is intensely prone to drift-problem. Also, tracking the arbitrary appearance of targets may lead to problems in recognizing different targets with varying appearances. Therefore, the performance of the classification-based visual tracking methods is also related to their object proposal method, which usually produces a considerable number of candidate BBs. On the other side, some recent DL-based methods utilize this objective function to take the optimal action on BB.

1.3.3.2 Regression-based Objective Function

Due to the continuous instinct of estimation space of visual tracking, regression-based methods usually aim to directly localize target in the subsequent frames by minimizing a regularized least-squares function. Generally, extensive training data are needed to train these methods effectively. The primary goal of regression-based methods is to refine the formulation of L2 or L1 loss functions such as utilizing shrinkage loss in learning procedure, modeling both regression coefficients and patch reliability to optimize a neural network efficiently, or applying the cost-sensitive loss to enhance unsupervised learning performance.

1.3.3.3 Both Classification and Regression-based Objective Function

To take advantages of both foreground-background/category classification and ridge regression (i.e., regularized least-squares objective function), some methods employ both classification- and regression-based objective functions for visual tracking (see Fig. 1.3), which their goal is to bridge the gap between the recent tracking-by-detection and continuous localization process of visual tracking. Commonly, these methods utilize classification-based methods to find the most similar object proposal to target and then the estimated region will be refined by a BB regression method [18, 17]. To enhance efficiency and accuracy, the target regions are estimated by classification scores and optimized regression/matching functions [32]. The classification outputs are mainly inferred for confidence scores of candidate proposals, foreground detection, response of candidate window, actions, and so forth.

1.3.4 Network Output

Based on their network outputs, the DL-based methods are classified into six main categories (see Fig. 1.3 and Table 1.1 to Table 1.3), namely confidence map (also includes score map, response map, and voting map), BB (also includes rotated BB), object score (also includes probability of object proposal, verification score, similarity score, and layer-wise score), action, feature maps, and segmentation mask. According to the network objective, the DL-based methods generate different network outputs to estimate or refine the estimated target location.

1.4 The Multiple Object Tracking Methodology

While in Single Object Tracking (SOT) the appearance of the target is known a priori, in Multiple Object Tracking (MOT), a detection step is necessary to identify the targets, that can leave or enter the scene. The main difficulty in tracking multiple targets simultaneously stems from the various occlusions and interactions between

objects, that can sometimes also have similar appearance. Thus, simply applying SOT models directly to solve MOT leads to poor results, often incurring in target drift and numerous ID switch errors, as such models usually struggle in distinguishing between similar looking intra-class objects. A series of algorithms specifically tuned to multi-target tracking have then been developed in recent years to address these issues, together with a number of benchmark datasets and competitions to ease the comparisons between the different methods.

The standard approach employed in most MOT algorithms is *tracking-by-detection*: a set of detections (i.e. bounding boxes identifying the targets in the image) are extracted from the video frames and are used to guide the tracking process, usually by associating them together in order to assign the same ID to bounding boxes that contain the same target. For this reason, many MOT algorithms formulate the task as an assignment problem. Modern detection frameworks [20, 19] ensure a good detection quality, and the majority of MOT methods (with some exceptions) have been focusing on improving the association; indeed, many MOT datasets provide a standard set of detections that can be used by the algorithms (that can thus skip the detection stage) in order to exclusively compare their performances on the quality of the association algorithm, since the detector performance can heavily affect the tracking results. The other branch is *template-matching*, which uses a template of the target object and tries to match it to the regions of the later images. Several works in the *template-matching* method [3, 32, 17, 18, 9] focus not only on deeper and wider networks, but also on searching as proposing regions to improve tracking robustness and accuracy on a single object.

MOT algorithms can also be divided into batch and online methods. Batch tracking algorithms are allowed to use future information (i.e. from future frames) when trying to determine the object identities in a certain frame. They often exploit global information and thus result in better tracking quality. Online tracking algorithms, on the contrary, can only use present and past information to make predictions about

the current frame. This is a requirement in some scenarios, like autonomous driving and robot navigation. Compared to batch methods, online methods tend to perform worse, since they cannot fix past errors using future information. It is important to note that while a real-time algorithm is required to run in an online fashion, not every online method necessarily runs in real-time; quite often, in fact, with very few exceptions, online algorithms are still too slow to be employed in a real-time environment, especially when exploiting deep learning algorithms, that are often computationally intensive.

Despite the huge variety of approaches presented in the literature, the vast majority of MOT algorithms share part or all of the following steps (summarized in Fig. 1.4):

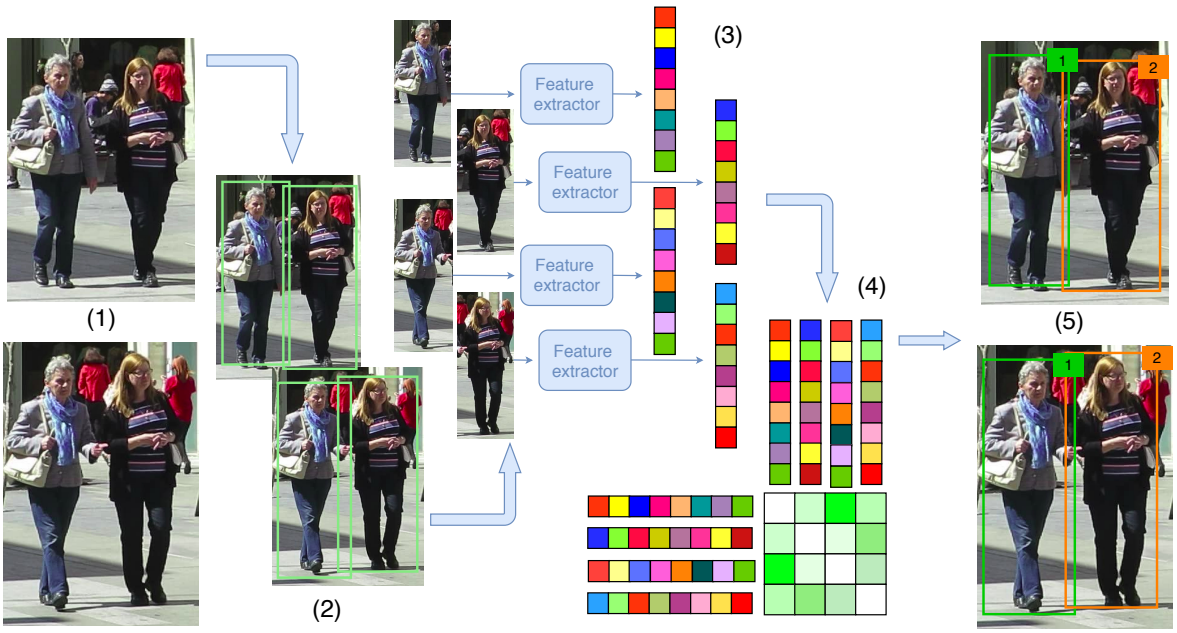

Figure 1.4: Usual workflow of a MOT algorithm. [7]

- Detection stage: an object detection algorithm analyzes each input frame to

[7] Source: Deep Learning in Video Multi-Object Tracking: A Survey [5]
Given the raw frames of a video (1), an object detector is run to obtain the bounding boxes of the objects (2). Then, for every detected object, different features are computed, usually visual and motion ones (3). After that, an affinity computation step calculates the probability of two objects belonging to the same target (4), and finally an association step assigns a numerical ID to each object (5).

identify objects belonging to the target class(es) using bounding boxes, also known as 'detections' in the context of MOT;

- Feature extraction/motion prediction stage: one or more feature extraction algorithms analyze the detections and/or the tracklets to extract appearance, motion and/or interaction features. Optionally, a motion predictor predicts the next position of each tracked target;
- Affinity stage: features and motion predictions are used to compute a similarity/distance score between pairs of detections and/or tracklets;
- Association stage: the similarity/distance measures are used to associate detections and tracklets belonging to the same target by assigning the same ID to detections that identify the same target.

While these stages can be performed sequentially in the order presented here (often once per frame for online methods and once for the whole video for batch methods), there are many algorithms that merge some of these steps together, or intertwine them, or even perform them multiple times using different techniques (e.g. in algorithms that work in two phases). Moreover, some methods do not directly associate detections together, but use them to refine trajectory predictions and to manage initialization and termination of new tracks; nonetheless, many of the presented steps can often still be identified even in such cases.

CHAPTER 2

BACKGROUND

In this chapter, I present the background knowledge related to Siamese Tracker, including the basic architecture of Siamese Network. This knowledge plays an important role as a theoretical foundation to follow and apply to this thesis. The used evaluation metrics also introduced.

2.1 Siamese Network

2.1.1 Siamese Neural Network

Siamese neural network was first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem. A siamese neural network consists of twin networks, which accept distinct inputs but are joined by an energy function at the top. This function computes some metrics between the highest-level feature representation on each side. The parameters between the twin networks are tied.

This strategy has two key properties:

- It ensures the consistency of its predictions. Weight tying guarantees that two extremely similar images could not possibly be mapped by their respective networks to very different locations in feature space because each network computes the same function.
- The network is symmetric: if I present two distinct images to the twin networks, the top conjoining layer will compute the same metric as the result from the same two images but to the opposite twins.

[1] Source: Siamese Neural Networks for One-Shot Image Recognition [11]

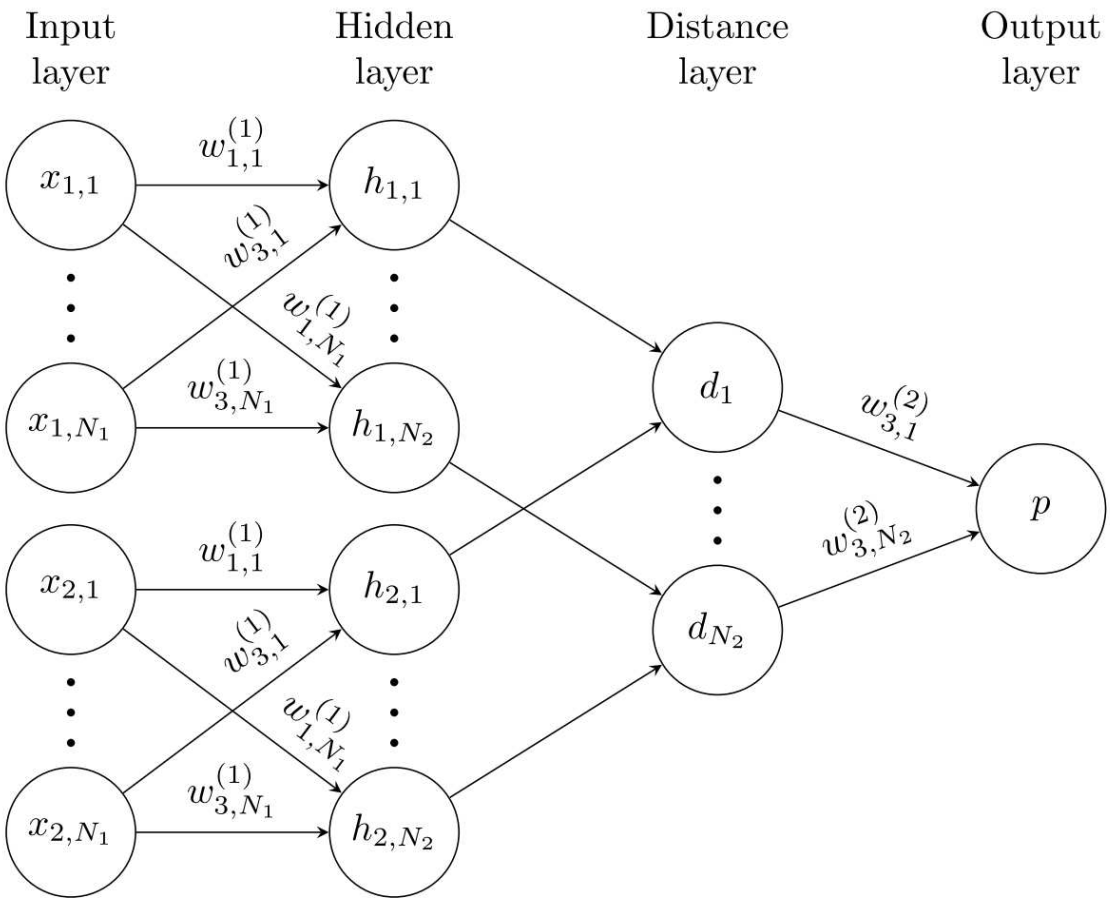

Figure 2.1: A simple 2 hidden layer siamese network for binary classification with logistic prediction p. [1]

The structure of the network is replicated across the top and bottom sections to form twin networks, with shared weight matrices at each layer. A cross-entropy objective is a natural choice for training the network.

