南京大學

研 究 生 毕 业 论 文

（申请硕士学位）

论　文　题　目　基于孪生网络的轻量级目标跟踪及应用

作　者　姓　名　姜少魁

学科、专业方向　计算机科学与技术

指　导　教　师　申富饶教授、宋方敏教授

研　究　方　向　计算机视觉

2021 年 5 月

计算机视觉论文集

作　　者：姜少魁
　　李雪健
　　邵玥
　　王绪冬

指导教师：申富饶教授
　　宋方敏教授
　　吴楠副教授

人工智能

论文集

作　　者：卜宇轩
董学文
毛乐坤
赵加成

指导教师：申富饶教授
吴楠副教授

论文集

搜索时间序列推荐强化学习

作　　者：高可攀
郝鸿延
刘凤山
刘雅辉

指导教师：申富饶教授
宋方敏教授

学　　号：MG1833033

论文答辩日期：2021 年 5 月 20 日

指 导 教 师：（签字）

Lightweight Object Tracking and its Application Based on Siamese Networks

by

Shaokui Jiang

Directed by

Professor Furao Shen, Professor Fangmin Song

Department of Computer Science and Technology
Nanjing University

May 2021

Submitted in partial fulfilment of the requirements
for the degree of Master in Computer Science and Technology

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于孪生网络的轻量级目标跟踪及应用

计算机科学与技术 专业 2018 级硕士生姓名：姜少魁

指导教师（姓名、职称）：申富饶教授、宋方敏教授

摘 要

目标跟踪是计算机视觉领域核心的问题之一，在智能视频监控、视频分析、无人驾驶等场景发挥着重要的作用。近些年来，得益于卷积网络相关研究的日益发展，单目标跟踪也迎来了一次新的发展浪潮，基于孪生网络的单目标跟踪方法便是一个很典型的例子。然而，越来越深的网络带来的不一定是精度的提高，可能是更大的模型以及更慢的运行速度，从而使跟踪无法在整个系统中更加高效紧凑地完成工作。所以，针对系统运行设备和速度都受限制的场景，由于主流的跟踪算法无法使用，本文设计了一种轻量级的单目标跟踪网络，其较少的参数量可大大降低模型的体积，方便部署；较快的运行速度可以提高系统的上限，完成更多的工作。

本文设计了一种轻量级的基于孪生网络的单目标跟踪方法FSSiamese，并将其应用在了视频目标检测中。通过与传统目标检测算法相融合，FSSiamese 能够在简单的视频场景下改善目标检测效果。最后，我们将本文提出的算法成功地应用到了实际场景中。本文的主要内容如下：

1. 本文提出了一种全新的单目标跟踪网络Faster and Simple Siamese Network (FSSiamese)。首先，在FSSiamese 中，我们引入了基于通道域的视觉注意力机制，通过对特征图的不同通道赋予不同的权重来代表通道与关键信息的相关度，使网络更加容易关注特定的通道，从而更容易收敛以及达到更好的效果；其次，我们使用了全新的回归方式得到最终目标的位置，新的回归方式可以在减小网络模型复杂度的同时提高运行效率，并且网络的准确度不会受到太大影响。为了更好地训练出FSSiamese，我们设计了一种新的训练样本生成方法，可以实时地生成随机训练样本，节省模型训练所需的空间。

2. 本文将FSSiamese算法成功应用到了视频目标检测中。在巧妙地与轻量级目标检测算法结合之后，FSSiamese对于场景简单、运行环境苛刻且对运行速度有一定的要求的视频检测效果提升较为明显。在FSSiamese的加持下，该方法可以实现1+1大于2的效果，降低目标检测漏报率的同时提高整体的运行速度。

3. 本文提出的FSSiamese算法在实际的系统中也得到了验证，系统运行的结果显示，FSSiamese确实能在具体的应用中发挥良好的作用。

通过实验，本文提出的轻量级单目标跟踪网络FSSiamese相比较于传统的单目标跟踪算法在运行效率上有巨大的提升，并且准确度也较为理想。同时，FSSiamese被应用在了视频检测中，多个实验证明，在简单场景下FSSiamese可以显著提升视频检测效果。最后，FSSiamese在实际的辅助驾驶系统中也发挥了重要作用，可以带来运行速度和运行效果的双提升，充分证明了其应用价值。

关键词： 目标跟踪　孪生网络　轻量级模型

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Lightweight Object Tracking and its Application Based on Siamese Networks

SPECIALIZATION: Computer Science and Technology

POSTGRADUATE: Shaokui Jiang

MENTOR: Professor Furao Shen, Professor Fangmin Song

Abstract

Object tracking is one of the key issues in computer vision, which plays an important part in intelligent video surveillance, video analyse, driverless cars and so on. In these years, thanks to the increasing development of research related to convolutional networks, single object tracking has also ushered in a new wave of development. It is typical that the single object tracking methods based on siamese network. However, deeper and deeper networks not necessarily bring about increases in accuracy. It may be a larger model and a slower running speed, which makes it impossible to complete the work more efficiently and compactly in system. Therefore, for scenarios where the equipment and the running speed are limited, since mainstream tracking algorithms cannot be used, we design a lightweight single object tracking network whose fewer parameters can greatly reduce the size of the model and facilitate deployment, and the fast running speed can increase the upper limit of the system and complete more work.

This paper designs a lightweight single object tracking method, FSSiamese, which is based on the siamese network. We apply it to video object detection, the performance of object detection is improved in simple video scenes by combining the object tracking algorithm with the object detection algorithm. Finally, we successfully apply these two algorithms to actual scenes. The main contents of this paper are as follows:

- This paper proposes a new single object tracking network Faster and Simple Siamese Network (FSSiamese). First of all, in FSSiamese, we introduce a visual attention mechanism based on the channel domain, and assign different weights to the different channels of the feature map to represent the correlation between the channel and the key information, making it easier for the network to pay

attention to a specific channel, so as to make the network easier to converge and achieve better results. Secondly, we use a new method to regress the position of the target. The new regression method can greatly reduce the complexity of the network model and improve the running efficiency, and the accuracy of the network is still excellent. In order to train FSSiamese better, we designed a new method of generating training samples, which can generate random training samples in real time, saving the space required for training the model

- In this paper, the FSSiamese algorithm has been successfully applied to video object detection. This algorithm cleverly combines single object tracking and object detection, and is suitable for simple scenes with poor environments, which has certain requirements for running speed. FSSiamese generates more than simply pooling their efforts together. With the support of FSSiamese, the false negative rate of the object detection can be reduced while the running operating speed be increased.

- The two algorithms in this paper have been verified in the actual system. The results of the system show that these two algorithms can indeed play a good role in specific applications.

Through experiments, the lightweight tracking network FSSiamese proposed in this paper has a huge improvement in efficiency compared with the traditional ones, and the accuracy is also ideal. At the same time, FSSiamese is applied in video detection, and many experiments show that it improves the detection significantly in simple scenes. Finally, FSSiamese also plays an important role in the actual auxiliary driving system, which improves the running speed and running effect. It fully proves its application value.

Keywords: Object Tracking Siamese Network Lightweight Model

目　录

表　格

插　图

第一章　绪论

1.1　研究背景及意义

1.1.1　研究背景

当前，单目标跟踪是计算机视觉中最重要的任务之一，在视频监控[31]、虚拟现实[19]、人机交互、无人驾驶[6]等领域中被广泛应用。传统的单目标跟踪方法主要通过对目标建模得到的目标特征进行跟踪，其运行速度快，但是特征通常难以设计，导致效果不是很理想。随着深度学习的发展以及针对单目标跟踪的一系列大规模数据集的发布，出现了一些性能远超传统跟踪方法的研究，孪生网络相关的跟踪方法便是一个很有代表性的分支。但是，为了获得更好的性能，孪生网络变得越来越深，运行速度越来越慢。这些方法可能在数据集上可以得到很好的表现，但是在实际的应用场景上却很难发挥作用。

另外，辅助视频目标检测是目标跟踪比较重要的用途之一，但由于主流的跟踪算法越来越复杂，运行效率越来越低下，和原本就是复杂任务的目标检测结合起来也并非易事，特别是在实际的应用场景中，更加难以达到理想的效果。因此，设计出轻量级的跟踪算法对于视频检测效果的提升也有很大的帮助，同时也能够使单目标跟踪算法的作用发挥得更加充分。

1.1.2　研究意义

作为计算机视觉的基本任务之一，目标跟踪在学术研究和工业应用中都具有非常重要的地位。例如近些年来比较火爆的无人驾驶，目标跟踪则是其中至关重要的模块，目标跟踪的质量直接决定了整个驾驶系统能否在多变的场景中准确并且快速地作出反应；目标跟踪还被广泛应用于各种视频分析中，通过分析目标的轨迹，挖掘出目标信息。例如在安保方面，目标跟踪能够发现人员的可疑行动轨迹，进行自动锁定以及报警。在日常生活中，这种分析也可以被用作交通规划，通过分析路口人员流动情况，为城市路线以及交通流动等提供数据支持；另外，目标跟踪也经常与其他视觉任务配合，发挥重要作用。例如，在某些场景下与目标检测相结合，可以有效提高目标检测效果。或者和步态识别相结合，使其识别更加精准。

同时，随着目标跟踪研究的逐渐深入，越来越多的研究专注于提高目标跟踪的准确度，而忽视了其在应用场景中的使用。因为在大部分场景中，目标跟

踪都需要与其他视觉任务相结合，单纯地追求跟踪的高指标带来的往往是复杂的网络结构以及缓慢的运行速度，从而在实际的应用场景，特别是实时场景中可能无法获得太高的收益。因此，轻量级的目标跟踪模型是很有必要的，特别是对于一些边缘场景，比如机器人或者移动设备上，现如今好多精准度高的跟踪算法由于硬件限制无法发挥作用。对此，另一种解决方案是将模型部署在云端，但这样仍然不能从根本上解决问题。第一，是因为不能解决跟踪算法运行速度慢的问题，反而由于网络延迟等原因会带来更长的处理时间；第二是由于此方法对运行环境有一定要求，需要在跟踪时有网络连接，假设系统到了没有网络的环境则不能够正常工作，这种情况会使得整个系统的安全性和可用性大幅度下降。

现如今大部分应用场景使用的跟踪算法都属于多目标跟踪，一方面是因为大部分场景下，目标数量足够多，多目标跟踪更加容易满足这些场景的需求；另一方面是单目标跟踪需要耗费一定的资源，如果给每个目标分配一个单目标跟踪器来达到多目标的效果，这样的计算量和资源的消耗是呈倍数上升的，即使进行并行化加速也会造成计算资源的大量消耗，这导致在目标较多的场景单目标跟踪无法使用。上述这些原因都造成了单目标跟踪在实际应用中的潜力没有被完全发挥出来，然而，我们发现很多场景下目标数量少，用单目标跟踪反而可以取得更好的效果。这是因为多目标跟踪更多的是在目标检测的基础上做关联，单目标跟踪相比起来对检测的依赖程度较低。这些场景下使用单目标跟踪可以获得更好的效果，同时，使用轻量级的单目标跟踪网络可以使系统整体的运行速度得到提升。

1.2　研究现状与难点

单目标跟踪的核心任务在于检测出目标在当前帧所处的位置，其难点是该目标形状、位置可能会实时发生变化。为了很好地解决该问题，从传统方法时代到现如今，出现了许多跟踪算法，其种类繁多、效果各异。我们一般将其分为两大类方法，第一类属于生成类算法，第二类属于判别类算法。生成类算法主要是传统方法居多，由于其缺点过于明显，现在已经很少被使用；判别类算法是现在被研究最多、使用最广泛的。但判别类算法也可以被分为若干分支，我们这里主要按滤波类算法和深度学习算法这两种最具代表性的判别类算法讨论其现状及难点。

（1）生成类算法：主要是在当前帧找出目标可能出现的位置，分别计算其出现在该处的可能性，然后根据这些计算结果确定最终的目标位置；（2）滤波类算法：训练得到目标模板，利用该模板与要搜索的范围进行运算，最大响应

值即为当前的目标位置；（3）深度学习算法：利用深度学习方法，通过训练而来的模型回归得到目标最终的位置信息。

另外，也存在很多目标跟踪算法符合不止一个上述分类的描述，比如某些滤波类算法会使用深度学习训练好的特征，这种特征往往比传统手工设计的特征效果更好。

1.2.1　生成类算法

生成类算法最根本的特征在于事先确定若干目标可能出现的位置，这些位置根据先验知识决定并且符合某种概率分布，同时需要一个足够优秀的算法确定该位置有多大概率正是最终的结果，然后不断重复这个过程。最典型的传统单目标跟踪算法均值偏移[69]、粒子滤波[58, 68]等都是通过这样的方式来实现目标跟踪。

粒子滤波中，在每一帧都需要先撒粒子，粒子的数目是超参数之一，通常粒子数目越多，找到目标正确位置的概率越大，即跟踪的结果越准确。最后通过计算每一个粒子与目标的相似度来确定当前帧目标的位置，同时相似度也用来为下一次撒粒子提供先验信息，均值偏移也是类似的步骤。

生成类算法最大的问题在于需要对每一个待选区域计算其为目标在当前帧位置的可能性，为了保证最终的跟踪效果，判断的模型不可能过于简单，因此每一次计算都是对计算资源和时间的消耗，更多的计算次数意味着更多的资源开销以及更大的运行成本。况且，这些代价都是跟踪单个目标在某一帧中所付出的，这样的开销对于跟踪系统而言肯定是难以接受的；另一方面，一个能准确计算出候选区域是最终目标位置概率的模型也是难以实现的。利用一些传统特征，例如直方图等速度虽然较快，但准确性和鲁棒性都较差，无法满足跟踪需要。后面有些工作，例如ASMS[62]和DAT[51]，在颜色直方图的基础上加入了尺度不剧变和可能偏最大作为正则项，一定程度上提高了准确度，但效果和滤波类方法和深度学习方法相比仍具有较大差距。该类方法还有另一个比较大的问题，即很难确定要生成多少候选区域，例如粒子滤波中每次要撒多少粒子，以及均值偏移中要生成多少偏移向量，这些都是需要依靠经验以及结合具体场景得到的，这也给生成类算法添加了很大的不确定性。

1.2.2　滤波类算法

滤波类算法全称为相关滤波（correlation filter或者discriminative filter）。其主要方法是将目标所在图像块训练为相关滤波器，然后对当前帧搜索的区域进行滤波操作，得到一张置信图或者响应图，这张图最大值对应的位置正是当前

帧目标的坐标位置。之后通过新位置的图像块，更新相关滤波器。为了加快滤波操作的运行速率，相关滤波器的检测、训练和更新使用离散傅立叶变换实现，整个过程在频域执行。

传统的相关滤波算法存在的问题主要是特征选择和表示。最初的MOSSE[9]使用灰度像素值作为图像特征，但灰度值表达的信息有限，且在背景相对复杂，颜色与目标相似的情况下很难区分目标。后来的KCF[28]引入了多通道特征表示，特征包括多种颜色特征以及对边缘信息敏感的方向梯度直方图特征（HOG）[14]，利用自适应降维策略，将多维特征进行压缩，加快了运行速度。但KCF在实际场景中的鲁棒性一般，特别在识别非刚性物体的形变上，效果并不如意。

后续出现了一些改进滤波器特征的工作，例如引入新的专门设计过的特征提高跟踪性能，比如判别颜色描述子（DD）[57]、多颜色通道提取的方向梯度直方图（MC-HOG）[70]等等。还有一些工作将深度学习的特征引入进来，例如融合深度特征的空间正则化判别相关滤波（DeepSRDCF）[16]、基于卷积特征的相关滤波算法（CF2）[47]等等。这些深度特征通常可以获得较好的性能，但过于复杂的特征网络会给跟踪带来较多的资源消耗。

1.2.3　深度学习类算法

虽然深度特征经常被应用在其他滤波类跟踪算法上，但这并没有完全发挥出深度网络强大的适应能力与拟合能力。因此，很多研究针对目标跟踪设计了端到端的网络，可以直接利用深度网络回归出目标在当前帧的位置。

较早的工作例如GOTURN[26]，通过增广数据集增加跟踪器的鲁棒性，利用卷积网络和全连接层回归目标的坐标。GOTURN网络结构简单，运行速度较快，然而准确率和鲁棒性都较低，和很多相关滤波类算法相比并无太大优势。

牛津大学提出的SiamFC[7]网络，打开了孪生网络实现目标跟踪的大门。利用卷积操作实现搜索检测的思想也非常巧妙，并且跟踪器的效果也来到了一个新的高度。随后，商汤和中科院做了一系列围绕孪生网络的工作，包括SiamRPN[41]、DaSiamRPN[72]、SiamMask[63]等等，他们将目标检测和语义分割的工作借鉴到了目标跟踪中，并通过大规模的数据集训练，得到了非常好的效果。然而，随着网络逐渐加深，虽然效果有小幅度提升，但其运行速度有明显下降，对于整个跟踪任务而言，牺牲运行速度去换取小幅度的精度明显是不明智的。另外，将其他视觉任务的复杂结构转移到单目标跟踪上来，并没有充分发挥孪生网络的优势，因为直观上来讲大部分视觉任务都是为多目标而生的，而单目标跟踪明显不需要如此大费周章，我们后面的实验结果也会证明这

一点。

1.3 研究内容

本文在孪生网络的基础上，设计了一种结构更加简单、运行速度更快并且能够保证跟踪效果的单目标跟踪网络FSSiamese。为了充分发挥FSSiamese的优势并且增加其实用性，本文将其应用在视频目标检测中，大大提高了视频的检测率。FSSiamese也被成功地应用到了实际的生产环境和场景中，验证了算法的有效性。其中本文的主要研究内容如下：

- 本文提出了一种新的基于孪生网络的单目标跟踪网络Fast and Simpler Siamese Network（FSSiamese）。首先在FSSiamese中，为了更好地跟踪到目标在当前帧的所在位置，我们开创性地将视觉注意力机制引入到了孪生网络结构中，其结构简单，给网络增加的开销基本可以忽略不计，但可以大幅度提高跟踪的效果以及网络的训练速度。为了简化网络结构，实现轻量级的目标跟踪网络，FSSiamese使用了全新的回归方法，并设计了新的训练样本生成方法来与之相匹配，使其能更好地发挥作用。VOT评测指标上的结果显示，FSSiamese在大幅度提高运行速度的同时，仍然能够保持良好的跟踪效果。

- 本文将FSSiamese跟踪网络应用于视频目标检测算法中，可以在具有少数目标的简单视频场景中呈现出非常显著的效果，同时将单目标跟踪的优势完全发挥了出来。该方法主要分为两个阶段，第一阶段是通过目标检测确定场景中有哪些目标，并可以自动区分不同帧之间的相同目标，将目标信息保存，为跟踪阶段提供输入；第二阶段是运行本文中提出的FSSiamese算法，根据第一阶段的输入对各个目标进行跟踪，由于单目标跟踪的连续性可以大大降低检测算法的漏检率。我们在多个数据集上进行了实验，也都证实了该方法的可行性。

- 本文提出的轻量级孪生网络被成功地应用在了实际的系统环境中。在辅助驾驶系统中，FSSiamese算法被封装起来作为系统的一个独立模块，实际场景中的结果显示FSSiamese跟踪效果良好，并且提高了系统的运行速度，这也充分说明本文提出的轻量级跟踪网络在实际的生产环境中，也发挥了巨大的作用。

1.4　本文组织结构

本文主要研究了基于孪生网络的目标跟踪算法，提出了轻量级的单目标跟踪网络FSSiamese，并将其应用在了视频目标检测和实际的辅助驾驶系统中。全文共分为六章：第一章是绪论，主要介绍了本文的研究背景和意义，以及目标跟踪的一些现状和存在的问题；第二章介绍了目标跟踪相关的研究工作；第三章主要介绍本文提出的单目标跟踪网络的设计细节以及实验；第四章主要介绍了如何将FSSiamese应用在视频检测中，包括具体流程以及实验；第五章主要介绍本文提出的轻量级跟踪网络在实际场景中的应用；第六章总结全文，并对未来工作进行了展望。

第二章　相关工作

单目标跟踪作为计算机视觉的基本任务之一，在很多场景下发挥着重要的作用，其本身也是个富有挑战性的问题。目标跟踪通常在各种各样的应用场景中出现，根据实际场景的不同，跟踪本身或许存在着些许差别。为了方便描述，我们将单目标跟踪定义为：一个输入为连续图片序列或者视频，给定目标的起始位置，输出为后序每一帧中该目标的位置，如图2.1所示。

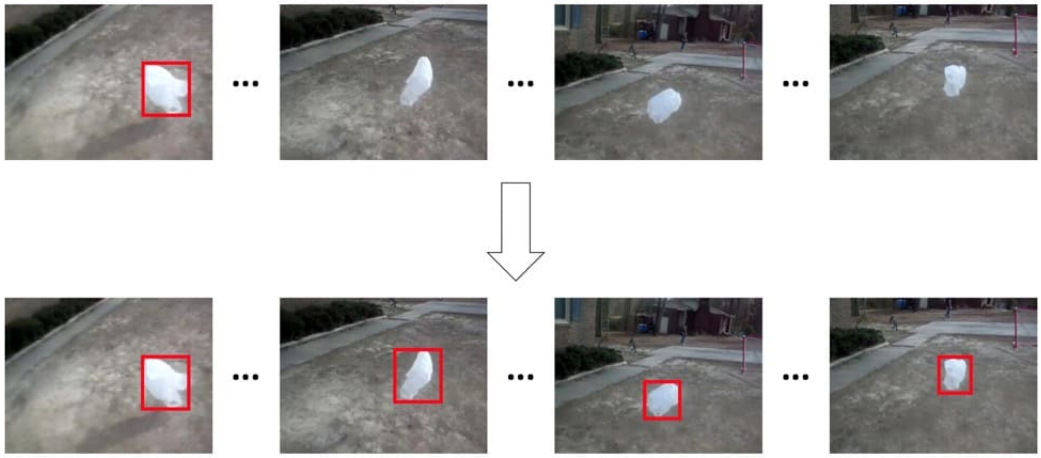


图 2.1: 单目标跟踪的输入输出

早期已经有很多针对单目标跟踪算法的研究。本章将主要介绍传统的目标跟踪算法，以及最近几年比较主流的滤波类以及深度学习类的单目标跟踪算法，分析这些算法的优势及局限性。

