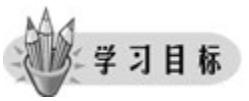
第3章

目标跟踪



知识目标	• 理解目标跟踪的基本概念，包括单目标跟踪（SOT）和多目标跟踪（MOT）的定义和区别。 • 了解目标跟踪的关键技术，包括运动模型、特征提取、观测模型和模型更新策略。 • 了解目标跟踪的评价指标，包括 IDF1 分数、MOTA 等
技能目标	• 能够使用代码实现基本的目标跟踪算法。 • 掌握如何使用深度学习框架，如 TensorFlow、PyTorch 或 Keras，搭建和训练目标跟踪模型。 • 能够针对特定目标跟踪任务，调整现有模型结构，如引入孪生网络、Transformer 等模块以提升性能。 • 能够分析和解决目标跟踪中的特定问题，如遮挡、快速运动、目标形变等
素质目标	• 培养读者的分析能力，使其能够理解和评估不同目标跟踪算法的优缺点。 • 培养读者的批判精神，能够对现有算法提出改进意见和独到见解。 • 培养读者发现问题的能力，使其能够运用合理的方法和技能分析并解决实际问题
思政目标	• 强调在目标跟踪领域的研究和实践中遵守伦理和法律规定，如保护个人隐私和数据安全。 • 鼓励读者在目标跟踪技术的发展中考虑社会责任，如避免技术滥用和促进技术公平性

本章深入目标跟踪领域，强调其在计算机视觉中的重要性。我们将从基础概念讲起，区分 SOT 与 MOT，并讨论其在视频中的应用。探讨关键技术如运动模型、特征提取等，比较传统与深度学习方法。通过 OTB、LaSOT 等数据集，揭示挑战如目标遮挡。案例研究和练习提升问题解决能力，鼓励批判性思维，促进算法改进。强调遵守法规、伦理，保护隐私，确保技术正面影响。读者将掌握目标跟踪的挑战和潜力，激发探索热情。

3.1 目标跟踪简介

目标跟踪（Object Tracking）是计算机视觉领域的一个重要研究方向，它主要关注的是如何在视频序列中实时地定位和跟踪目标对象（见图 3-1）。目标跟踪技术在许多实际应用中都发挥着重要作用，如视频监控、自动驾驶、人机交互、运动分析等。目标跟踪算法的核心任务是在每一帧图像中准确地识别出目标的位置，并在整个视频序列中保持对目标的持续跟踪。

图 3-1　目标跟踪示意图

3.2　目标跟踪关键技术

主流视频目标跟踪系统的架构可从图 3-2 中得到直观展示，整体上划分为运动模型、特征提取、观测模型、模型更新 4 个主要环节。运动模型主要基于上一帧目标的状态，生成可能包含目标的候选区域，用于进一步的处理。特征提取指的是利用相应的特征提取方法，表征目标图像。观测模型基于提取到的目标特征，来判断它是否属于目标。模型更新指的是根据不同时间序列下的目标状态，调整跟踪器模型以适应目标变换和环境变换。

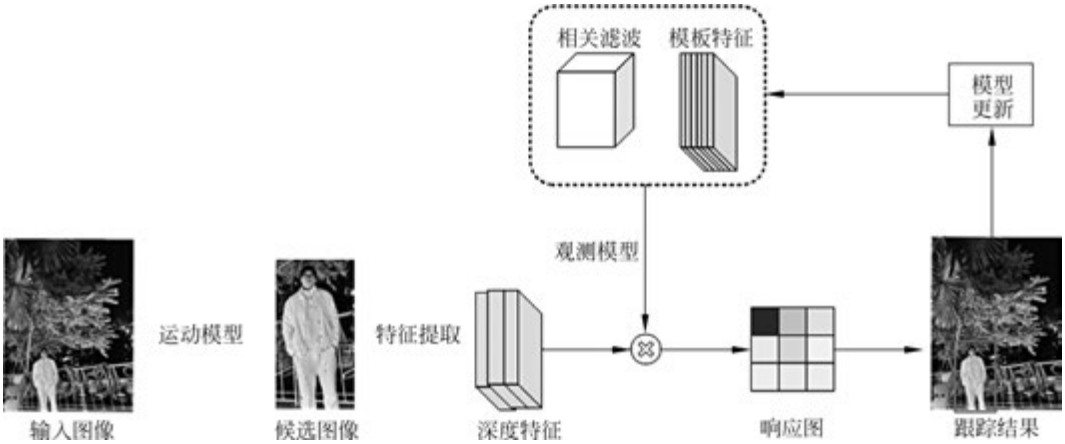


图 3-2　目标跟踪系统的架构图

3.2.1　运动模型

在目标跟踪领域，运动模型是用来预测目标在下一帧或未来几帧中的位置和运动状态的模型。运动模型的介绍主要集中在以下几个方面。

1. 卡尔曼滤波器

这是一种线性动态系统的状态估计方法，对视频进行运动目标检测，通过简单的匹配方法来给出目标的第一个和第二个状态，从第三个状态开始，先使用卡尔曼滤波器预测出当前状态，再用当前帧图像的检测结果作为观测值输入给卡尔曼滤波器，得到的校正结果就被认为是目标在当前帧的真实状态。它使用反馈控制的概念来估计目标的位置，能够提供目标状态的最优估计，比较适合估计行人、车辆等目标，因此在多目标跟踪器领域运用很广，如经典的 SORT[1] 和 ByteTrack[2] 方法。

2. 粒子滤波器

与卡尔曼滤波器相比，粒子滤波器是一种非参数化的概率估计方法，它使用一组随机样本（粒子）来表示目标的状态分布。粒子滤波器适用于非线性和非高斯噪声的场景，能够提供更加灵活的状态估计，因此相较卡尔曼滤波器，粒子滤波器也更受欢迎。

3. 滑动窗口方法

这种方法通常与相关滤波跟踪器一起使用，通过在上帧预测的目标中心位置周围滑动窗口来生成候选区域，便于寻找目标位置和估计目标形状。得益于机器学习和深度学习良好的特征表达，滑动窗口往往不需要在原始图像上滑动，而是先对原始图像下采样提取特征，在特征图上滑动窗口，在特征图上预测目标位置和形状。这避免了粒子滤波需要在原始图像中生成大量稠密的候选框，减少样本生成和处理时间。上述方法并未真正地基于目标的运动规律建立运动模型。

4. 光流法

光流法通过计算图像序列中的像素运动来估计目标的运动。这种方法可以学习目标的运动轨迹，但计算复杂度较高，且在单目标跟踪中应用较少。例如 SINT[3]，这些研究工作引入了光流算法来学习目标的运动轨迹，进而生成基于运动模型的候选框。然而，由于单目标跟踪中目标种类不定、缺乏统一的运动规律，建立一个适应性强的运动模型非常具有挑战性。

近年来，深度学习方法在目标跟踪领域也得到了应用，通过学习目标的历史运动信息来预测目标的未来状态。在目标跟踪中，运动模型的建立对于预测目标的下一位置至关重要，尤其是在目标遮挡、快速运动或形状变化等复杂情况下。不同的运动模型适用于不同的跟踪场景和需求，选择合适的运动模型可以显著提高跟踪准确性和鲁棒性。

3.2.2 特征提取

特征提取是目标跟踪中一个重要的步骤，它涉及从目标候选框中提取有助于跟踪的特征。特征提取主要经历了两个阶段：手工设计特征和深度特征。

1. 手工设计特征

在这个阶段，学者们会使用各种手工设计的特征，并将其与不同的机器学习算法结合起来使用。

（1）哈尔特征：通常与支持向量机结合使用进行目标跟踪。

（2）单通道灰度特征：早期滤波跟踪器如 MOSSE[4] 特征。

（3）ColorNames 特征：该特征由 Danelljan 和 Martin 等引入[5]，与灰度特征结合使用，提升了相关滤波器的精度。

（4）Histogram of Oriented Gradients（HOG）：这是一个广泛使用的特征，许多性能优异的相关滤波跟踪器如 AutoTrack[6] 和 RCF[7] 等都不同程度地使用了 HOG 和 Color Names 特征。

2. 深度特征

随着深度学习的流行，基于卷积神经网络（CNN）的特征开始被用来取代手工特征。这些特征不需要人为设计，而是通过大量数据驱动学习，具备良好的语义特征表达能力。

（1）预训练的 CNN 特征：由于缺乏大规模基于跟踪的训练数据，许多跟踪器如 HCF[8] 和 RPCF[9] 直接使用在 ImageNet 数据集上预训练的 CNN 特征。

（2）特征融合选择策略：学者探索了特征融合选择策略，以提高目标定位的准确性。

（3）端到端学习：一些跟踪器如 TrackerNano 构建了轻量级的骨干特征网络用于训练相关滤波器，实现了端到端的学习，性能可以与使用现成 CNN 的相关滤波器相媲美，同时运行速度更快。

（4）更先进的网络：跟踪方法的特征提取网络从 ResNet[10] 到最近的 Transformer[11] 模型。

特征提取是目标跟踪中的一个核心环节，其宗旨在于捕捉目标的本质属性，从而赋予跟踪算法在多变环境下准确识别和定位目标的能力。这些特征不仅要具有高度的区分性，以确保目标在众多潜在干扰中独树一帜，还要具备足够的鲁棒性，以抵御光照变化、目标形变等环境因素的挑战。

这些方法能够从大量数据中自动提取和优化特征，显著提升了目标跟踪的准确性和适应性。此外，特征提取技术还在不断探索新的维度，如时空特征的联合学习、多模态特征的融合等，以期在动态和复杂的视觉场景中实现更加精准和鲁棒的目标跟踪。

3.2.3 观测模型

在目标跟踪的领域中，观测模型扮演着至关重要的角色，它主要负责对候选区域进行特征提取和评分。这一过程是目标跟踪算法的核心，因为观测模型的准确性直接影响到跟踪效果。观测模型通过深入分析候选区域的特征，对它们进行精确评分，从而识别出与目标最为匹配的候选框。

根据观测模型的不同，跟踪算法可以被划分为两大类：生成式和判别式。生成式模型在目标跟踪的早期发展阶段占据了主导地位，它们通过学习目标的外观模型来预测目标在新帧中的位置。然而，自 2012 年 TLD 方法的提出[12]，判别式跟踪方法开始崭露头角，并逐渐成为主流。判别式模型不直接学习目标的外观模型，而是通过比较候选框与目标之间的相似度来进行跟踪，这种方法在处理复杂场景和目标变化时展现出了更高的灵活性和准确性。

(1) 生成式模型在目标跟踪中扮演着寻找与目标模板高度相似候选对象的角色，其过程本质上是一种模板匹配策略。这类模型致力于在复杂场景中识别与预定义目标模板最为契合的对象。为了实现这一目标，生成式模型依赖多种理论方法，包括但不限于子空间学习、稀疏表示和字典学习等。这些方法通过构建目标的特征表示，使得模型能够在新帧中有效地定位目标。