2.1.2 Siamese Convolutional Neural Network

The most widely used methods for image feature extraction are based on subtle modifications of convolutional neural networks. The Siamese CNN idea is training CNNs with loss functions that combine information from different images, in order to learn the set of features that best differentiates examples of different objects (an example of the architecture is shown in Fig. 2.2).

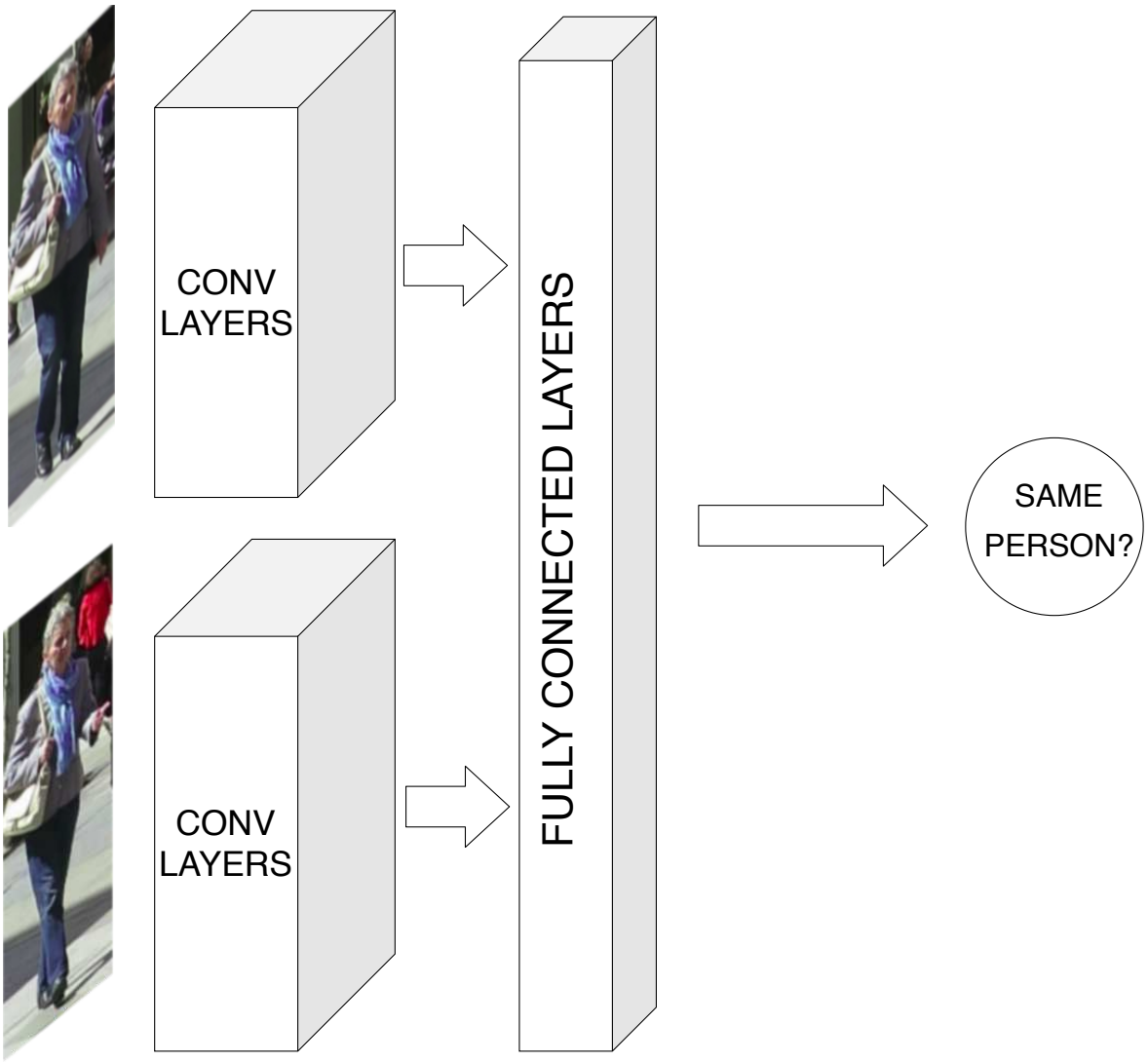

Figure 2.2: Example of a Siamese CNN architecture. For feature extraction, the network is trained as a Siamese CNN, but at inference time the output probability is discarded, and the last fully connected layer is used as feature vector for a single candidate. When the network is used for affinity computation, the whole structure is preserved during inference. [2]

Denote the exemplar image patch as z, the candidate image as x, the network is a nonlinear projection from image space to a Euclidean vector space. Finally, each of the patches is represented by a $\boldsymbol{d}$-dimensional feature vector using the same deep

[2]Source: Deep Learning in Video Multi-Object Tracking: A Survey [5]

nonlinear projection:

$$Z = \phi(z), X = \phi(x) \tag{2.1}$$

After this CNN layer, a function $f(Z,X)$ that compares two patches and returns a high score if the two images depict the same object and a low score otherwise. Several approaches are used to define a *similarity score*. Here are a few functions:

2.1.2.1 The cosine similarity

Object patches require to be cropped and resized to the same size. A pair of patches pass the same convolutional net and then two fully connected layers with a nonlinear activation layer between them. The two output vectors are compared to each other under the cosine similarity (CS) metric as shown in Fig. 2.3.

$$f(Z,X) = \frac{Z^T X}{||Z|| \cdot ||X||} \tag{2.2}$$

where $||\cdot||$ is the L2 norm of a vector, Z and X differentiate two branches.

During training, every pair has a groundtruth label indicating if they belong to the same object according to their object identification *ID*.

$$label(Z,X) = \begin{cases} +1 & , ID(Z) = ID(X) \\ -1 & , otherwise \end{cases} \tag{2.3}$$

At the same time, the groundtruth indicator *label* is the ideal similarity value, given $-1 \leq f \leq 1$. The training goal is to draw the similarity of every pair in the dataset (or a sampled set when using mini-batch learning) towards its label. Here, Mean Squared

Error is used as loss function:

$$L_{MSE} = \sum (f - label)^2 \tag{2.4}$$

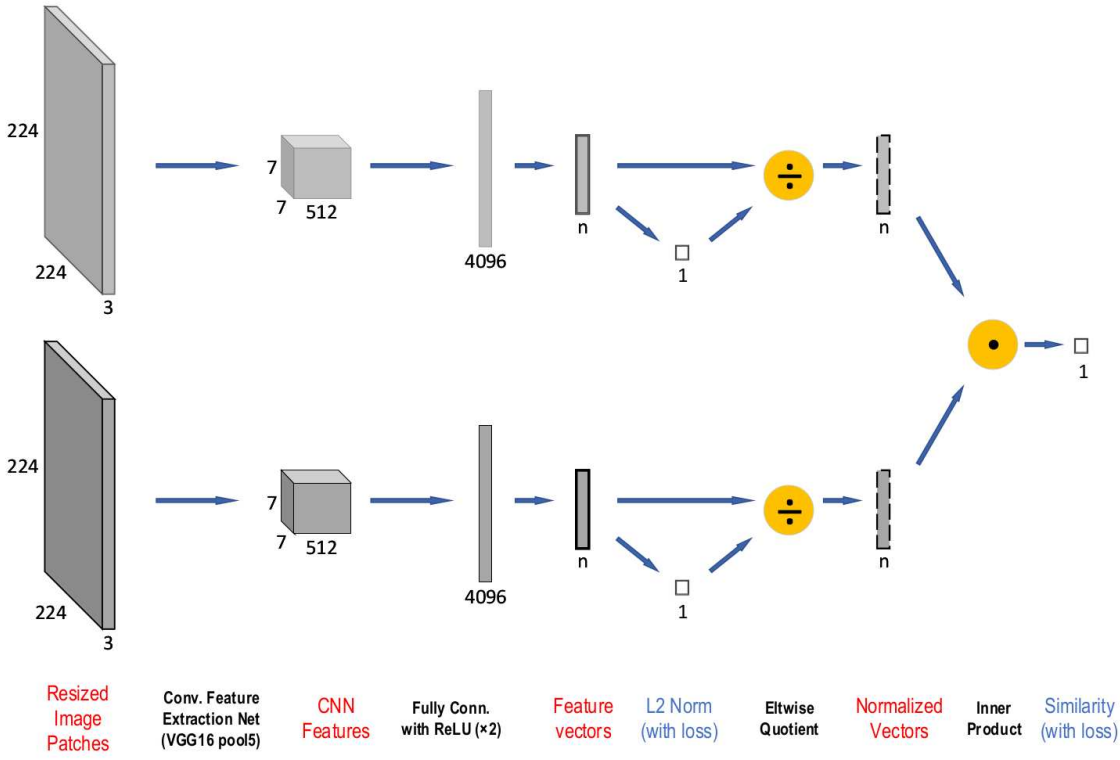

Figure 2.3: Pipeline of deep Siamese network for cosine similarity metric. [3]

2.1.2.2 The similarity as a cross-correlation layer

The advantage of a convolutional network is that, instead of a candidate image of the same size, a much larger search image X can be provided as input to the network and it will compute the similarity at all translated sub-windows on a dense grid in a single evaluation. To achieve this, a convolutional embedding function is used ϕ and combined the resulting feature maps using a cross-correlation layer:

[3] Source: Deep Siamese Network for Multiple Object Tracking [6]

$$f(Z,X) = \phi(z) * \phi(x) + b \tag{2.5}$$

where b denotes a signal which takes value $b \in \mathbb{R}$ in every location. The output of this network is not a single score but rather a score map defined on a finite grid as illustrated in Fig. 2.4.