2.1　传统单目标跟踪算法

2.1.1　传统单目标跟踪原理

在滤波类和深度学习类的目标跟踪算法出现之前，大部分跟踪算法的原理是利用传统视觉特征结合搜索算法以达到目标跟踪的目的。通常，先根据一定的规则确定要搜索的若干区域，然后设计判别算法判断目标存在于这些位置的可能性，最后根据这些可能性计算出当前帧目标的位置，并且更新算法的参数，为下一次跟踪做准备。传统视觉特征包括颜色特征、灰度特征、梯度特征、边

缘特征等等。搜索算法常见的有卡尔曼滤波、粒子滤波、均值漂移等等。后面两个小节，会分别介绍这些内容。

2.1.2　传统视觉特征

颜色特征属于全局特征，通常在视觉中用颜色空间（如RGB，HSV等）的值表示物体的颜色，但是通常不会直接把物体的颜色像素值当作特征，因为图像的像素值较多，如此多的特征值难以被模型学习，所以一般会通过一定的统计方法来挖掘颜色信息当作特征。例如颜色直方图，将图像颜色分为多个小区间（bin），统计落在每一个小区间的像素个数，最后将统计结果画成直方图，该直方图的数值，即每个bin里面的频数组合起来正是一个多维的直方图特征。

颜色特征的优点是对于描述难以进行分割或者对位置不敏感的图像效果较好，缺点也很明显，该特征只能反映图像的颜色分布，但无从得知图像颜色所对应的位置。

灰度特征和颜色特征类似，不过颜色特征通常的来源是多维的颜色空间，而灰度特征只有一个大小范围在0到1之间的灰度值。灰度特征常被直接当作特征使用，在一些简单的应用中能够发挥很好的作用，例如MNIST手写数字的识别[39]。除此之外，灰度特征可以引申出区域灰度变化特征（Haar特征），灰度特征的优点是实现简单、特征稳定、计算速度快，即使稍微复杂一些的Haar特征也可以通过计算积分图进行加速；缺点是灰度值丢失了颜色信息，只保留了大概的明暗信息，表达能力较差。

边缘特征是指包含了图像边缘信息的特征。首先对图像（通常来说是灰度图）进行边缘检测，通常图像像素发生剧烈变化的位置正是图像边缘，然后将原图边缘对应位置和非边缘对应位置表示在一张图上构成了边缘特征，

梯度特征是指利用图像梯度分布作为特征描述图像，不同的实现方式有着不同的梯度计算方法，常见的有SIFT[45]、SURF[5]、HOG等等。

2.1.3　搜索算法

卡尔曼滤波用于在动态系统中，通过处理观测值并对噪声进行估计，从而在一定程度上纠正观测值，使其更接近真实的数据。卡尔曼滤波被应用在很多系统中，例如航天中飞行器的引导和跟踪，机器人运动的规划和控制等等。图2.2所示[1]是使用卡尔曼滤波对观测到的速度进行校正。 Dae-Sik Jang等人[30]在1998年提出了一种使用卡尔曼滤波进行跟踪的动态模型，但该模型在

[1]图片来自于https://www.kalmanfilter.net/alphabeta.html

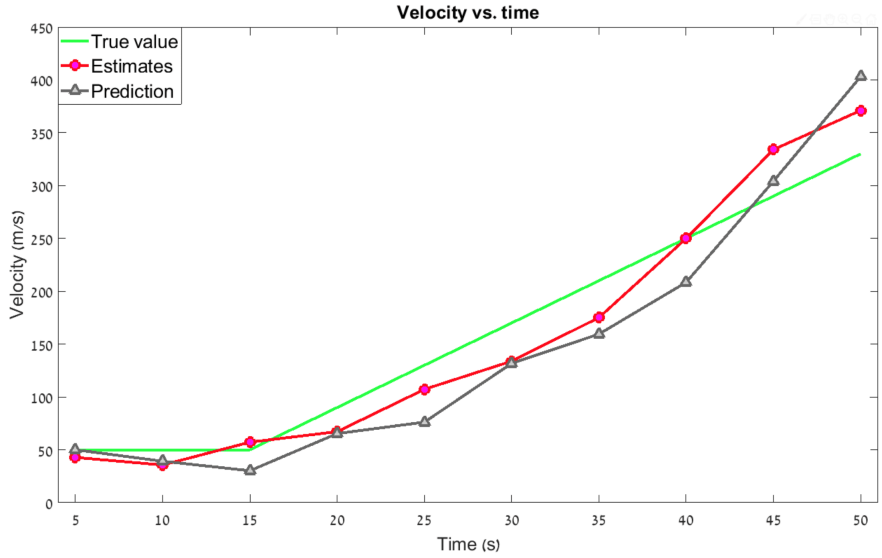


图 2.2: 利用卡尔曼滤波矫正速度检测

非理想场景下效果堪忧。为了解决该问题, Dae-SikJang在几年后针对人体跟踪设计了结构化的卡尔曼滤波器，该卡尔曼滤波器由区域卡尔曼滤波器（cell Kalman filter）和联合卡尔曼滤波器（relation Kalman filter）两部分组成。其主要思想是将目标切分为若干区域，区域卡尔曼滤波负责预测每个子区域的运动信息，联合滤波器负责将相邻两个区域之间的滤波器运动信息结合起来。结构化卡尔曼滤波器提出了一个很好的跟踪思路，不过缺点是过于复杂，使用场景容易受到限制，很多情况下很难按照某种标准将特征按区域划分成块。

另外，有一些针对人体跟踪的研究，里面融合了多种跟踪算法，卡尔曼滤波器在这些文章中发挥了一定的作用。Shiuh-Ku Weng等人[64]提出的适应性卡尔曼滤波跟踪器中同样构造了动态模型，相对于传统卡尔曼滤波，引入了遮挡率（occlusion ratio）来自适应预测参数，该算法与其他传统算法相比稳定性更好，对有移动物体、光照改变等场景也有不错的效果。但由于卡尔曼滤波器固有的性质，该类算法更适合符合高斯分布的系统，对于非线性甚至非高斯的运动系统效果则会大大折扣。

Fukunaga等人[21]在1975年的一篇估计概率密度梯度函数的文章中提出了均值偏移（Mean Shift），均值偏移通过不断迭代的方法在概率密度函数的局部最大值收敛。随后有文章对均值偏移做了推广，一是定义了核函数，使得偏移点的距离和对应偏移向量的贡献成反比；二是通过权重值代表不同样本点的重要性不同。

经过改进，均值偏移开始被应用在人脸的跟踪中，CamShift算法[11]从目标

区域提取出颜色直方图特征，对于跟踪的每一帧图像计算概率分布图，概率分布图中的信息用来表示目标是否包含当前像素，然后通过Mean Shift均值偏移算法不断计算搜索框的中心和大小，收敛时的结果正是目标在该帧的跟踪结果。Camshift之后，均值偏移被推广到更加广义的目标跟踪任务中，2000年Dorin Omaniciu等人[12]在CVPR上发表的一篇文章发现Mean Shift在目标部分遮挡以及目标处于复杂场景下有着不错的效果。

随后的一系列工作主要是针对颜色直方图容易受背景干扰做的一系列优化，例如加入纹理特征，与RGB特征组合成联合直方图[3]。但这种联合直方图的弊端之一是维度过高，影响算法速率，王路等人[4]利用多个特征直方图建立目标生成模型，从而降低直方图维度。

均值偏移用作目标跟踪有着很多优点，例如实现简单、速度快、参数少等等，但由于均值偏移算法限制了其特征只能是类似于直方图的形式，并且搜索的是局部最优值，因此对于快速移动或者形状外观发生突变的场景，很难发挥作用。

为了解决这个问题，随后的一系列研究提出了更多样化的特征，例如Maggio E等[48]在颜色特征的基础上融入了方向特征，颜色直方图加上梯度直方图能够应对大部分复杂背景，但如果背景发生变化，该类方法就会失效。金志刚等[2]将更复杂的SURF特征和颜色特征图组合起来进行跟踪，类似的还有杨智雄等[1]将运动特征与核函数直方图融合起来，这些方法在一定程度上提高了跟踪的鲁棒性。

然而这些改进并没有从根本上缓解粒子滤波的缺点带来的问题，较大的计算量带来的系统负担是不可忽视的，另外同其他传统算法一样，这些跟踪方法对特征的选择依赖较大，做到在不同场景下稳定、准确地跟踪仍然具有很大的困难。

2.2　滤波类跟踪算法

2.2.1　滤波类算法发展过程

相关滤波（correlation filter或者correlation）用来在一个场景下生成目标的响应图，其中响应图峰值对应的位置正是目标在原图中的位置，响应图值越低的区域对应原图中区分于目标的背景。

相关滤波的方法在很早之前就开始被使用，最早可以追溯到上世纪80年代。这些滤波器包括SDF[50]、MVSDF[37]、MACE[49]、OTF[65]、MSESDF[38] 等等。这些滤波器通过具有变化外观的样本进行训练，为了使响应图峰值足够高，

这些训练通常是在约束较强的条件下进行的。但是这种滤波器的最大问题是对于形变不够敏感，例如MACE类的方法，响应图都有很高的峰值，但目标发生较大形变时跟踪效果会变差。

为了解决这个问题，一些研究指出要去掉强约束进行训练，结果表明这种方法确实很好地缓解了形变引起的问题。不论是强约束还是弱约束，这些方法训练时只指定了峰值，ASEF[10]提出了应该将整张图的响应值都加入训练，该方法在人眼定位以及行人检测上取得了很好的效果。

然而上面这些研究没有被广泛的重要原因是它们需要大量的在线训练，这对目标跟踪的速度影响是致命的。MOSSE是如今主流相关滤波类方法的开山之作，提出了使用最小平方和误差来训练滤波器。同时，MOSSE和其他滤波器一样，利用快速傅立叶变换将图像和滤波模版转化到频域内进行计算，但MOSSE改进了滤波器的求解形式，使其能够更加快速和鲁棒地跟踪目标。在MOSSE代表相关滤波取得目标跟踪的较好效果之后，很多基于该方法的研究涌现出来，两年之后，CSK[27]将循环矩阵结构和核方法引入了MOSSE方法中，同时也在最小化函数中首次加入了正则项，防止滤波器过拟合，CSK虽然速度不及MOSSE，但准确度来到了一个新的层次。

后续的研究基本都是沿着这个方向，KCF则是将MOSSE的单通道灰度特征改进为多通道梯度特征，另外加入了密集采样。CN[18]则是在CSK的基础上扩展了多通道颜色特征等等。这些方法都是在MOSSE的基础上进行改进，总体的框架并没有发生改变。

前面的工作都是基于传统特征，随着深度学习的发展，越来越多的滤波算法将传统特征与深度特征融合起来，现如今准确率高的相关滤波跟踪算法基本都是这种思路。具有开创性质的研究当属Martin的CCOT算法[17]，CCOT在原有传统特征的基础上，使用VGG[60]提取图像深度特征，然后将不同层的结果通过插值操作扩展到连续的空间域中，CCOT的跟踪效果相比之前传统特征有了明显的提升。ECO[15]对CCOT做了一些小的改进，一方面是CCOT由于特征众多导致运行速度过慢，ECO通过对这些特征降维，加快运行速率；另一方面优化模型训练，通过高斯混合模型对样本分类，使样本更加多样化，最后迭代的模型鲁棒性更强。之后的UPDT[8]则指出，深度特征和浅层特征应该区分开来，深度特征的高层激活鲁棒性强，对各种形变有着很好的适应能力，能够更强地定位到目标，但缺点是由于抽象程度高，空间分辨率低，无法确定目标的准确位置。而浅层特征与之相反，虽然鲁棒性低但是适合高精度定位目标。UPDT将两种特征区分开来，分别训练一个响应图，最后将响应图做融合。从CCOT开始，相关滤波类的跟踪算法来到了一个新的高度，在目标跟踪的竞

赛中取得了很好的成绩。

2.2.2　滤波类算法原理

上一节提到的CCOT以及之后的算法在原理上比CCOT之前的相关滤波算法更加复杂，同时算法中包含了很多细节以及技巧，但是万变不离其宗，这些方法并没有改变相关滤波最基本的算法框架。下面会介绍相关滤波的基本原理。

假设有两个信号f和g，互相关表示他们之间的相关性，其值越大，相关性越强，意味着两个信号越相似。相关性通常分为连续信号和离散信号，用以下公式计算：

$$(f \otimes g)(\tau) = \int_{-\infty}^{\infty} f^*(t)g(t+\tau)dt$$

$$(f \otimes g)(n) = \sum_{-\infty}^{\infty} f^*[m]g(m+n)$$

在图像领域，f表示输入图像，h表示目标，g代表计算出来的响应结果，二者做卷积操作有：

$$g = f \otimes h$$

为了加快运算速度，会将上式做傅立叶变换，卷积操作变为点乘操作，极大地减少了计算量：

$$F(g) = F(f \otimes h) = F(f) \cdot F(h)^*$$

通常简写为：

$$G = F \cdot H^*$$

因此对于跟踪任务而言，目的是得到足够好的h或者H^*，相关滤波的算法也是围绕求解最优的H^*展开来，例如MOSSE提出通过最小平方和误差来求解，也就是：

$$min_{H^*_{wv}} \sum_{i=1}^{m} |H * F_i - G_i|^2$$

KCF与之类似，但定义形式不太相同，并且加入了正则化项，定义训练集为（x_i，y_i）线性回归函数，公式如下：

$$\min_w \sum_i (f(x_i) - y_i)^2 + \lambda ||w||^2$$

网络训练时需要计算的正是回归函数里面的权重w。同样地，这个过程需要用傅立叶变换转化到频域进行计算，并且KCF中引入的核函数将线性函数映射到非线性空间，提高了模型的表达能力，因此最后求解的是傅立叶变换之后核函数下的w。

2.2.3　深度学习类目标跟踪

传统跟踪算法以及相关滤波类跟踪算法的难点之一是选择合适的图像特征，这些图像特征至少需要满足两点要求，一是特征能够充分表达图像中的关键信息，因为大多数跟踪算法其实都是在一个局部图像中搜索匹配目标的过程，这要求图像特征在表示有限的像素内更好地识别到目标信息，才能在各种复杂的场景以及非刚性物体的跟踪中达到较好的效果；二是特征不能过于复杂，过于简单的图像特征固然难以表达复杂的图像信息，这一点显而易见，然而过于复杂的图像特征不一定能够达到预期的效果，并且特征过于复杂，特征提取以及特征相关的运算都需要花费更多的时间和资源，这对于大部分单目标跟踪任务而言代价过大。

因此使用传统特征的跟踪算法在选择特征时都足够谨慎，这些特征的选择通常也是大多数算法最核心的区别。下表列举了若干使用传统特征的跟踪算法以及所选择的特征：

表 2.1: 跟踪算法及使用的特征

跟踪算法	使用的特征
CFLB、MOSSE、CSK	灰度图
ASMS	颜色直方图
Struck	哈尔特征
KCF、DCF	方向梯度直方图/图像像素
CN	多颜色通道特征
SAMF	方向梯度直方图+多颜色通道特征
BACF	多通道方向梯度直方图
Staple	模板特征+颜色统计特征
DAT	颜色统计特征

在深度学习发展起来之后，卷积层带来的深度特征在计算机视觉的各个领域被广泛使用。由于相关滤波类算法的设计较为灵活，很多研究将深度特征引入到了相关滤波之中，例如SRDCF[16]中使用的特征是梯度方向直方图特征，DeepSRDCF将其替换为了ImageNet-VGG-2048网络的第一层，算法的其他部分与SRDCF完全相同，然而算法的跟踪效果有了较大提升。表2.2汇总了若干使用深度特征的相关滤波算法以及其使用的卷积层特征。

表 2.2: 跟踪算法及使用的深度特征

跟踪算法	使用的卷积层
DeepSRDCF	Imagenet-VGG-2048 Layer1
HCF	VGG-Net-19 Conv5-4, Conv4-4, Conv3-4
CCOT	VGG Layer0, Layer1, Layer5
ECO	VGG Conv-1, Conv-5

同传统特征相比，深度特征设计起来相对容易一些，普遍做法是选择比较通用的骨干网络，对不同的骨干网络以及不同的卷积层进行测试，根据测试的结果判断出所选卷积层是否提取出了合适的特征，进而确定网络的结构。这个过程自然要比选择传统特征，甚至是手动设计的特征容易得多，能够大大降低算法难度，简化算法结构。

深度特征另一大优点是效果通常比较鲁棒，传统特征大多是根据研究者相关经验设计得来，比较符合人类的视觉特点，但不一定带来很好的效果。而深度特征则是从模型训练的角度，目标正是模型获得最好的效果，因此使用深度特征获得更优解的可能性更大。但这也是深度学习的弊端所在，传统特征以及其相互间的组合使得模型具有很强的可解释性，研究者也可以根据不同的场景不同的任务选择合适的特征。而深度特征只能使用数据去训练，表现不好时无法对特征本身进行任何调整，当然不只是深度特征，这是深度学习整个领域存在的问题。

还有一点便是深度特征带来的模型复杂度问题，传统特征例如颜色直方图、方向梯度直方图等都大多属于运算量较小的特征，深度特征带来的往往是更大的模型，这意味着参数量更大并且特征提取的时间更长。例如前面提到的DeepSRDCF，虽然相比SRDCF效果得到了很大的提升，但运行速度比之前慢了若干倍。但这一点并不是无解，ECO的运行速度便很快，这得益于ECO筛选之后的特征以及降低了的模型更新率，因此如何巧妙地使用深度特征也至关重要。

深度特征除了应用在相关滤波类跟踪算法中，越来越多的工作将单目标跟踪的过程设计为端到端的深度学习网络。通过使用目标跟踪数据集对模型进行训练，使模型能够在图像中直接回归出目标所在的位置，不同于相关滤波类算法，深度学习类目标跟踪算法的训练过程完全离线，不需要在推断阶段对模型进行更新，从而大大提高了模型在推断阶段的运行效率。由于网络基本均为端到端网络，其离线训练的步骤也被大大简化，从另一个角度来讲，深度学习类

的目标跟踪算法将目标跟踪完全转变成了模型回归问题。

GOTURN是出现较早并且非常典型的深度学习类目标跟踪算法，从结构上看，GOTURN是足够简单的，仅仅使用了若干卷积层和全连接层便完成了每一帧的跟踪任务。GOTURN没有使用主流的骨干网络作为特征提取器，而是自行设计了若干卷积层分别对当前帧和上一帧的图像进行特征提取。特征提取完成之后，网络将两帧提取到的特征进行级联操作，然后通过若干全连接层，将其转化为最终目标在当前帧的位置。

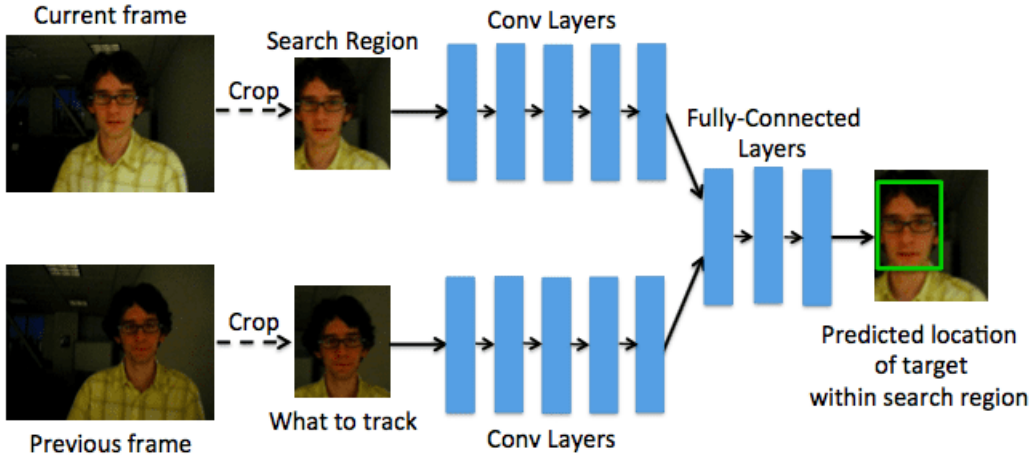


图 2.3: GOTURN结构图[26]

GOTURN的效果并不是非常理想，但速度较快，更加重要的是这种端到端网络实现跟踪的思路打开了深度学习类跟踪的大门。牛津大学提出的SiamFC网络，如图2.4所示，和GOTURN 结构大体类似，输入都是两个分支，分别进行特征提取后进行融合处理，然后通过特定结构回归得到最后的位置。然而SiamFC更加容易训练，并且跟踪效果要远好于GOTURN，这就归功于SiamFC中的几个特殊的技巧。