(2) 与生成式模型不同，判别式模型的核心在于区分目标与背景，通过训练一个分类器来实现目标的分离和跟踪。判别式模型的关键在于学习一个能够明确区分目标和背景的分类器，即使在目标与背景之间存在显著差异的情况下，也能保持跟踪的准确性。这类模型涵盖了多种算法，包括支持向量机、随机森林、相关滤波、孪生网络以及分类网络等。这些判别式方法通过直接学习目标与背景的差异，提高了跟踪算法在复杂环境下的鲁棒性和适应性。

3.2.4 模型更新

在目标跟踪的领域中，目标的运动状态和姿态是动态变化的，这对跟踪算法提出了更高的要求。由于目标的外观模型是基于先前捕获的样本信息构建的，它可能无法及时适应目标当前的外观和姿态变化。为了解决这一挑战，外观模型必须能够根据目标姿态的实时变化进行适当的更新，以确保跟踪的连续性和准确性。

移动平均更新策略提供了一种简洁而高效的解决方案。这种策略允许跟踪器根据最新的跟踪结果，按照预定的权重更新模型，从而快速适应目标的最新状态。这种方法因其有效性，在多种跟踪器模型中得到了广泛应用。在跟踪过程中，历史帧的跟踪结果可能会包含大量背景像素。一些先进的方法通过分析跟踪器分类响应图的分布特性，智能地决定是否对历史帧结果进行更新，以提高跟踪的准确性。此外，目标跟踪过程中会产生大量的易于分类的负样

本，而难以分类的负样本相对较少。判别式跟踪算法特别注重挖掘这些困难负样本，以增强跟踪器的分类能力。例如，离线算法 Da-SiamRPN[13] 和在线算法 MDNet[14] 都采用了这种策略。通常，深入研究和关注困难负样本是模型更新的关键，这对于提升基于判别式跟踪器的鲁棒性至关重要。对于离线跟踪模型，如孪生网络跟踪器，UpdateNet[15] 引入了一个轻量级的更新网络，它能够根据上一帧的跟踪结果自适应地更新目标模板图像。在最新的基于 Transformer 的跟踪方法中，STARK[16] 和 MixFormer[17] 等研究工作提出了分数预测头部，用以判断当前帧预测的目标是否可靠。如果预测结果可靠，则更新模板图像；否则，将其丢弃，不予考虑。这些创新的方法展示了目标跟踪领域在模型更新策略上的多样性和深度，它们共同推动了跟踪技术的发展，使其更加适应目标的动态变化和复杂场景的挑战。

3.3 单目标跟踪算法

目标跟踪是计算机视觉领域的关键研究领域之一，专注于在视频序列中实现对一个或多个目标对象的持续定位和追踪。这一技术可以根据不同的应用场景和需求，被精确地划分为两个主要类别：单目标跟踪（Single Object Tracking，SOT）和多目标跟踪（Multi-Object Tracking，MOT）。单目标跟踪致力于在视频序列中锁定并追踪单一对象的运动轨迹，而多目标跟踪则面临更为复杂的挑战，它需要同时处理和追踪视频中出现的所有目标对象。这两种跟踪方法在监控系统、自动驾驶、体育分析等多个领域都有着广泛的应用。随着技术的不断进步，目标跟踪算法的精确度和鲁棒性也在不断提升，以适应更加多变和复杂的现实世界场景。在单目标跟踪领域，深度学习框架的应用分类如下。

1. 基于卷积神经网络

CNN 是计算机视觉中最常用的一种深度学习架构，它能够自动提取图像特征，并在目标检测和识别任务中表现出色。在目标跟踪中，CNN 被用于学习目标外观的表示，以及用于区分目标和背景的分类器。

2. 基于循环神经网络

循环神经网络（Recurrent Neural Network，RNN）的思想是将上一个循环块的输出值作为下一个循环块输入的一部分，建立不同输入间的联系。在计算机视觉领域通常作为建立输入时序的模型。RFL[18] 利用长短时记忆单元（LSTM）[19] 保留目标空间与时序信息。在训练资料学习之后，在线过程不再需要微调。相对于其他 CNN 与 SNN 框架有较好的时序适应性。循环神经网络可以很好地建立时空间关系，考虑上下文信息，但是由于其训练困难，一般只有在长时间跟踪领域上有较好的表现。

3. 基于孪生网络

孪生网络是一种独特的神经网络架构，它由两个或多个结构相同且权重共享的子网络构成。这些子网络分别处理不同的输入数据，并通过对比学习的方法来识别和学习输入之间的相似性。在目标跟踪的应用中，孪生网络的这一特性被用来处理模板图像（即目标的初始图像）和搜索区域图像，通过比较两个子网络的输出，算法能够准确地定位目标对象。

孪生神经网络（Siamese Neural Network，SNN）的概念起源于目标识别领域[20]。其核心思想在于，对于两个相似的物体，它们在经过相同的特征提取过程后，产生的特征图应当是相似的，并且在空间分布上具有一致性。这一理念与模板匹配技术，如相关滤波方法，有着天然的契合点。因此，SNN 在目标跟踪领域迅速崛起，成为继相关滤波框架 DCF 之后，最受关注的跟踪框架之一，如图 3-3 所示。在实现上，孪生网络通常采用卷积神经网络（CNN）作为特征提取的基础架构。大多数基于孪生网络的跟踪方法也确实采用了 CNN 来实现特征提取。因

此，孪生网络在很大程度上可以被视为 CNN 的一个特殊变种，它们在目标跟踪任务中展现出了卓越的性能和灵活性。通过这种方式，孪生网络能够学习到目标的深层特征表示，可在复杂多变的视觉环境中实现稳定而准确的目标跟踪。

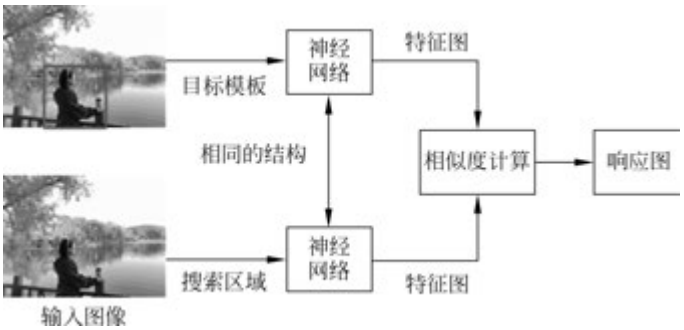


图 3-3　Siamese 类方法框架

自 2016 年起，SINT[3]将孪生网络的概念引入了目标跟踪领域，这一创新为跟踪技术的发展带来了新的视角。SiamFC[21]在此基础上进一步发展，构建了一个完整的孪生网络框架，如图 3-4 所示。SiamFC 的核心思想与相关滤波技术中的特征判别相似，即在搜索区域中，置信度分数最高的点即为目标的预测位置。由于 SiamFC 在速度上显著超越了其他基于 CNN 的跟踪方法，一经推出便受到了广泛关注。这个框架虽然简单，但却蕴含着巨大的潜力，它不仅提升了目标跟踪的效率，还为该领域引入了跨学科的创新思路。

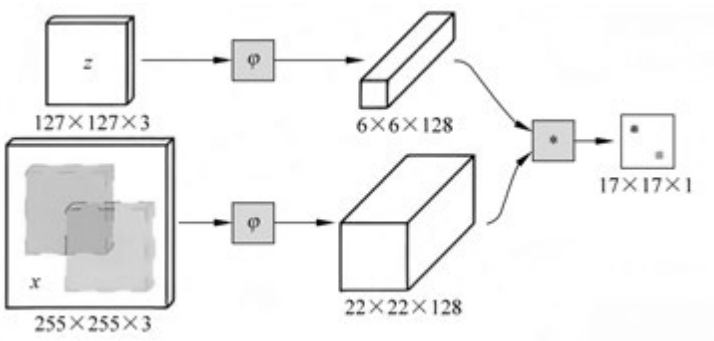


图 3-4　SiamFC 跟踪算法原理

在 SiamFC 的基础上，SiamRPN[22]进一步创新，受到目标检测领域的启发，引入了区域候选网络(Region Proposal Network，RPN)。这一改进有效解决了 SiamFC 在处理目标尺度变化时的不足，通过边界框回归技术替代了传统的多尺度检测方法。SiamRPN 将跟踪任务巧妙地分解为两个并行的分支：一个负责目标定位，另一个负责边界框回归，如图 3-5 所示。这种设计不仅提高了跟踪的准确性，也增强了算法对不同尺度目标的适应能力，使得 SiamRPN 在目标跟踪领域中占据了重要地位。

4. 其他深度学习方法

在目标跟踪领域，多种先进的机器学习技术被用来提升跟踪算法的性能。这些技术包括但不限于以下几个。

(1) 生成对抗网络(GAN)：这种深度学习模型能够生成高逼真的数据样本，用于增强训练数据集，提高模型对新场景的泛化能力。在目标跟踪中，GAN 可以用来生成更多样化的目标模板，以应对目标外观的快速变化。

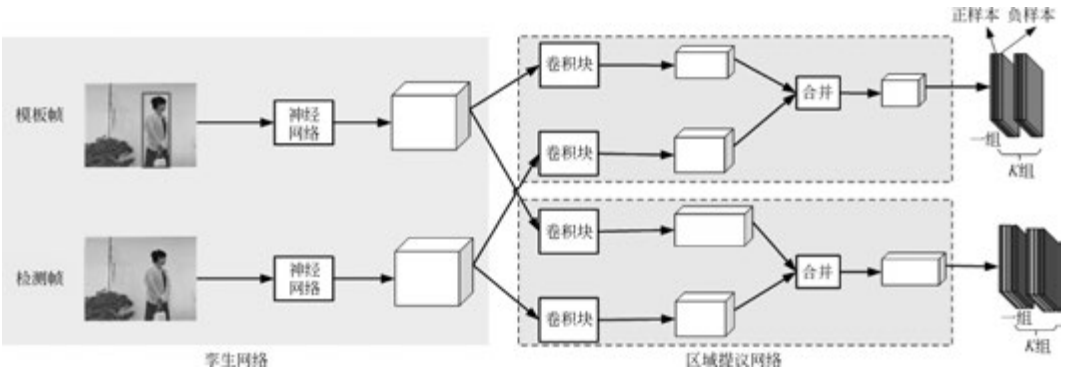


图 3-5 SiamRPN 跟踪算法原理

(2) 变分自编码器(VAE)：VAE 是一种无监督学习模型，能够学习数据的有效表示。在目标跟踪中，VAE 可以用来学习目标的动态模型，从而更好地预测目标在未来帧中的位置和姿态。

(3) 强化学习(RL)：RL 是一种学习策略的方法，通过与环境的交互来优化决策过程。在目标跟踪中，RL 可以用来优化跟踪策略，使跟踪器能够根据环境反馈动态调整其行为，以实现更精确的跟踪。