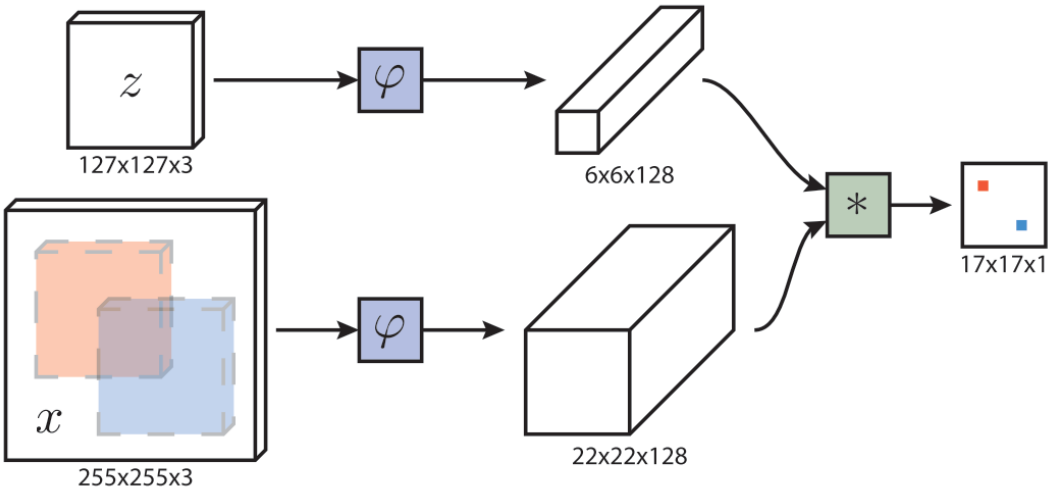

Figure 2.4: The similarity as a cross-correlation layer in Siamese architecture. The output is a scalar-valued score map whose dimension depends on the size of the search image x. This enables the similarity function to be computed for all translated sub-windows within the search image in one evaluation. In this example, the red and blue pixels in the score map contain the similarities for the corresponding sub-windows. [4]

Also, discriminative training is employed, training the network on positive and negative pairs and adopting the logistic loss:

$$l(\alpha,\beta) = \log(1+\exp(-\alpha\beta)) \tag{2.6}$$

[4]Source: Fully-Convolutional Siamese Networks for Object Tracking [3]

where α is the real-valued score of a single exemplar-candidate pair and $\beta \in \{+1, -1\}$ is its ground-truth label. The loss of a score map can be considered as the mean of the individual losses:

$$L(f(Z,X), label) = \frac{1}{|f(Z,X)|} \sum_{u \in |f(Z,X)|} l(f(Z,X)[u], label[u]) \tag{2.7}$$

Pairs are obtained from a dataset of annotated videos by extracting exemplar and search images that are centered on the target. The scale of the object within each image is normalized without corrupting the aspect ratio of the image. The elements of the score map are considered to belong to a positive example if they are within radius R of the center c (accounting for the stride k of the network):

$$label[u] = \begin{cases} +1 & , \text{if } k|u-c| \leq R \\ -1 & , otherwise \end{cases} \tag{2.8}$$

The losses of the positive and negative examples in the score map are weighted to eliminate class imbalance. Note that the network is symmetric $f(Z,X) = f(X,Z)$.

2.2 Siamese Tracker

2.2.1 Fully-Convolutional Siamese Network (SiamFC)

SiamFC [3] combines feature maps using cross-correlation and evaluates the network once on the larger search image. The cross-correlation layer provides an incredibly simple method to implement this operation efficiently within the framework of existing conv-net libraries. Also, an extremely simplistic algorithm performs tracking. Unlike more sophisticated trackers, it does not update a model or maintain a memory of past appearances, it does not incorporate additional cues such as optical flow or colour histograms, and it does not refine prediction with bounding box regression.

SiamFC only searches for the object within a region of approximately four times the previous size, and a cosine window is added to the score map to penalize large displacements. Tracking through scale space is achieved by processing several scaled versions of the search image. Any change in scale is penalized and updates of the current scale are damped.

2.2.2 Siamese Region Proposal Network (SiamRPN)

The SiamRPN [18] framework consists of a Siamese subnetwork for feature extraction and a region proposal subnetwork for proposal generation. Specifically, there are two branches in RPN subnetwork, one is in charge of the foreground-background classification, another is used for proposal refinement as shown in Fig. 2.5.

The region proposal subnetwork consists of a pair-wise correlation section and a supervision section. The supervision section has two branches, one for foreground-background classification and the other for proposal regression. If there are k anchors, network needs to output $2k$ channels for classification and $4k$ channels for regression. So the pair-wise correlation section first increase the channels of $\phi(z)$ to two branches $[\phi(z)]_{cls}$ and $[\phi(z)]_{reg}$ which have $2k$ and $4k$ times in channel respectively by two convolution layers. $\phi(x)$ is also split into two branches $[\phi(x)]_{cls}$ and $[\phi(x)]_{reg}$ by two convolution layers but keeping the channels unchanged. The correlation is computed on both the classification branch and the regression branch:

$$\begin{aligned} REG &= [\phi(z)]_{reg} \star [\phi(x)]_{reg} \\ CLS &= [\phi(z)]_{cls} \star [\phi(x)]_{cls} \end{aligned} \tag{2.9}$$

The template feature maps $[\phi(z)]_{cls}$ and $[\phi(z)]_{reg}$ are used as kernels and $\star$ denotes the convolution operation. The REG branch returns the output vector whose shape is $(17,17,4,k)$, each point whose shape is $(17,17,4)$ represents the distance between

[5]Source: High Performance Visual Tracking with Siamese Region Proposal Network [18]

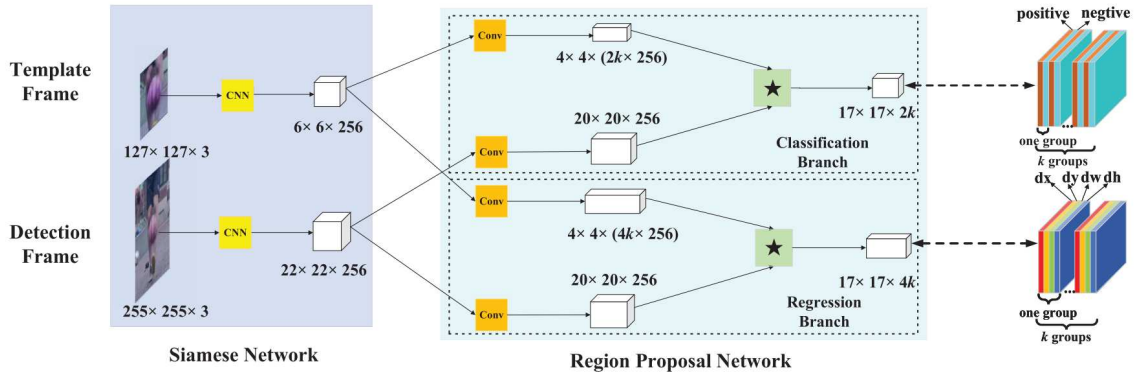

Figure 2.5: Main framework of Siamese-RPN: left side is Siamese subnetwork for feature extraction. Region proposal subnetwork lies in the middle, which has two branches, one for classification and the other for regression. Pair-wise correlation is adopted to obtain the output of two branches. Details of these two output feature maps are in the right side. In classification branch, the output feature map has $2k$ channels which corresponding to foreground and background of k anchors. In regression branch, the output feature map has $4k$ channels which corresponding to four coordinates used for proposal refinement of k anchors. In the figure, $\star$ denotes correlation operator. [5]

bounding boxes generated and a single relative anchor box. Similarly, the *CLS* branch returns the output vector whose shape is $(17,17,2,k)$ and each point in the *CLS* branch has $(17,17,2)$ shape, representing the positive score (denoted as *ps*) and negative score (denoted as *ns*) of these bounding boxes. Softmax loss is adopted to supervise the classification branch.

$$\begin{aligned} REG &= \{(dx_{\alpha\beta\kappa}, dy_{\alpha\beta\kappa}, dw_{\alpha\beta\kappa}, dh_{\alpha\beta\kappa})\} \\ CLS &= \{(ps_{\alpha\beta\kappa}, ns_{\alpha\beta\kappa})\} \end{aligned} \tag{2.10}$$

where $\alpha, \beta \in [0,17)$, $\kappa \in [0,k)$.

From the highest positive score index in the *CLS* branch, the next location of the

object is estimated in the *REG* branch as the same index as the score.

2.2.3 Other Trackers

Kim et al. [10] proposed a Siamese network which was trained using a contrastive loss. The network took two images, their IoU score and their area ratio as input, and produced a contrastive loss as output. After the net was trained, the layer that computed the contrastive loss was removed, and the last layer was used as a feature vector for the input image. The similarity score was later computed by combining the Euclidean distance between feature vectors, the IoU score and the area ratio between bounding boxes. The association step was solved using a custom greedy algorithm. Wang et al. [31] also proposed a Siamese network which took two image patches and computed a similarity score between them. The score at test time was computed comparing the visual features extracted by the network for the two images, and including temporally constrained information. The distance employed as similarity score was a Mahalanobis distance with a weight matrix, also learned by the model.

2.3 Evaluation metrics

In order to provide a common experimental setup where algorithms can be fairly tested and compared, a group of metrics has been *de facto* established as standard, and they are used in almost every work. The most relevant ones are metrics defined by Wu and Nevatia [34], the so-called CLEAR MOT metrics [2], and recently the ID metrics [24]. These sets of metrics aim to reflect the overall performance of the tested models, and point out the possible drawbacks of each one. Therefore, those metrics are defined as follows:

2.3.1 Classical metrics

These metrics, defined by Wu and Nevatia [34], highlight the different types of errors a MOT algorithm can make. In order to show those problems, the following values are computed:

- *Mostly Tracked* (MT) trajectories: number of ground-truth trajectories that are correctly tracked in at least 80% of the frames.
- *Fragments*: trajectory hypotheses which cover at most 80% of a ground truth trajectory. Observe that a true trajectory can be covered by more than one fragment.
- *Mostly Lost* (ML) trajectories: number of ground-truth trajectories that are correctly tracked in less than 20% of the frames.
- *False trajectories*: predicted trajectories which do not correspond to a real object (i.e. to a ground truth trajectory).
- *ID switches*: number of times when the object is correctly tracked, but the associated ID for the object is mistakenly changed.

2.3.2 CLEAR MOT metrics

The CLEAR MOT metrics were developed for the *Classification of Events, Activities and Relationships* (CLEAR) workshops held in 2006 [27] and 2007 [28]. The workshops were jointly organized by the European CHIL project, the U.S. VACE project, and the National Institute of Standards and Technology (NIST). Those metrics are MOTA (Multiple Object Tracking Accuracy) and MOTP (Multiple Object Tracking Precision). They serve as a summary of other simpler metrics which compose them. A detailed description of how to match the real objects (ground truth) with the tracker hypothesis can be found in [2], as it is not trivial how to consider when a hypothesis

is related to an object, and it depends on the precise tracking task to be evaluated. The most used metric to decide whether an object and a prediction are related or not is Intersection over Union (IoU) of bounding boxes, as it was the measure established in the presentation paper of MOT15 dataset [15]. Specifically, the mapping between ground truth and hypotheses is established as follows: if the ground truth object o_i and the hypothesis h_j are matched in frame $t-1$, and in frame t the $IoU(o_i, h_j) \geq 0.5$, then o_i and h_j are matched in that frame, even if there exists another hypothesis h_k such that $IoU(o_i, h_j) < IoU(o_i, h_k)$, considering the continuity constraint. After the matching from previous frames has been performed, the remaining objects are tried to be matched with the remaining hypotheses, still using a 0.5 IoU threshold. The ground truth bounding boxes that cannot be associated with a hypothesis are counted as *false negatives* (FN), and the hypotheses that cannot be associated with a real bounding box are marked as *false positives* (FP). Also, every time a ground truth object tracking is interrupted and later resumed is counted as a *fragmentation*, while every time a tracked ground truth object ID is incorrectly changed during the tracking duration is counted as an *ID switch*. Then, the simple metrics computed are the following:

- FP: the number of false positives in the whole video;
- FN: the number of false negatives in the whole video;
- Fragm: the total number of fragmentations;
- IDSW: the total number of ID switches.

The MOTA score is then defined as follows:

$$MOTA = 1 - \frac{(FN + FP + IDSW)}{GT} \quad \in (-\infty, 1]$$

where GT is the number of ground truth boxes. It is important to note that the score can be negative, as the algorithm can commit a number of errors greater than the

number of ground truth boxes. Usually, instead of reporting MOTA, it is common to report the percentage MOTA, which is just the previous expression expressed as a percentage. On the other hand, MOTP is computed as:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$$

where c_t denotes the number of matches in frame t, and $d_{t,i}$ is the bounding box overlap between the hypothesis i with its assigned ground truth object. It is important to note that this metric takes less information about tracking into account, and rather focuses on the quality of the detections.