（1）特征提取网络采用孪生网络进行参数共享，GOTURN两个分支特征提取网络虽然相同，但是没有使用同一组参数，而使用同一组参数可以让网络更加容易训练，实验证明也可以得到更好的跟踪效果；（2）提取特征之后没有直接进行级联，而是使用卷积操作，有些类似于相关滤波类算法的处理方式；（3）没有直接回归位置，而是回归出一张响应图，对应目标在当前帧对应位置的可能性；（4）使用多尺寸回归目标大小，跟踪结果更加准确。

SiamFC提出的这几点基本为后续孪生网络类目标跟踪算法奠定了基础，中科院大学王强等人在此基础上提出了SiamRPN网络，巧妙地将目标检测算

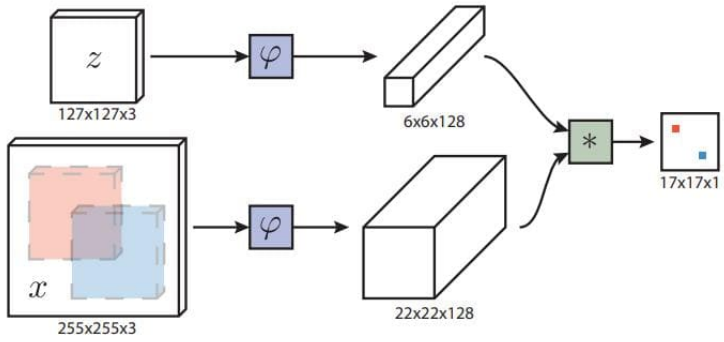


图 2.4: SiamFC结构图[7]

法Faster-RCNN[56]中的RPN网络移植到了孪生网络当中。之前SiamFC只是通过回归响应图大致确定目标位置，然后使用多尺度方法拟合目标大小，该方法不容易过拟合但是对于大小变化频繁的目标容易丢失，解决方法是使用更多尺度来回归目标，但运行速率会随着尺度的增加明显下降。另外，使用这种方法无法精准地回归目标的位置，因为特征图的大小有限，回归之后等比例放大可能会存在位置精度不够的问题。

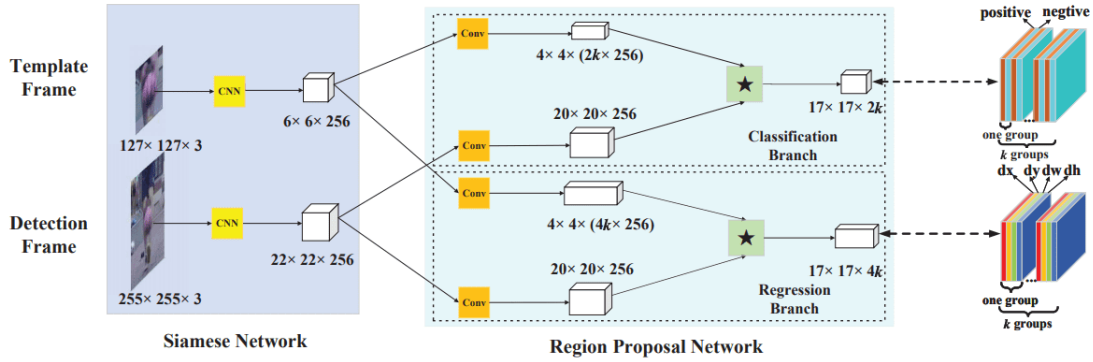


图 2.5: SiamRPN结构图[41]

SiamRPN使用了更加复杂的方式回归目标的位置，与目标检测中的RPN网络基本完全相同。特征图中每个点不仅仅是目标中心的概率信息，而是通过卷积生成另外两个特征图，通道数分别为$2k$和$4k$，其中k表示的是在特征图的每个点对应的锚框个数，如图2.6所示。对于特征图上的每个点，每个锚框都要回归出正负背景的可能性，因此共有$2k$个值；回归位置同理，每个点对应的是目标与每个锚框位置的偏移，位置需要4个变量来确定，因此共有$4k$个值。

在推断阶段，得到了最终的两个特征图之后，也就得到了原图对应位置每

种锚框所对应的前后景概率以及与目标位置的偏差。然后选择前景概率最高的若干锚框作为正样本，在另一个特征图中找到对应位置以及对应锚框的偏差值，得到了不同锚框估计到的目标位置，最后利用NMS算法将这些位置处理成最终的目标结果。区域候选算法由于包含前后景概率信息，所以可以通过前后景概率大小来判定有无目标，再加上有很多锚框参与回归位置，这使得跟踪结果更加稳定。但同时结构更复杂意味着更大的计算量和更慢的运行速度，因为原本的RPN网络是被用来进行目标检测的，通常情况下图像中会有多个目标出现，而单目标跟踪和检测有很大的区别，因此SiamRPN的结构依然有很大的优化空间。

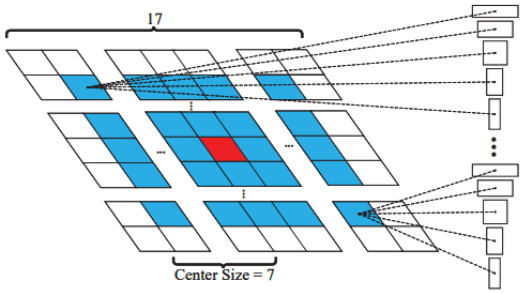


图 2.6: 特征图对应的锚框[41]

SiamRPN取得了很好的效果，在当年的VOT测试中拿到了非常好的名次，无论跟踪的准确度还是跟踪器的运行速度都相当出色。后续该团队又在此基础上进行了一系列改进，主要包括SiamRPN++[40]与DaSiamRPN。SiamRPN++探讨了使用更深的特征提取网络为何会引起跟踪效果下降，为了解决这个问题，SiamRPN++在训练样本上做了额外的工作，即对于每个正样本在其搜索区域的中心位置随机偏移一段距离，通过这种方式减轻更深的特征提取网络带来的过拟合问题。SiamRPN++使用Resnet50[25]作为骨干网络，为了充分利用深网络的图像特征，对不同特征层的特征通过Depthwise Cross Correlation 融合在一起，如图2.7所示，这样的优点是能够可以简化网络结构并且更加容易训练。

DaSiamRPN则是在训练数据集中做文章，在之前的ImageNet-VID数据集[59]中加入了视频数量更多的YOUTUBEBB[52]数据集，以及部分ImageNet的静态检测图片，丰富数据集物体的种类，使训练出来的网络更加鲁棒。另一方面，在训练样本中加入了更加复杂的负样本，增强网络的判别能力，提高模型的泛化性能。

另外还有一些其他关于孪生网络的目标跟踪算法，大致思路都相同。例如图2.8中的SiamVGG[43]将SiamFC中的骨干网络替换成了VGG-16。

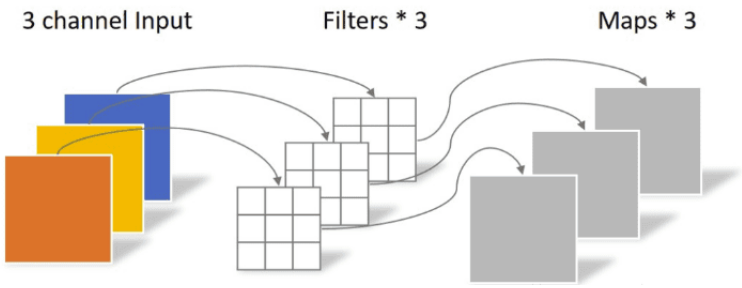


图 2.7: Depthwise Cross Correlation特征融合

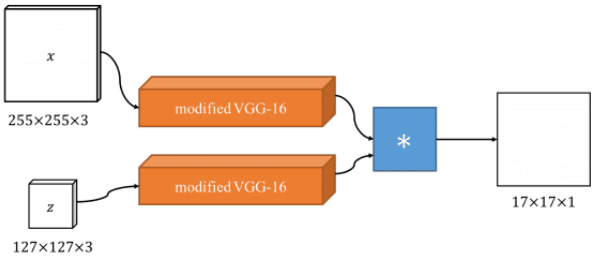


图 2.8: SiamVGG结构图[43]

图2.9所示的SiamMask将图像分割加入了孪生网络当中，利用分割训练出更加精细的跟踪网络，使得物体框变成可旋转矩形，跟踪精度得到了进一步的提升。

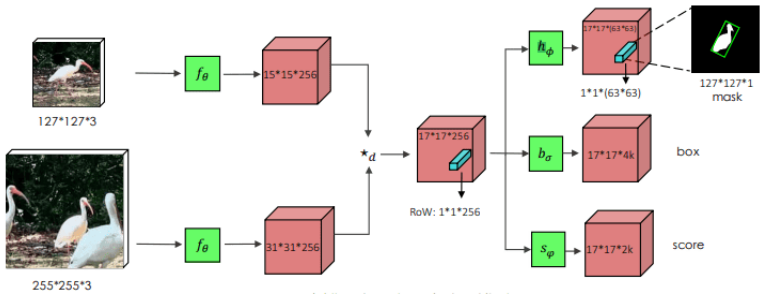


图 2.9: SiamMask结构图[63]

随着相关研究的进行，目标跟踪效果逐渐达到了一个新的高度。比较具有代表性的是以ECO为代表的滤波类方法和以SiamMask为代表的深度学习方法，其准确率相比于传统的目标跟踪方法有了显著提高。表2.3是ECO和SiamMask在普通终端PC的运行速度对比，运行环境如表2.4。

从表中可以看出来，虽然这些方法在测试指标上达到了较为理想的效果，但是算法复杂度和运行速度却并不理想。表2.3的结果是在配置低端的PC 得到

表 2.3: ECO和SiamMask在PC上的运行速度

跟踪算法	运行速度（**fps**）
ECO	5.9
SiamMask	6.0

表 2.4: 终端PC环境配置

GPU	处理器
GeForce 840M	i7-4510U CPU @ 2.00GHz

的，如果运行环境是机器人或者无人机之类，得到的结果可能会差。另外，目标跟踪通常是作为系统的一部分发挥作用，如果其模型过大或者跟踪速度过慢，必定会成为整个系统的短板，影响整个系统的工作。如果将整个跟踪系统部署到云端，即处理数据时将输入信息发送给专门的服务器，整个跟踪工作在服务器处理完成，最后将跟踪结果回传到本地。这样可以一定程度上缓解本地的计算压力，但其运行速度可能仍然遇到瓶颈，这取决于服务器端的配置。一方面会由于服务器端的部署造成成本上升，另一方面对于整个系统的运行会增加对网络的需求，如果网络不稳定或者是在根本没有网络的情况下，整个系统将无法使用。因此，目标跟踪在准确运行的同时，能够满足轻量级、快速运行的要求也很有必要。

为了满足上述要求，本文设计了一种轻量级的快速的单目标跟踪网络，并在视频检测和辅助驾驶系统中都发挥了重要的作用。

第三章　基于轻量级孪生网络的跟踪模型FSSiamese

在前面的绪论部分和相关背景介绍中，我们提到了当前的单目标跟踪算法过于追求指标，而忽视了跟踪的性能，导致其在实际的应用中受到更多的限制。针对这一问题，本文提出的FSSiamese模型在原有孪生网络架构的基础上，设计了新的网络架构，其结构简单，容易部署，在拥有良好运行速度的同时仍保持着较高的准确度。

本章共分为四个部分，第一部分先从整体架构上介绍FSSiamese的网络结构，然后分别就特征提取子网络、视觉注意力模块、卷积模块、回归模块这四个子模块介绍其详细结构以及发挥的作用；第二部分会介绍FSSiamese的训练以及如何结合现实场景中的视频使用FSSiamese进行跟踪；第三部分会在单目标跟踪公认的两个测试标准上运行FSSiamese，并将与其他优秀的跟踪器进行分析对比；最后一个部分是本章小结。

3.1　FSSiamese整体架构

FSSiamese整体架构如下图，主要由以下四个部分构成：（1）特征提取网络：提取图像的特征，为后续模块的操作提供基础；（2）视觉注意力模块：给特征图不同通道赋予不同的权重，强化特征的表达能力；（3）卷积模块：将两个分支的特征图进行卷积操作，模拟图像的搜索过程；（4）回归模块：由一个1×1大小的卷积核和全连接层组成，将卷积的结果转化为用来表示目标位置的向量。

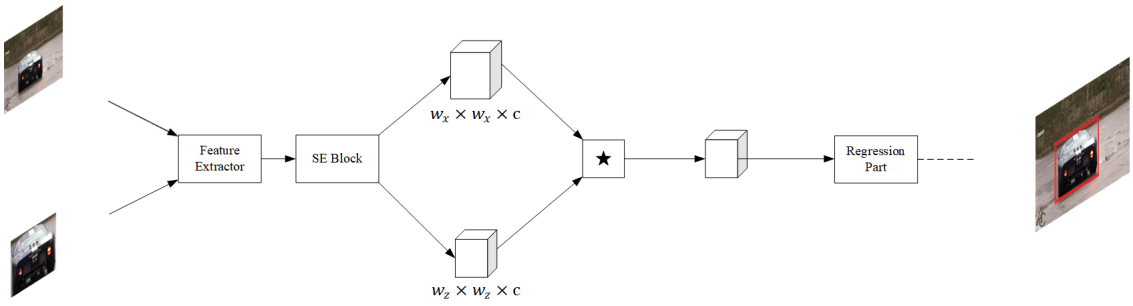


图 3.1: FSSiamese整体结构图

如图3.1所示，我们将上面搜索区域所在的分支称为搜索分支，目标作为输入所在的分支称为模板分支。结构和参数完全共享的特征提取网络分别将两个

分支的RGB三通道图片作为输入，搜索分支输出大小为$w_x \times w_x \times c$的特征图，模板分支输出的大小为$w_z \times w_z \times c$。然后将这两张特征图分别输入到同样是结构和参数共享的视觉注意力模块（SE模块）中，输出与各自的输入分别保持一致。$\star$代表卷积操作，即模板分支输出的特征图在搜索分支的特征图上做卷积操作，模拟了在搜索区域寻找目标的过程。由于这里做的是分层卷积，输出的特征图大小为$(w_x - w_z + 1) \times (w_x - w_z + 1) \times c$。最后，回归模块将输出的特征图转化为搜索区域上目标所在的相对位置信息。整个网络的训练过程是完全离线的，这意味着在进行跟踪的过程中不会涉及模型的更新训练，每两帧之间除了位置信息，其他都是完全独立的。

3.1.1　特征提取网络

特征提取器网络包括两个结构和参数完全共享的分支，模板分支将目标的图像作为输入，本文中用z表示；搜索分支将当前帧的搜索区域图像作为输入，文中用x 表示。特征提取器旨在提取适当的特征，从而在回归位置时得到准确的结果。我们将提取特征的操作写作F，两个分支的输入在经过特征提取网络之后的输出分别为$F(z)$、$F(x)$。

本文中特征提取器的网络设计为可更换的，主流的骨干网络都可以使用，例如AlexNet、Resnet18、Resnet34、SqueezeNet等等。只需要注意网络的复杂程度，通过实验可以看到越复杂的骨干网络最后效果越好，不过复杂网络的模型大小和运行时间都会增加，这里本文使用的是Resnet18，根据测试，Resnet18能够在准确度和运行速度上达到一个很好的平衡。

3.1.2　视觉注意力机制

受到SE（Squeeze-and-Exciation）网络的启发，我们在特征提取网络后面引入了SE模块。SE模块作为一种可以进行特征调整的注意力机制，可以使网络训练速度加快以及得到更好的效果。SE模块结构较为简单，只包括一个平均池化层、两个全连接层和一个激活函数层，并且两个分支的SE模块同特征提取网络一样，结构相同并且参数共享，如图3.2所示。

SE模块将特征提取网络的输出作为输入，其输出与输入的特征图尺寸完全一致。我们将SE模块的变换记为S，有：

$$\begin{cases} F_x = S(F(x)) \\ F_z = S(F(z)) \end{cases} \tag{3.1}$$

其中F_x表示搜索分支的输出，F_z表示模板分支的输出。

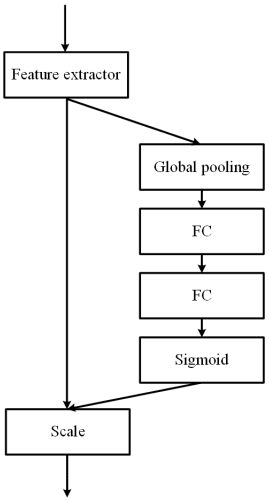


图 3.2: SE模块结构图

3.1.3　卷积模块

在分别得到模板分支和搜索分支的特征图之后，将模板分支的输出F_z作为卷积核，在搜索分支的输出F_x做步长为1的分层卷积操作。分层卷积指的是对应通道分别进行卷积，和传统卷积对应位置所有通道值相加不同，每个通道的结果都要保留下来。公式如下：

$$Conv_c(F_x, F_z) = F_x^c \star F_z^c \tag{3.2}$$

其中，F_x^c 表示F_x中第c个通道的特征图，F_z^c 表示F_z中第c个通道的特征图，$\star$表示卷积操作。经过特征提取模块和SE模块之后，搜索分支的输出F_x大小为$w_x \times w_x \times c$，同样地，模板分支的输出F_z大小为$w_z \times w_z \times c$。卷积操作在每个通道分别进行，最后的输出记做M，大小为$(w_x - w_z + 1) \times (w_x - w_z + 1) \times c$，里面包含着目标在搜索区域内的位置信息。

3.1.4　基于小卷积核和全连接层的回归模块

回归模块基于卷积模板的结果生成目标在搜索区域内的相对位置，这是跟踪步骤中最重要的部分，也是其他基于孪生网络的跟踪算法最大的不同之处。通常不同的跟踪算法有不同的回归方式，例如SiamFC将卷积后的结果作为热度图来定位目标在该搜索区域的位置，SiamRPN用Faster-RCNN中的RPN网络生成多个回归框来回归目标的x、y、w、h以及前后景的可能性。

在我们的回归模块中，我们提出了全新的回归方式，结构更加简单，如图3.3所示。

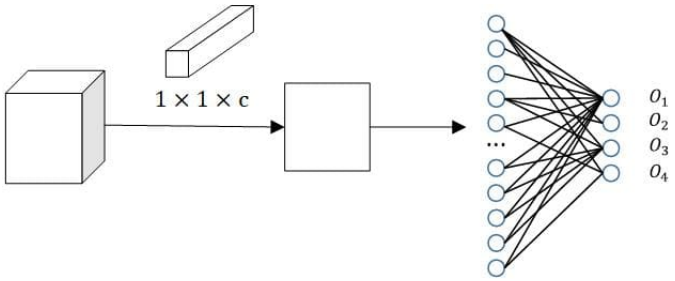


图 3.3: 回归模块结构图

我们使用一个1×1大小的卷积核在M上做卷积，这相当于对M的每一个通道重新训练一个权重，之后按权重相加起来，压缩成单通道的特征图。之后再将其展开为一维的向量，训练一个全连接层，输出为目标在搜索区域内的相对坐标，我们记做$[O_1, O_2, O_3, O_4]$。最终目标在搜索区域的位置计算如下：

$$\begin{cases} x_1 = O_1 * w_f + w_f/2 \\ y_1 = O_2 * h_f + h_f/2 \\ x_2 = O_3 * w_f + w_f/2 \\ y_2 = O_4 * h_f + h_f/2 \end{cases} \tag{3.3}$$

其中，x_1, y_1, x_2, y_2 分别代表搜索区域中目标左上角和右下角的x坐标和y坐标，h_f 和w_f 代表搜索区域的高和宽。

3.2 FSSiamese的训练与使用

3.2.1 训练过程

在训练阶段，我们选择了ImageNet-VID和GOT[29]两个数据集生成检测帧和模板帧的样本。其中检测帧对应网络输入的搜索分支，里面的图片正是搜索目标的范围；同样地，模板帧对应网络输入的模板分支，里面的图片是跟踪的目标图像。为了使模型足够鲁棒，我们每个训练样本都由不同的检测帧和模板帧组成，这些训练样本全部是从两个数据集中的视频中随机生成出来的。

GOT提供了一个包含多种目标，并且专门用来训练目标跟踪器的数据集。GOT中移动物体的种类超过560种，运动模式高达87种，一共有1万多个视频片段，标注的物体框多达150多万；ImageNet-VID包含30种物体标注，3000多个视频片段。这样的数据集规模可以保证生成差别足够大的训练样本。

上述数据集提供了大量的图片序列，对于序列中的每张图片，都有标注目标在该帧中的位置。在生成训练样本时，我们并不是直接使用这些图片以及标

注，而是通过固定的步骤从每个图片序列中生成若干样本，大致步骤如图3.4 所示。每次样本的生成会在选定图片序列中挑选两张不同的图像，一张生成模板帧，另一张生成搜索帧。