这些方法的引入，为解决目标跟踪中的挑战提供了新的途径。例如，GAN 可以用于生成更丰富的训练样本，VAE 有助于捕捉目标的动态特性，而 RL 则可以用于设计更加智能的跟踪决策过程。通过这些技术的融合和创新，目标跟踪算法能够更加灵活和鲁棒，以适应复杂多变的现实世界。

3.4 多目标跟踪算法

多目标跟踪(Multi-Object Tracking，MOT)是一项在视频中同时捕捉和追踪多个目标的复杂任务。它不仅要求算法能够准确识别和锁定每个目标的轨迹，还需要考虑目标之间的复杂交互作用，例如，目标的相互遮挡、合并和分离等现象。在 MOT 的应用中，跟踪的对象可以多样化，包括但不限于行人、车辆以及其他各类物体。随着计算机视觉技术的飞速进步，MOT 技术已经被广泛地应用于多个关键领域，包括视频智能监控、人机交互、智能导航等[23]。

此外，MOT 作为一项基础而关键的技术，对于高级计算机视觉任务的发展起到了支撑作用。它为姿态估计、行为识别、行为分析以及视频内容分析等任务提供了不可或缺的数据基础和分析前提。这些高级任务往往依赖 MOT 提供的精确目标定位和轨迹信息，以进一步提取和分析目标的行为模式或交互关系。

基于深度学习的 MOT 算法根据检测和跟踪阶段是否独立，可以分为基于检测的跟踪(Detection Based Tracking，DBT)和联合检测跟踪(Joint Detection Tracking，JDT)两个类别[24]。图 3-6 为 DBT 和 JDT 算法流程图。从图中可以看出，DBT 算法主要分为检测和跟踪两个阶段。第一阶段使用基于卷积神经网络的检测器对每一帧图像中感兴趣的目标用矩形框进行标注，第二阶段跟踪器使用检测器的输出结果再次提取目标的外观、运动等特征，并计算和 $t-1$ 帧中已存在轨迹特征之间的相似度以进行目标和轨迹的关联。JDT 算法则是将检测和跟踪融合到一个框架中，对检测器中提取的特征进行了复用，跟踪器可以直接利用检测器输出目标的外观特征，避免了跟踪器接收检测结果的过程。

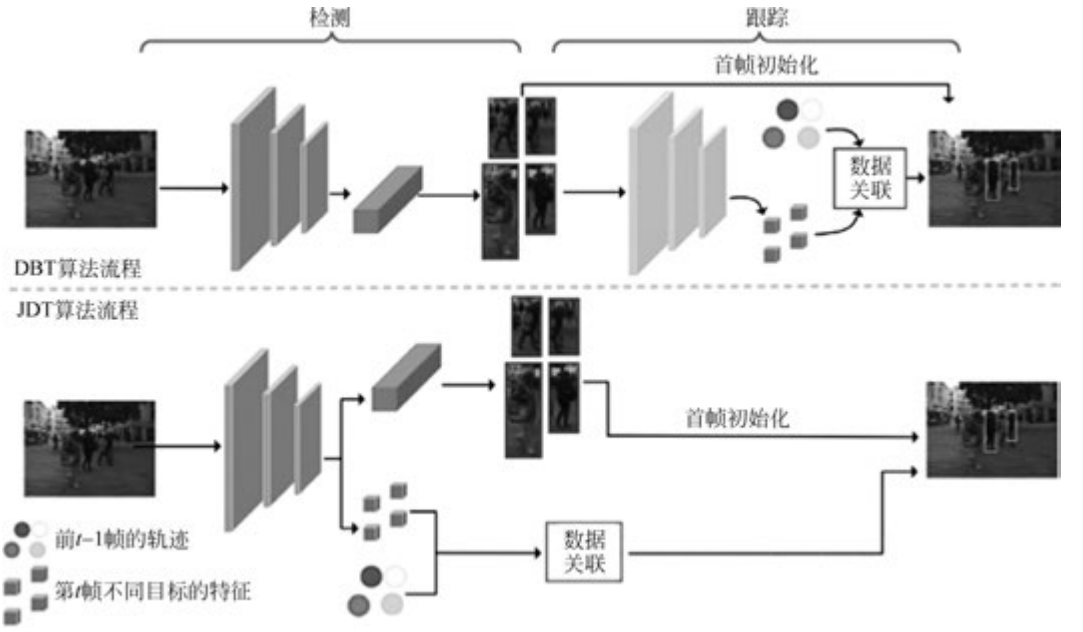


图 3-6 DBT 和 JDT 算法流程图

近些年，很多 MOT 算法都是基于上述两种范式，下文将介绍常见的基于深度学习的 MOT 方法（见图 3-7）。

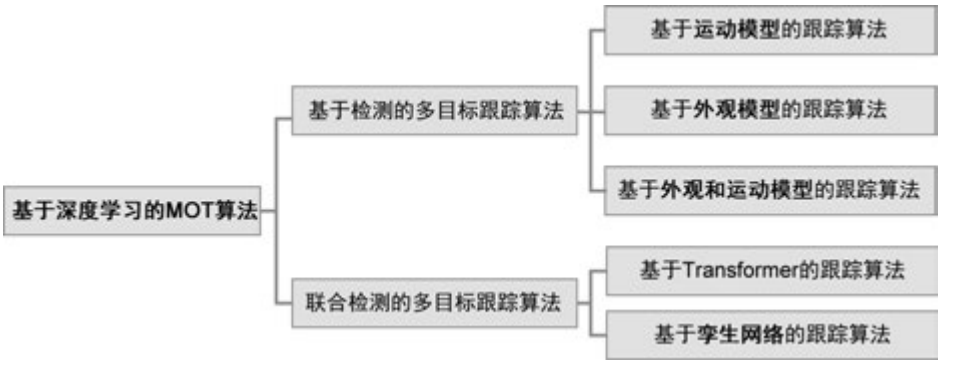


图 3-7 常见的基于深度学习的 MOT 方法

3.4.1 基于检测的多目标跟踪算法

基于检测的多目标跟踪算法（Detection-Based Tracking，DBT）是多目标跟踪（MOT），它将跟踪过程分为两个主要阶段：检测和关联。在检测阶段，利用深度学习驱动的目标检测算法来识别视频中每一帧中潜在的目标物体位置。这一阶段的输出为跟踪阶段提供了丰富的候选目标信息。随后，在跟踪阶段，算法采用检测结果作为输入，通过精心设计的关联策略来构建目标在不同帧之间的轨迹。由于跟踪器完全依赖检测器的输出，检测的准确性对最终的跟踪性能有着决定性的影响。

为了对跟踪算法的性能进行公正的评估，并消除检测性能波动带来的影响，许多 MOT 数据集提供了标准化的检测结果。同时，一些先进的跟踪算法采用自定义检测器，通过提升检测的准确性来增强整体跟踪性能，例如，采用 Faster R-CNN[25] 和 YOLO 系列等高质量的检测器。

跟踪阶段是MOT算法的精髓所在，在这一阶段，算法会利用检测阶段的输出，结合运动、位置、外观等线索或这些线索的综合，来实现检测框与现有轨迹之间的跨帧关联。以下是对不同类型跟踪算法的详细分析。①基于外观模型的跟踪：这类算法侧重于目标的视觉特征，如颜色、纹理或深度学习提取的特征，来区分和关联目标。②基于运动模型的跟踪：这类算法主要依赖目标的运动信息，如速度和方向，来预测和更新目标的轨迹。③基于外观和运动模型结合的跟踪：这类算法综合了外观和运动信息，以实现更鲁棒和准确的目标跟踪。下面将分别从基于外观模型、基于运动模型、基于外观和运动模型分别进行算法分析。

1. 基于外观模型的多目标跟踪算法

目标的外观特征是多目标跟踪(MOT)中计算相似度矩阵的重要线索。在单目标跟踪(SOT)算法中也有不少基于外观模型，其主要区别在于SOT侧重于构建复杂的外观模型以区分目标和背景的不同，而MOT中则大多是利用目标的外观信息来区分不同目标。Sun等[26]提出了一个深度亲和网络(Deep Affinity Network，DAN)用于对不一定相邻帧之间目标的外观进行建模并评估其相似性(见图3-8)，以便后续进行可靠的轨迹匹配。

其亲和估计的计算过程如图3-9所示，Nm=5。图3-9(a)和图3-9(b)中的帧1和帧30共同包含5个检测对象。图3-9(c)创建一个中间矩阵，考虑虚拟对象(带0的行和列)，以实现每帧Nm=5。图3-9(d)在两帧之间增加额外的列和行，以包括未识别(UI)目标(分别为离开和进入的对象)。

行人重识别(Person Re-identification)，简称为Re-ID，是利用计算机视觉技术判断图像或者视频序列中是否存在特定行人的技术，受Re-ID任务的启发，许多MOT任务也采用了基于Re-ID的方法来提取目标的外观特征。Li等[27]使用了自监督外观模型进行Re-ID特征的提取，在不使用跟踪注释的情况下就能训练一个新的外观嵌入模型。基于外观模型的MOT算法可以很好地提取检测框内具有鉴别力的特征以便对目标进行更好的描述，从而提高算法性能，在目标运动模式复杂或相机运动场景下表现更好。缺点是外观模型容易受到光照、遮挡等情况的影响，进而造成误判。

2. 基于运动模型的多目标跟踪算法

运动特征是MOT跟踪阶段另一个常用的特征。运动模型通过捕捉目标的动态行为来预估目标在未来帧的潜在位置，从而缩小搜索空间。其中最常用的是线性运动模型，它假设目标在每个时间段内的运动速度是恒定的，但是在复杂场景下，如物体突然加速、变向、遮挡等情况下，线性运动模型就不再适用了。

SORT算法[1]把目标帧间的位移视作线性匀速运动，通过卡尔曼滤波器基于前一帧的目标位置来预测当前帧的目标位置，当视频帧率较高时，即使运动目标在整个运动过程中是非线性的，目标的运动依然可以在很短的时间间隔内视为线性运动，故算法也能很好地工作，但是在目标跟踪的过程中物体发生遮挡，导致目标在视频序列上的时间间隔变大，即使一个目标在丢失一段时间后可以通过SORT算法重新关联，但是由于时间误差放大，卡尔曼滤波参数已经远远偏离真实值，因此可能会再次丢失。此外，SORT算法在高帧率视频的连续帧中物体的位移和噪声可能有相同的量级，且噪声会累积到后续位置估计中。为了缓解这一问题，Cao等[28]提出OC-SORT(见图3-10)，提出以观测为中心的更新策略来减少累积误差，即一旦一个跟踪轨迹在丢失跟踪一段时间后又重新与观测结果相关联，便会根据丢失目标期间开始和结束的观测值来生成虚拟轨迹，用虚拟轨迹重新更新丢失期间的卡尔曼滤波器的参数，这样就避免了误差的积累，在图3-10中，红色框是检测到的，橙色框是活动轨道，蓝色框是未跟踪的轨道，虚线框是卡尔曼滤波器的估计。在关联过程中，使用OCM增加速度一致性代价。由于