2.3.3 ID scores

The main problem of MOTA score is that it takes into account the number of times a tracker makes an incorrect decision, such as an ID switch, but in some scenarios (e.g. airport security) one could be more interested in rewarding a tracker that can follow an object for the longest time possible, in order to not lose its position. Because of that, in [24] a couple of alternative new metrics are defined, that are supposed to complement the information given by the CLEAR MOT metrics. Instead of matching ground truth and detections frame by frame, the mapping is performed globally, and the trajectory hypothesis assigned to a given ground truth trajectory is the one that maximizes the number of frames correctly classified for the ground truth. In order to solve that problem, a bipartite graph is constructed, and the minimum cost solution for that problem is taken as the problem solution. For the bipartite graph, the sets of vertices are defined as follows: the first set of vertices, V_T, has a so-called regular node for each true trajectory, and a false positive node for each computed trajectory. The second set, V_C, has a regular node for each computed trajectory and a false negative for each true one. The costs of the edges are set in order to count the number of false negative and false positive frames in case that edge was chosen (more information can

be found in [24]). After the association is performed, there are four different possible pairs, attending to the nature of the involved nodes. If a regular node from V_T is matched with a regular node of V_C (i.e. a true trajectory is matched with a computed trajectory), a *true positive ID* is counted. Every false positive from V_T matched with a regular node from V_C counts as a *false positive ID*. Every regular node from V_T matched with a false negative from V_C counts as a *false negative ID*, and finally, every false positive matched with a false negative counts as a *true negative ID*. Afterwards, three scores are calculated. IDTP is the sum of the weights of the edges selected as *true positive ID* matches (it can be seen as the percentage of detections correctly assigned in the whole video). IDFN is the sum of weights from the selected *false negative ID* edges, and IDFP is the sum of weights from the selected *false positive ID* edges. With these three basic measures, another three measures are computed:

- Identification precision: $IDP = \frac{IDTP}{IDTP+IDFP}$
- Identification recall: $IDR = \frac{IDTP}{IDTP+IDFN}$
- Identification F1: $IDF1 = \frac{2}{\frac{1}{IDP}+\frac{1}{IDR}} = \frac{2IDTP}{2IDTP+IDFP+IDFN}$

Usually, the reported metrics in almost every piece of work are the CLEAR MOT metrics, mostly tracked trajectories (MT), mostly lost trajectories (ML) and IDF1, since these metrics are the ones shown in MOTChallenge leaderboards. Additionally, the number of frames per second (FPS) the tracker can process is often reported, and is also included in the leaderboards. However, this metric is difficult to compare among different algorithms, since some of the methods include the detection phase while others skip that computation. Also, the dependency on the hardware employed is relevant in terms of speed.

CHAPTER 3

MULTIPLE PEDESTRIAN TRACKING WITH SIAMESE TRACKER

In this chapter, I present my strategy to track pedestrians in detail. Inspired by SORT [4] and DeepSORT [33], I extend the target of Siamese Tracker into multiple object, especially on human, by adding the matching stage periodically after tracking. This approach can be applied to all common types of variations in appearance from tracking examples. I use the ***NgocHa Retail Store*** *dataset, which focuses on heavily occluded humans and unpredictable movement trajectories, and* ***TownCentre*** *dataset to evaluate the proposed strategy.*

3.1 Simple Feature Updating

Since human pose changes all the time, I adopt a simple feature update step to an existing tracker when there is a new detected position. At the *j*th frame, the detect stage returns the cropped image of the *i*th object (denoted as z_{ij}). Let $\phi(z_{ij})$ denotes the deep convolutional appearance template from the Siamese Network and $Z_{ij} = \phi(z_{ij})$. In order to adapt object appearance, I compute $\phi(z_{ij})$ and update the new template feature with a small coefficient to balance two parts whenever it gets the detector to correct. This action may not always be continuous but periodical after tracking.

$$\widehat{Z}_{ij} = \begin{cases} Z_{ij} & , j = 0 \\ \gamma \widehat{Z}_{ij-1} + (1-\gamma) Z_{ij} & , j > 0 \end{cases} \tag{3.1}$$

The update function is a very simple linear function by using previous appearance templates [35]. The update rate γ is assigned to a fixed small value in range $[0,1]$, following the assumption that the human pose changes quite much. During my ex-

periments, I found that setting γ in mid-range gives an excellent result.

3.2 Trackers and detections associating strategy

At any tracking step, the Siamese Network takes x_{ij} as the search field that is samples around the previous position of the target and then predicts the location of the target in the current frame. I employ the Region Proposal Network generation and selection framework that is first introduced in Faster-RCNN [23] and also used in SiamRPN [18]. Let function $\zeta(\widehat{Z}_{ij}, \phi(x_{ij}))$ denotes for the correlation function, which is computed on both two frameworks and returns result in two branches:

$$REG, CLS = \zeta(\widehat{Z}_{ij}, \phi(x_{ij})) \tag{3.2}$$

Let bounding box regression branch denotes as REG, the classification branch denotes as CLS, and the number of anchor box is k. The REG branch returns the output vector whose shape is $(21, 21, 4, k)$, each point whose shape is $(21, 21, 4)$ represents the distance between bounding boxes generated and a single relative anchor box. Similarly, the CLS branch returns the output vector whose shape is $(21, 21, 2, k)$ and each point in the CLS branch has $(21, 21, 2)$ shape, representing the positive score (denoted as ps) and negative score (denoted as ns) of these bounding boxes.

$$\begin{aligned} REG &= \{(dx_{\alpha\beta\kappa}, dy_{\alpha\beta\kappa}, dw_{\alpha\beta\kappa}, dh_{\alpha\beta\kappa})\} \\ CLS &= \{(ps_{\alpha\beta\kappa}, ns_{\alpha\beta\kappa})\} \end{aligned} \tag{3.3}$$

where $\alpha, \beta \in [0, 21)$, $\kappa \in [0, k)$.

From the highest positive score index in the CLS branch, I get the next location of the ith object in the REG branch as the same index as the score:

$$
\begin{aligned}
A,B,K &= argmax(softmax(CLS)[:,:,0,:]) \\
t_l &= convert\_boundingbox(REG[A,B,:,K]) \\
t_s &= softmax(CLS)[A,B,0,K]
\end{aligned}
\tag{3.4}
$$

where t_l is the tracker location and t_s is the tracker score. The 0 index in the *CLS* branch represents the positive scores *ps*.

I use the softmax function to normalize each pair (ps, ns) into a vector of two values that follows a probability distribution whose total sums up to 1 and convert the distance with the corresponding anchor to the bounding box of the target. I map a collection of objects whose each object t is tracked as mentioned above in parallel, combine to set T and denote $|T| = n$.

Besides, the object detector detects object bounding boxes (denotes as set D and $|D| = m$) in the following format: $D = \{(x_\alpha, y_\alpha, w_\alpha, h_\alpha) | \alpha \in [0, m)\}$. To match each pair from the tracker predictions set T and detections set D, I compute an intersection over union association cost:

$$
c(t_l, d) = 1 - \frac{t_l \cap d}{t_l \cup d} \tag{3.5}
$$

where $t \in T$ and $d \in D$.

The global cost matrix can be considered to formulate as:

$$
\begin{aligned}
C(T,D)_{n\times m} &= \begin{pmatrix}
c(t_{l1}, d_1) & c(t_{l1}, d_2) & \cdots & c(t_{l1}, d_m) \\
c(t_{l2}, d_1) & c(t_{l2}, d_2) & \cdots & c(t_{l2}, d_m) \\
\vdots & \vdots & \ddots & \vdots \\
c(t_{ln}, d_1) & c(t_{ln}, d_2) & \cdots & c(t_{ln}, d_m)
\end{pmatrix} \\
&= (c(t_{l\alpha}, d_\beta)) \in \mathbb{R}^{n\times m}
\end{aligned}
\tag{3.6}
$$

Additionally, the positive score is considered as an important factor to prioritize pre-

dictions. I select the subset of trackers $\{t \in T | t_s = score\}$ that have the same score and unmatched subset of detections $\widehat{D}$ that have not been associated with any trackers in the previous matches. Selected *score* is descending sorted from $max(\{t_s | t \in T\})$ to $min(\{t_s | t \in T\})$. Then the sub cost matrices between each pair of two sets at considered score can be formulated as:

$$\begin{gathered} C(\{t \in T | t_s = score\}, \widehat{D}) \\ \textbf{for } score \in descending\_sort(\{t_s | t \in T\}) \end{gathered} \tag{3.7}$$

In the implementation, I linearly rescaled values $t_{l\alpha}$ into a new arbitrary range 0 to 100 so that values can be observed easily. After computing sub cost matrices, I also mark certain impossible pairings. By separating global cost matching problem into several sub cost matches, I reduce the computation cost of linear assignment problem from $O(|T|^3)$ to $O(a^3) + O(b^3) + \ldots$ where $a + b + \ldots = n$. The values of $a, b, \ldots$ depend on the degree of occlusion of data scenes.

A linear sum assignment problem solver for dense matrices is applied to get suitable pairs on each matrix. After getting matched pairs, I update detections to trackers as described in (3.1). On the other hand, trackers that are not matched with any detections are considered to be occluded and still predict shortly until the correction is made or after a long time. Also, the unmatched observations will be initialized as new trackers.

The strategy process flow is depicted in Fig. 3.1. The Track stage leverages the Siamese Tracker's work and is along with the Update phase to adapt human pose change by a simple update function. $\widehat{Z}_{ij} = Z_{ij} + Z_{ij-1}$ is a simple representation of (3.1). In implementation, I compute the accumulated $\widehat{Z}_{ij-1}$ instead of Z_{ij-1}. Each object in T set associates with each object in D set by (3.5). The global cost matrix, which may contain impossible match pairs, is split into many sub-matrices and assignment sum cost is optimized on these matrices. Compatible pairs, which are

marked by **X** letter in Fig. 3.1, will be used to update the status of the objects.

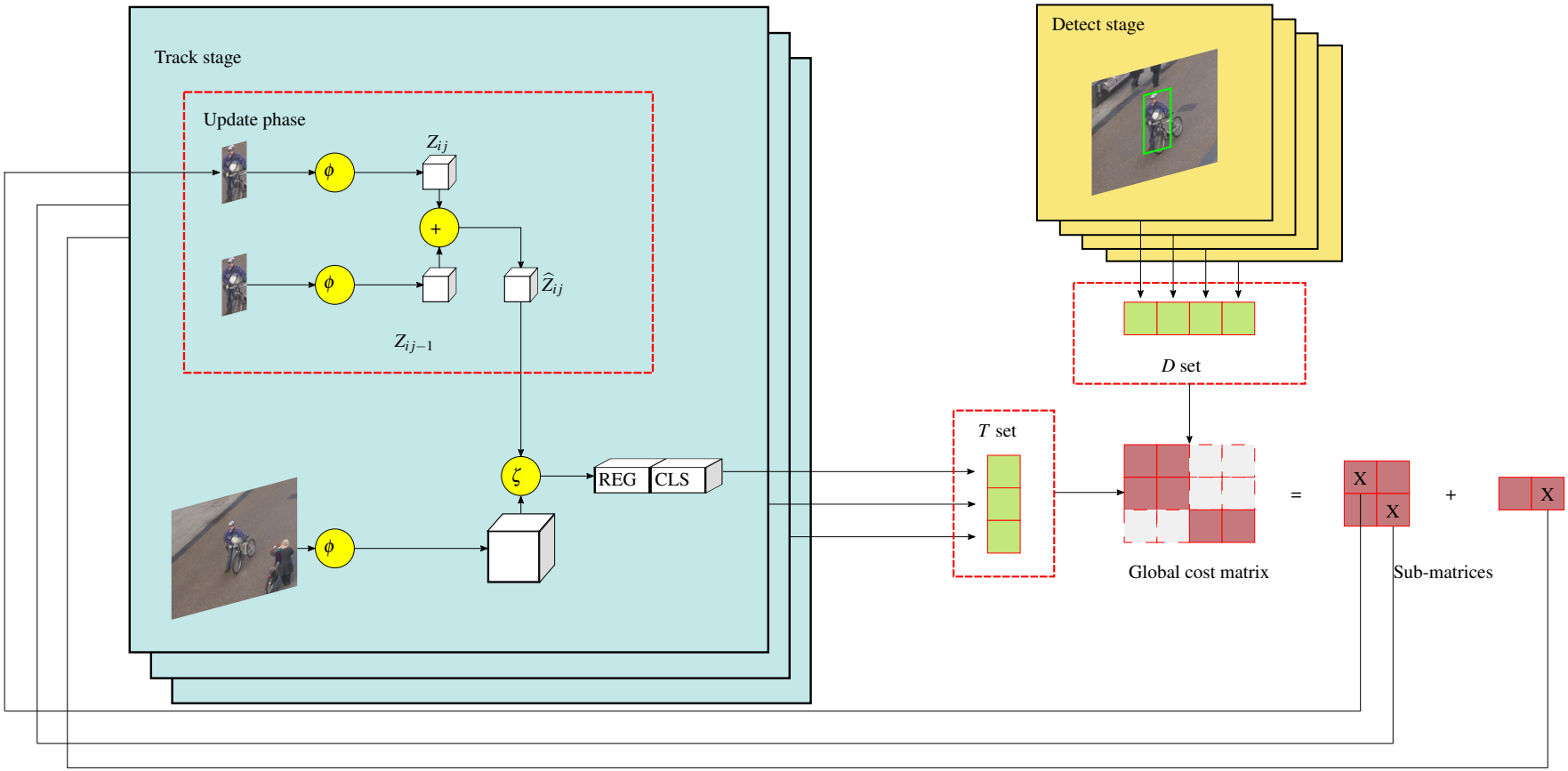

Figure 3.1: The detector-tracker associating strategy. [1]