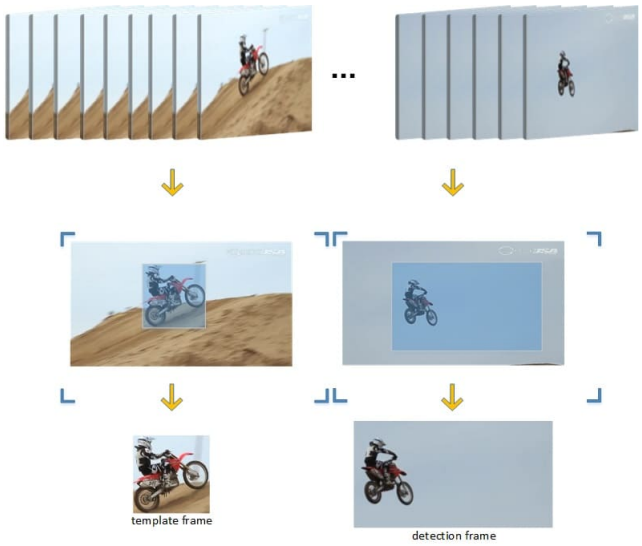


图 3.4: 训练样本生成示意图

在这几步中，最重要的是搜索帧图像的生成。为了使网络更加充分地学习到目标的位置，搜索帧中目标的位置需要具有一定的随机性，本文设计了一种新的生成算法生成搜索帧图像。生成算法的大致思路是能满足网络收敛的前提下，尽可能地丰富训练样本中的目标在搜索范围内出现的位置。虽然在实际的跟踪过程中，相邻两帧之间物体的位置不会发生太大变化，但如果训练样本的分布与实际跟踪的分布完全相同，会造成由于偏差过小导致网络过拟合。因此要保证搜索帧中目标出现的位置尽可能随机并且偏差足够大，以及搜索帧区域与目标的尺寸比例也有一定的变化。经过我们的尝试，我们使用的方式是先确定搜索帧图像的大致范围，然后在该范围内生成包含目标的图像。

大致范围的确定有三种情况，其边界分别由公式3.4、3.5和3.6确定。

$$\begin{cases} \hat{left} = max(0, x_1 - w_b) \\ \hat{right} = min(x_2 + w_b, w_f) \\ \hat{top} = max(0, y_1 - h_b) \\ \hat{bottom} = min(y_2 + h_b, h_f) \end{cases} \tag{3.4}$$

$$\begin{cases} \hat{left} = max(0, x_1 - w_b/2) \\ \hat{right} = min(x_2 + w_b/2, w_f) \\ \hat{top} = max(0, y_1 - h_b) \\ \hat{bottom} = min(y_2 + h_b, h_f) \end{cases} \tag{3.5}$$

$$\begin{cases} \hat{left} = max(0, x_1 - w_b) \\ \hat{right} = min(x_2 + w_b, w_f) \\ \hat{top} = max(0, y_1 - h_b/2) \\ \hat{bottom} = min(y_2 + h_b/2, h_f) \end{cases} \tag{3.6}$$

为了更加形象地说明上述三个公式所代表的含义，我们绘制了图3.5来展示三个公式分别对应生成的搜索帧图像。

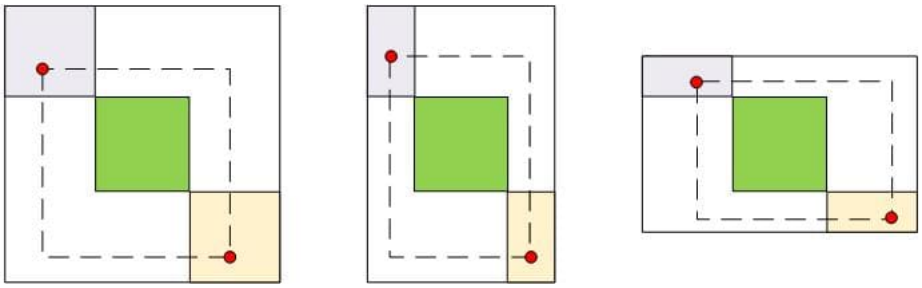


图 3.5: 三个公式对应的搜索帧图像生成示意图

其中绿色区域为搜索帧图像中目标的位置，搜索帧区域的左上角顶点在灰色区域中随机生成，右下角顶点在黄色区域中随机生成。三个公式分别代表了三种不同比例的搜索帧图像，并且目标的位置随机性更强。确定了生成范围，顶点的具体位置通过3.7得到。其中，标注x_1表示目标左上角的x值，x_2 表示右下角的x值，y值同理，w_b和h_b代表该目标的宽和高。

$$\begin{cases} left = random(\hat{left}, x_1) \\ top = random(\hat{top}, y_1) \\ right = random(x_2, \hat{right}) \\ bottom = random(y_2, \hat{bottom}) \end{cases} \tag{3.7}$$

在确定了搜索帧图像的两个顶点之后，按照虚线裁剪下来便是最终的搜索帧图像，具体算法步骤见算法1。

算法 1 FSSiamese训练样本生成算法

输入: 图片序列fm，其中$fm_{i,j}$表示第i个图片序列的第j帧，一共有m个图片序列，每一个序列长度为l_i，每一个序列中图片的宽和高分别为w_{fm_i}和h_{fm_i}

输出: 样本集合$Samples$，大小为N

初始化结果集合$Samples = \emptyset$
for $i = 1 : m$ **do**
    while $Samples$大小小于N/m **do**
        随机选取两帧fm_{i,r_1}，fm_{i,r_2}
        if r_1与r_2相差大于100帧 **then**
            continue
        end if
        将fm_{i,r_1}中的目标裁剪下来，作为模板图片z
        在公式3.4、3.5和3.6中随机选择一个，通过计算得到$\hat{left}$、$\hat{right}$、$\hat{top}$、$\hat{bottom}$和确定裁剪的大致范围
        确定了裁剪的大致范围后，根据这个范围通过公式3.7随机生成搜索区域
        按照left、top、right、bottom的值在fm_{i,r_2}上裁剪作为搜索图片x，在x中，目标的左上角和右下角坐标用$\hat{x_1}, \hat{y_1}, \hat{x_2}, \hat{y_2}$表示，宽和高分别为$\hat{w_f}, \hat{h_f}$
        $O_1 = \frac{\hat{x_1} - \frac{\hat{w_f}}{2}}{\hat{w_f}} = \frac{\hat{x_1}}{\hat{w_f}} - \frac{1}{2}$
        $O_2 = \frac{\hat{y_1} - \frac{\hat{y_f}}{2}}{\hat{h_f}} = \frac{\hat{y_1}}{\hat{h_f}} - \frac{1}{2}$
        $O_3 = \frac{\hat{x_2} - \frac{\hat{w_f}}{2}}{\hat{w_f}} = \frac{\hat{x_2}}{\hat{w_f}} - \frac{1}{2}$
        $O_4 = \frac{\hat{y_2} - \frac{\hat{y_f}}{2}}{\hat{h_f}} = \frac{\hat{y_2}}{\hat{h_f}} - \frac{1}{2}$
        $Samples \leftarrow \{x, z, [O_1, O_2, O_3, O_4]\}$
    end while
end for

在算法1中，输入是所有用来生成样本的视频或者图片序列（后面我们统一称为“图片序列”），输出则是大小为设定值的样本集合。图片序列中除了图像，还包含每一帧中目标在图像中的具体位置信息以及图像本身的一些属性，例如图片的分辨率等信息。对于每一组图片序列，首先判断是否已经生成了足够数量的样本，如果是，则跳过当前图片序列，处理下一个图片序列；否则，在当前图片序列中随机挑选间隔超过100帧的两张图像，这里的间隔可以根据参数具

体调节，如果大部分图片序列长度较短，可以适当降低该值。接下来的工作便是从这两幅图像中生成一组样本。其中一张图像用来生成模板帧，只需要根据图像的标注将目标裁剪出来即可；另一张图像生成搜索帧的过程要稍微复杂一些，从公式3.4、3.5和3.6中随机选择一个确定裁剪边界的顶点范围，这三个公式分别代表不同长宽比的裁剪结果，在确定顶点范围之后，利用公式3.7在顶点和目标边界之间随机确定搜索帧的左上、右下两个顶点，然后裁剪即可。至于生成样本的标注信息，正是目标在搜索帧中的顶点坐标与搜索帧中心的偏差值，只需要简单运算即可得出。

在生成足够多的训练样本之后，便可以开始训练本文提出的网络。损失函数我们选择SmoothL1损失函数，batch大小为80，学习率为0.001。损失函数计算方式如下：

$$Smooth_{L1}(x,\sigma)=\begin{cases}0.5\sigma^2x^2,|x|\leq\frac{1}{\sigma^2}\\|x|-\frac{1}{2\sigma^2},|x|\geq\frac{1}{\sigma^2}\end{cases}\tag{3.8}$$

因此，对于每个batch，最终的损失为：

$$loss=\sum_{i=1}^{n}Smooth_{L1}(O_i^\star-O_i,\sigma)\tag{3.9}$$

其中，$O_i^\star$表示网络的输出值。损失函数是根据实验验证得到的，在对比了平方损失函数和L1损失之后，L1损失的结果要稍微好一些，但二者差距不是很大。L2损失同L1损失相比，更加鲁棒一些，不容易受异常值影响。在我们的训练样本中，样本多样性相比其他端到端跟踪网络要更强一些，因此我们推测这可能是L1损失表现更好的原因所在。选择SmoothL1而不是普通L1损失的原因是，当预测值与真值差距足够小时，SmoothL1的导数更小，网络收敛会更加稳定一些。

3.2.2　推断阶段

在推断阶段，对于跟踪过程中的每一帧，我们将前面提到的模板帧和搜索帧作为网络的输入，输出是计算出来的相对位置$[O_1,O_2,O_3,O_4]$.整个算法的流程见算法2。

算法 2 FSSiamese跟踪算法

输入: 图片序列fm，其中fm_j表示当前图片序列的第j帧，len表示当前图片序列的长度，序列中图片的宽和高分别为w_{fm}和h_{fm}，标注x_{i_1}表示第i 个图片序列中第一帧目标左上角的x值，x_{i_2}表示右下角的x值，y值同理

输出: 每一帧目标的位置Pos

初始化结果集合$Pos = \emptyset$
初始化目标起始位置$x_1 = x_{i_1}, y_1 = y_{i_1}, x_2 = x_{i_2}, y_2 = y_{i_2}$
根据x_1, y_1, x_2, y_2裁剪目标作为模板帧z
for $j = 2 : len$ **do**
    $l = max(0, x_1 - (x_2 - x_1) * \delta)$
    $t = max(0, y_1 - (y_2 - y_1)) * \delta)$
    $r = min(x_2 + (x_2 - x_1) * \delta, w_{fm})$
    $b = min(y_2 + (y_2 - y_1) * \delta, h_{fm})$
    根据l, t, r, b裁剪图像作为搜索帧x
    将z和x传入FSSiamese网络，得到结果$[O_1, O_2, O_3, O_4]$
    更新目标位置：
        $x_1 = x_1 + O_1 * (r - l)) + (r - l)/2$
        $y_1 = y_1 + O_2 * (b - t)) + (b - t)/2$
        $x_2 = x_2 + O_3 * (r - l)) + (r - l)/2$
        $y_2 = y_2 + O_4 * (b - t)) + (b - t)/2$
    $Pos \leftarrow \{x_1, y_1, x_2, y_2\}$
end for

FSSiamese跟踪算法同其他端到端跟踪算法一样，使用起来较为方便。首先从图片序列的第一帧中裁剪模板帧保存下来，模板帧在整个跟踪的过程中一直保持不变。从第二帧开始，根据上一帧目标的位置，将搜索区域扩充为长和宽分别是原来的$1 + 2\delta$倍。不同于在生成训练样本时搜索范围的大小比例随机生成，在推断阶段搜索区域与上一帧区域目标的比例是固定值，本文中我们的δ取值为1.45，该值与训练样本生成时定义的范围有一定关系，也需要通过对应的测试确定该值。

3.3　实验与分析

VOT和OTB是单目标跟踪最常用的两个评测标准。我们在下面两个小节分别介绍两个评测标准的指标详情，包括指标的内容以及如何进行计算，然后我

们会将本文提出的FSSiamese在这两个评测指标上进行测试，并和其他优秀的跟踪器做分析对比。最后，我们还测试了不同跟踪模型的尺寸大小，以及使用不同特征提取网络对FSSiamese效果以及模型大小造成的影响。

3.3.1 VOT数据集实验

3.3.1.1 测试指标

评价跟踪器是否优秀并不是一件容易的事，这取决于很多因素。例如，在大部分应用场景中，目标跟踪只是其中的一个模块，在这种场景下，对跟踪器是否能够在长时间下完成跟踪没有太高要求。所以，我们这里选择的都是VOT的短时跟踪测试，我们使用的指标有Accuracy、Robustness、EAO和EFO。

Accuracy用来衡量跟踪结果A_t^T和真值A_t^G的平均重叠率，第t个时刻的值可以通过公式3.10计算。

$$\phi = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T} \tag{3.10}$$

如果跟踪器丢失掉了目标（重叠率为0），为了充分利用测试数据，这时候VOT系统会在跟踪失败5帧后重新初始化跟踪器，但为了公平起见，初始化之后10帧的重叠率不计入Accuracy指标。

Robustness指标用来衡量跟踪器跟踪失败的平均次数（VOT2015[34]是个例外，表示的是总次数，其他都是平均次数）。N个测试视频的Robustness指标可以用公式3.11计算。

$$Robustness = \frac{1}{\sum l_i} \sum_{0}^{N-1} f_i \tag{3.11}$$

其中，f_i和l_i表示第i个视频的失败次数和视频长度。

EAO是在VOT2015提出来的一个新指标，计算起来比前两个稍微复杂一些。简单来说，EAO建立了一条关于视频长度和原始准确率（$\hat{\Phi_{N_s}}$）的曲线，其中N_s指测试视频的长度。$\hat{\Phi_{N_s}}$ 可以通过在长度为N_s的视频集合上求Φ_{N_s}的平均值获得。EAO指标$\hat{\Phi}$是曲线关于区间$[N_{lo}, N_{hi}]$的平均值, 公式如3.12所示。

$$\begin{cases} \Phi_{N_s} = \frac{1}{N_s} \sum_{i=1:N_s} \phi_i \\ \hat{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s = N_{lo}:N_{hi}} \hat{\Phi_{N_s}} \end{cases} \tag{3.12}$$

EFO（Equivalent Filter Operations）是用来衡量跟踪器运行速度的指标。EFO首先计算在特定大小的图像上做滤波操作所花费的时间，通过计算速度与该值的比值来消除因为硬件不同造成的速率差异。EFO能够从一定程度上反

应跟踪器的速度，但并不完全准确，在VOT2017[33]之前一直用这个指标。到了VOT2017，VOT提出了实时实验，去掉了EFO指标。实时实验的主要思路是要求跟踪器在特定时间内给出跟踪结果，如果超出规定时间，则系统用上一次跟踪的结果作为当前帧的结果，失败的情况和之前的处理方式一样，重新初始化跟踪器。

实时实验可以一定程度上反应跟踪器的运行速度，因为速度较慢的跟踪器在指标上会有较大的衰减。但是，实时实验的结果会将跟踪器大概分为两个部分，其中一部分是跟踪器速度足够快的，另一个部分是跟踪器速度相对慢的，对于同一个部分的跟踪器，实时实验无法判断究竟哪个更快，以及究竟快多少。为此，我们直接在同一个硬件环境下测试了VOT2017上一些优秀的跟踪器，对比了它们的运行时间，弥补实时实验的缺陷。

本次VOT实验主要有四部分，分别是在VOT2015、VOT2016和VOT2017测试视频上，FSSiamese与对应指标最前面的跟踪器进行对比；最后一个实验是在同一平台下与VOT2017中表现较好并且开源的跟踪器进行的速度对比。

3.3.1.2　VOT2015测试结果

VOT2015提供了用作视觉跟踪的完整标注视频，这些视频包含多种场景，大部分对应了一个或者多个跟踪下的困难问题。我们基于VOT2015的测试规则对FSSiamese进行了测试，其他对比的跟踪器结果均来自VOT2015官方的测试报告，结果如表3.1所示。

表 3.1: VOT2015实验结果

Tracker	Accuracy	Robustness	EAO	EFO
DeepSRDCF	**0.56**	**1.0**	**0.32**	0.38
EBT	0.47	1.02	0.31	1.76
SRDCF	0.56	1.24	0.29	1.99
LDP	0.51	1.84	0.28	4.36
sPST	0.55	1.48	0.28	1.01
SC-EBT	0.55	1.86	0.25	0.80
NSAMF	0.53	1.29	0.25	5.47
Struck	0.47	1.61	0.25	2.44
Ours	0.55	1.95	0.20	**52.37**

我们将FSSiamese同VOT2015 EAO排名前八的跟踪器进行了对比（VOT2015中原本MDNet排在第一，但由于其训练集和VOT2015测试集重合，所以大部分文献做对比时会去掉该跟踪器）。其中粗体表示在对比的几个跟踪算法中，当前跟踪器在该指标中表现最优。Accuracy、EAO和EFO数值越大说明效果越好，Robustness 本质是失败率，数值越小结果越好。可以看到，虽然我们的跟踪器属于轻量级网络，但和VOT2015最顶尖的跟踪器相比并不逊色，跟踪准确率上和最好的DeepSRDCF 相差可以忽略不计，鲁棒性比其他几个稍差一些，但几个跟踪器的数值相差并不大。更重要的是，这些跟踪器都不是轻量级跟踪模型，这一点可以从EFO 的数值上看出来，大部分跟踪器都保持在1.0 附近，FSSiamese 经过测试EFO 值为52.37，这代表大部分跟踪器运行速度较慢，FSSiamese 在速度上有更加明显的优势并且准确率相当。

3.3.1.3　VOT2016测试结果

VOT2016[35]重新标定了VOT2015视频的物体框，并且出现了更多优秀的跟踪器，在指标上要远超VOT2015，或许与VOT2016采用了同样的测试视频有关系，我们仍然选择了EAO排名最高的几个跟踪器，结果如表3.2所示。

表 3.2: VOT2016实验结果

Tracker	Accuracy	Robustness	EAO	EFO
C-COT	0.54	0.24	**0.33**	0.51
TCNN	0.56	0.27	0.33	1.05
SSAT	**0.58**	0.29	0.32	0.48
MLDF	0.49	**0.23**	0.31	1.48
Staple	0.54	0.38	0.30	11.11
DDC	0.54	0.35	0.29	0.20
EBT	0.47	0.25	0.29	3.01
Ours	0.51	0.53	0.19	**53.05**

可以看到，FSSiamese在VOT2016中的表现和VOT2015接近，这主要是由于两个测试集基本相同造成的。其中CCOT 在VOT2016 中表现最佳，FSSiamese在准确度上和其他几个最好的跟踪器水平基本持平，鲁棒性上由于是轻量级模型有一定差距，这一点也导致EAO 指标的结果一般。EFO 指标上和之前一样呈依然呈碾压趋势，这也是轻量级模型的主要优势，但准确度的稳定可以保证其在大部分场景下很好地工作。VOT2016中新出现了一

些效果更好的跟踪器，一方面是更多优秀的跟踪器的出现，另一方面是由于VOT2016的测试视频与VOT2015的完全相同，大部分参加VOT2016测评的跟踪器针对VOT2015做了更多的调校工作，这样的设置对于通用的目标跟踪器是不合适的。在VOT2017更换了部分测试视频后，可以观察到大部分跟踪器的效果明显不如VOT2016，这也从另一方面印证了我们的猜想。

3.3.1.4　VOT2017测试结果

VOT2017更换了VOT2016中的部分测试视频，这些新的视频跟踪起来难度更大，导致VOT2017中整体测试的结果都有所下降。我们这里选择了VOT2017的实时实验对FSSiamese进行了测试，对比的几个跟踪器分别是实时实验中排名最靠前的七个。之前提到，实时实验无法准确地对比跟踪算法的运行速度，为此，我们添加了补充实验，在表3.3所示的低配置硬件条件下进行测试。

表 3.3: 实验硬件配置

GPU	Processor
GeForce 840M	i7-4510U CPU @ 2.00GHz

这里采用较低配置的原因有两个，一方面是较低配置的实验平台做实验更加方便，直接选用个人PC测试即可，同时个人PC进程较少，比较容易控制无关变量，从而使测试结果更加准确；另一方面是较低配置更能体现出跟踪器在实际场景中的运行效果，因为大部分实际场景对于实时性的要求更高，并且不太可能运行在理想环境，特别是对于成熟的产品，会更加考虑运行环境的成本问题。通过我们的实验，我们发现不同的配置，不同跟踪器的运行速率并不完全成正比。通常来讲，配置环境越好，轻量级跟踪器的运行速率会更加明显。因此，我们无法根据理想环境中的实验结果准确地推测一般环境中的运行结果，只能给出一个大致参考，在特定的硬件环境下进行实验才能得到更有说服力的结果。

跟踪器在VOT2017的测试结果如表3.4所示：

同样地，VOT2017的测试指标去除了EFO，只剩下Accuracy、Robustness、EAO三个指标，其含义和之前的完全相同。这里我们对比时选择的是VOT2017的实时实验中排名较高的几个跟踪器，这意味着几个跟踪器可以在大部分场景下实时运行，后面我们会通过实验测试具体的运行速度。在该表中的三个测试指标中，加粗的仍然是当前指标下测量出的最好结果。VOT2017实时实验

表 3.4: VOT2017实验结果

Tracker	Accuracy	Robustness	EAO
CSRDCF++	0.46	**0.40**	**0.21**
SiamFC	0.50	0.60	0.18
ECOhc	0.49	0.57	0.18
Staple	**0.53**	0.69	0.17
KFebT	0.45	0.68	0.17
ASMS	0.49	0.63	0.17
SSKCF	0.53	0.66	0.16
Ours	0.45	0.81	0.13

的整体指标相比VOT2015和VOT2016相比下降明显，原因之一是之前提到的更换了新的视频序列，导致跟踪难度有所上升，同时部分跟踪器在测试前无法根据新的视频序列进行参数调整，也会导致效果变差；另一个原因是由于这里实时测试表现较好的跟踪器，基本都属于轻量级跟踪器，在Accuracy、Robustness、EAO三个指标上同最顶级的算法会稍微低一些。我们的实验结果显示，FSSiamese在一众优秀的实时跟踪算法中保持着不错的跟踪效果。为了更进一步比较不同算法之间的速度差异，我们在表3.3的环境下，测试了若干跟踪器的具体速度，如图4.8所示。