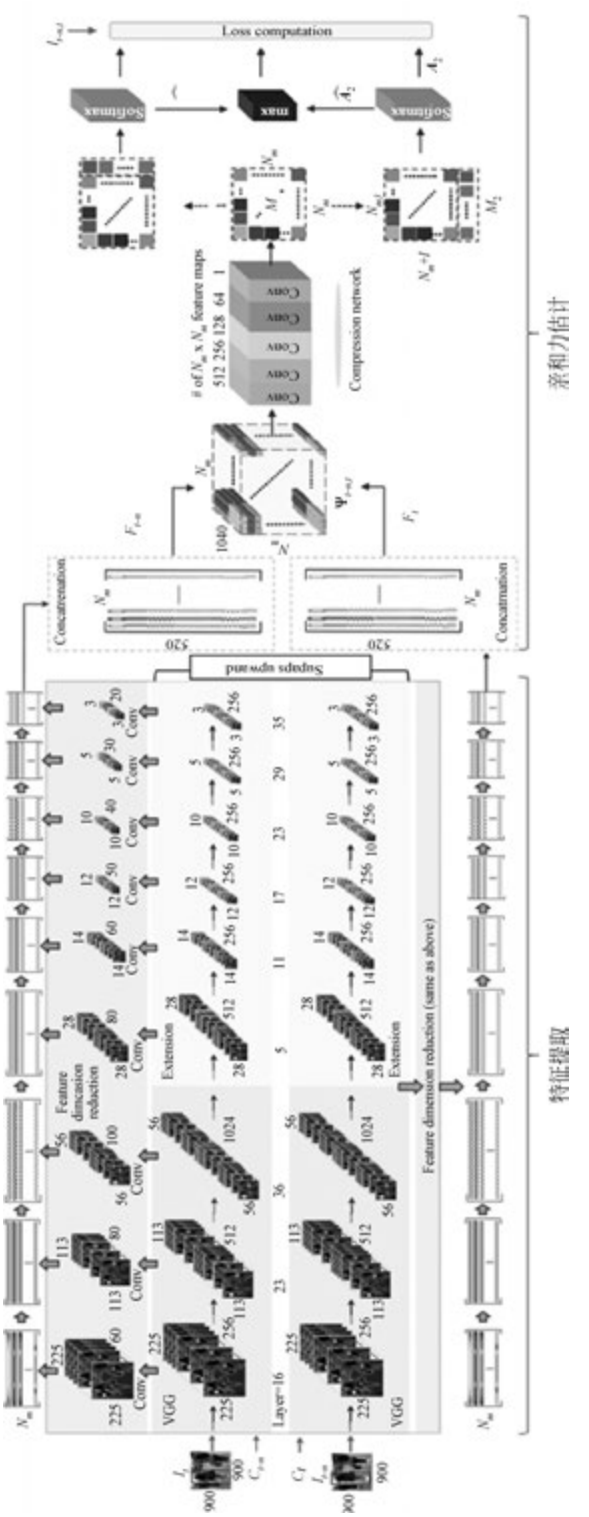


图 3-8 深度亲和网络架构图

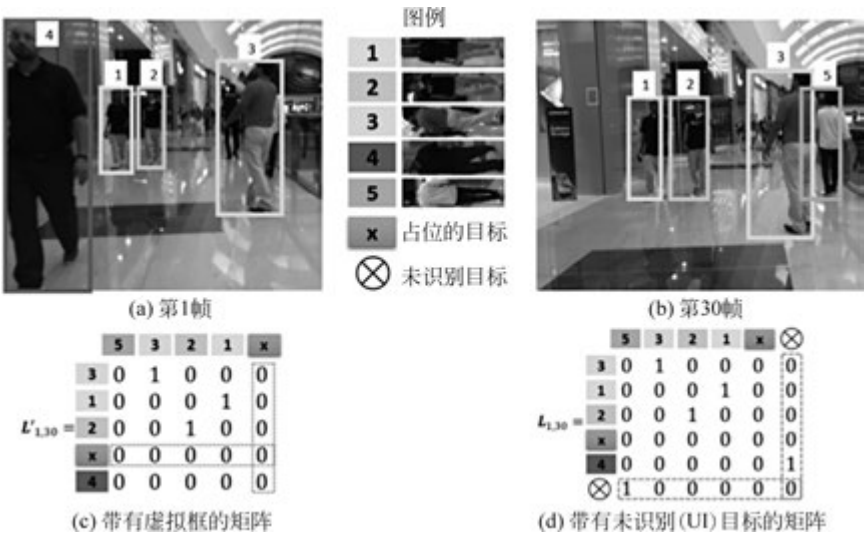


图 3-9 第 1 帧到第 30 帧的数据关联矩阵示意图

遮挡,1 号目标在第 $t+1$ 帧中丢失。但是在下一帧中,它通过引用 OCR 对帧 t 的观察得到恢复。它被重新跟踪触发 ORU 从 t 到 $t+2$ 帧的 KF 参数。

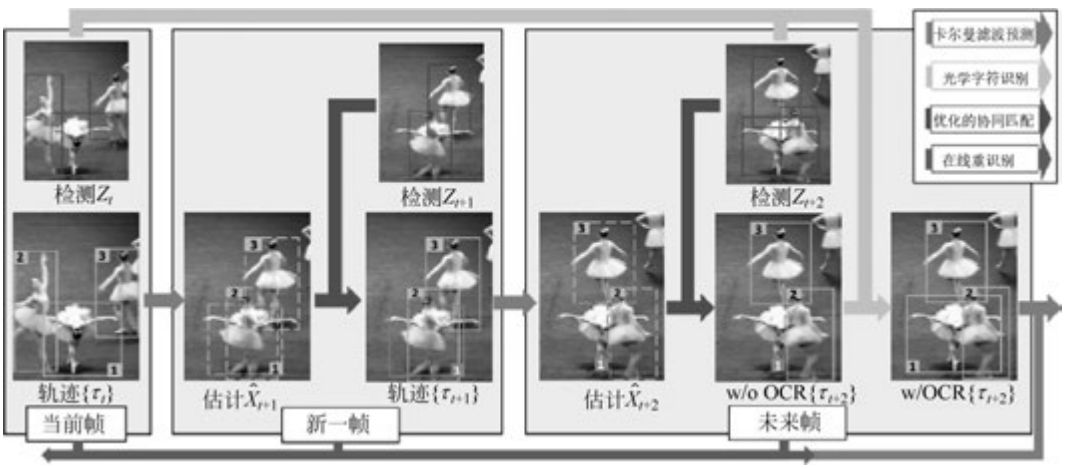


图 3-10 OC-SORT 网络框架

另外,文章还提出用观测值代替估计值以降低运动方向计算的噪声,并引入运动方向的一致性项来帮助关联,提出以观测为中心的恢复来将轨迹最后一次观测与新检测到的观测相关联来恢复轨迹。实验表明,此方法相较于 SORT 对遮挡和非线性物体运动具有更强的鲁棒性(见图 3-11),目标在第二步和第三步之间被遮挡,跟踪器在第三步找到它。黄框是探测器的状态观测。白色恒星是没有 ORU 的估计中心。黄色的星星是由 ORU 固定的估计中心。第四步的灰色恒星是估计的中心,没有 ORU,与观测结果不匹配。

基于运动模型的 MOT 算法可以很好地改善目标遮挡和检测器漏检而导致的轨迹碎片化问题,相比于利用外观特征的跟踪器,速度更快,对检测器的性能依赖更小,在一些外观特征提取不可靠的复杂场景中更具鲁棒性。但这类算法非常依赖运动模型预测目标的位置,所以对于运动轨迹复杂场景的跟踪能力会有所下降。

3. 基于外观和运动模型的多目标跟踪算法

在面对诸如目标遮挡或摄像机抖动等复杂场景时,单一依赖目标的外观特征或运动特征

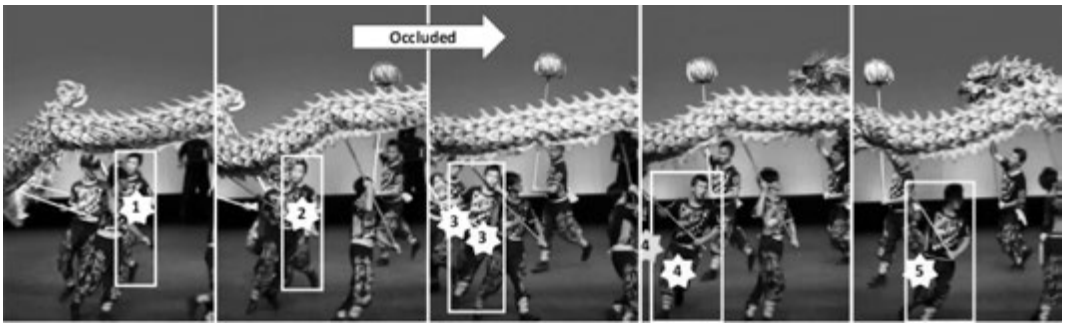


图 3-11　以观察为中心的重新更新(ORU)如何在跟踪中断时减少错误积累的示例

进行跟踪往往难以实现稳定而准确的跟踪。为了克服这些挑战,研究人员开始将目标的外观特征与运动信息相结合,融入多目标跟踪(MOT)算法之中,以此提升跟踪的精确度和鲁棒性。SORT[1]算法是一种基础而直观的方法,它采用 Faster R-CNN[25]进行目标检测,结合卡尔曼滤波器和匈牙利算法来实现跟踪。然而,SORT 算法的局限性在于其运动模型的简化——它假设目标以线性恒速运动,这在实际应用中可能不够准确,尤其是在未考虑摄像机非线性运动的情况下。此外,SORT 算法未能妥善处理目标在视野中消失后再次出现时的重识别(Re-ID)问题。

为了解决这些问题,DeepSORT[29]算法在 SORT 的基础上进行了显著的改进。DeepSORT 通过引入深度学习提取的特征,有效减少了身份切换(ID switch)的发生,从而缓解了重识别问题(见图 3-12)。DeepSORT 算法的创新之处在于其关联度量机制,它不仅考虑了 SORT 中的 IoU(Intersection over Union)距离,还融合了基于深度特征的马氏距离和余弦相似度,以实现更精确的目标匹配。DeepSORT 算法的实现依赖传统的单假设跟踪方法,其中包括递归卡尔曼滤波和逐帧数据关联。其跟踪过程大致分为以下 4 个步骤。