[1]Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

3.3 Kalman Filter Penalty

I break the assignment problem in a global cost matrix into a series of sub-matrices as described in 3.2. Additionally, with some uncertainty or inaccuracy in prediction, the rescaled *score* is multiplied by a penalty to decrease the priority of suddenly direction changed trackers, also expand the score range. I use a simple Kalman Filter with the assumption that those objects are moving at the constant velocity motion. I take the bounding box center position as direct observation of the object state. The distance between the Kalman Filter prediction $p = (x, y)$ and Siamese Tracker prediction center $center(t_l) = (x, y)$ over the frame size (w, h), which map to the interval $[0, 1]$, is computed as the punishment weight:

$$\widehat{t_s} = t_s \times (1 - norm(\frac{center(t_l) - p}{image\_size})) \tag{3.8}$$

The punishment is illustrated in Fig. 3.2. For simplicity, I plot ground truth trajectory with solid curve and the predictions with dashed ones. The gray area indicates the matching threshold. The red curve represents the Siamese Tracker, and the blue one is the Kalman Filter output.

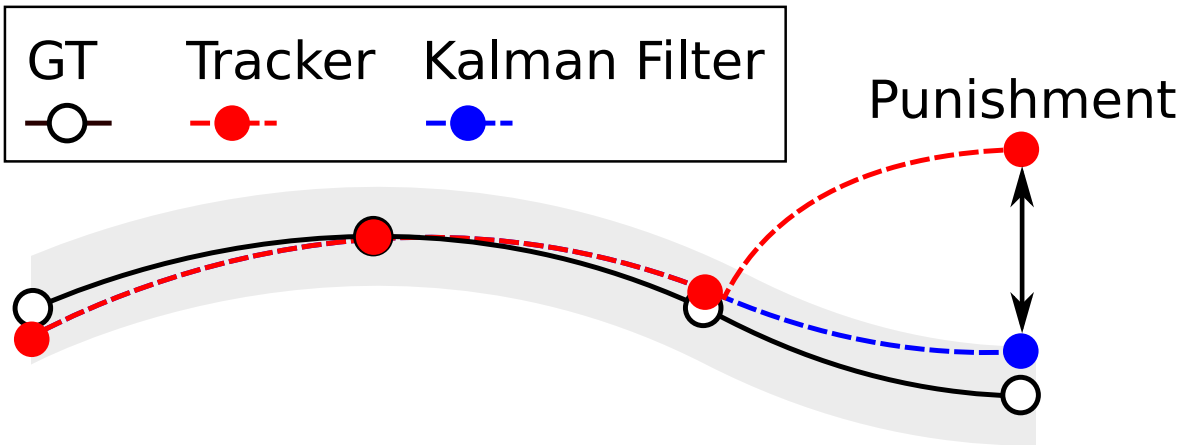

Figure 3.2: A punishment occurs when the tracker suddenly changes direction. [2]

3.4 Training phase

Since all pretrained Siamese convolutional appearance networks (denoted as ϕ) are trained on a large number of object types, I retrain Siamese convolutional network on the **Market-1501** [36] dataset with AlexNet [14]. Each positive pair is generated by 2-permutations of n intra-trackers, where n is the length of a tracker. From 751 trackers, 593876 positive pairs are generated with a large number of human poses, cloth colors. Negative samples are also picked by randomly choosing two people. Data augmentation methods are considered applicable, I apply horizontal flip, grayscale, blur, shift scale, and color distortions on both template images and search images.

I use the loss function in Faster R-CNN [23] to train the network on both the classification and the regression branch. The loss for classification is the cross-entropy loss and smooth L1 loss for regression. Finally, I optimize the two losses combination:

$$L = L_{cls} + \lambda L_{reg} \tag{3.9}$$

3.5 Experiments

3.5.1 State-of-the-art algorithms comparison

I evaluate my proposed method on two datasets. **NgocHa Retail Store**, which is recorded at retail shopping sites and focuses on heavily occluded humans and unpredictable movement trajectories. Sample is shown as Fig. 3.3. This dataset contains 38015 human bounding boxes within 7580 images. And **TownCentre** [1] dataset is recorded in a busy town center street. Up to 230 pedestrians were tracked simultaneously in 4500 frames, contains 71460 bounding boxes. Also, I only focus on my matching framework by using ground truth bounding boxes as input. I further compare my method results with the original SiamRPN [18], DeepSORT [33] and

[2]Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

Figure 3.3: Sample of my **NgocHa Retail Store** dataset. All frames are marked by the bounding box representing different ID. [3]

GOTURN [9] tracking methods on the MOT metrics [24] as shown in Table 3.1.

[3]Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

Table 3.1: Performance comparisons on two datasets. [4]

Methods	**NgocHa Retail Store**									
	$frame\_interval = 1$					$frame\_interval = 10$				
	IDF1	IDP	IDR	MOTA	MOTP	IDF1	IDP	IDR	MOTA	MOTP
GOTURN	22.0	22.0	22.0	-4.5	0.349	-	-	-	-	-
SiamRPN	25.9	25.9	25.9	-25.1	0.330	-	-	-	-	-
DeepSORT	64.0	62.6	65.5	94.9	0.033	53.1	52.0	54.2	84.5	**0.084**
Modified SiamRPN (Ours)	**69.8**	**69.9**	**69.8**	**99.8**	**0.000**	**56.1**	**56.5**	**55.7**	**95.6**	0.117
	TownCentre									
Methods	$frame\_interval = 1$					$frame\_interval = 10$				
	IDF1	IDP	IDR	MOTA	MOTP	IDF1	IDP	IDR	MOTA	MOTP
GOTURN	28.1	28.2	27.9	-2.1	0.343	-	-	-	-	-
SiamRPN	39.1	39.1	39.1	-12.5	0.376	-	-	-	-	-
DeepSORT	80.5	76.5	**85.0**	87.7	0.019	70.4	69.3	71.6	76.6	**0.104**
Modified SiamRPN (Ours)	**85.4**	**85.8**	**85.0**	**98.6**	**0.000**	**79.9**	**81.8**	**78.2**	**89.4**	0.167

[4]Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

Seeing detector as a semi-advisor, I evaluate the tracker predictions within a $frame\_interval$ then get the matching strategy as the correction, so it only occurs on detection-based methods. Since the original idea of GOTURN [9] and SiamRPN [18] only works on a single object, I have to add some modifications to make it work on multiple object. By adding the initialization phase based on the first time new object appears from the ground truth and letting it track till the end without any correction, I see it change ID easily in crowded environments and it is vulnerable at occlusion scenes.

Thanks to the deep convolutional feature of Siamese Network, my method can perform not only a better quantitative result on ID precision, ID recall, ID F1-score, but also better qualitative comparison[5], as demonstrated in Fig. 3.4.

[5]Full tracking visualization by DeepSORT [33] is uploaded at `https://youtu.be/Jp-mjmrO0uU`, and my modified Siamese Tracker is at `https://youtu.be/OeN46gICi2w`.

Figure 3.4: Qualitative comparison result on two methods. Red bounding boxes are DeepSORT's visualization and green bounding boxes are the visualization of my method. My method performs tracking better than DeepSORT after a complete occlusion. The 9th ID changes to the 11th ID, compares with the 8th ID, whose prediction is more stable. Best viewed on color display. [6]

[6]Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

3.5.2 Time complexity

I benchmark the assignment time complexity of my cascade matching strategies on the **TownCentre** [1] dataset as described in 3.2. In Fig. 3.5 and Fig. 3.6, the blue area and the blue square mark represent the matching strategy which applies with (3.7) and Kalman Filter penalty (3.8), the red area and red triangle mark represent the strategy applies with (3.7) but does not use penalty (3.8), and the black area and black dot use the naivest approach, which simply assigns on the global cost matrix (3.6) without penalty (3.8). The time complexity, which is counted by the number of loops, significantly decreases from the black to the red to the blue. The number of iterations of the highest density point in the blue area is less than the number of two others. However, this results to a trade-off for a slight drop of F1-score as in Fig. 3.6. In this work, I only benchmark on small dataset. On a large and extreme occlusion dataset, the gap could be wider.

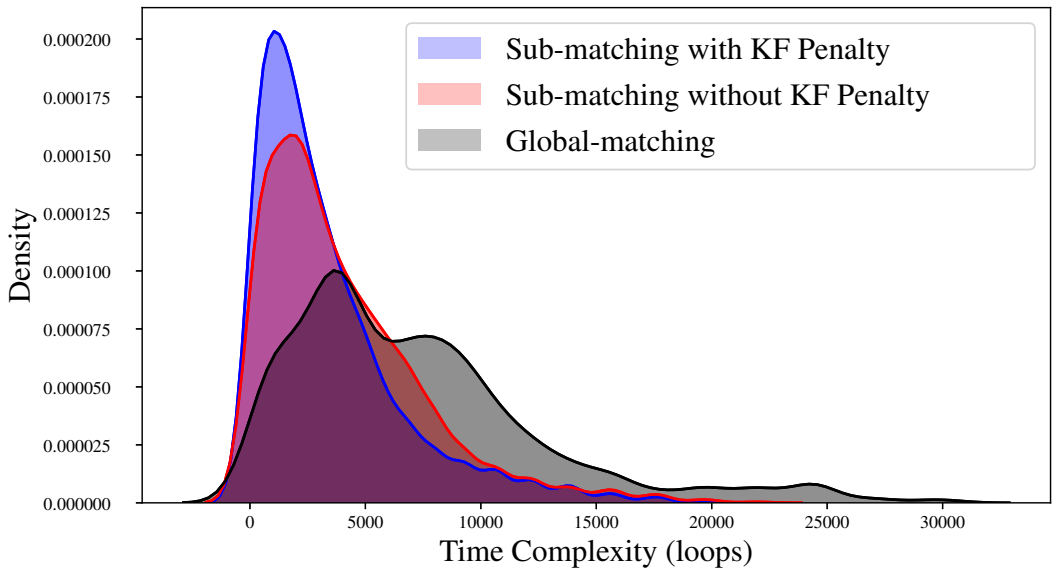

Figure 3.5: Histogram time complexity distribution between three types of matching. Best viewed on color display. [7]