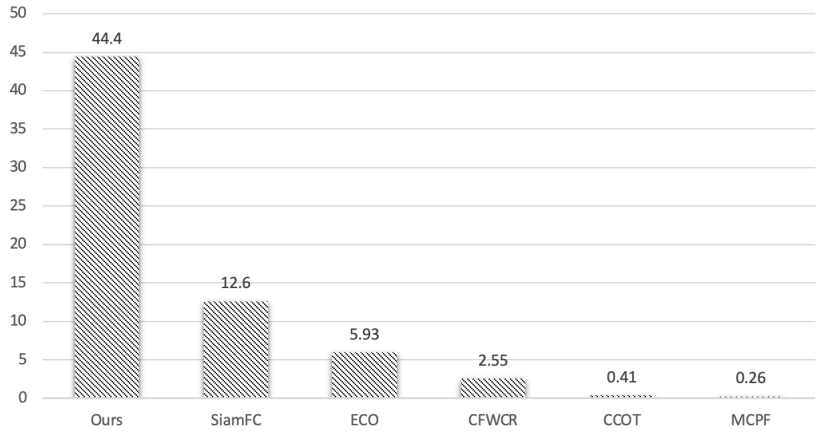


图 3.6: VOT2017跟踪器速度对比

我们测试运行了若干在VOT2017中表现较为突出，并且完全开源的跟踪器。不难发现，在硬件环境不理想的情况下，大部分跟踪器的运行速度都在每秒20 帧以下，运行帧率超过5FPS便基本可以通过VOT2017实时测

试。本文提出的FSSiamese在本次测试环境下运行超过40FPS，但仍保持着较好的准确率。其他在非实时场景下取得不错成绩的跟踪器，例如CFWCR、CCOT和MCPF则运行速度过慢，与本文进行的速度测试结果一致，这几个跟踪器在VOT2017的实时测试中表现也并不理想。与FSSiamese在实时测试中指标接近的SiamFC和ECO在运行速度上，和FSSiamese仍有较大的差距，这也进一步证明了FSSiamese作为轻量级网络的速度优势。

3.3.2　OTB数据集实验

OTB[66]测试集主要有两个版本，分别是OTB50以及OTB100，二者的测试指标完全相同，不同的是OTB50拥有50个测试视频，OTB100则拥有包括OTB50所有视频在内的100个视频。OTB数据集中包含了目标跟踪领域常见的难点，主要有低分辨率、相似背景、平面内外旋转、快速移动、目标形变、目标遮挡问题、尺度变化等等。

为了得到更加稳定的结果，我们选择了OTB100作为测试数据集，和VOT不同，OTB采用的是“One Pass”的测试方法，意思是跟踪器失败之后不进行重启，而是继续跟踪直至视频结束。这种测试方法对于我们的轻量级网络是一种较大的挑战，另外这种方法实际上对测试集而言有些浪费，因为任何跟踪器一旦跟踪失败，后面的所有帧大概率都会一直处于失败状态，从而无法利用起来。

我们会先介绍OTB的两个测试指标，然后分别与近些年一些跟踪效果较好的跟踪器和一些运行速率快的跟踪器进行对比。

3.3.2.1　测试指标

精准图（Precision Plot）用来衡量预测位置和真实位置中心点的欧式距离，给定不同的阈值，对应不同的准确率，最终得到这样一条曲线。然而，精准图只能衡量中心点的位置偏差，而不能反映目标的大小和长宽比例的准确度。

成功图（Success Plot）和VOT中的accuracy相似，计算预测框和真实框之间的重叠率，设定不同的阈值可以获得不同的准确率，从而得到一条曲线。

3.3.2.2　实验结果

我们对比了DeepSRDCF、Staple、MOSSE、KCF、CN、CSK、DAT这几种跟踪算法，其中DeepSRDCF和Staple属于比较典型的主流跟踪算法，效果较好但运行速度不太理想，难以满足实时运行的要求；MOSSE、KCF、CN、CSK和DAT属于同FSSiamese一样属于轻量级跟踪网络，运行速度快但效果和主

流算法相比有一定差距，最后的对比结果绘制在精准图和成功图中，如3.7 所示。

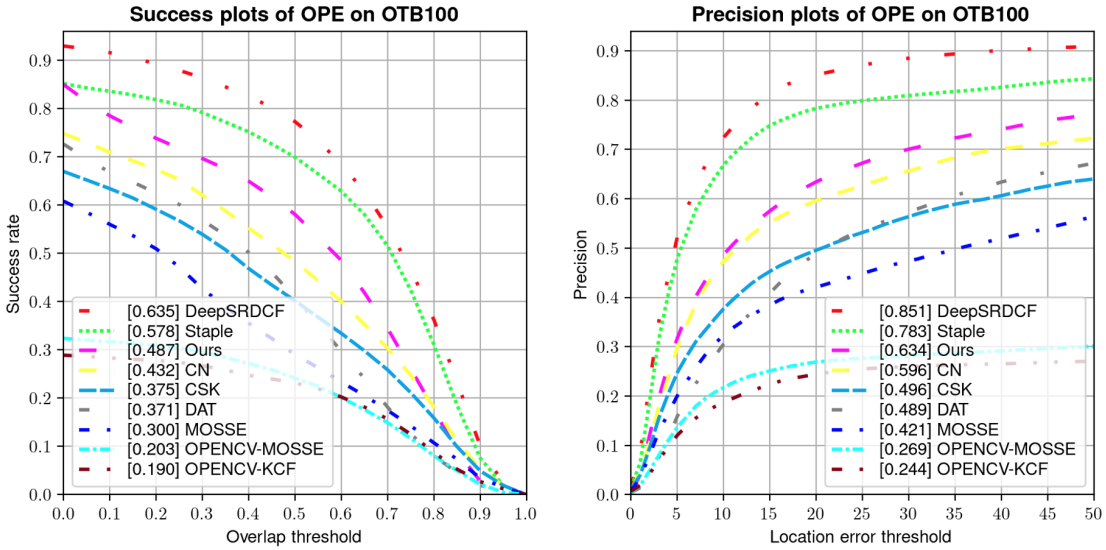


图 3.7: OTB100实验结果

成功图的曲线代表了预测框和真实框的重叠程度，曲线越靠近右上方代表着算法跟踪的目标位置与真实的位置重叠率越高。OTB数据集中的测试方法偏向于测试长时跟踪算法，本文中涉及的算法大部分属于短时跟踪算法，短时跟踪算法大多是在每一帧中做目标的搜索工作，而长时跟踪更偏向于提出一系列策略，保证跟踪算法能够在更长的图片序列中定位到目标。因此OTB数据集的测试在长时跟踪算法上效果会更好一些，短时跟踪中的主流算法也会受到测试方法的影响，而对于本文提出的轻量级跟踪算法挑战较大。从成功图上也可以看出这一点，两个典型的主流算法成功图上曲线更靠近右上方，FSSiamese和一众轻量级算法相比起来指标更低一些。但在这些轻量级算法中，FSSiamese的表现最好，这意味着在轻量级的跟踪算法中，FSSiamese的跟踪准确率更高。

精准图代表了预测框中心点和真实位置中心点的偏离程度，曲线越靠近左上方代表着算法跟踪的目标位置中心点与真实的中心点偏差更小。进行对比的几个算法中，精准图中的变化趋势和成功图基本保持一致，这里不再赘述。同样地，在这些轻量级跟踪算法中，FSSiamese的跟踪效果在精准图上的表现最好。

3.3.3　模型大小测试

得益于网络的简单结构，FSSiamese的模型大小是足够小的。这使得模型在边缘设备甚至是移动设备上使用成为可能，我们对比了其他基于孪生网络的深度学习跟踪模型，这些模型都属于端到端模型，模型大小也反应了模型参数量，结果如3.8所示。

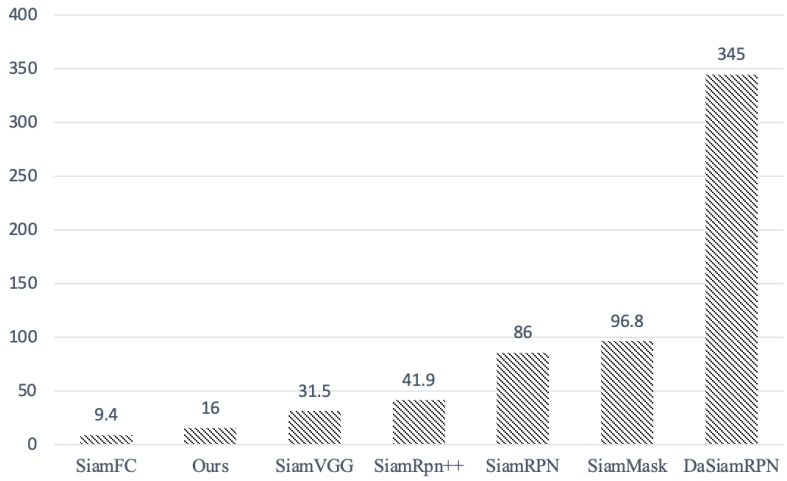


图 3.8: 模型所占体积大小对比

可以看到，大部分基于孪生网络的端到端跟踪模型大小都在几十兆附近，最大的DaSiamRPN由于使用了较大的骨干网络，模型大小超过了300M，其他几个网络大小相差不多。在大部分视觉检测任务中，几十兆大小的模型其实并不算大，常见的检测网络模型大部分都是上百兆。但考虑到一个单目标跟踪网络处理的任务有限，在很多场景下可能需要多个跟踪器并发使用，因此模型的尺寸问题在实际系统的使用过程中会被放大。另外，移动设备和边缘设备上使用网络处理任务的场景也逐渐出现，作为轻量级跟踪网络，较小的尺寸也能够被很好地部署在这些场景中。我们的模型只有15M，比大部分的孪生网络都要轻很多。模型最小的SiamFC，前面的内容有介绍到，虽然模型较小，但由于SiamFC 需要多尺度回归来确定目标大小，因此运行速度较慢，和FSSiamese相比有很大差距。

3.3.4　特征提取网络对比

在上一小结我们提到，FSSiamese中的特征提取网络可以根据实际应用场景进行更换，只需要注意特征提取网络得到的特征图尺寸与后面的全连接层输入

保持一致即可。我们分别将四种常见的骨干网络作为特征提取网络，将得到的模型在VOT2016中测试EAO指标以及参数量，结果如图3.9所示。

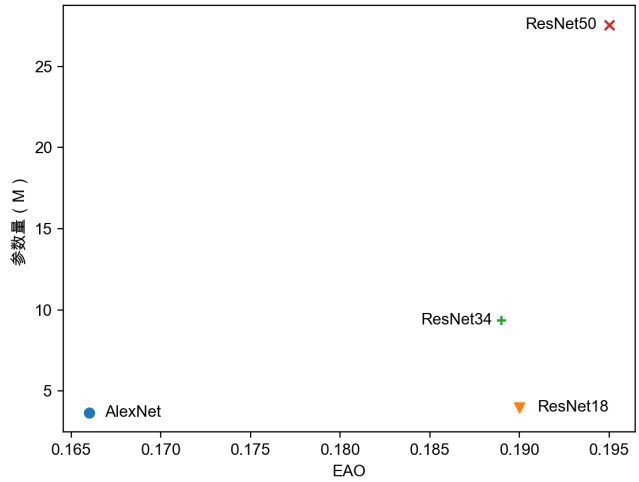


图 3.9: 骨干网络对比

我们测试了AlexNet、ResNet18、ResNet32和ResNet50四种骨干网络，网络的复杂程度按照顺序依次增大，其中AlexNet最简单，ResNet50结构最复杂。图中横坐标轴代表的是模型在VOT2016测试中得到的EAO值，EAO值越大代表跟踪结果更好；纵坐标轴代表模型的参数量，单位为M。最终实验的结果与之前的分析基本一致，特征网络越复杂，提取到的图像信息更丰富，因此总体上跟踪效果会有所提高。但根据实验也可以发现，在分别选择ResNet18和ResNet34作为特征提取网络时，二者的EAO值差距基本可以忽略不计，但参数量上ResNet34基本是ResNet18的二倍，参数量除了会影响模型大小之外，还决定了算法的运行速度，因此我们认为在这四个骨干网络中，ResNet18更具性价比，这也正是我们在本文中将其作为特征提取网络的原因。ResNet50虽然跟踪效果最好，但参数量是ResNet18的五倍之多，对于轻量级跟踪网络的FSSiamese而言不太合适。当然，如果在某些场景下，对于跟踪的高实时性没有过高要求，也可以选用其他更加复杂的骨干网络，跟踪效果确实会得到一定提高。

3.4　本章小结

在本章中，我们提出了一种基于孪生网络的单目标跟踪跟踪器FSSiamese。FSSiamese是通过ImageNet-VID和GOT数据集训练得到的端到端网络，可以回归

得到目标在当前帧左上角以及右下角的坐标，从而确定目标位置。FSSiamese由于其特殊的网络的设计，即使在不理想的硬件环境下也可以实时运行，同时还保持着优秀的跟踪准确率。

第四章　FSSiamese在视频检测算法中的应用

本章介绍了如何将本文提出的FSSiamese算法应用在视频检测中。本章共分为四个部分，第一部分介绍了视频目标检测的相关背景知识，主要包括视频目标检测与传统目标检测的区别以及主流的视频目标检测算法；第二部分介绍了应用FSSiamese的视频检测算法的详细流程；第三部分是实验部分，将新的视频检测算法在相关数据集上进行测试对比，并对实验结果进行了分析；最后一个部分是本章小结。

4.1　背景介绍

4.1.1　目标检测背景

目标检测的目的是在一张图像中找到事先给定类别（例如人、车辆、猫、狗等等）的所有物体，如果找到，则给出每个物体的种类、在图片中的位置以及置信度。长期以来，目标检测都是计算机视觉最基础和最具有挑战性的任务之一，在无人驾驶、机器人视觉、人机交互等领域有着广泛的应用，同时，目标检测还是众多其他计算机视觉领域的基础，例如目标跟踪、图像分割、异常检测、看图说话（image caption）等等。

在2012年AlexNet[36]被提出，打开深度神经网络大门之前，目标检测大多采用传统机器学习方法，我们会在第一小节介绍传统检测算法的大致思路。在AlexNet取得当年ImageNet大规模视觉识别挑战赛冠军之后，深度学习算法很快就被用于进行目标检测，效果相比传统算法有了质的提升，所以现在大多数场景下的目标检测算法都是以深度学习算法为主。

在众多的基于深度学习的检测算法中，除了比较经典的Faster-RCNN、YOLO[53–55]等算法，近些年来，也有很多新的目标检测算法脱颖而出，取得了很好的检测效果。基于深度学习的目标检测算法可以大致分为两类，一类是以RCNN系列为代表的基于分类的方法，我们会在第二小节中介绍，这类算法通常能够达到不错的效果但过程较为繁琐；另一类是基于回归的检测算法，以YOLO、SSD[44]为代表，这类算法结构简单，效率通常更高一些。

4.1.1.1　传统检测算法

时间追溯到20年前，深度学习还没有被广泛应用起来，视觉任务无法使用卷积网络提取特征，此时大部分的目标检测算法使用都是基于手工设计的特

征。首先，通过设计一系列复杂的特征来提炼出图像上的关键信息，然后加上一些优化技巧来提速以及降低资源的开销。传统检测算法主要分为两个核心内容，一是特征的表示，当时以手工设计的特征居多；二是分类器，通过提取到的特征判断是否包含指定类别的物体，通常使用的是Adaboost分类器[20]。

AdaBoost分类器基于Boost思想，Boost指的是一种集成学习的训练思路，在若干弱学习器中每次训练通过结果挑选出最好的，然后针对这些较好的训练器再进行改进并且挑选，最后将每个分类器的结果依次叠加起来，得到最终分类的结果。在AdaBoost中每次训练都会根据结果改变弱分类器训练数据的权值与分布，使结果不好的数据更加被分类器关注，得到更好的效果。训练完成后，根据各个弱分类器的误差率确定每个分类器决定最终结果时的权重。

出现较早并且效果较好的检测算法是VJ算法（Viola Jones Detectors），由P. Viola 和M. Jones提出[61]。VJ算法被设计做人脸检测，该算法也成功将计算机进行人脸检测引入到了实际的生产场景中。VJ算法中通过提取类Haar特征来捕捉人脸信息，利用滑动窗口机制对各个位置依次使用不同大小的窗口进行人脸检测。虽然这种方式较为粗暴并且计算量较大，但是VJ算法巧妙地使用积分图来加快计算，并且类Haar特征比传统的Haar特征长度更短，使得计算量更小。另外，VJ算法还提出了级联检测，如图4.1所示，不是所有的位置和窗口都需要使用全部的弱分类器进行检测，在某一层检测失败的窗口会被直接删除，这也大大减少了运算量。

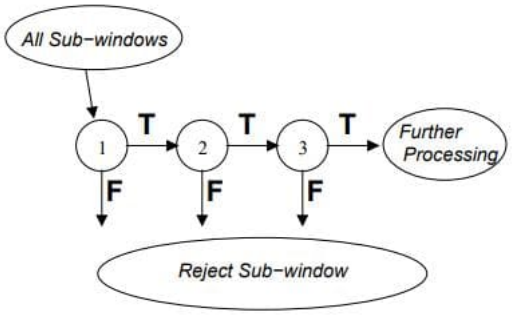


图 4.1: VJ算法中的级联检测[61]

4.1.1.2　基于分类的检测算法

基于分类的检测算法大体流程分为两步，第一步是提取出候选区域，所谓的候选区域是指可能存在目标的区域；第二步是得到这些区域中是否存在目标的概率、目标的具体位置以及目标的种类。最终检测的结果很大程度上取决于提取的特征，而这里使用的特征都是卷积神经网络提取得到的。

RCNN系列是最具代表性的基于分类的检测算法，其后一系列的Fast-RCNN[22]和Faster-RCNN都是在其基础上做了改进。RCNN算法大概分为四个步骤：（1）使用选择性搜索（selective search）算法在图像上生成足够多的候选区域；（2）使用卷积神经网络提取每个候选区域的特征；（3）将特征输入每一类的SVM分类器判断每个候选区域是否属于该类；（4）使用回归网络回归候选区域位置与真实位置的偏差。

随后的Fast-RCNN将每个区域提取特征改为了整张图片先提取特征，然后根据区域位置取出对应的特征图，从而减少计算。其次提出了ROI Pooling将所有的特征图固定为统一大小，去除了RCNN中的SVM分类器和回归器，提高了运算速度。Faster-RCNN则是提出了RPN网络，输入图像特征则可以回归出感兴趣区域的位置，后面的步骤同Fast-RCNN基本一致，将提取出的特征再次进行卷积，回归得到物体为各个种类的置信度以及位置精修值。Faster-RCNN的结构图如下所示：

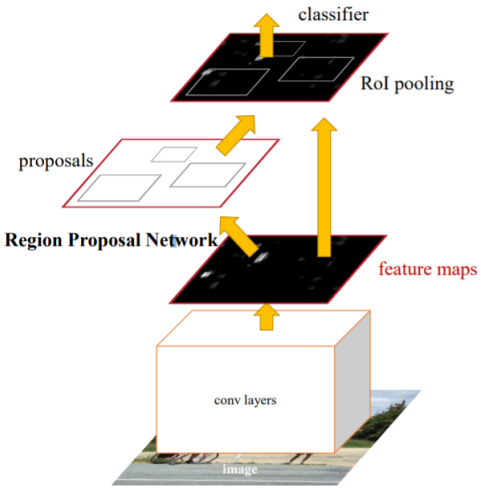


图 4.2: Faster-RCNN结构图[22]

从RCNN系列的结构中不难看出，基于分类的检测算法需要先找到候选区域，然后在候选区域的基础上回归得到物体的类别以及位置信息。这样的设计可以使网络更专注于每一步的任务，结合起来之后检测的效果更好。但缺点也很明显，两步操作会分别耗时，导致基于分类的检测算法通常运行速度不是很快。另外，其他的一些基于分类的检测算法，例如R-FCN[13]、Mask R-CNN[24]，在Faster-RCNN的基础上做了一些改进，包括使用了更大的ResNet作为骨干网路，Mask R-CNN提出了ROI Align提高回归精度，但同时也带来了更大的运行代价。R-FCN速度稍好一些，但仍然无法满足高实时性的需求。

4.1.1.3　基于回归的检测算法

为了解决基于分类的检测算法在速度上的不足，一系列基于直接回归的检测算法开始涌现出来。YOLO算法不需要事先生成候选区域，而是将图像划分为若干网格，每个网格负责预测中心点落在该网格的物体框。YOLO最初的版本设计是对于每张图片网络预测一个长度为$S*S*(B*5+C)$的向量，其中$S*S$代表网格的个数，B代表每个网格预测框的个数，C代表种类数（不管有多少个预测框，只能预测一组类别概率值），如图4.3所示。