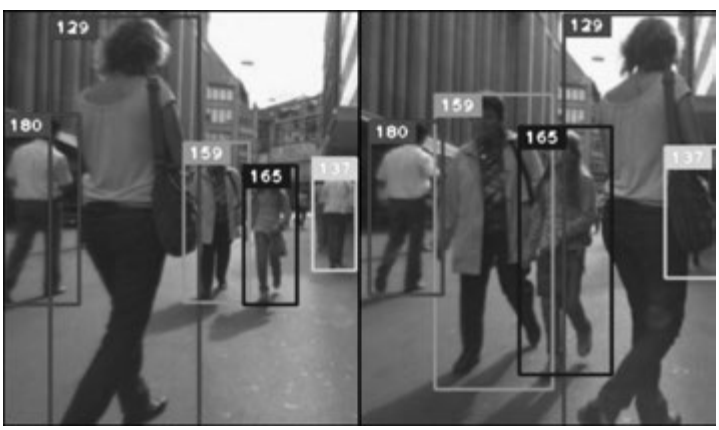


图 3-12　DeepSORT 在具有频繁遮挡的常见跟踪情况下对 MOT 挑战数据集的示例输出

Step1：航迹处理和状态估计

在 DeepSORT 中,确实使用了扩展的状态向量来包含目标的长宽比(γ)及其变化速率(γ'),这是与 SORT 算法中的一个主要区别。SORT 算法通常假设目标的长宽比是恒定的,而 DeepSORT 通过引入长宽比的变化速率,能够更好地适应目标在相机运动或其他因素影响下长宽比可能发生的变化。

在 DeepSORT 中,卡尔曼滤波器的状态向量 $\boldsymbol{X}$ 是一个 8 维向量,包括 u,v(目标边界框中心点的坐标),γ(目标边界框的长宽比),h(目标边界框的高度),u',v'(目标中心点坐标的变

化速率),γ'(长宽比的变化速率),h'(高度的变化速率)。

这种设计使得 DeepSORT 在处理目标外观变化较大的场景,如相机运动或目标旋转时,能够提供更准确的跟踪性能。通过引入长宽比的变化速率,DeepSORT 能够更灵活地适应目标外观的变化,从而提高多目标跟踪的准确性和鲁棒性。

Step2:分配问题

运动特征的关联度量:解决预测的卡尔曼状态和新到达的测量值之间的关联的传统方法是建立分配问题,该问题可以使用匈牙利算法来解决。在这个问题公式中,作者通过结合两个适当的度量来整合运动和外观信息。为了结合运动信息,作者使用预测的卡尔曼状态和新到达的测量值之间的(平方)马氏距离[如式(3-1)所示]:

$$d^{(1)}(i,j)=(d_j-y_i)^{\mathrm{T}}\boldsymbol{S}_i^{-1}(d_j-y_i) \tag{3-1}$$

其中,i 表示卡尔曼预测的状态变量的序号,j 表示检测的序号。$\boldsymbol{S}_i^{-1}$ 表示第 i 个卡尔曼预测的状态变量的协方差矩阵,y_i 表示卡尔曼预测的状态变量的均值。该式本质上就是计算了第 j 个检测和第 i 个预测的状态变量之间的距离。

此外,设置一个马氏距离的阈值,来排除不可能的匹配,定义如式(3-2)所示。

$$b_{i,j}^{(1)}=I\{d_{i,j}^{(1)}\leqslant t^{(1)}\} \tag{3-2}$$

外观特征的关联度量:如果有相机运动等情况,卡尔曼滤波的不确定性会变大,这样仅用运动信息来进行关联度量就不太可靠了。DeepSORT 用一个小型的残差网络来提取目标边界框的外观特征描述符,该网络结构如表 3-1 所示。

表 3-1 DeepSORT 提取外观特征的 CNN 架构

名 称	卷积核大小	输 出 尺 寸
卷积层 1	3×3/1	32×128×64
卷积层 2	3×3/1	32×128×64
最大池化层 3	3×3/2	32×64×32
残差网络层 4	3×3/1	32×64×32
残差网络层 5	3×3/1	32×64×32
残差网络层 6	3×3/2	64×32×16
残差网络层 7	3×3/1	64×32×16
残差网络层 8	3×3/2	128×16×8
残差网络层 9	3×3/1	128×16×8
全连接层 10		128
批处理和归一化层		128

对于每个检测边界框 d_j,将其对应的图像切片输入预训练好的 Re-ID 网络中,输出 128 维的向量,即为外观特征描述符 r_j,满足 $\|r_j\|=1$(范数为 1)。此外,用一个 gallery$R_i=\{r_k^{(i)}\}_{k=1}^{L_k}$ 来存放第 i 个轨迹在最近的 L_k 帧中的所有外观特征描述符(L_k 默认为 100)。公式如式(3-3)所示。

$$d_{i,j}^{(2)}=\min\{1-r_j^{\mathrm{T}}r_k^{(i)}\mid r_k^{(i)}\in R_i\} \tag{3-3}$$

第 j 个检测的外观特征描述符和第 i 个轨迹的 gallery 中所有外观特征描述符之间的最小余弦相似度。

同理,也可以用一个指示函数式(3-4)来排除不可能的匹配。

$$b_{i,j}^{(2)}=I\{d_{i,j}^{(2)}\leqslant t^{(2)}\} \tag{3-4}$$

Step3:结合运动特征和外观特征的关联度量

运动特征和外观特征的关联度量可以在匹配问题中相互补充。一方面,马氏距离对运动

特征的度量，对于短期预测比较有用；另一方面，余弦距离对外观特征的度量，对长时间遮挡后的 id 恢复比较有用。

为了构建关联问题，将两个度量加权求和，公式(3-5)如下。

$$c_{i,j}=\lambda d_{i,j}^{(1)}+(1-\lambda)d_{i,j}^{(2)} \tag{3-5}$$

每个度量对关联成本矩阵的影响可以通过超参数 λ 来控制。当存在大量相机运动时，设置 $\lambda=0$，即仅使用外观信息。

Step4：匹配级联

引入一个级联来解决一系列子问题，而不是在全局分配问题中解决测量到航迹的关联。为了推动这种方法，考虑到当物体被遮挡较长时间时，随后的卡尔曼滤波器预测增加了与物体位置相关的不确定性。因此，概率质量在状态空间中展开，并且观察可能性变得不太突出。因此，作者引入了一个匹配级联，优先考虑更频繁出现的对象，以编码关联似然中的概率分布概念，匹配算法流程如下。

Listing Matching Cascade

Input: Track indices $t=\{1,2,\cdots,N\}$, Detection indices $D=\{1,2,\cdots,N\}$, Maximum age Amax

1: Compute cost matrix $\boldsymbol{C}=[c_{i,j}]$ using(3-5) ＃计算检测和轨迹之间的相似度或代价

2: Compute gate matrix $\boldsymbol{B}=[b_{i,j}]$ using(3-6) ＃确定每个检测和轨迹是否应该被考虑在匹配过程中，＃通常基于它们的相对位置或其他属性

3: Initialize set of matches $M\leftarrow 0$ ＃初始化一个空的匹配集合，用于存储最终的匹配对

4: Initialize set of unmatched detections $u\leftarrow D$ ＃初始化一个包含所有检测的集合，这些检测在开始时＃都被认为是未匹配的

5: for $n\in\{1,2,\cdots,A_{\max}\}$ **do** ＃开始一个循环，循环次数由 $A_{\max}$ 决定

6: **Select tracks by age** $T_n\leftarrow\{\boldsymbol{i}\in\boldsymbol{T}\mid\boldsymbol{a}_i=\boldsymbol{n}\}$ ＃选择 n 的轨迹

7: $[\boldsymbol{x}_i,\boldsymbol{j}]\leftarrow$**min_cost_matching**$(C,T_n,U)$ ＃最小代价匹配算法找到当前轨迹和未匹配检测间最佳＃匹配

8: $M\leftarrow M\cup\{(i,j)\mid b_{i,j}\cdot x_{i,j}>0\}$ ＃更新匹配集合，确定为匹配的检测和轨迹对

9: $U\leftarrow U\backslash\{j\mid\sum_i b_{i,j}\cdot x_{i,j}>0\}$ ＃更新未匹配检测集合，移除那些已经被匹配的检测

10: end for ＃结束循环，继续下一步

11: return M,U ＃返回匹配集合和未匹配检测集合

在多目标跟踪(MOT)的研究领域，Yang 等[30]提出了一个重要的观察：大多数 MOT 方法依赖强线索，如空间和外观信息，来实现目标的区分。这些线索在区分不同实例时发挥着关键作用。然而，当两个或多个目标在当前帧中高度重叠时，基于检测和轨迹位置估计的交集变得模糊不清，因为此时外观特征可能会被前景目标所掩盖。为了解决这一挑战，学者们开始探索结合弱线索，如置信度状态、高度状态和速度方向，来补充强线索的不足。这种方法能够有效地减轻目标间相互遮挡的问题，提高跟踪的准确性。为了更有效地捕捉目标的动态特性，一些学者开始研究长短时记忆(LSTM)网络在 MOT 中的应用。LSTM 网络能够存储和处理时间序列信息，这对于理解目标在视频序列中的长期行为模式非常有用。

3.4.2 基于联合检测跟踪的多目标跟踪算法

联合检测跟踪(JDT)是另一类常见的基于深度学习的 MOT 算法，是将检测阶段和跟踪阶段结合在一起，有的 JDT 算法是将检测和跟踪阶段设计成一个端到端的网络，联合优化；还有的 JDT 算法是将检测阶段的网络和跟踪阶段特征提取部分的网络设计成一个网络，或者

通过特定设计直接用检测部分网络去生成跟踪阶段所需的运动特征，它们的区别是检测阶段和跟踪阶段融合程度的不同。下面将分别介绍基于 Transformer 和基于孪生网络的 JDT 算法。

1. 基于 Transformer 的 JDT 算法

Transformer 是谷歌提出的一种基于注意力机制的深度学习模型，能够有效捕捉序列数据中的长距离依赖关系，最初被应用于自然语言处理领域，后由于其出色的特征表征能力被广泛地应用于计算机视觉领域。Sun 等[31]提出的 TransTrack 首次将 Transformer 应用到 MOT 任务中，设计了两个并行的解码器分别用于检测和跟踪，并利用 Query-Key 机制来跟踪当前帧中已存在的目标并检测新的目标，如图 3-13 所示。