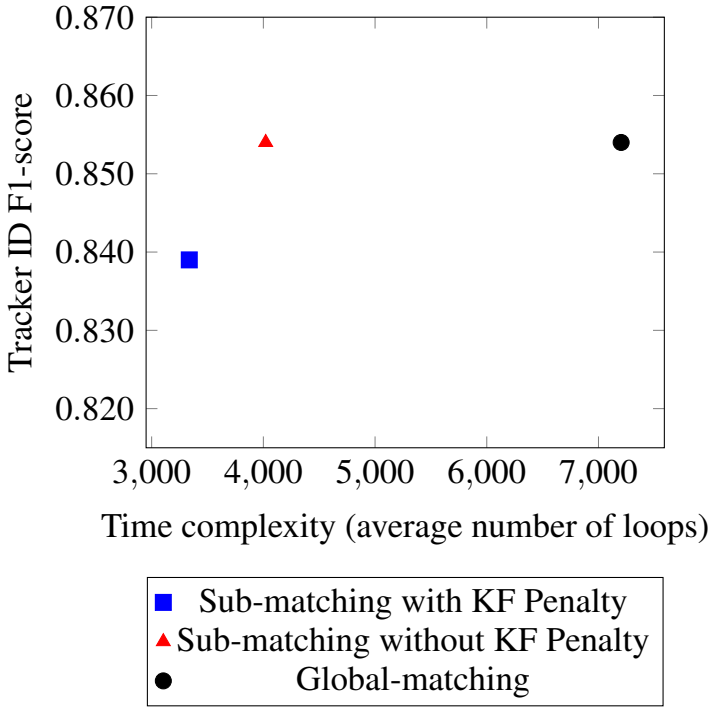

Figure 3.6: Complexity and F1-score compares between three methods. [8]

3.5.3 The update rate observation

The update rate γ in (3.1) is crucial to learn a good appearance for human tracking. In Fig. 3.7, I observe the change of F1-score by the increase of γ in the range $[0,1]$. By choosing a value which makes two parts balance and focusing on new features, I achieve the highest F1-score on **NgocHa Retail Store** at $\gamma = 0.4$. And with the dataset whose human moving trajectories are stable and linear as in the **TownCentre** [1] dataset, the γ coefficient does not affect to excess, the best result is also obtained when γ is in the mid-range. Oppositely, F1-score decreases dramatically when γ reaches 1. It means that the template feature is only initialized for the first time, and it never gets updated later.

[7] Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]
[8] Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]
[9] Source: Tracking customers in crowded retail scenes with Siamese Tracker [1]

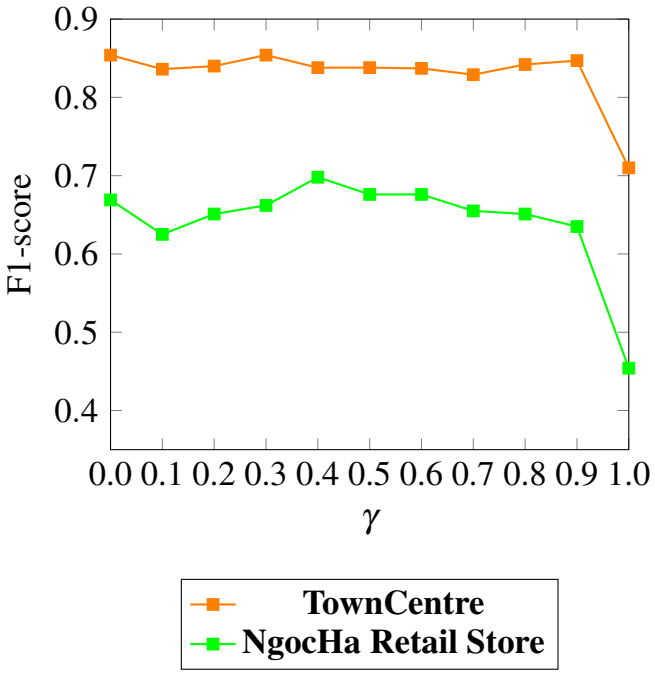

Figure 3.7: The change of F1-score is affected by γ. Best viewed on color display. [9]

CHAPTER 4

CONCLUSION AND FUTURE WORK

This research work propose a new algorithm to track multiple object and successfully set up the tracking system identifying customer's behaviors at the retailing sites based on the theoretical basis of Siamese Neural Network. Result of the implementation fulfills the essential requirements that are previously set up by the dissertation. This research work is approved to publish in the Proceedings of the 2020 IEEE-RIVF International Conference on Computing And Communication Technologies [1].

4.1 Conclusion

With a good method to track in-store customers, these questions are easier to approach: What time is the store most crowded? How much time do customers spend? And how will these affect sales? I focus in this work not only on the real in-store data but also on the efficient method to handle the Siamese Tracker and the object detector in a crowded environment. During the process, the feature extraction branch is trained by a custom dataset. This provides the additional exotic information of human pose.

Achievements:

- Examining Siamese Network structure and training its backbone which is fine-tuned with the custom human dataset.
- Studying the methodology of tracking algorithms and applying it.
- Building an end-to-end tracking system to analyze customer's behaviors, and especially work well in crowded environment.
- The potential for extending to be applied for a variety of organizations such as bank, hospital, traffic controlling department, office security department.

4.2 Future Work

Identifying customer's behaviors at the counter is a practical application of AI technology which helps the retailer to enhance their service quality. Based on them, the decision makers could propose different plans to improve customer satisfaction, reallocate their resources, cut down inefficient expenditure. In the future, this project is expected to reach a diversity of organizations such as coffee shop, airport, train station, hospital, etc, where queueing and length of customer's waiting time are still an imperative problem. To actualize such intention, the system must be gone through several researches to enhance its accuracy. The following methods are in consideration for the upcoming update:

- My method has shown good performance in tracking precision and recall, but the number of identity switches is still relatively high. I remark that the object deep feature might be an important factor to be considered when matching tracker prediction with detection besides the overlapping area. A new branch in the network architecture with a fuse loss function training may be used to perform appearance representation besides classification and regression branch, hence the accuracy would be amended.

- In this work, I trained my backbone on the **Market-1501** [36] dataset, which contains cropped pedestrian images. By collecting more dataset with full-frame availability, I am hopeful that the regression branch can perform better results.

- Since several deeper and wider networks are created and boost not only the neural network depth but also the performance of all the computer vision tasks, I also regard them as major improvements in tracking problem. But with the limitation of resources, I only present AlexNet [14] performance in this work. In the future, I will attempt to take full advantage of the neural network.

REFERENCES

[1] B. Benfold and I. Reid. Coarse gaze estimation. https://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html#datasets, 2009.

[2] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008, 01 2008. doi: 10.1155/2008/246309.

[3] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H.S. Torr. Fully-convolutional Siamese networks for object tracking. In *Proc. ECCV*, pages 850–865, 2016.

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. *arXiv e-prints*, art. arXiv:1602.00763, Feb 2016.

[5] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. DEEP LEARNING IN VIDEO MULTI-OBJECT TRACKING: A SURVEY A PREPRINT. Technical report, 2019.

[6] Bonan Cuan, Khalid Idrissi, and Christonhe Garcia. Deep Siamese Network for Multiple Object Tracking. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, aug 2018. ISBN 978-1-5386-6070-6. doi: 10.1109/MMSP.2018.8547137. URL https://ieeexplore.ieee.org/document/8547137/.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE CVPR*, pages 886–893, 2005.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE CVPR*, pages 770–778, 2016.

[9] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 FPS with deep regression networks. In *Proc. ECCV*, pages 749–765, 2016.

[10] Minyoung Kim, Stefano Alletto, and Luca Rigazio. Similarity mapping with enhanced siamese network for multi-object tracking, 2016.

[11] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proc. ICML Deep Learning Workshop*, 2015.

[12] Matej Kristan and et al. The seventh visual object tracking VOT2019 challenge results. In *Proc. ICCVW*, 2019.

[13] Matej Kristan, Aleš Leonardis, Jiří Matas, Michael Felsberg, Roman Pflugfelder, and et al. The sixth visual object tracking VOT2018 challenge results. In *Proc. ECCVW*, pages 3–53, 2019.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, volume 2, pages 1097–1105, 2012.

[15] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv e-prints*, art. arXiv:1504.01942, Apr 2015.

[16] Hankyeol Lee, Seokeon Choi, and Changick Kim. A memory model based on the Siamese network for long-term tracking. In *Proc. ECCVW*, pages 100–115, 2019.

[17] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of Siamese visual tracking with very deep networks, 2018. URL `http://arxiv.org/abs/1812.11703`.

[18] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with Siamese region proposal network. In *Proc. IEEE CVPR*, pages 8971–8980, 2018.

[19] Tsung Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE CVPR*, pages 936–944, 2017.

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Proc. ECCV*, pages 21–37, 2016.

[21] Christopher Lovelock and Evert Gummesson. Whither services marketing? in search of a new paradigm and fresh perspectives. *Journal of Services Research*, 7:20–41, 08 2004. doi: 10.1177/1094670504266131.

[22] Seyed Mojtaba Marvasti-Zadeh, Student Member, Li Cheng, Senior Member, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep Learning for Visual Tracking: A Comprehensive Survey. Technical report, 2019.

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

[24] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. 09 2016.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, pages 1–14, 2014.

[27] Rainer Stiefelhagen and John Garofolo. Multimodal technologies for perception of humans, first international evaluation workshop on classification of events, activities and relationships, clear 2006, southampton, uk, april 6-7, 2006, revised selected papers. 01 2007.

[28] Rainer Stiefelhagen, Rachel Bowers, and Jonathan Fiscus. *Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, Revised Selected Papers*, volume 4625. 01 2008. ISBN 978-3-540-68584-5. doi: 10.1007/978-3-540-68585-2.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE CVPR*, pages 1–9, 2015.

[30] Joost Van De Weijer, Cordelia Schmid, and Jakob Verbeek. Learning color names from real-world images. In *Proc. IEEE CVPR*, pages 1–8, 2007.

[31] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. L. Chan, and G. Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 386–393, June 2016. doi: 10.1109/CVPRW.2016.55.

[32] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach, 2018. URL `http://arxiv.org/abs/1812.05050`.

[33] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime

Tracking with a Deep Association Metric. *arXiv e-prints*, art. arXiv:1703.07402, Mar 2017.

[34] Bo Wu and Ram Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. volume 1, pages 951– 958, 07 2006. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.312.

[35] Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers, 2019.

[36] Liang Zheng, Liyue Shen, Lili Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015.

[37] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware Siamese networks for visual object tracking. In *Proc. ECCV*, volume 11213 LNCS, pages 103–119, 2018.

APPENDICES

[1] **Pha A. Nguyen** and Son T. Tran. Tracking customers in crowded retail scenes with Siamese Tracker. In *Proceedings of the 2020 IEEE-RIVF International Conference on Computing And Communication Technologies*, April 2020. **(Oral presentation)**.

APPENDICES

Tracking customers in crowded retail scenes with Siamese Tracker

Pha A. Nguyen
University of Science
Vietnam National University Ho Chi Minh City
Ho Chi Minh City, Vietnam
nganhpha.hcmus@gmail.com

Son T. Tran
University of Science
Vietnam National University Ho Chi Minh City
Ho Chi Minh City, Vietnam
ttson@fit.hcmus.edu.vn

***Abstract*—Object tracking is a fundamental domain in computer vision, especially multiple human tracking, which is an important task to solve many key problems, including traffic counting, behavior analysis of in-store customers. It is very difficult to handle these problems in heavily crowded scenes because of pedestrian occlusions. Recent research works focus on tracking-by-detection, which breaks tracking problem down into two steps: detect objects then associate them with groups by motion or features. Recent object detectors still do not have the accuracy-speed balance and do not work perfectly in a crowded area. In this paper, we propose a new method for human tracking, which is less depended on the instability of the object detector and inherits the success of Siamese Visual Tracking in recent years. While Siamese-family trackers work on a single object, we extend the problem by proposing a strategy that manages across multiple tracker and combines a human detector as a semi-supervisor for tracker location correction. Additionally, the detector-tracker associating strategy adapts the transformation of human poses and the acquisition of new pedestrians during the tracking process. Our method outperforms the state-of-the-art algorithms on our crowded scenes dataset.**

***Index Terms*—Siamese Network, Multiple Object Tracking, Computer Vision**

I. Introduction

The problem of object detection, localization, and tracking has got significant attention in different research areas. Modern trackers could be roughly divided into two branches. The first one is *tracking-by-detection*, which has become the leading paradigm of multiple object tracking. SORT [1] and DeepSORT [2] are popular methods for online tracking algorithms. The simple idea is to employ a frame-by-frame data association using the Hungarian method with an association metric that measures the deep feature distance and bounding box overlap. However, they need an object detector that works all the time.