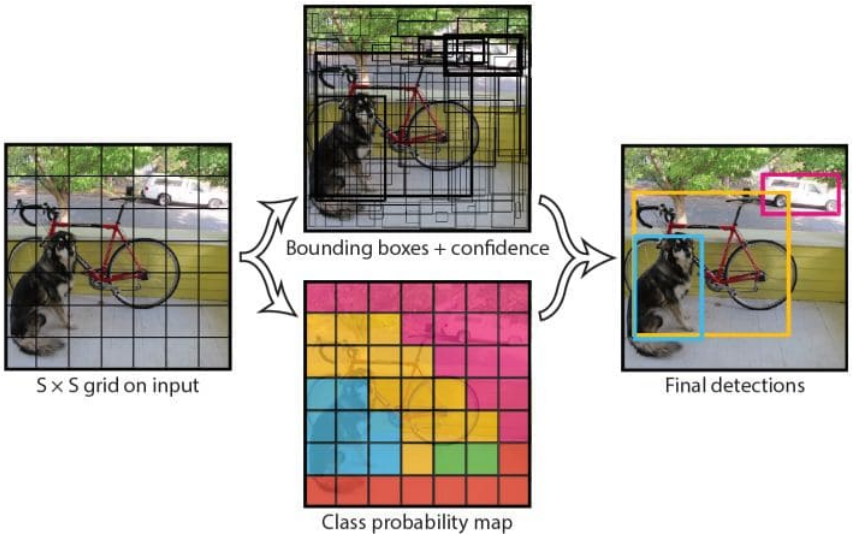


图 4.3: YOLO结构图[53]

SSD算法和YOLO类似，区别在于SSD借鉴了Faster-RCNN，每个网格的处理类似于RPN网络，回归物体的预测框与各个锚框的偏差值。另外，SSD使用VGG网络作为骨干网络，并将不同层次的特征图分别处理，层次不同的特征图对应着不同的感受野，使网络更加容易找到图像中的物体。

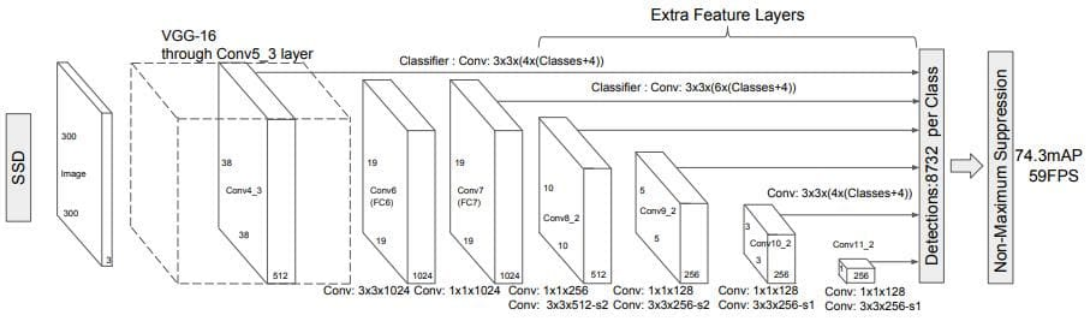


图 4.4: SSD结构图[44]

SSD做到了YOLO和Faster-RCNN相平衡，从鲁棒性上讲，Faster-RCNN要比SSD和YOLO都要强一些，这也是基于分类的检测算法的优势所在。但代价是速度稍慢一些，SSD 的运行速度和Faster-RCNN相比提升有足足六倍。

4.1.2　视频检测存在的问题

随着卷积网络的发展，以及一系列优秀检测算法的提出，包括Faster-RCNN，YOLO，SSD这些非常经典的算法，以及后来被提出的CenterNet、EfficentDet等等，物体检测准确度已经达到了比较理想的效果。这些算法通过提取图像的深度特征，训练得到图片中的目标位置以及其对应的类别属性。但是大部分检测算法都是设计用来识别单张静止图片上的物体，而不是针对连续的视频进行检测。

一种可行的方法是将视频中的每一帧进行检测，但由于视频中可能会由于某些特定的角度、遮挡等原因造成物体漏检或者误报。具体有如下四种情况：

- 外观不同：对于非刚性物体，不同角度不同姿态都会导致不同的外观，某些角度可能比较少见，一般的检测算法没有学习过类似的图像，导致检测失败

图 4.5: 外观变化[71]

- 遮挡：目标或者摄像机由于移动导致目标出现在另一个物体后方，造成部分或者完全遮挡，这时候检测算法通常会检测到前方物体或者完全漏检

图 4.6: 遮挡[71]

- 运动模糊：由于摄像机只能捕捉一帧一帧的画面，但物体运动是连续的，如果曝光时间过长就会导致多个画面被合到一张图片上面去，这种图像一般的目标检测也难以发挥作用

图 4.7: 运动模糊[71]

- 虚焦：在拍摄过程中由于摄像设备没有对焦准确，导致视频某些部分出现或者逐渐出现模糊的情况

图 4.8: 虚焦[71]

4.1.3　视频检测算法思路

针对这些检测场景，有研究提出了视频检测算法，例如Han等人[23]提出了利用NMS算法关联图像序列中静态图片的检测结果，并非只在每一帧的图像上做NMS，从而充分利用相邻帧间的信息。

更加复杂一些的方法，例如T-CNN[32]用深度学习方法将多目标跟踪和检测结合起来，设计了两个模块：融合了目标检测与目标跟踪的管道模块和对管道分类与重打分的模块。第一个模块先对所有图像进行检测，然后对高置信度的候选框进行跟踪；第二个模块通过时域卷积对管道进行重新打分。该方法计算量较大，并且需要处理所有帧来挑选高置信度的候选框，这意味着该算法无法检测在线视频。另外，该算法的跟踪模块在时域上是双向的，这也进一步加大了算法的复杂度。

还有一些研究将光流法引入到检测当中，例如FGFA[71]将相邻帧间的光流当作特征的一部分，同当前帧的图像特征结合起来进行检测。但光流也需要放入网络中进行训练，导致该方法复杂度较高，难以满足实时要求。并且计算多

帧的光流需要结合当前帧之前以及之后的图像，所以该算法也无法在线进行检测。类似的还有Associate LSTM算法[46]，利用LSTM提取学习视频序列中的时间信息来增强目标检测算法的鲁棒性，通过设计好的匹配损失函数可以将不同帧间相同的物体更好地关联起来；STMN[67]也是类似的做法，不过为了达到较好的鲁棒性，使用了双向的循环卷积网络，但也带来了更低的训练效率。另外，该类算法都过于依赖检测和跟踪算法，如果使用轻量级的检测模型和跟踪模型则很难发挥作用。

4.2　FSSiamese在视频检测算法中的应用

为了能够使视频检测算法能够在线使用，并且满足实时运行的要求。我们将FSSiamese应用到了视频检测算法中，该算法只需在原来轻量级目标检测算法的基础上，引入FSSiamese跟踪算法，便能够在不降低原检测运行速率的情况下，提高视频的检测效果，并且方便部署，其主要原理如图4.9所示。

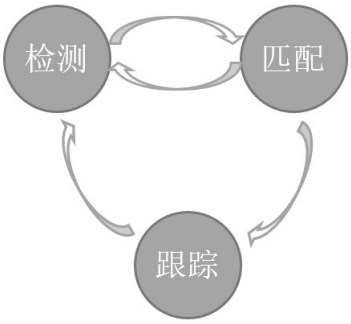


图 4.9: 引入了FSSiamese的视频检测算法框架图

新的视频检测算法分为三个模块，第一个模块是检测模块，用来负责单张静态图像的目标检测。由于该方法的特性，这里的目标检测算法使用的是轻量级检测算法，这样子才能完全发挥该方法的作用，我们下面的实验中选用的是SSD算法；在检测完成之后，进入第二个模块，也就是匹配模块。匹配模块负责将前几帧的检测结果进行融合，为了避免检测算法在某一帧漏检，匹配算法可以通过多帧的检测结果，最大可能地还原出这几帧中所有的物体，并对其进行跟踪；因此最后是跟踪模块，将前面匹配完成的物体作为输入，对每个物体调用FSSiamese跟踪器进行跟踪，将跟踪的结果作为检测结果返回。详细的算法流程见算法3。

4.2.1　算法流程

算法 3 引入了FSSiamese的视频检测算法

Input: 图片序列fm，长度为l，其中fm_i表示第i帧图片

Output: 检测结果集合D，其中D_i代表第i帧的检测结果

初始化轨迹集合T
初始化当前帧计数器$i = 0$
repeat
    初始化检测计数器$d = 0$
    初始化跟踪计数器$t = 0$
    repeat
        调用检测算法得到检测结果Det
        将Det保存到D_i
        for Det中的每个目标 **do**
            对当前目标和轨迹集合T上调用匹配算法得到id
            if $id == -1$ **then**
                将该目标加入T
            else
                if $id! = -2$ **then**
                    将T中对应的轨迹更新为当前目标的信息
                end if
            end if
        end for
        $d : d + 1$
    until $d < d_m$
    repeat
        for T中的每个轨迹$track$ **do**
            使用FSSiamese算法进行跟踪，结果保存到D_i
        end for
        $t : t + 1$
    until $t < t_m$
until $i < l$

算法一开始先初始化轨迹集合和计数器，轨迹集合用来存储当前需要跟踪

的所有目标，计数器用来记录当前帧数。在算法的执行过程中，还需要维护检测计数器和跟踪计数器，算法的核心思路是检测若干帧之后进行跟踪，这两个计数器用来判断当前处于检测阶段还是跟踪阶段。如果是检测阶段，那么调用传统目标检测算法，得到检测结果，并将结果中的每个目标同轨迹集合中的目标进行匹配，如果匹配算法返回结果为-1，表示匹配失败，将其加入轨迹集合；返回-2则表示找到匹配目标，但无需更新；返回其他数值表示需要更新轨迹集合中对应序号的目标信息。如果当前是检测阶段，则只需要调用FSSiamese算法对目标进行跟踪，并将跟踪到的结果更新到轨迹集合即可。

算法在检测阶段将检测的结果直接返回，同时记录检测阶段的所有物体，并将不同帧之间的物体进行匹配。最后在跟踪阶段，通过跟踪已经记录的物体作为目标检测的结果。这样做的原因，是由于在大部分轻量级检测网络中，漏检的现象出现频率更高，跟踪作为一个连续的过程，则可以很好地解决这个问题。但对于离线场景，脱离目标检测的跟踪难以单独达到理想的效果，原因在于跟踪更加容易丢失目标，相反，检测可以弥补这个不足。因此检测和跟踪相互交替可以有效降低轻量级检测模型的漏报问题。

4.2.2　双阈值匹配算法

融合之后的算法中最核心的部分当属匹配算法，匹配算法的作用是找到相邻帧之间所有的物体，确保相同的物体被正确匹配到，并且每个物体不重复，以便这几帧中出现的物体可以在后续被正确地跟踪。我们设计了双阈值匹配算法在轨迹集合中寻找与当前物体最匹配的轨迹。

计算相似度时，可以根据不同的场景选择不同的特征来计算。例如，对于某些目标较少的场景，可以使用简单的位置信息作为特征，通过两个位置的交叉比计算相似度，如公式4.1所示。其中A_t^T表示检测出来的物体框，A_t^G表示真实的物体框。这样做的优点是运行开销极小，在场景简单、目标稀疏的情况下可以有很好的效果。

$$\phi = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T} \tag{4.1}$$

对于场景较为复杂的情况，可以使用更加复杂特征计算相似度。例如前面几个章节提到的，传统特征中的HOG 特征、SIFT 特征，以及专门针对人脸的Haar 特征等等。也可以深度学习中的骨干网络提取特征，比较常见的有VGG、ResNet等等。例如DeepSORT中使用WRN网络作为深度特征来完成目标关联。通过最小化类似于行人重识别中的三元组损失进行训练，可以得到专门用于计算物体相似度的特征提取器。但这种方法稍微复杂一些，针对不同的

场景需要特定的数据集进行训练才能达到更好的效果。例如对于行人重识别场景，使用专门的行人数据集表现尚好。但如果是通用场景，因为物体的种类较多，训练难度会更大，难以达到理想的效果。

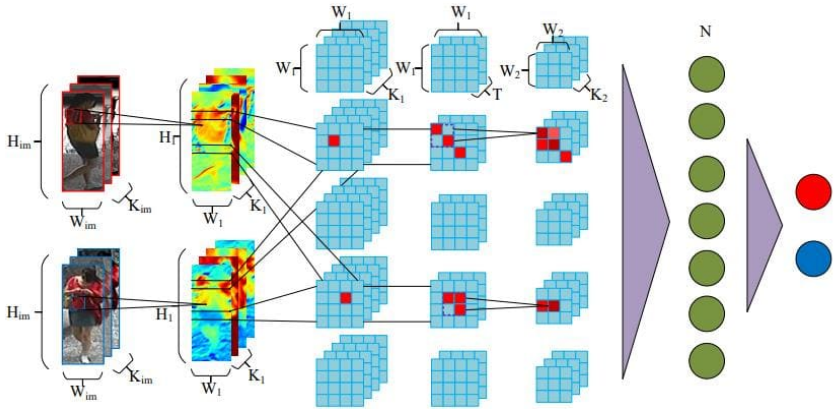


图 4.10: 行人重识别场景下使用深度学习特征度量图像相似度[42]

所谓的双阈值，是指在对比目标的相似度和面积之比时分别设置一个阈值，第一个阈值用来判断相似度最高的情况下，寻找到的目标是否可信，如果高于该阈值则直接完成匹配，否则通过第二个阈值进行判断；第二个阈值是用来弥补第一个阈值造成的匹配遗漏，因为对于相似度过低的匹配可能是由于两帧之间物体框差异较大导致的。为了更加清楚地解释上面的情况，我们用图4.11表示三种场景。

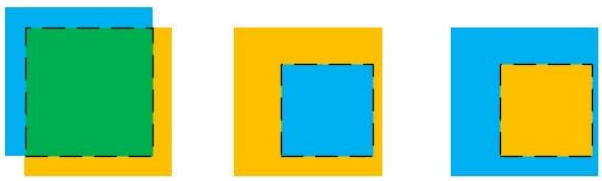


图 4.11: 计算位置相似度可能出现的几种情况

其中第一种情况指大部分最优匹配的情形，两个物体框大小相近且大部分重合；第二种情况和第三种分别是当前物体框过小和轨迹中的物体框过小，这两种情况都会导致重叠率过低，造成匹配失败。而使用第二个阈值可以弥补这种情况，从而提高相邻帧之间的匹配率。基于该思想，双阈值匹配算法对于当前帧检测到的每个目标，在给定的轨迹集合中尝试寻找一个与目标最匹配的轨迹，返回值包括如下三种：-1：表示未找到匹配的轨迹，后续将新建轨迹；-2：找到了合适的轨迹，但尺寸相比之前过小，可能是检测不准确造成，故忽略；

pos：找到了合适的轨迹，且在轨迹集合中下标为pos，后续将使用当前目标更新该轨迹。

具体的匹配算法如4所示：

算法 4 双阈值匹配算法

Input: 轨迹集合T，当前目标t，阈值θ_1，阈值θ_2

Output: 对应的轨迹集合下标i

初始化$Output = -1$
初始化重叠率最大值$res = 0$
初始化重叠率最大值所对应的下标$pos = -1$
for T中的每个轨迹 **do**
    计算当前轨迹与目标t的重叠率tmp
    if $tmp > res$ **then**
        $res : tmp$
        pos :当前轨迹的下标
    end if
end for
if $res < \theta_1$ **then**
    计算当前轨迹与目标面积的最小值与最大值之比$ratio$
    if $ratio < \theta_2$ **then**
        if 目标的面积比得到的轨迹面积更小 **then**
            $i = -2$
        else
            $i : pos$
        end if
    else
        $i = -1$
    end if
else
    $i : pos$
end if

双阈值匹配的算法核心在于通过检测框的位置信息匹配关联轨迹以及检测

结果。首先初始化返回值和返回值对应的下标，返回值默认为-1，代表这轨迹集合中找不到合适的匹配目标，返回值下标保存的是最佳匹配轨迹在轨迹集合中的位置，方便后续通过该位置定位到对应轨迹进行更新操作。算法首先遍历轨迹集合，寻找与目标重合率最高的轨迹，并保存该重叠率以及轨迹对应的下标。接下来，通过双阈值判断是否该结果符合要求，第一个阈值用来界定重叠率是否足够高，如果足够高说明可以直接返回结果；第二个阈值用来判断重叠率低是否是由于两个检测框大小差异导致的，如果确实是，即使当前轨迹和目标的重叠率低于阈值，也将其当作最佳匹配，通过这个技巧可以有效地提高匹配率。

4.3　实验与分析

实验部分，我们选择了两个完全标注的视频数据集进行测试，分别是ImageNet-VID和GOT数据集。为了更加清晰地展示引入FSSiamese的目标检测算法在简单场景下的表现，我们对于每个数据集挑选了常见的几种类别进行检测，每种类别包含多个简单场景下的视频，分别计算出每个类别的AP值后，再计算出整个数据集中所有测试视频的mAP值。

我们首先使用传统的轻量级目标检测算法，对每个数据集中的视频进行检测，然后在检测算法中引入本文提出的FSSiamese算法，其中的检测算法与前者保持一致，最后对比两种方法在不同IOU阈值下的AP值、mAP值以及在相同平台下的运行速度。

4.3.1　评价指标

下面我们简单介绍一下使用的指标mAP，mAP表示的是mean Average Precision，常用来衡量目标检测的结果是否准确，mAP是根据每一个类别的AP值（Average Precision）通过求平均得到的结果。首先，我们需要明确机器学习算法中比较常见的衡量指标TP、FP、FN、TN、Precision和Recall在目标检测任务中代表的含义。

TP（True Positive）：指被正确预测的正样本。在目标检测中，当检测结果与真值的重叠率大于等于阈值时，表示检测正确。

FP（False Positive）：指被错误预测的正样本。在目标检测中，当检测结果与真值的重叠率小于阈值时，表示检测错误。

FN（False Negative）：指被错误预测的负样本。在目标检测中，指的是本应该被检测到但被漏检的样本。

TN（True Negative）：指判断正确的负样本，但由于检测任务得到的结果都看作是正样本，所以这一项不会在求AP中出现。

Precision：通过$TP/(TP+FP)$计算得来，用来衡量预测正确的正样本在所有预测结果是正样本中所占的比率，也被称作查准率。

Recall：通过$TP/(TP+FN)$计算得来，用来衡量预测正确的正样本在所有正样本中所占的比率，也被称作查全率。

由于目标检测的结果是众多的候选框，为了方便我们这里只讨论某一类的情况，对每一类求得AP后取均值则是mAP。当设置重叠率阈值之后，可以根据候选框与真值的重叠率将候选框分为TP、FP、FN三类，然后根据上面的公式计算出Precision和Recall值。为了更加全面地评价检测结果是否准确，一般会根据检测结果的置信度对候选框进行筛选，这意味着候选框置信度低于所设置信度阈值的候选框将不被统计为TP、FP或FN，因此不同的置信度阈值对应着不同的Precision和Recall。我们将每一组置信度阈值得到的Precision值和Recall值画在二维坐标系中，可以得到P-R曲线，如图4.12所示[1]。

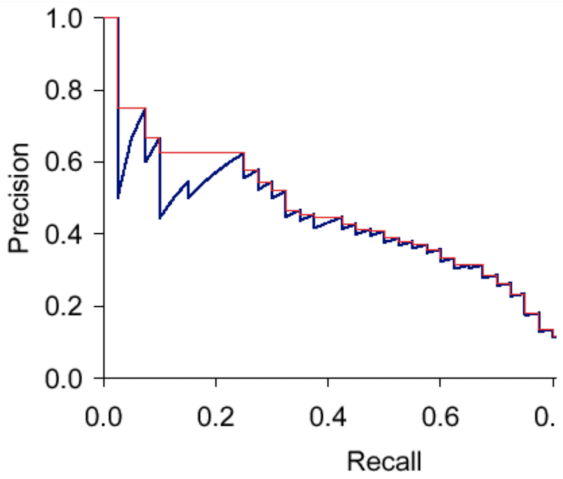


图 4.12: P-R曲线

坐标轴中横坐标的不同取值代表着不同的Recall值，Recall值的取值范围是0到1之间；纵坐标的不同取值代表着不同的Precision值，取值范围同样是0到1之间。不同的置信度阈值在坐标轴中对应一个点，当我们取足够多个点

[1]图片来自于http://robertlexis.github.io

时，便可以得到一条曲线，曲线与坐标轴包围形成图形的面积，正是检测结果的AP值。AP 值越高意味着更多的置信度阈值下，可以同时获得更高的Precision和Recall 值，即检测的结果更加准确。

4.3.2　实验配置及参数设置

在本次实验中，目标检测算法我们使用了Pytorch版本实现的SSD算法，其结构如图4.4所示，骨干网络为VGG网络。目标跟踪算法我们使用了上一章提出的FSSiamese，骨干网络为ResNet18。基于目标的双阈值匹配算法中，阈值θ_1设为0.4、θ_2设为0.6。

4.3.3　在GOT数据集上的结果

GOT数据集提供了数目更加庞大的标注视频，我们在训练FSSiamese时也将该数据集作为了主要的训练集。本次实验我们在GOT的验证集中挑选了若干类别的视频，如表4.1所示。

表 4.1: GOT类别划分

大类	人	动物	交通工具
小类	男人、女人、男孩、女孩	猫、狗、牛、羊、马、鸟	飞机、公交、汽车、船

对于每种类别，在引入与未引入FSSiamese的SSD检测算法上，我们分别测试了若干视频运行的AP值。最后汇总所有类别计算出了对应的mAP值，结果如表4.2所示。

在GOT数据集中，我们选择了11个常见类别作为测试的内容。在未引入FSSiamese跟踪算法时，视频检测每一帧调用传统的目标检测算法。GOT是专门设计做单目标跟踪的通用数据集，而大部分检测网络是由专门用于检测的通用数据集训练而来，并且没有专门针对GOT数据集做微调，因此检测的效果较为一般。因为视频中帧数过多，许多相邻帧的差异其实并不大，但网络容易造成某些帧的目标漏报，特别是在轻量级的目标检测网络下。我们设置了三个置信度阈值，置信度阈值越高意味着对检测结果的精度要求越高，通常将置信度设为0.5，为了更加全面地展现引入FSSiamese后检测精度的提升，我们分别对比了三个置信度下各个类别的AP 值。