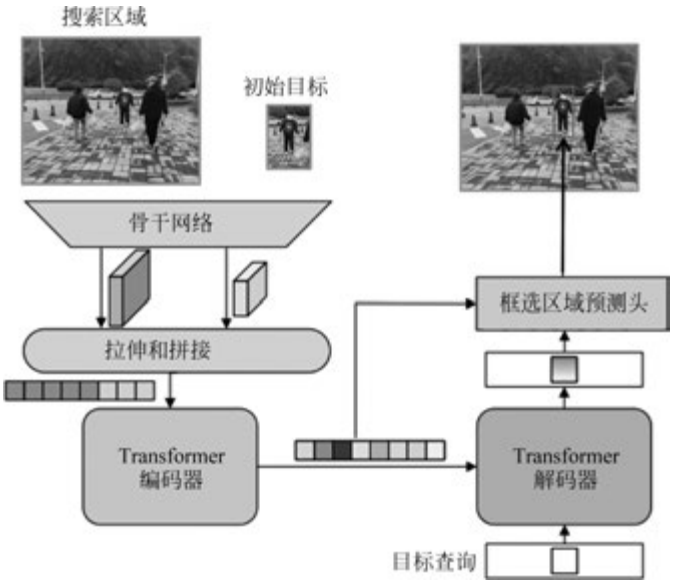


图 3-13 TransTrack 网络框架

Meinhardt 等[32]提出了一种全新的 MOT 框架 TrackFormer，该框架参考 DETR[33]思路，是一个端到端并且以 Tracking-by-attention 为全新思路实现了帧间的数据关联。注意力机制则确保了该模型同时考虑位置、遮挡和目标的识别特征。以上涉及的 MOT 算法大多是基于目标框表示方法，但在极度稠密的场景下，一个目标框可能包含多个人的外观信息，这就导致训练的基于视觉外观表示模型不准确，造成关联歧义。

为了解决此问题，Xu 等[34]提出 TransCenter 方法抛弃了以往从稀疏查询输出稀疏目标框的方式，采用像素级多尺度密集查询预测目标中心点的方式，一定程度上解决了目标框重叠问题。总的来讲，基于 Transformer 的 JDT 算法具有强大的建模能力和全局信息处理能力，但计算复杂度较高且实时性较差。在实际应用中，需要根据具体场景和需求来选择是否采用基于 Transformer 的 JDT 算法。

2. 基于孪生网络的 JDT 算法

孪生网络是基于两个人工神经网络构建的耦合架构，起初常用于单目标跟踪领域。基于孪生网络的跟踪算法是将跟踪问题转换为检测目标与目标模板的相关性匹配问题。

首次将孪生网络应用于单目标跟踪领域的是 SiamFC[21]，图 3-14 为算法中孪生网络的结构示意图，该网络具有两个权值共享的分支：一个分支以目标模板图像作为输入，另一个分支利用目标模型在待搜索区域进行滑动密集搜索，将这些候选区域作为输入，然后计算目标模板

与各个候选区域的相关性,相关性最大的候选区域就是待跟踪目标在当前帧的位置。把单目标跟踪迁移到多目标跟踪,最直接的方法就是将 MOT 中的每一个目标都当作一个独立的 SOT,但是 MOT 场景目标数量太多,同时维持大量的单目标跟踪器会严重影响跟踪的速度,且目标之间相互遮挡,会进一步影响目标外观建模,影响跟踪性能。

为了深入挖掘多目标跟踪(MOT)任务中目标间的外观相似性,Pang 等[35]最近提出了一种创新的网络架构,名为 QDTrack。该架构通过精心挑选视频序列中的相邻帧对,首先利用区域建议网络(Region Proposal Network,RPN)来生成大量的候选区域。这些候选区域为后续的特征提取和目标识别提供了丰富的选择。紧接着,QDTrack 采用一个轻量级的卷积神经网络(CNN),该网络通过权重共享机制高效地提取图像特征,为精准的目标跟踪打下基础。为了进一步提升嵌入特征的区分能力,QDTrack 引入了一种正负样本指定策略,并结合非参数化的 softmax 交叉熵损失函数,通过对抗学习的方式进行网络训练。这样的训练策略极大地增强了特征的辨别力,从而显著提升了跟踪的性能。此外,QDTrack 还创新性地引入了基于双端 softmax 的目标相似度计算方法,这一方法确保了目标匹配过程的一致性和准确性。通过设定合理的相似度阈值,QDTrack 能够高效地将检测到的目标与现有轨迹进行准确关联,实现了精确的多目标跟踪(见图 3-14)。QDTrack 的 Embedding Head 部分代码实现如下,它展示了如何通过深度学习技术来提取和优化目标特征,以实现卓越的跟踪效果。这一部分的代码是 QDTrack 架构中的关键,它直接关系到特征提取的质量和跟踪的准确性。通过这种创新的方法,QDTrack 在多目标跟踪领域展现了巨大的潜力和应用价值。

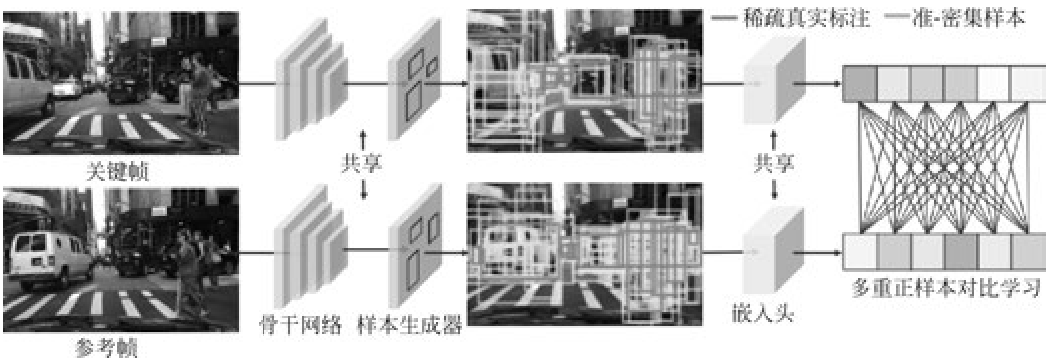


图 3-14 QDTrack 网络架构图

```
#计算预测值和标签之间的多位置交叉熵损失
def multi_pos_cross_entropy(pred,label, weight = None, reduction = 'mean',avg_factor = None):
##########################################
# pred (Tensor): 预测的未归一化概率(logits),形状为 (N, C),其中,N 是样本数量,C 是类别数量
# label (Tensor): 真实的标签,形状为 (N,),其中,1 表示正样本,0 表示负样本
# weight (Tensor, optional): 每个样本的权重,默认为 None
# reduction (str, optional): 指定如何降低损失的维度,选项有 'none', 'mean', 'sum',默认为 'mean'
# avg_factor (int, optional): 用于除以损失的平均因子,默认为 None
##########################################
    #通过标签确定正负样本的索引
pos_inds = (label == 1)
    neg_inds = (label == 0)
    #根据正负样本索引选择预测值
    pred_pos = pred * pos_inds.float()
```

```
        pred_neg = pred * neg_inds.float()
        #使用负无穷大来掩盖不需要的元素,以提高数值稳定性
        pred_pos[neg_inds] = pred_pos[neg_inds] + float('inf')
        pred_neg[pos_inds] = pred_neg[pos_inds] + float('-inf')
    #扩展正样本预测值以匹配负样本预测值的形状
        _pos_expand = torch.repeat_interleave(pred_pos, pred.shape[1], dim=1)
        _neg_expand = pred_neg.repeat(1, pred.shape[1])
        #对负样本预测值进行填充,以便在计算 logsumexp 时包括所有可能的正负样本对
    x = torch.nn.functional.pad((_neg_expand - _pos_expand), (0, 1), "constant", 0)
        loss = torch.logsumexp(x, dim=1)
        #应用权重并执行降低维度的操作
        if weight is not None:
            weight = weight.float()
        loss = weight_reduce_loss(loss, weight=weight, reduction=reduction, avg_factor=avg_factor)
        return loss
    @LOSSES.register_module()
    class MultiPosCrossEntropyLoss(nn.Module): #用于计算多位置交叉熵损失
        def __init__(self, reduction='mean', loss_weight=1.0):
            super(MultiPosCrossEntropyLoss, self).__init__()
            self.reduction = reduction
            self.loss_weight = loss_weight
        def forward(self, cls_score, label, weight=None, avg_factor=None, reduction_override=
    None, **kwargs):
            assert cls_score.size() == label.size() #确保预测值和标签的形状相同
            assert reduction_override in (None, 'none', 'mean', 'sum')  #确保 reduction_override 是有效的
            reduction = (reduction_override if reduction_override else self.reduction)
            loss_cls = self.loss_weight * multi_pos_cross_entropy( #计算损失并乘以损失权重
                cls_score, label, weight, reduction=reduction, avg_factor=avg_factor, **kwargs)
            return loss_cls
```

3.5 目标跟踪数据集及评价指标

在目标跟踪领域,数据集和评价指标对于评估算法的性能至关重要。以下是对一些广泛认可的单目标跟踪(SOT)数据集(见表 3-2)和多目标跟踪(MOT)数据集(见表 3-3),以及评价指标的概述。

表 3-2 常见单目标跟踪数据集及属性

数据集	序列数	平均帧长	目标种类	目标属性	总帧数
OTB2013[36]	51	578	10	11	30k
OTB2015[37]	100	588	16	11	59k
UAV123[38]	123	915	9	12	112k
TrackingNet[39]	30 643	451	21	15	14M
LaSOT[40]	1400	2506	70	14	3.5M

表 3-3 MOT 典型数据集概况

数据集	年份	视频序列	大小	特点
KITTI Tracking[41]	2012	50	15GB	允许同时追踪人和车辆
MOT15	2015	22	1.3GB	场景丰富,密集度低
MOT16	2016	14	1.9GB	场景丰富,规范标注,目标密集程度高
MOT17	2016	14	5.5GB	增加更多场景,提供了更多的公开检测器
MOT20	2020	8	5.0GB	人群更加密集,相互遮挡情况更严重
DanceTrack[42]	2022	100	16.5GB	目标外观相似且存在大量遮挡,运动模式复杂

3.5.1 常见单目标跟踪数据集介绍

1. OTB 单目标跟踪数据集

OTB 数据集有两个版本：OTB-2013[36]和 OTB2015[37]。后面的数字指的是数据集提出的年份，又因为 OTB2015 数据集包含 100 个视频序列，所以又叫作 OTB100。作者又将 100 个视频序列分成 11 个挑战，包括光照变化、尺度变化、遮挡、目标形变、运动模糊、快速运动、平面内旋转、平面外旋转、离开视野、背景干扰和低分辨率，便于评估跟踪器应对不同挑战的能力。

2. NFS 单目标跟踪数据集

该数据集包含 100 个视频[38]，包含 17 个类别。数据集提供了两个版本：以每秒 30 帧采样视频(NFS30)和以每秒 240 帧采样视频(NFS240)。高帧率视频可很好地反映目标分配运动属性，对跟踪性能有一定的提升。

3. UAV123 单目标跟踪数据集

该数据集包含 123 个视频[38]，不同于以上三个数据集，该视频全部采集于无人机，因此视角是高空俯拍，目标较小，且目标形变大。该数据集上不同跟踪器性能差别开始扩大，是验证跟踪器是否足够强大的重要基准。