The other branch is *template-matching*, which uses a template of the target object and tries to match it to the regions of the later images. Several works in the *template-matching* method focus not only on deeper and wider networks, but also on searching as proposing regions to improve tracking robustness and accuracy on a single object. SiamFC [3] uses template embedding as the correlation filter for the search image to allow real-time performance. SiamRPN [4] uses region proposal networks that have been used by Faster R-CNN [5] to extract proposals from the correlation feature maps. Siamese instance search tracking (SINT) [6] trains a Siamese Network using the margin contrastive loss for the representation of features. By using these extracted features, a learned matching function compares the initial patch of the target image with the later patches that are centered at the previously detected location. The patch with the highest positive score is considered to be matching. GOTURN [7] centers around the previous location and regresses the next position of the bounding box, assuming that the object does not move too far [8].

Fig. 1. Sample of our **NgocHa Retail Store** dataset. All frames are marked by the bounding box representing different ID.

Inspired by SORT [1] and DeepSORT [2], we extend the target of Siamese Tracker into multiple object, especially on human, by adding the matching stage periodically after tracking. This approach can be applied to all common types of variations in appearance from tracking examples.

II. Multiple Pedestrian Tracking with Siamese Tracker

In this section, we present our strategy to track pedestrians in detail.

A. Simple Feature Updating

Since human pose changes all the time, we adopt a simple feature update step to an existing tracker when there is a new detected position. At the jth frame, the detect stage returns the cropped image of the ith object (denoted as z_{ij}). Let $\phi(z_{ij})$ denotes the deep convolutional appearance template from the Siamese Network and $Z_{ij} = \phi(z_{ij})$. In order to adapt object appearance, we compute $\phi(z_{ij})$ and update the new template

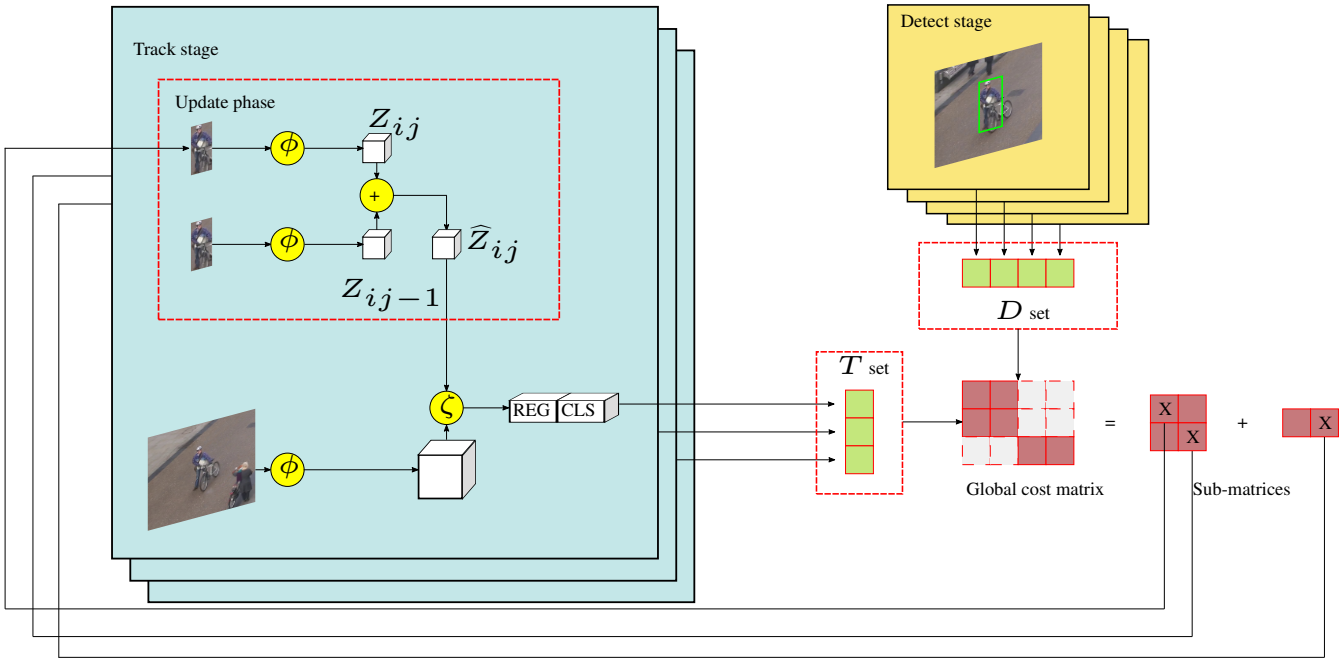

Fig. 2. The detector-tracker associating strategy.

feature with a small coefficient to balance two parts whenever it gets the detector to correct. This action may not always be continuous.

$$\widehat{Z}_{ij} = \begin{cases} Z_{ij} & , j = 0 \\ \gamma \widehat{Z}_{ij-1} + (1-\gamma) Z_{ij} & , j > 0 \end{cases} \tag{1}$$

The update function is a very simple linear function by using previous appearance templates [9]. The update rate γ is assigned to a fixed small value in range $[0, 1]$, following the assumption that the human pose changes quite much. During our experiments, we found that setting γ in the mid-range gives an excellent result.

B. Trackers and detections associating strategy

At any tracking step, the Siamese Network takes x_{ij} as the search field that is samples around the previous position of the target and then predicts the location of the target in the current frame. We employ the Region Proposal Network generation and selection framework that is first introduced in Faster-RCNN [5] and also used in SiamRPN [4]. Let function $\zeta(\widehat{Z}_{ij}, \phi(x_{ij}))$ denotes for the correlation function, which is computed on both two frameworks and returns result in two branches:

$$REG, CLS = \zeta(\widehat{Z}_{ij}, \phi(x_{ij})) \tag{2}$$

Let bounding box regression branch denotes as REG, the classification branch denotes as CLS, and the number of anchor box is k. The REG branch returns the output vector whose shape is $(21, 21, 4, k)$, each point whose shape is $(21, 21, 4)$ represents the distance between bounding boxes generated and a single relative anchor box. Similarly, the CLS branch returns the output vector whose shape is $(21, 21, 2, k)$ and each point in the CLS branch has $(21, 21, 2)$ shape, representing the positive score (denoted as ps) and negative score (denoted as ns) of these bounding boxes.

$$\begin{aligned} REG &= \{(dx_{\alpha\beta\kappa}, dy_{\alpha\beta\kappa}, dw_{\alpha\beta\kappa}, dh_{\alpha\beta\kappa})\} \\ CLS &= \{(ps_{\alpha\beta\kappa}, ns_{\alpha\beta\kappa})\} \end{aligned} \tag{3}$$

where $\alpha, \beta \in [0, 21)$, $\kappa \in [0, k)$.

From the highest positive score index in the CLS branch, we get the next location of the ith object in the REG branch as the same index as the score:

$$\begin{aligned} A, B, K &= argmax(softmax(CLS)[:, :, 0, :]) \\ t_l &= convert\_boundingbox(REG[A, B, :, K]) \\ t_s &= softmax(CLS)[A, B, 0, K] \end{aligned} \tag{4}$$

where t_l is the tracker location and t_s is the tracker score. The 0 index in the CLS branch represents the positive scores ps.

We use the softmax function to normalize each pair (ps, ns) into a vector of two values that follows a probability distribution whose total sums up to 1 and convert the distance with the corresponding anchor to the bounding box of the target. We map a collection of objects whose each object t is tracked as mentioned above in parallel, combine to set T and denote $|T| = n$.

Besides, the object detector detects object bounding boxes (denotes as set D and $|D| = m$) in the following format: $D = \{(x_\alpha, y_\alpha, w_\alpha, h_\alpha) | \alpha \in [0, m)\}$. To match each pair from the tracker predictions set T and detections set D, we compute an intersection over union association cost:

$$c(t_l, d) = 1 - \frac{t_l \cap d}{t_l \cup d} \tag{5}$$

TABLE I
PERFORMANCE COMPARISONS ON TWO DATASETS

Methods	NgocHa Retail Store									
	$frame\_interval = 1$					$frame\_interval = 10$				
	IDF1	IDP	IDR	MOTA	MOTP	IDF1	IDP	IDR	MOTA	MOTP
GOTURN	22.0	22.0	22.0	-4.5	0.349	-	-	-	-	-
SiamRPN	25.9	25.9	25.9	-25.1	0.330	-	-	-	-	-
DeepSORT	64.0	62.6	65.5	94.9	0.033	53.1	52.0	54.2	84.5	**0.084**
Modified SiamRPN (Ours)	**69.8**	**69.9**	**69.8**	**99.8**	**0.000**	**56.1**	**56.5**	**55.7**	**95.6**	0.117
Methods	**TownCentre**									
	$frame\_interval = 1$					$frame\_interval = 10$				
	IDF1	IDP	IDR	MOTA	MOTP	IDF1	IDP	IDR	MOTA	MOTP
GOTURN	28.1	28.2	27.9	-2.1	0.343	-	-	-	-	-
SiamRPN	39.1	39.1	39.1	-12.5	0.376	-	-	-	-	-
DeepSORT	80.5	76.5	**85.0**	87.7	0.019	70.4	69.3	71.6	76.6	**0.104**
Modified SiamRPN (Ours)	**85.4**	**85.8**	**85.0**	**98.6**	**0.000**	**79.9**	**81.8**	**78.2**	**89.4**	0.167

where $t \in T$ and $d \in D$.

The global cost matrix can be considered to formulate as:

$$C(T,D)_{n\times m} = \begin{pmatrix} c(t_{l1},d_1) & c(t_{l1},d_2) & \cdots & c(t_{l1},d_m) \\ c(t_{l2},d_1) & c(t_{l2},d_2) & \cdots & c(t_{l2},d_m) \\ \vdots & \vdots & \ddots & \vdots \\ c(t_{ln},d_1) & c(t_{ln},d_2) & \cdots & c(t_{ln},d_m) \end{pmatrix} = (c(t_{l\alpha},d_\beta)) \in \mathbb{R}^{n\times m} \tag{6}$$

Additionally, the positive score is considered as an important factor to prioritize predictions. We select the subset of trackers $\{t \in T | t_s = score\}$ that have the same score and unmatched subset of detections $\widehat{D}$ that have not been associated with any trackers in the previous matches. Selected $score$ is descending sorted from $max(\{t_s | t \in T\})$ to $min(\{t_s | t \in T\})$. Then the sub cost matrices between each pair of two sets at considered score can be formulated as:

$$\begin{gathered} C(\{t \in T | t_s = score\}, \widehat{D}) \\ \textbf{for } score \in descending\_sort(\{t_s | t \in T\}) \end{gathered} \tag{7}$$

In the implementation, we linearly rescaled values $t_{l\alpha}$ into a new arbitrary range 0 to 100 so that values can be observed easily. After computing sub cost matrices, we also mark certain impossible pairings. By separating global cost matching problem into several sub cost matches, we reduce the computation cost of linear assignment problem from $O(|T|^3)$ to $O(a^3) + O(b^3) + \ldots$ where $a + b + \ldots = n$. The values of $a, b, \ldots$ depend on the degree of occlusion of data scenes.

A linear sum assignment problem solver for dense matrices is applied to get suitable pairs on each matrix. After getting matched pairs, we update detections to trackers as described in (1). On the other hand, trackers that are not matched with any detections are considered to be occluded and still predict shortly until the correction is made or after a long time. Also, the unmatched observations will be initialized as new trackers.