通过实验结果可以看到，在各个常见的类别上，置信度阈值分别为0.4、0.5和0.6 的情况下，AP 均有显著提高。对应地，GOT数据集上的mAP值也有非常

表 4.2: GOT数据集mAP对比

算法	未引入$FSSiamese$			引入$FSSiamese$		
	0.4(%)	0.5(%)	0.6(%)	0.4(%)	0.5(%)	0.6(%)
飞机	58.51	58.51	57.99	66.46	64.69	63.63
鸟	34.19	27.99	22.06	44.05	37.56	31.72
船	41.24	40.33	38.40	48.64	48.47	48.21
公交	60.05	56.11	52.94	66.32	64.07	61.66
汽车	19.83	19.36	18.50	19.99	19.89	18.41
猫	45.82	44.83	40.04	62.51	59.70	53.24
牛	19.50	19.23	18.79	19.84	19.56	19.31
狗	31.90	30.92	29.63	40.73	39.73	37.74
马	79.21	78.84	78.23	89.22	87.95	85.92
人	0.39	0.11	0.08	1.04	0.60	0.45
羊	38.66	37.70	34.27	40.89	38.97	35.24
全部类别	24.59	22.67	20.97	29.44	27.67	25.61

明显的提升，平均每个阈值下mAP值提升了5%左右。这说明，在目标数目较少的简单场景下，在检测的同时引入FSSiamese算法，即使使用简单的关联算法，也可以大幅度提高检测准确率。

4.3.4　在ImageNet-VID数据集上的结果

ImageNet数据集被广泛应用在图像分类和目标检测上，是计算机视觉最经典的数据集之一，它由两个部分组成，分别是ImageNet-DET与ImageNet-VID。其中，ImageNet-DET提供了多种类型的静态图片，以及图片中目标的相关信息。ImageNet-VID区别于前者的地方在于，提供的不再是一张张静止的图片，而是一段段的视频，每一段视频的每一帧都标注了存在目标的信息，包括目标是否出现在图片中、目标是否被遮挡以及目标的具体位置。

同样地，我们在ImageNet-VID的验证集中挑选了若干类别的视频，如表4.3所示。

表 4.3: ImageNet-VID类别划分

大类	人	动物	交通工具
小类	男人、女人、男孩、女孩	猫、狗、牛、羊、马、鸟	飞机、摩托车、船

同样地，对于每种类别，在引入与未引入FSSiamese的SSD检测算法上，我们分别测试了若干视频运行的AP值。最后汇总所有类别计算出了对应的mAP值，结果如表4.4所示。

表 4.4: ImageNet-VID数据集mAP对比

模型	未引入$FSSiamese$			引入$FSSiamese$		
	0.4(%)	0.5(%)	0.6(%)	0.4(%)	0.5(%)	0.6(%)
飞机	45.66	44.24	42.35	51.23	49.51	47.00
鸟	48.67	45.70	39.62	53.17	49.11	41.09
船	30.69	29.88	28.95	32.17	31.14	25.31
猫	51.68	47.15	42.59	61.86	57.02	49.48
牛	47.74	46.38	41.15	58.34	57.63	53.20
狗	50.69	48.63	44.83	55.07	51.30	46.63
马	39.01	38.73	36.48	44.75	44.60	42.73
摩托	24.29	24.08	22.42	30.93	25.90	20.10
人	14.25	12.50	10.37	17.49	13.89	10.47
羊	54.34	54.26	53.34	63.11	62.92	62.02
全部类别	34.26	32.18	28.99	39.92	36.72	32.12

ImaeNet-VID数据集的测试结果和GOT上的测试结果类似。从表格上不难看出，所有10个类别的AP值均有所提高，各个阈值下的mAP值也提高了大约5%左右。另外，使用传统目标检测网络时，ImageNet-VID中各个类别的检测效果比GOT明显要好一些。造成这种现象的原因是检测网络的训练集通常是包含ImageNet-DET 的，ImageNet-DET是ImageNet数据集中最核心的部分，主要用来训练目标检测任务（近些年有些跟踪网络在训练时加入部分ImageNet-DET的样本，用来提升样本的多样性），而ImageNet-VID和ImageNet-DET两个数据集的物体具有一定的相似性，因此在ImageNet-VID上的目标检测结果整体上稍好一些。

不过无论原先的传统目标检测效果如何，实验结果告诉我们，在引入FSSiamese跟踪网络之后，视频目标检测在各个常见类别下都有了显著提高。即使在置信度阈值设为0.6的苛刻条件下，FSSiamese的引入也带来了AP的提升，这一方面是由于FSSiamese作为跟踪网络弥补了传统检测漏报造成的指标下降，另一方面也意味着FSSiamese 在大部分视频中的跟踪效果较好，能够满足准确检测的需求，这两点都证明了FSSiamese可以应用在简单的视频目标检测场景中，并且在不影响性能的前提下提高检测效果。

4.3.5　运行速度测试

我们对比了两种算法的运行速度，为了使对比更加具体，我们选择了不同尺寸的视频并且在不同的硬件条件下进行了测试。硬件配置我们选择了中低配置的GeForce 840M和高配置2080Ti两种环境，两种环境的具体配置如表4.5所示。

表 4.5: 两种环境下的配置对比

GPU	处理器
GeForce 840M	i7-4510U CPU @ 2.00GHz
Nvidia 2080Ti	Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz

至于视频尺寸方面，我们选择了1920*1080和1280*720两类图像尺寸的视频，由于检测算法的花费时间除了与检测算法和图像大小有关，还和图像中检测出来的目标数量有一定关系，这是由不同目标数量在执行NMS算法时的开销不同所导致的。我们用完全相同的图片序列分别测试了两种算法，以消除算法除外的其他因素对速度造成的影响。通过多次测试，实验结果如图4.13所示。

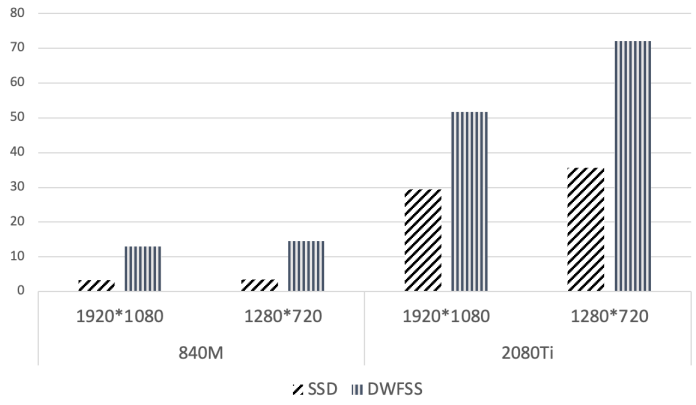


图 4.13: 不同环境不同分辨率下算法速度对比

在速度测试中，不同环境的硬件配置不同，这也是最影响算法运行速度的因素，图中左右两侧的柱状图也说明了这一点。另外还不难看出，在不同的输入尺寸下，即使同一平台速度仍有所不同，这是由于算法在读入图片以及处理图片时会进行一系列操作，这些操作花费的时间与图片尺寸有一定的关系。引入FSSiamese 之后，算法速度提升将近一倍左右，这一方面得益于实验中所使用的双阈值算法没有依靠复杂的特征进行计算，关联部分的时间开销在整个系统中基本可以忽略不计；另一方面则是得益于本文提出的轻量级跟踪网络FSSiamese，能够在更短的时间内完成任务。

4.4　本章小结

本章详细介绍了一种如何将FSSiamese引入到视频目标检测算法中。更加准确地说，该方法是一种通用的算法框架，可以选择合适的跟踪算法以及检测算法来适应不同的场景需求。通过我们的对比实验，引入FSSiamese之后的视频检测算法，能够在目标较少的简单场景下发挥出良好的效果。相比于传统的轻量级目标检测算法，不仅检测的准确率能够大幅度提升，同时还可以将检测花费的时间降低数倍。

第五章　FSSiamese在辅助驾驶系统中的应用

为了验证FSSiamese算法在实际生产中的实用性，我们将其成功地引入到了真实的辅助驾驶系统中。本章主要介绍辅助驾驶系统以及FSSiamese在其中发挥的重要作用。第一小节会介绍辅助驾驶系统的背景以及在码头发挥的重要作用；第二小节开始详细介绍辅助驾驶系统，包括整体的系统架构以及运行过程；第三小节会单独介绍FSSiamese跟踪模块，主要包括该模块的架构以及在系统中发挥的作用；第四小节我们会演示辅助驾驶系统的编译和运行；最后一个小节是本章总结。

5.1　辅助驾驶系统背景

辅助驾驶系统部署的场景是泰国的一个港口，其中的轮胎式集装箱门式起重机（RTG，Rubber Tyre Gantry）和轨道式集装箱门式起重机（RMG）（统称场桥）负责运输集装箱和集装箱卡车（自动引导运输车AGV）。场桥的工作对于整个码头有着至关重要的作用。同时，由于码头环境相对复杂，危险程度高，极度依赖司机的技术操作。但由于场桥具有较高的高度，驾驶室所处的位置也相对较高，所以在前方道路上有着较大的视觉盲区。

图 5.1: 辅助驾驶系统应用场景

为减少场桥的安全隐患，辅助驾驶系统设计了一套基于双目视觉的障碍物检测方法，利用场桥下方安装的双目摄像头来捕捉道路上面的画面，系统检测到前方的障碍物之后反馈给司机准确的预警信号。另外，为了保证系统的可扩展性，系统对不同场景不同运行环境提出了一定的要求，本文提出的算法完全满足实时运行的需求，对于例如码头这种目标少的简单场景，即使在不够理想的硬件条件下也可以达到不错的效果。

5.2　辅助驾驶系统设计

辅助驾驶系统是一个运行在Linux系统下的完整软件，基于C++与Python语言开发而成。输入双目摄像头传递进来的画面，辅助驾驶系统给出画面中感兴趣区域内的障碍物信息，包括障碍物位置、种类以及和摄像机之间的距离。同时，辅助驾驶系统中还有一些其他的功能，例如可以判断当前摄像头是否被遮挡。

辅助驾驶系统的核心在于障碍物的识别，为了适应不同的场景以及运行环境，内部实现了多套模型方案，包括在理想硬件条件下使用的准确率高但运行速率较低的普通方案，以及在非理想条件下使用的轻量级方案。由于不同解决方案使用的整体架构完全相同，所以我们这里着重介绍轻量级的算法，也就是应用了本文提出的FSSiamese跟踪算法的方案。为了更加直观地展示本文提出的算法的作用，我们将算法运行在测试数据集上进行分析，这个数据集同前两章的数据集不同，从实际生产环境中采集而来的，并且同样进行了完全标注，对算法在实际场景的表现有着更加直观的体现。

5.2.1　系统架构

辅助驾驶系统的整体架构如图5.2所示，总体可以分为三个模块，分别为障碍物检测（目标识别与目标跟踪）、双目相关（防遮挡、距离判断）、第三目识别。

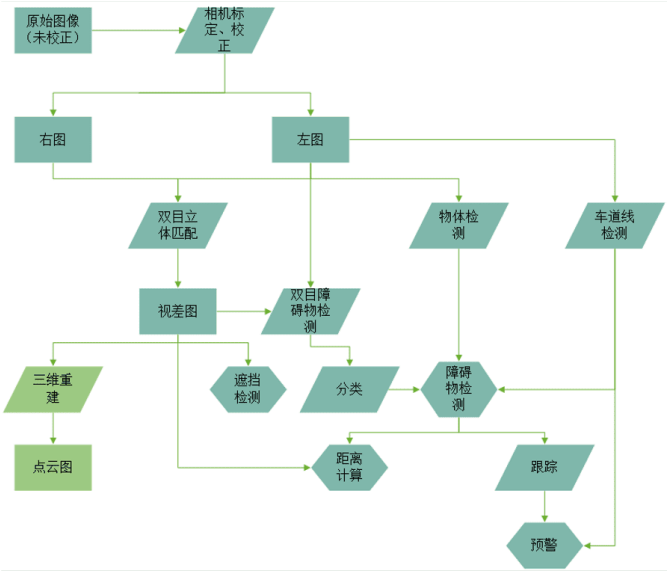


图 5.2: 辅助驾驶系统架构图

其中障碍物识别包括ROI区域选定、目标检测、目标跟踪。ROI区域选定用来过滤系统检测出来的障碍物，位于感兴趣区域之外的障碍物系统会自动忽略。目标检测使用多种方法来检测障碍物，主要有深度学习模型、边缘检测、视差图检测、Stixel检测和光流法检测，在目标数量较少并且使用轻量级检测网络时，可以通过跟踪模块改进检测模块的效果。检测模块的最终输出是障碍物的位置以及类别。区域选定用来确定感兴趣区域，识别画面中的障碍物，对于感兴趣区域外的障碍物，系统将自动过滤。

双目相关包括使用双目视觉算法寻找障碍物，与障碍物检测的结果相融合，减少漏报。前面提到的视差图检测和stixel检测，都需要运用双目视觉得到结果。另外，可以通过双目视觉生成的视差图判断摄像头是否被遮挡，以及判断障碍物距离。

第三目识别与双目识别类似，区别在于只有一个摄像机的画面，通过该画面进行ROI区域的选定，以及对画面上的物体进行检测。

5.2.2　系统运行步骤

辅助系统系统任务较为复杂，我们将各个功能封装成各个模块，启动系统之后，各个模块会按照顺序开始执行。具体步骤如下：

（1）由双目摄像头捕捉场桥前方的图像，第三目捕捉俯视图像，利用张氏标定法对这三张图像分别进行标定，得到摄像头的内参数，具体包括三个摄像头的焦距、畸变参数、驻点坐标以及坐标轴倾斜参数；（2）通过立体标定方法标定两个摄像头，得到摄像头的旋转矩阵、平移矩阵以及外参数；（3）对于每次双目输入的图像，通过前两步的参数进行校正，第三目的图像利用第一步的参数进行校正；（4）将左图送入车道线检测模块，读取配置文件中的车道线信息，如果没有读取到则利用车道线检测算法得到车道线。两条车道线内侧作为感兴趣区域，车道线向外延伸出的两部分作为预警区域，其他部分作为无关区域；（5）计算左右图的视差图，将右图与视差图送入障碍物检测模块，得到场桥前方的障碍物位置以及分类。将第三目矫正过的图像送入障碍物检测模块，得到第三目的障碍物位置以及分类；（6）通过场桥前方检测出来的障碍物位置，利用透视变换或者双目测距的方法计算出障碍物距离；（7）将视差图送入判遮挡模块，返回当前摄像头是否被遮挡，如被遮挡，给出遮挡信号；（8）将场桥前方以及俯视图检测出的障碍物位置以及感兴趣区域信息传入位置过滤模块，如果位于感兴趣区域内，系统会显示出来并给出信号。

其中，最为核心的是检测模块，在系统调用检测模块时，会依据配置文件自动选择系统的检测模式，具体检测步骤如下：

（1）使用Tensorflow深度模型检测图片中的障碍物（$tf\_detector\_$）。模型可根据配置文件中指定的方案自动匹配，如果场景要求检测效果足够好，但对速度要求不高，可以使用较为复杂的目标检测模型；如果对性能要求较高，可以使用轻量级的检测模型，在某些情况下可以在检测时引入FSSiamese提高运行效率；（2）通过深度图检测障碍物（$depth\_detector\_$）。原理是使用U/V视差图检测出障碍物，从而得到候选障碍物物体框；（3）通过边缘检测获得障碍物（$edge\_detector\_$）。原理是使用Canny边缘检测算法对图像做轮廓检测，通过障碍物的轮廓求得候选障碍物的物体框；（4）通过stixel柱状像素检测障碍物（$stixel\_detector\_$），柱状像素是从双目视觉得到的视差图中，分割出若干棒状像素，然后利用投票法合成较宽的像素，这些较宽的像素通常是检测出的障碍物，我们将其作为候选障碍物物体框；（5）通过光流法检测障碍物（$feature\_detector\_$）。这里并没有使用复杂的深度学习类光流算法，而是选择了传统光流法，其原理是利用相邻帧之前图像的变化，计算图像光流并得到特征点，利用这些特征点的位置得到候选障碍物物体框；（6）将前几步检测到的障碍物通过一定的规则进行融合。由于不同检测方法得到的候选物体框质量有所不同，根据经验，项目中通常会优先使用上述方法中基于视差图或者基于深度模型的方法得到的候选物体框。边缘检测模块和光流法模块则是用来做某些极端场景下的辅助工作，在主流检测方法无法检测出障碍物时发挥一定作用。

5.3　FSSiamese跟踪模块

前面提到辅助驾驶系统由多个模块组成，每个模块各司其职，负责实现系统中相对应的功能。这一小节我们详细介绍踪模块，也就是FSSiamese在辅助驾驶系统中的实现以及其发挥的作用。上一个章节提到，将FSSiamese应用到视频目标检测之后，能够在特定场景下，实现更高检测率的同时降低系统的运行开销。因此，在系统的设计方面，跟踪模块被分成两个部分，其中一部分是跟踪器，里面用C++的接口实现了本文提出的FSSiamese算法；第二部分是管理器，不同的管理器负责实现使用FSSiamese做不同的工作，例如将FSSiamese应用在辅助驾驶系统中时，负责实现将FSSiamese与检测模块相结合。为了提高系统的灵活程度，我们将其解耦作为系统的一个可选模式。同时，在软件实现方面，普通检测模块和引入了FSSiamese 的跟踪模块实现了相同的接口，以便简化逻辑，检测器和跟踪器的接口设计如图5.3 所示。

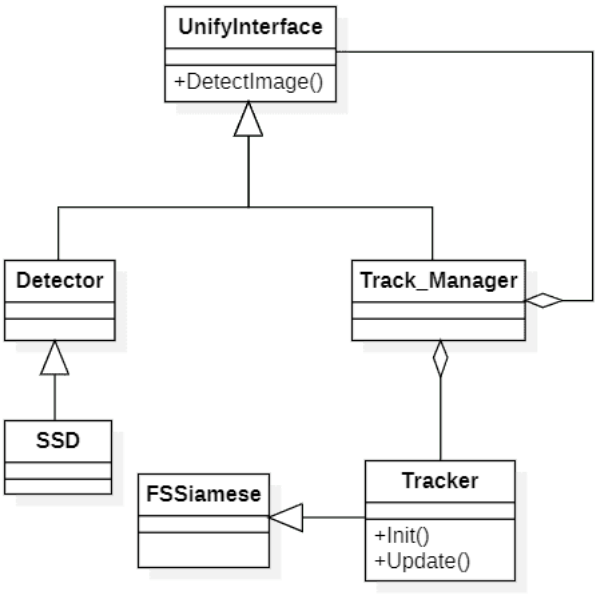


图 5.3: 接口设计图

Detector类与Track_Manager类都实现了统一的检测接口，其中Detector类也是所有目标检测算法的基类，图中只给出了SSD算法作为示例。Track_Manager类中实现了将FSSiamese应用于目标检测的逻辑，从设计的层面讲，其聚合了检测与跟踪。在图中，跟踪器的接口是Tracker接口，接口定义了单目标跟踪常用的Init 方法和Update方法，本文中实现该接口的类是FSSiamese类，实现了FSSiamese算法的功能，当然也可以根据系统配置，选择使用其他的跟踪算法。

5.3.1　跟踪模式测试

为了验证引入FSSiamese后辅助驾驶系统检测障碍物效果是否有所提高，我们设计了如下对比测试：对照组一是关闭了跟踪算法，$tf_detector_$只使用轻量级检测模型SSD；对照组二是打开跟踪算法，$tf_detector_$使用和对照组一相完全相同的检测模型，但会在检测时引入FSSiamese算法，测试指标与上一章在通用数据集上做的实验完全一致。

至于测试集部分，我们选择了18个在码头实际录得的视频片段，并且这些视频片段已经被完全标注。如图5.4所示，图中选取了一部分视频片段的截图，可以看到这些视频包含了车道线内或者附近通过行人时的各种情况，包括行人的站姿、躺姿、跑姿和跪姿，因此实验结果对算法在实际场景中的作用有很强

的指导意义。

(a) 站立　(b) 侧站

(c) 跪　(d) 躺倒

(e) 向前走　(f) 向后走

(g) 向前跑　(h) 向后跑

图 5.4: 码头视频截图

5.3.2　测试结果及分析

本次测试的测试指标和上一章完全相同，我们测试了两个对照组下检测的AP值以及运行速度。AP值的对比结果如表5.1所示。

在辅助驾驶系统运行的实际场景中，由于条件限制无法提供现场障碍物的标注训练集，这使得一般使用通用检测训练集训练得到的模型很容易出现漏报

表 5.1: 辅助系统下算法AP值对比

Model	未引入$FSSiamese$			引入$FSSiamese$		
	0.4(%)	0.5(%)	0.6(%)	0.4(%)	0.5(%)	0.6(%)
站立	29.77	29.66	28.34	32.69	32.45	30.94
侧站	10.34	10.20	8.38	13.16	12.10	9.01
跪姿	2.68	2.43	1.42	4.33	4.18	3.41
躺姿	23.64	23.51	22.54	24.79	24.79	24.64
向前走	20.53	17.98	10.21	24.14	22.13	13.95
向后走	22.86	20.30	15.18	31.49	30.00	22.17
向前跑	27.96	23.90	17.98	45.89	41.33	25.70
向后跑	32.68	24.87	12.90	40.26	26.55	15.34
所有类别	23.43	22.80	20.28	26.70	25.92	23.01