4. TrackingNet 单目标跟踪数据集

该数据集[39]是一个大规模数据集，总共收集了 3 万个标注过的视频数据，共有 27 个目标类别。事实上，基于神经网络的目标跟踪器非常依赖大规模的数据集，该数据集已经成为基于深度学习跟踪器的必备训练集之一，对跟踪方法的发展起重要作用。

5. LaSOT 单目标跟踪数据集

LaSOT[40]是近年来 Fan 等提出的大规模单目标跟踪数据集，共 1400 个长视频序列，其中 280 个用于测试。视频种类共 70 个，平均视频长度大于 2000 帧，非常适合考验跟踪器的长期跟踪能力，是验证跟踪器性能的必备数据集之一。

3.5.2 常见多目标跟踪数据集介绍

一个典型的 MOT 数据集是由多个视频帧序列组成的。在这些序列中，对于每个视频帧数据集通常会提供目标的位置信息，为每个目标分配唯一的标识符，以便跟踪同一目标在不同帧之间的运动轨迹。

1. KITTI 数据集多目标跟踪数据集

KITTI 数据集[41]包含用于自动驾驶和目标跟踪的大量现实世界驾驶场景的图像序列，可以提供包含车辆、行人和自行车等目标的标注边界框及关联轨迹。KITTI 数据集以其真实性和多样性而闻名，被广泛应用于评估 MOT 在复杂真实场景下的表现。

2. MOTChallenge 系列多目标跟踪数据集

MOTChallenge 系列数据集是常用 MOT 数据集，目前包括 4 个行人多目标跟踪数据集，即 MOT15、MOT16、MOT17 和 MOT20。这些数据集目标密集程度不断提高，涵盖了不同条件下的跟踪场景，有摄像机固定的跟踪场景，也有摄像机放在移动车辆的场景，此外还包含俯视、平视和仰视不同的拍摄视角；同时，这些视频是在不同的光照条件下拍摄的，包括晴天、阴天、夜晚等。

3. DanceTrack 多目标跟踪数据集

DanceTrack 数据集[42]搜集了 100 段视频，内容包括集体舞蹈、功夫、体操等。视频具

有如下特点：①目标人物穿着相似甚至一致；②目标之间有大量的遮挡和位置交错；③目标的运动模式非常复杂多样，呈现明显的非线性，并且时常伴随多样的肢体动作。以上用于评估的 MOT 数据集提供了丰富的视频序列，包括行人、车辆等多种类型目标，并提供准确的标注信息如目标边界框和关联轨迹，常用于评估多目标跟踪算法的准确性、鲁棒性和实时性。

3.5.3 常见目标跟踪评价指标介绍

1. 单目标跟踪常用评价指标

(1) 单次评估。

在单次评估(One-Pass Evaluation，OPE)方法中，跟踪器仅在初始帧进行初始化，随后在后续帧中预测目标的位置。评估过程中不涉及对跟踪过程的中间结果进行监督。常用的性能指标包括精确率(Precision)和成功率(Success Rate)。

(2) 精确率。

精确率(Precision)通过距离精确率(Distance Precision，DP)来衡量，这通常基于预测目标中心位置与真实目标中心位置之间的欧几里得距离差(Center Location Error，CLE)。设定一个阈值范围(通常为 0～50px)，并统计不同阈值下的 DP，绘制精确率曲线(Precision Plots)。在精确率曲线上，当阈值设定为 20px 时，对应的 DP 值被特别关注。

(3) 规范化精确率。

为减少图像分辨率和目标大小对精确率指标的影响，Müller 等提出了规范化精确率(Normalized Precision)。规范化中心位置误差(Normalized Center Location Error)定义如式(3-6)所示。

$$\text{Normalized CLE}=\frac{\text{CLE}}{\max(W,H)} \tag{3-6}$$

其中，W 和 H 分别表示真实目标框的宽度和高度。

(4) 重合率。

重合率(Overlap Rate)也称为交并比(Intersection over Union，IoU)，定义如式(3-7)所示。

$$\text{IoU}=\frac{\text{Area}(B_E \cap B_G)}{\text{Area}(B_E \cup B_G)} \tag{3-7}$$

其中，B_E 和 B_G 分别代表预测的矩形框和真实的矩形框。通过设定不同的 IoU 阈值，可以得到不同的成功率。设置置信度范围为 0～1，可以绘制相应的成功率曲线(Success Plots)。成功率曲线下的面积(Area Under Curve，AUC)是评估跟踪器性能的一个重要指标，它能准确反映跟踪器在目标定位和边界框预估方面的质量。

2. 多目标跟踪常用评价指标

(1) 多目标跟踪准确度。

最能体现 MOT 算法综合性能的指标是多目标跟踪准确度(Multiple Object Tracking Accuracy，MOTA)，它主要评估的是检测能力，如式(3-8)所示。

$$\text{MOTA}=1-\frac{\text{FN}+\text{FP}+\text{IDSW}}{\text{GT}} \in (-\infty,1] \tag{3-8}$$

其中，FN 为未被正确检测的真实目标数目，即漏检数；FP 为错误的检测数目，即虚警数；IDSW 为跟踪过程中目标身份标识切换次数；GT 是真实框个数。由于算法中身份标识切换次数可能会比真实框还多，因此 MOTA 取值可能是负数，具体代码实现如下。

```
#计算多目标跟踪精度(MOTA)
def multiple_object_tracking_accuracy(pred_boxes, gt_boxes, iou_threshold=0.5):
    #pred_boxes 是一个列表,长度为 t,表示 t 个时间步的预测的 3D 边界框
    #gt_boxes 是一个列表,长度为 t,表示 t 个时间步的真实的 3D 边界框
    #iou_threshold 是一个浮点数,表示重叠比例的阈值,默认为 0.5
    #返回多目标跟踪精度(MOTA)值
    #如果没有预测或真实边界框,返回 0
    if len(pred_boxes) == 0 or len(gt_boxes) == 0:
        return 0.0
    miss_count = 0                          #初始化漏检数、误检数、ID 切换数和总真实数为 0
    false_count = 0
    switch_count = 0
    total_count = 0
    prev_matches = {}                       #初始化上一时间步的匹配结果为空字典
   for t in range(len(pred_boxes)):         #遍历每个时间步
        #获取当前时间步的预测和真实边界框
        pred_box = pred_boxes[t]
        gt_box = gt_boxes[t]
        #计算当前时间步的真实边界框的数量,并累加到总真实数中
        total_count += gt_box.shape[0]
        #如果当前时间步没有预测或真实边界框,跳过该时间步
        if pred_box.shape[0] == 0 or gt_box.shape[0] == 0:
            continue
        #计算当前时间步的预测和真实边界框之间的重叠比例矩阵,大小为 p×g
        iou_matrix = np.zeros((pred_box.shape[0], gt_box.shape[0]))
        for i in range(pred_box.shape[0]):
            for j in range(gt_box.shape[0]):
                iou_matrix[i][j] = iou_3d(pred_box[i], gt_box[j])
        #使用匈牙利算法进行数据关联,得到匹配结果
        matches = data_association(-iou_matrix)
        curr_matches = {}                   #初始化当前时间步的匹配结果为空字典
        for i in range(pred_box.shape[0]): #遍历每个预测边界框
         #如果没有匹配的真实边界框,或者重叠比例低于阈值,累加误检数,并跳过该预测边界框
            if matches[i] == -1 or iou_matrix[i][matches[i]] < iou_threshold:
                false_count += 1
                continue
            j = matches[i]                  #获取匹配的真实边界框的索引
            #如果上一时间步有匹配的真实边界框,并且 ID 不同,累加 ID 切换数
            if j in prev_matches and prev_matches[j] != i:
                switch_count += 1
            curr_matches[j] = i             #将当前的匹配结果保存到字典中
        #遍历每个真实边界框
        for j in range(gt_box.shape[0]):
            #如果没有匹配的预测边界框,累加漏检数
            if j not in curr_matches:
                miss_count += 1
        #更新上一时间步的匹配结果为当前的匹配结果
        prev_matches = curr_matches
    #计算并返回多目标跟踪精度(MOTA)值
    mota = 1 - (miss_count + false_count + switch_count) / total_count
    return mota
```

（2）多目标跟踪精确度（Multiple Object Tracking Precision，MOTP）。

多目标跟踪精确度可以衡量跟踪算法在正确跟踪的情况下目标位置估计的精确度，即跟踪算法在空间上的定位精确度。MOTP 计算公式如式(3-9)所示。

$$\mathrm{MOTP}=\frac{\sum_{t,i} d_{t,i}}{\sum_{t} C_t} \tag{3-9}$$

其中，C_t 表示第 t 帧中目标和轨迹匹配数目；$d_{t,i}$ 表示第 t 帧中第 i 对检测框中心点和真实框中心点之间的距离，具体代码实现如下。

```
#计算多目标跟踪精确度(MOTP)
def multiple_object_tracking_precision(pred_boxes, gt_boxes, iou_threshold = 0.5):
    #pred_boxes 是一个列表，长度为 t，表示 t 个时间步的预测的 3D 边界框
    #gt_boxes 是一个列表，长度为 t，表示 t 个时间步的真实的 3D 边界框
    #iou_threshold 是一个浮点数，表示重叠比例的阈值，默认为 0.5
    if len(pred_boxes) == 0 or len(gt_boxes) == 0:
        return 0.0
    match_count = 0                          #初始化总匹配数和总重叠率为 0
    sum_iou = 0.0
    for t in range(len(pred_boxes)):         #遍历每个时间步，获取当前时间步的预测和真实边界框
        pred_box = pred_boxes[t]
        gt_box = gt_boxes[t]
        #如果当前时间步没有预测或真实边界框，跳过该时间步
        if pred_box.shape[0] == 0 or gt_box.shape[0] == 0:
            continue
        #计算当前时间步的预测和真实边界框之间的重叠比例矩阵，大小为 p×g
        iou_matrix = np.zeros((pred_box.shape[0], gt_box.shape[0]))
        for i in range(pred_box.shape[0]):
            for j in range(gt_box.shape[0]):
                iou_matrix[i][j] = iou_3d(pred_box[i], gt_box[j])
        matches = data_association(-iou_matrix) #使用匈牙利算法进行数据关联，得到匹配结果
        for i in range(pred_box.shape[0]): #遍历每个预测边界框
            #如果没有匹配的真实边界框，或者重叠比例低于阈值，跳过该预测边界框
            if matches[i] == -1 or iou_matrix[i][matches[i]] < iou_threshold:
                continue
            j = matches[i]                   #获取匹配的真实边界框的索引
            match_count += 1                 #累加匹配数和重叠率
            sum_iou += iou_matrix[i][j]
       motp = sum_iou / match_count          #计算并返回多目标跟踪精确度(MOTP)值
    return motp
```