The strategy process flow is depicted in Fig. 2. The Track stage leverages the Siamese Tracker's work and is along with the Update phase to adapt human pose change by a simple update function. $\widehat{Z}_{ij} = Z_{ij} + Z_{ij-1}$ is a simple representation of (1). In implementation, we compute the accumulated $\widehat{Z}_{ij-1}$ instead of Z_{ij-1}. Each object in T set associates with each object in D set by (5). The global cost matrix, which may contain impossible match pairs, is split into many sub-matrices and assignment sum cost is optimized on these matrices. Compatible pairs, which are marked by **X** letter in Fig. 2, will be used to update the status of the objects.

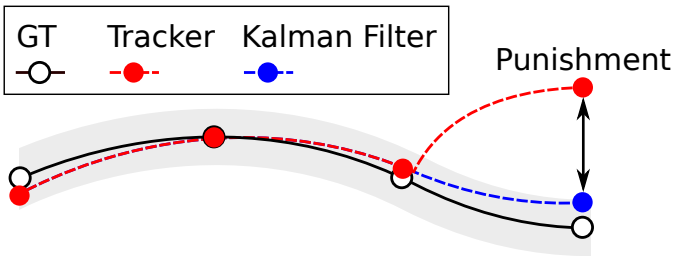

Fig. 3. A punishment occurs when the tracker suddenly changes direction.

C. Kalman Filter Penalty

We break the assignment problem in a global cost matrix into a series of sub-matrices as described in II-B. Additionally, with some uncertainty or inaccuracy in prediction, the rescaled $score$ is multiplied by a penalty to decrease the priority of suddenly direction changed trackers, also expand the score range. We use a simple Kalman Filter with the assumption that those objects are moving at the constant velocity motion. We take the bounding box center position as direct observation of the object state. The distance between the Kalman Filter prediction $p = (x, y)$ and Siamese Tracker prediction center $center(t_l) = (x, y)$ over the frame size (w, h), which map to the interval $[0, 1]$, is computed as the punishment weight:

$$\widehat{t_s} = t_s \times (1 - norm(\frac{center(t_l) - p}{image\_size})) \tag{8}$$

The punishment is illustrated in Fig. 3. For simplicity, we plot ground truth trajectory with solid curve and the predictions with dashed ones. The gray area indicates the matching threshold [10]. The red curve represents the Siamese Tracker, and the blue one is the Kalman Filter output.

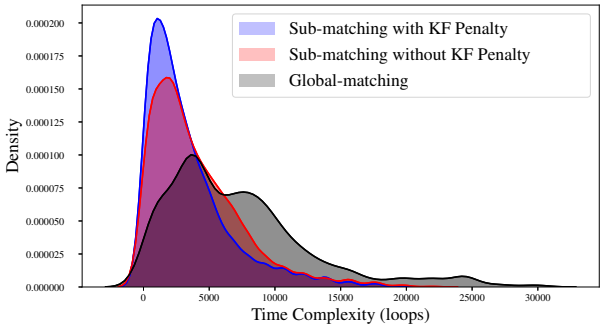

Fig. 4. Histogram time complexity distribution between three types of matching. Best viewed on color display.

D. Training phase

Since all pretrained Siamese convolutional appearance networks (denoted as ϕ) are trained on a large number of object types, we retrain Siamese convolutional network on the **Market-1501** [11] dataset with AlexNet [12]. Each positive pair is generated by 2-permutations of n intra-trackers, where n is the length of a tracker. From 751 trackers, 593876 positive pairs are generated with a large number of human poses, cloth colors. Negative samples are also picked by randomly choosing two people. Data augmentation methods are considered applicable, we apply horizontal flip, grayscale, blur, shift scale, and color distortions on both template images and search images.

We use the loss function in Faster R-CNN [5] to train the network on both the classification and the regression branch. The loss for classification is the cross-entropy loss and smooth L1 loss for regression. Finally, we optimize the two losses combination:

$$L = L_{cls} + \lambda L_{reg} \tag{9}$$

III. Experiments

A. State-of-the-art algorithms comparison

We evaluate our proposed method on two datasets. **NgocHa Retail Store**, which is recorded at retail shopping sites and focuses on heavily occluded humans and unpredictable movement trajectories. Sample is shown as Fig. 1. This dataset contains 38015 human bounding boxes within 7580 images. And **TownCentre** [13] dataset is recorded in a busy town center street. Up to 230 pedestrians were tracked simultaneously in 4500 frames, contains 71460 bounding boxes. Also, we only focus on our matching framework by using ground truth bounding boxes as input. We further compare our method results with the original SiamRPN [4], DeepSORT [2] and GOTURN [7] tracking methods on the MOT metrics [10] as shown in Table I.

Seeing detector as a semi-advisor, we evaluate the tracker predictions within a $frame\_interval$ then get the matching

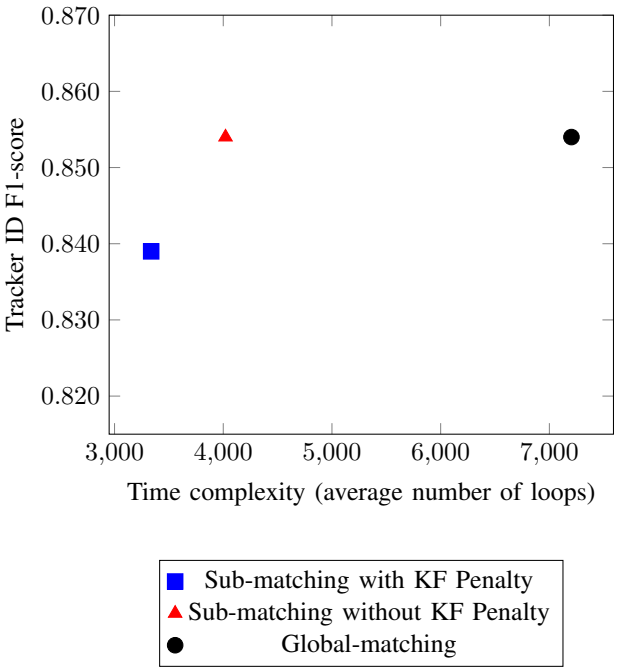

Fig. 5. Complexity and F1-score compares between three methods.

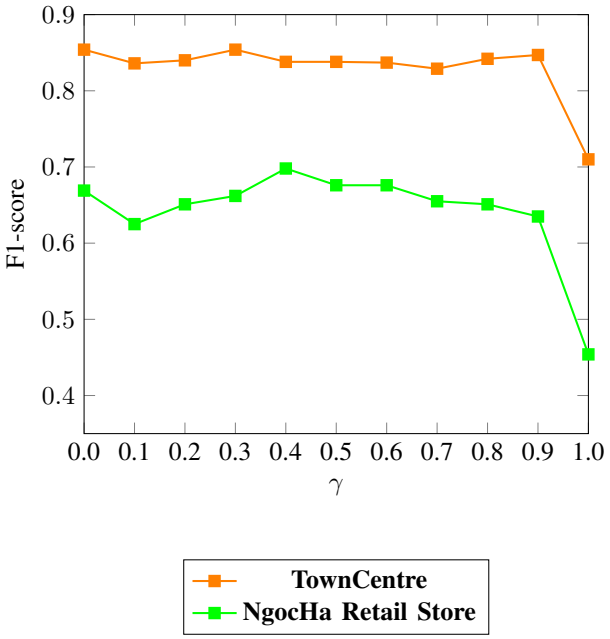

Fig. 6. The change of F1-score is affected by γ. Best viewed on color display.

Fig. 7. Qualitative comparison result on two methods. Red bounding boxes are DeepSORT's visualization and green bounding boxes are the visualization of our method. Our method performs tracking better than DeepSORT after a complete occlusion. The 9th ID changes to the 11th ID, compares with the 8th ID, whose prediction is more stable. Best viewed on color display.

strategy as the correction, so it only occurs on detection-based methods. Since the original idea of GOTURN [7] and SiamRPN [4] only works on a single object, we have to add some modifications to make it work on multiple object. By adding the initialization phase based on the first time new object appears from the ground truth and letting it track till the end without any correction, we see it change ID easily in crowded environments and it is vulnerable at occlusion scenes.

Thanks to the deep convolutional feature of Siamese Network, our method can perform not only a better quantitative result on ID precision, ID recall, ID F1-score, but also better qualitative comparison[1], as demonstrated in Fig. 7.

B. Time complexity

We benchmark the assignment time complexity of our cascade matching strategies on the **TownCentre** [13] dataset as described in II-B. In Fig. 4 and Fig. 5, the blue area and the blue square mark represent the matching strategy which applies with (7) and Kalman Filter penalty (8), the red area and red triangle mark represent the strategy applies with (7) but does not use penalty (8), and the black area and black dot use the naivest approach, which simply assigns on the global cost matrix (6) without penalty (8). The time complexity, which is counted by the number of loops, significantly decreases from the black to the red to the blue. The number of iterations of the highest density point in the blue area is less than the number of two others. However, this results to a trade-off for a slight drop of F1-score as in Fig. 5. In this work, we only benchmark on small dataset. On a large and extreme occlusion dataset, the gap could be wider.

[1]Full tracking visualization by DeepSORT [2] is uploaded at https://youtu.be/Jp-mjmrO0uU, and our modified Siamese Tracker is at https://youtu.be/OeN46gICi2w.

C. The update rate observation

The update rate γ in (1) is crucial to learn a good appearance for human tracking. In Fig. 6, we observe the change of F1-score by the increase of γ in the range $[0, 1]$. By choosing a value which makes two parts balance and focusing on new features, we achieve the highest F1-score on **NgocHa Retail Store** at $\gamma = 0.4$. And with the dataset whose human moving trajectories are stable and linear as in the **TownCentre** [13] dataset, the γ coefficient does not affect to excess, the best result is also obtained when γ is in the mid-range. Oppositely, F1-score decreases dramatically when γ reaches 1. It means that the template feature is only initialized for the first time, and it never gets updated later.

IV. Conclusion and Future Work

With a good method to track in-store customers, these questions are easier to approach: What time is the store most crowded? How much time do customers spend? And how will these affect sales? We focus in this work not only on the real in-store data but also on the efficient method to handle the Siamese Tracker and the object detector in a crowded environment. During the process, the feature extraction branch is trained by a custom human database **Market-1501** [11]. Meanwhile, our tracking system shows good performance in tracking precision and recall.

We remark that the object deep feature might be an important factor to be considered when matching tracker prediction with detection besides the overlapping area. A new branch in the network architecture with a fuse loss function training may be used to perform appearance representation besides classification and regression branch.

In this paper, we trained our backbone on the **Market-1501** [11] dataset, which contains cropped pedestrian images. By collecting more dataset with full-frame availability, we are hopeful that the regression branch can perform better results.

Since several deeper and wider networks are created and boost not only the neural network depth but also the performance of all the computer vision tasks, we also regard them as major improvements in tracking problem. But with the limitation of resources, we only present AlexNet [12] performance in this work. In the future, we will attempt to take full advantage of the neural network.

ACKNOWLEDGMENT

We would like to thank Tuan Hue Thi for his great support and discussion.

REFERENCES

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," *arXiv e-prints*, p. arXiv:1602.00763, Feb 2016.

[2] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *arXiv e-prints*, p. arXiv:1703.07402, Mar 2017.

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," *arXiv e-prints*, p. arXiv:1606.09549, Jun 2016.

[4] B. Q. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8971–8980, 2018.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[6] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese Instance Search for Tracking," *arXiv e-prints*, p. arXiv:1605.05863, May 2016.

[7] D. Held, S. Thrun, and S. Savarese, "Learning to Track at 100 FPS with Deep Regression Networks," *arXiv e-prints*, p. arXiv:1604.01802, Apr 2016.

[8] A. Sauer, E. Aljalbout, and S. Haddadin, "Tracking holistic object representations," 2019.

[9] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," 2019.

[10] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," *arXiv e-prints*, p. arXiv:1603.00831, Mar 2016.

[11] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1116–1124, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[13] B. Benfold and I. Reid, "Coarse gaze estimation," https://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html#datasets, 2009.