的情况，轻量级检测模型更是如此。可以看到，在辅助驾驶系统中检测的AP值比GOT数据集和ImageNe-VID数据集都要低一些，主要原因是码头上拍摄的视频基本都是工作人员，工作人员身着特殊的工作服，检测模型的训练集中并没有类似的样本，给检测造成了一定的困难。这样的情形在视觉算法落地的过程中经常发生，FSSiamese算法的引入为改善这类问题提供了一种新的思路。在引入FSSiamese 之后，所有情形的检测AP值都有所提高，并且由于FSSiamese 的高实时性，辅助驾驶系统能够在不影响整体运行效率的前提下，更加准确地定位到障碍物，为驾驶员提供信号，避免危险的发生，在现场视频的运行的结果也充分证实了这一点。

由于系统中视频图像输入尺寸均完全相同，因此对于每种场景下相同的算法运行速度基本保持一致，表5.2给出了引入FSSiamese前后系统在上一章中提到的840M环境下的平均速度。

表 5.2: 两种检测算法速度对比

算法	平均运行速度
未引入FSSiamese	8.9FPS
引入FSSiamese	15.2FPS

辅助驾驶系统中检测和跟踪都只是其中的一个模块，在运行时还有其他模块在同时工作，因此辅助驾驶系统在840M环境下的运行速度较低，只有不够9FPS。在引入FSSiamese跟踪模块之后，检测的一部分工作交给了跟踪模块

进行处理，由于FSSiamese的高实时性，运行速度即使和轻量级检测网络相比也要快上许多，因此交给跟踪处理的帧花费时间更短，运行完整个视频的总时长也比未引入FSSiamese时短很多。

5.4　系统运行效果展示

在编译阶段，使用指令”cmake -DUSE_TRACKING=ON/OFF .. && make example”对代码完成编译，其中”ON/OFF”表示是否开启跟踪，如图5.5所示。

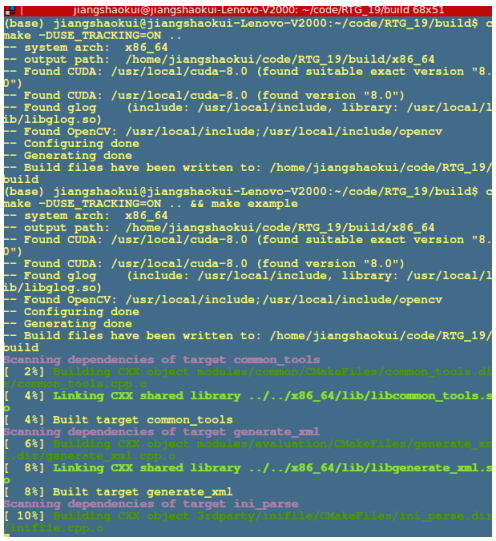


图 5.5: 命令行启动系统演示

使用cmake可以根据CMakeList.txt文件生成Makefile文件，CMakeList.txt中定义了各种项目文件的配置信息以及引用的多个子模块，其中USE_Tracking是定义好的变量，通过该变量控制编译文件时是否启动跟踪模块。调用cmake之后，会将辅助驾驶系统项目的所有模块都进行处理，生成多个Makefile文件，之后的make文件则通过Makefile文件编译链接，生成最终的可执行文件。项目中有许多可执行文件，例如主程序、各个模块的测试程序等等，我们这里只生成主程序，运行展示主程序界面。

编译完成后，使用”./x86/bin/example –flagfile=../config/flags.example” 启动系统，flagfile中指定了视频的输入（包括在线摄像头输入图像或者使用录制完成的视频输入）以及系统的一些其他配置参数，打开图像显示，可以看到系统处理完成的图像结果，如图5.6所示。

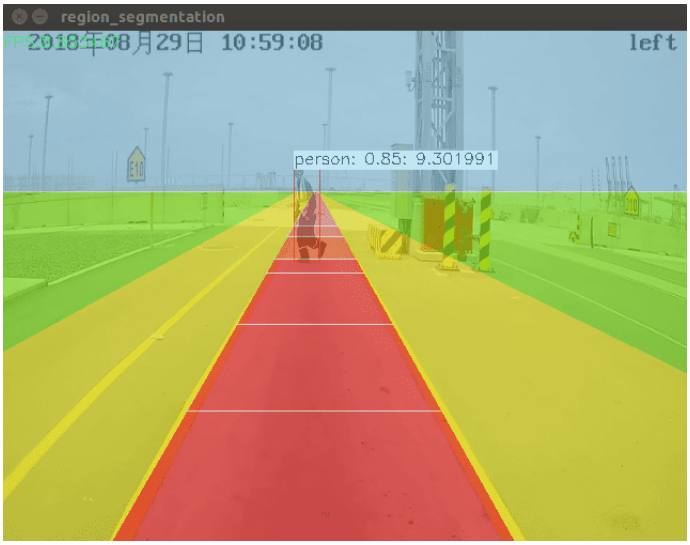


图 5.6: 系统处理结果截图

如上图，辅助驾驶系统会根据车道线检测的结果将画面分为若干区域，其中最中心的红色部分是感兴趣区域；左右两侧的黄色部分是车道线附近的区域，会在预警中发挥作用；绿色部分代表安全区域；最上方的蓝色部分是天空区域。其中安全区域和天空区域系统并不作处理。感兴趣区域中出现的障碍物会被标示出来，当有障碍物进入感兴趣区域时，系统会给出信号，定位到障碍物在画面中的位置，并且给出障碍物的类别以及和摄像机之间的水平距离；当障碍物移出感兴趣区域时，信号消失，对障碍物的标识也会停止出现。

5.5　本章小结

本章主要介绍了本文提出的FSSiamese算法在实际生产中的应用案例，即辅助驾驶系统。作为比较典型、主要依靠计算机视觉工作的系统，本文提出的轻量级目标跟踪算法FSSiamese体积小、易部署，能够很快地作为一个模块应用到辅助驾驶系统中。根据在实际场景中采集到的视频中进行测试，引入FSSiamese跟踪模块之后，辅助驾驶系统识别前方障碍物的准确率有了大幅度提高，并且系统的整体速度没有受到影响，这证明了本文提出的算法具有很强的实用性。

第六章　总结与展望

随着相关滤波算法被提出，以及深度学习在计算机视觉的各个领域被广泛应用，单目标跟踪的研究进入了一个崭新的时代。与传统的目标跟踪算法相比，跟踪的效果也来到了新的高度。然而，目标跟踪还有很多问题需要解决。本文主要针对主流跟踪算法，特别是基于深度学习的跟踪算法普遍运行效率较低的问题，设计了轻量级的孪生网络结构进行目标跟踪，在保证跟踪准确度的同时，也大大提高了目标跟踪的速率。并且通过将轻量级的单目标跟踪算法应用到轻量级的目标检测中，能够大幅度提升视频目标检测的准确度。经过在辅助驾驶系统上的实验可以证明，本文提出的算法在实际的生产中确实能够发挥很好的效果，具有很强的使用性。本文的主要贡献如下：

- 本文提出一种基于孪生网络的轻量级跟踪模型FSSiamese。相比传统的孪生网络跟踪模型，FSSiamese结构更加简洁，参数量少的同时，运行时的计算量也更小。并且由于网络的轻巧，相比于很多跟踪网络需要若干大型的数据集进行多轮训练，FSSiamese只需要两个数据集，训练五轮之内便可以得到理想的模型，使其在其他场景下的微调更加方便省时。能够达成这一点，很大程度上得益于本文提出的训练样本生成方法，可以快速使网络捕捉到图像中的关键信息。根据在多个数据集上的实验，FSSiamese能够在保持较好准确度的同时，大幅度提高跟踪速度。

- 本文将基于轻量级孪生网络的跟踪算法FSSiamese应用到了视频检测中。由于在实际场景中，轻量级检测目标检测算法非常容易发生漏检，再加上有些场景的硬件条件限制，导致其无法使用更加复杂的检测算法。在引入了本文提出的轻量级孪生跟踪网络FSSiamese后，很好地解决了这个问题的同时，运行速度相比之前也有很大提高。经过实验，在检测大部分的常见物体中，同原本轻量级的检测算法相比，引入FSSiamese之后确实能大幅度提高检测准确度。

- 本文提出的算法被应用到了实际的生产环境中。在辅助驾驶系统中，本文提出的FSSiamese算法在提高现场障碍物检测率上发挥了重要作用。另外，速度上的提升也提高了整个系统的运行上限，使系统可以实时完成更多任务成为可能。

按照本文的现有进展，可以在下面几个方面开展工作。首先，针对网络结构的设计，可以按照FSSiamese的思路尝试其他结构，在保持结构精简的同时，进一步降低跟踪的失败率。其次，可以设计一种针对复杂场景进行视频目标检测的算法，替换双阈值匹配算法，进一步增加FSSiamese在实际场景中的可用性。最后，对于如何将单目标跟踪算法引入到更多实际的生产应用中，使其发挥更大的作用，也是一个重要的研究方向。

参考文献

[1] 杨智雄, 余春超, 严敏, 袁小春, 曾邦泽, 粟宇路, 等, 2016. 基于特征融合的粒子滤波红外目标跟踪算法. 红外技术38, 211–217.

[2] 金志刚, 卫津津, 罗咏梅, 刘晓辉, 2015. 基于改进的颜色和surf 特征的粒子滤波目标跟踪. 计算机工程与应用22.

[3] 王保云, 范保杰, 2013. 基于颜色纹理联合特征直方图的自适应meanshift 跟踪算法. 南京邮电大学学报(自然科学版) 3, 18–25.

[4] 王路, 阳琳赟, 卓晴, 王文渊, 等, 2008. 光照鲁棒的mean shift 跟踪方法. 计算机应用28, 1672–1674.

[5] Bay, H., Tuytelaars, T., Van Gool, L., 2006. Surf: Speeded up robust features, in: European conference on computer vision, Springer. pp. 404–417.

[6] Benenson, R., Petti, S., Fraichard, T., Parent, M., 2008. Towards urban driverless vehicles. Journal of Vehicle Autonomous Systems 1/2, 4–23.

[7] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H., 2016. Fully-convolutional siamese networks for object tracking, in: European conference on computer vision, pp. 850–865.

[8] Bhat, G., Johnander, J., Danelljan, M., Khan, F.S., Felsberg, M., 2018. Unveiling the power of deep tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 483–498.

[9] Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M., 2010. Visual object tracking using adaptive correlation filters, in: 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE. pp. 2544–2550.

[10] Bolme, D.S., Draper, B.A., Beveridge, J.R., 2009. Average of synthetic exact filters, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 2105–2112.

[11] Bradski, G.R., 1998. Computer vision face tracking for use in a perceptual user interface .

[12] Comaniciu, D., Ramesh, V., Meer, P., 2000. Real-time tracking of non-rigid objects using mean shift, in: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), IEEE. pp. 142–149.

[13] Dai, J., Li, Y., He, K., Sun, J., 2016. R-fcn: Object detection via region-based fully convolutional networks. arXiv preprint arXiv:1605.06409 .

[14] Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection, in: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), Ieee. pp. 886–893.

[15] Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M., 2017. Eco: Efficient convolution operators for tracking, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6638–6646.

[16] Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M., 2015. Learning spatially regularized correlation filters for visual tracking, in: Proceedings of the IEEE international conference on computer vision, pp. 4310–4318.

[17] Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M., 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking, in: European conference on computer vision, Springer. pp. 472–488.

[18] Danelljan, M., Shahbaz Khan, F., Felsberg, M., Van de Weijer, J., 2014. Adaptive color attributes for real-time visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1090–1097.

[19] Dempski, K.L., 2006. Arbitrary object tracking augmented reality applications. US Patent 7,050,078.

[20] Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences 55, 119–139.

[21] Fukunaga, K., Hostetler, L., 1975. The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on information theory 21, 32–40.

[22] Girshick, R., 2015. Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, pp. 1440–1448.

[23] Han, W., Khorrami, P., Paine, T.L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., Huang, T.S., 2016. Seq-nms for video object detection. arXiv preprint arXiv:1602.08465 .

[24] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, pp. 2961–2969.

[25] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[26] Held, D., Thrun, S., Savarese, S., 2016. Learning to track at 100 fps with deep regression networks, in: European Conference on Computer Vision, Springer. pp. 749–765.

[27] Henriques, J.F., Caseiro, R., Martins, P., Batista, J., 2012. Exploiting the circulant structure of tracking-by-detection with kernels, in: European conference on computer vision, Springer. pp. 702–715.

[28] Henriques, J.F., Caseiro, R., Martins, P., Batista, J., 2014. High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis and machine intelligence 37, 583–596.

[29] Huang, L., Zhao, X., Huang, K., 2019. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[30] Jang, D., Choi, H.I., 1998. Moving object tracking using active models, in: Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269), IEEE. pp. 648–652.

[31] Joshi, K.A., Thakore, D.G., 2012. A survey on moving object detection and tracking in video surveillance system. International Journal of Soft Computing and Engineering 2, 44–48.

[32] Kang, K., Ouyang, W., Li, H., Wang, X., 2016. Object detection from video tubelets with convolutional neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 817–825.

[33] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Cehovin Zajc, L., Vojir, T., Hager, G., Lukezic, A., Eldesokey, A., 等, 2017. The visual object tracking vot2017 challenge results, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 1949–1972.

[34] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernandez, G., Vojir, T., Hager, G., Nebehay, G., Pflugfelder, R., 2015. The visual object tracking vot2015 challenge results, in: Proceedings of the IEEE international conference on computer vision workshops, pp. 1–23.

[35] Kristan M, L.A.e.a., 2016. The visual object tracking vot2016 challenge results. Proceedings of the European Conference on Computer Vision 9914, 777–823.

[36] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105.

[37] Kumar, B.V., 1986. Minimum-variance synthetic discriminant functions. JOSA A 3, 1579–1584.

[38] Kumar, B.V., Mahalanobis, A., Song, S., Sims, S.R.F., Epperson, J.F., 1992. Minimum squared error synthetic discriminant functions. Optical Engineering 31, 915–922.

[39] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

[40] Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J., 2019. Siamrpn++: Evolution of siamese visual tracking with very deep networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4282–4291.

[41] Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X., 2018. High performance visual tracking with siamese region proposal network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8971–8980.

[42] Li, W., Zhao, R., Xiao, T., Wang, X., 2014. Deepreid: Deep filter pairing neural network for person re-identification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 152–159.

[43] Li, Y., Zhang, X., 2019. Siamvgg: Visual tracking using deeper siamese networks. arXiv preprint arXiv:1902.02804 .

[44] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in: European conference on computer vision, Springer. pp. 21–37.

[45] Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 91–110.

[46] Lu, Y., Lu, C., Tang, C.K., 2017. Online video object detection using association lstm, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 2344–2352.

[47] Ma, C., Huang, J.B., Yang, X., Yang, M.H., 2015. Hierarchical convolutional features for visual tracking, in: Proceedings of the IEEE international conference on computer vision, pp. 3074–3082.

[48] Maggio, E., Cavallaro, A., 2005. Hybrid particle filter and mean shift tracker with adaptive transition model, in: Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005., IEEE. pp. ii–221.

[49] Mahalanobis, A., Kumar, B.V., Casasent, D., 1987. Minimum average correlation energy filters. Applied Optics 26, 3633–3640.

[50] Osher, S., Fedkiw, R., 2003. Signed distance functions, in: Level set methods and dynamic implicit surfaces. Springer, pp. 17–22.

[51] Pu, S., Song, Y., Ma, C., Zhang, H., Yang, M.H., 2018. Deep attentive tracking via reciprocative learning. arXiv preprint arXiv:1810.03851 .

[52] Real, E., Shlens, J., Mazzocchi, S., Pan, X., Vanhoucke, V., 2017. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video, in: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5296–5305.

[53] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788.

[54] Redmon, J., Farhadi, A., 2017. Yolo9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271.

[55] Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 .

[56] Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497 .

[57] Ruan, Y., Wei, Z., 2016. Discriminative descriptors for object tracking. Journal of Visual Communication and Image Representation 35, 146–154.

[58] Rui, Y., Chen, Y., 2001. Better proposal distributions: Object tracking using unscented particle filter, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, IEEE. pp. II–II.

[59] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 211–252.

[60] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

[61] Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, IEEE. pp. I–I.

[62] Vojir, T., Noskova, J., Matas, J., 2014. Robust scale-adaptive mean-shift for tracking. Pattern Recognition Letters 49, 250–258.

[63] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H., 2019. Fast online object tracking and segmentation: A unifying approach, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1328–1338.

[64] Weng, S.K., Kuo, C.M., Tu, S.K., 2006. Video object tracking using adaptive kalman filter. Journal of Visual Communication and Image Representation 17, 1190–1208.

[65] Williams, C.S., Becklund, O.A., 2002. Introduction to the optical transfer function. volume 112. SPIE Press.

[66] Wu, Y., Lim, J., Yang, M.H., 2013. Online object tracking: A benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[67] Xiao, F., Lee, Y.J., 2018. Video object detection with an aligned spatial-temporal memory, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 485–501.

[68] Yang, C., Duraiswami, R., Davis, L., 2005. Fast multiple object tracking via a hierarchical particle filter, in: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, IEEE. pp. 212–219.

[69] Zhou, H., Yuan, Y., Shi, C., 2009. Object tracking using sift features and mean shift. Computer vision and image understanding 113, 345–352.

[70] Zhu, G., Wang, J., Wu, Y., Zhang, X., Lu, H., 2016. Mc-hog correlation tracking with saliency proposal, in: Proceedings of the AAAI Conference on Artificial Intelligence.

[71] Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y., 2017. Flow-guided feature aggregation for video object detection, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 408–417.

[72] Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W., 2018. Distractor-aware siamese networks for visual object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 101–117.

简历与科研成果

基本情况

姜少魁，男，汉族，1997年3月出生，山西省大同市人。

教育背景

2018.9～2021.6　南京大学计算机科学与技术系　硕士

2014.9～2018.6　电子科技大学计算机科学与工程学院　本科

攻读硕士学位期间完成的学术成果

[1] **Shaokui Jiang**, Baile Xu, Jian Zhao, Furao Shen “Faster and Simpler Siamese Network for Single Object Tracking”, in arXiv.

攻读硕士学位期间的发明专利

1. 申富饶，**姜少魁**，李俊，赵健. “一种基于孪生网络的目标跟踪方法”（201910930500.8）

攻读硕士学位期间参与的科研课题

1. 国家自然科学基金面上项目“基于深度感知增量式联想记忆神经网络的信息融合系统研究，Information fusion system based on deep perception and incremental associative memory neural networks“（课题年限 2019.01～2022.12），负责神经网络模型相关研究。

攻读硕士学位期间的比赛奖项

1. 韩峰，李雪健，赵加成，**姜少魁**. 2019首届IKCEST“一带一路”国际大数据竞赛，国际二等奖

致　谢

时光飞逝，转眼间我的三年研究生生活即将结束。自己拿到南京大学研究生录取通知书的喜悦至今还历历在目，仿佛是昨天发生的事情一样，然而现在自己即将完成研究生学业，离开这所坐落在南京的美丽校园，顿时有些不舍与伤感。回首这三年，我的研究生生活格外充实，在这里我遇到了很多学识渊博的老师，也认识了很多志同道合的同学，学习到了很多知识，也经历了很多事情，老师和同学们教会了我很多很多，不只有渊博的知识，更多的是为人处世的道理和对科研严谨细致的态度。

感谢我的导师申富饶老师和宋方敏老师，申老师是一位德才兼备、为人和善的师长。在学术上，他一丝不苟、严谨认真，教导着我们带着质疑的眼光去做研究，要以问题为导向而不是为了做研究而做研究。申老师告诉我们要反思自己的研究对于生产生活是否有意义，而不仅仅为了能够发表文章出来，平时要多动脑筋多思考，不单单局限在前人的工作上做修修补补的工作，要勇于提出自己的想法和思路；在研究生生活中，申老师也用他丰富的人生经验教育着我们要积极向上、勇于面对。这三年，我的成长离不开申老师的教导与鼓励，今后我也将带着这些教诲走向社会，开始自己新的旅途。

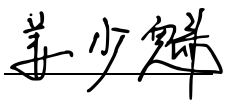
其次，我要感谢赵健老师。赵老师是一位德才兼备、对工作细致认真的好老师。组会报告上赵老师内容详实的论文指导给我留下了深刻的印象，对我论文写作能力的提升有很大的帮助。

此外，我还要感谢辅助驾驶系统项目组的各位同学。特别是高年级的韩峰师兄和已经毕业的黄羽佳师兄，在我刚进组时给予了我很多帮助，在这里祝两位今后的学习工作一切顺利。也感谢项目组的其他同学，和你们一起做事是我三年间最珍贵的回忆。

最后，我要感谢教研室的其他同学以及我的朋友、家人，感谢你们在我研究生期间对我的支持和帮助。在我遇到挫折和困难时，你们一直都是我坚强的后盾。

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名： 

2021 年 5 月 28 日

论文题名	基于孪生网络的轻量级目标跟踪及应用				
研究生学号	MG1833033	所在院系	计算机科学与技术	学位年度	2021
论文级别	✔学术学位硕士 □专业学位硕士 □学术学位博士 □专业学位博士 （请在方框内画钩）				
作者 Email	jiangshaokui@foxmail.com				
导师姓名	申富饶教授、宋方敏教授				

论文涉密情况：

✔ 不保密

□ 保密，保密期（_____年_____月_____日 至 _____年_____月_____日）

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。