（3）IDF1。

IDF1 表示在 ID 保持不变情况下跟踪准确率和召回率的 F_1 值，相比于考虑 IDSW 的 MOTA，IDF1 指标更注重跟踪能力。IDTP、IDFP 和 IDFN 分别为目标 ID 的正确匹配数、虚警数和漏检数。IDF1 计算公式如式(3-10)所示。

$$\mathrm{IDF1}=\frac{2\mathrm{IDTP}}{2\mathrm{IDTP}+\mathrm{IDFP}+\mathrm{IDFN}} \tag{3-10}$$

（4）HOTA 系列指标。

MOTA 和 IDF1 分别强调检测与关联的性能，因此 Luiter 等[43]提出了 HOTA 系列指标，该指标可以更好地兼顾考虑检测和关联的性能。DetA 是检测的准确率，计算公式如式(3-11)所示。

$$\mathrm{DetA}=\int_0^1 \mathrm{DetA}_\alpha \,\mathrm{d}\alpha,\quad \mathrm{DetA}_\alpha=\frac{|\ \mathrm{TP}\ |}{|\ \mathrm{TP}\ |+|\ \mathrm{FN}\ |+|\ \mathrm{FP}\ |} \tag{3-11}$$

其中，TP 为只考虑视频序列中目标的检测被正确检测的目标数目，即正样本数；FN 和 FP 和上述 MOTA 公式中的含义相同。α 为真实框和检测框是否匹配的阈值，最终的 DetA 是在不同的阈值下计算 DetA_α 然后求平均得到的。AssA 是关联的准确率，计算公式如式(3-12)和式(3-13)所示。

$$A(c)=\frac{|\ \mathrm{TPA}(c)\ |}{|\ \mathrm{TPA}(c)+|\ \mathrm{FNA}(c)\ |+|\ \mathrm{FPA}(c)\ |} \tag{3-12}$$

$$\mathrm{AssA}=\int_0^1 \mathrm{AssA}_\alpha \,\mathrm{d}\alpha,\quad \mathrm{AssA}_\alpha=\frac{1}{|\ \mathrm{TP}\ |}\sum_{c\in\{\mathrm{TP}\}} A(c) \tag{3-13}$$

其中，在多目标跟踪(MOT)的评估中，ID 一致性(AssA)是一个关键指标，它衡量了跟踪算法在维持目标身份标识一致性方面的表现。ID 一致性指标的计算考虑了多个因素，包括真正例(TP)、假正例(FP)和假负例(FN)，以及它们与预测的 ID 编号之间的关系。具体来说：

(1) c 表示预测的 ID 编号，它是算法为检测到的目标分配的标识符。

(2) α 是用于确定真实边界框和检测边界框是否匹配的阈值。

(3) TP 表示所有被正确检测为目标的正样本数量。

(4) TPA(c)表示在所有真正例中，真实框和检测框 ID 具有相同 c 的数量。

(5) FNA(c)表示在真正例中，真实框 ID 为 c 但预测框 ID 不为 c 的数量，以及在假负例中，真实框 ID 具有 c 的数量。

(6) FPA(c)表示在真正例中，检测框 ID 有 c 但真实框 ID 不为 c 的数量，以及在假正例中，检测框 ID 具有 c 的数量。

HOTA 是检测准确性和关联准确性的几何平均值(Geometric Mean，GM)，它综合了这两个方面的性能。HOTA 的计算公式如式(3-14)所示。

$$\mathrm{HOTA}=\int_0^1 \mathrm{HOTA}_\alpha \,\mathrm{d}\alpha,\quad \mathrm{HOTA}_\alpha=\sqrt{\mathrm{DetA}_\alpha\times\mathrm{AssA}_\alpha} \tag{3-14}$$

3.6 目标跟踪的应用

1. 视频监控与安全

在视频监控领域，目标跟踪技术使得实时监控和分析个体或多个目标的行为成为可能，这对于安全防护和交通管理等领域尤为重要，如图 3-15 所示。例如，在安防监控中，目标跟踪可以用于识别和跟踪可疑行为，如未经授权的入侵或暴力行为，从而及时响应潜在的安全威胁。在交通监控中，通过跟踪车辆和行人，可以优化交通流量，减少事故发生。

图 3-15 目标跟踪在视频监控中的应用场景[43]

2. 体育分析

在体育分析领域，目标跟踪技术可以用于实时捕捉和分析运动员的动作和表现，如图 3-16 所示。这为教练和训练师提供了宝贵的数据，帮助他们优化训练计划，提高运动员的表现，并

减少受伤风险。在体育赛事中，目标跟踪还用于自动生成精彩回放和统计数据，增强观众的观看体验。

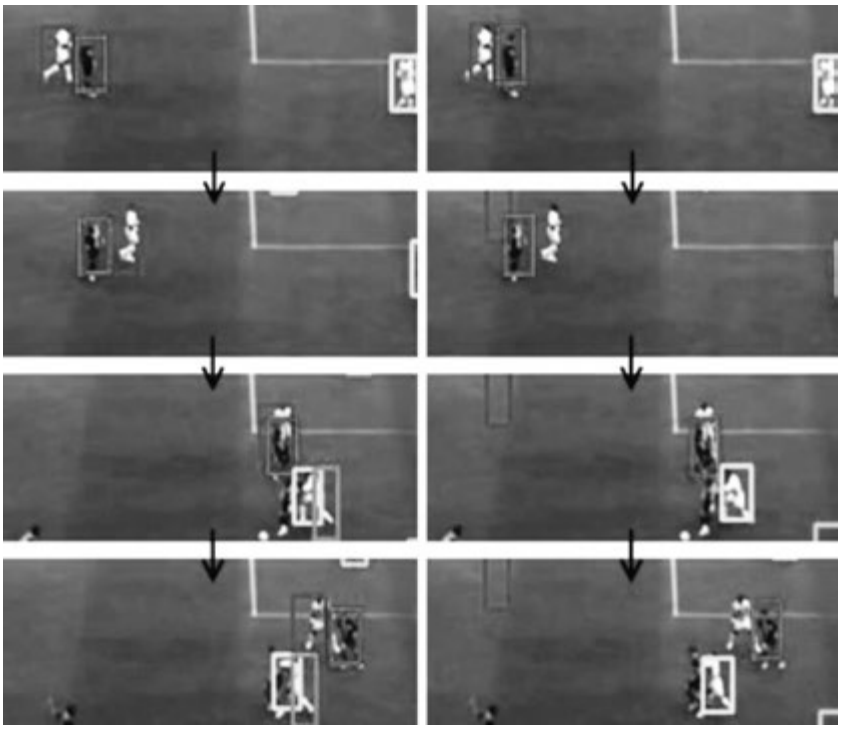


图 3-16 目标跟踪在体育领域的应用场景[44]

3. 自动驾驶

在自动驾驶技术中，目标跟踪是实现车辆对周围环境感知的关键，如图 3-17 所示。通过跟踪其他车辆、行人、自行车和其他障碍物，自动驾驶系统可以做出及时的决策，避免碰撞，确保行车安全。

图 3-17 目标跟踪在自动驾驶领域的应用场景[45]

除上述应用外，目标跟踪在无人机监控、工业自动化、医疗成像分析、零售分析等领域均有广泛应用。随着技术的不断进步，目标跟踪技术有望在更多领域展现其价值。

3.7 目标跟踪的挑战

1. 复杂场景下的目标跟踪

在目标跟踪领域，复杂场景如光照变化、目标遮挡、背景干扰等对跟踪算法的准确性提出了挑战。尽管现有的跟踪算法在简单场景下表现良好，但在复杂环境下，如夜间监控或拥挤的

公共场所，跟踪性能往往会显著下降。为了提高在这些场景下的跟踪效果，学者们正在探索利用深度学习模型来增强算法对复杂环境的适应能力。

2. 实时性与计算资源的平衡

目标跟踪在实时应用中的需求日益增长，如视频监控、自动驾驶等。然而，实时跟踪需要算法在极短的时间内完成目标的检测和跟踪，这对计算资源提出了较高的要求。学者们正致力于开发更高效的算法，通过优化网络结构、采用轻量级模型等方法，以实现在有限计算资源下的实时跟踪。

3. 多目标跟踪中的关联问题

在多目标跟踪(MOT)中，目标之间的关联是一个关键问题。在拥挤场景或目标快速移动的情况下，正确地将目标在不同帧之间进行匹配是一项挑战。为了解决这一问题，学者们正在研究基于图模型、概率模型和深度学习的方法，以提高目标关联的准确性。

4. 跨场景和跨摄像头的跟踪

目标跟踪在跨场景和跨摄像头的应用中面临挑战，因为场景变化和视角差异可能导致目标外观的显著变化。为了实现跨场景和跨摄像头的跟踪，学者们正在探索使用迁移学习和领域适应技术，以提高算法的泛化能力。

5. 小目标和快速运动目标的跟踪

在视频监控中，小目标和快速运动目标的跟踪是一个难题。小目标由于分辨率低，特征不明显，而快速运动目标则因为运动模糊和快速的位置变化，给跟踪算法带来了挑战。为了解决这些问题，学者们正在开发新的算法，如利用高帧率视频、采用多尺度分析和运动模型来提高跟踪性能。

6. 目标跟踪数据集的多样性和质量

目标跟踪算法的性能很大程度上依赖训练数据集的质量和多样性。目前，虽然已经有一些公开的数据集，但它们在场景、目标类型和摄像机设置等方面的多样性仍有待提高。为了推动目标跟踪技术的发展，学者们需要构建更加丰富和多样化的数据集，以覆盖更多的实际应用场景。

7. 目标跟踪的可解释性和鲁棒性

随着目标跟踪技术在安全监控等领域的广泛应用，算法的可解释性和鲁棒性变得越来越重要。学者们正在努力提高算法的透明度，使其能够提供关于跟踪决策的更多信息，并且能够在面对恶意攻击或数据污染时保持稳定。这涉及算法的鲁棒性测试和可解释性分析。

小结

未来的发展趋势可能包括开发更加鲁棒和准确的目标跟踪算法，尤其是在处理复杂场景和目标变化时。此外，轻量化架构、端到端学习、基于注意力机制的方法也是当前研究的热点。随着深度学习技术的不断进步，未来的跟踪算法有望在效率和准确性上都取得更大突破。

习题

1. 描述目标跟踪技术在哪些实际应用中发挥着重要作用，并解释它在这些应用中的作用。

2. 讨论目标跟踪算法中的关键技术，并解释每个技术在目标跟踪过程中的作用。

3. 比较卡尔曼滤波器、粒子滤波器、滑动窗口方法和光流法在目标跟踪中的优缺点，并讨论它们各自适用的场景。

4. 描述目标跟踪中特征提取方法的发展历程，从手工设计特征到深度特征，并讨论它们各自的优势和局限性。

5. 解释目标跟踪中的生成式和判别式观测模型，并比较它们在实际应用中的性能和适用性。

参考文献