清 华 大 学

综 合 论 文 训 练

题目：基于深度学习的多物体实时跟踪方法研究与实现

系 别：计算机科学与技术系

专 业：计算机科学与技术专业

姓 名：何蔚然

指导教师：姚海龙

2016年 06月 07日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签 名： 导师签名： 日 期：

中文摘要

现今物体检测、分割、识别仍是计算机视觉三大难题。物体追踪不论在学术上还是商业上都处于蓬勃发展的阶段，应用有行人追踪、人脸追踪等。而物体追踪仍有很大的进步空间，现今绝大部分算法都不能做到实时在线追踪，深度学习与传统方法的配合仍大有所为。本文基于传统以及深度学习提出了几种现代物体追踪算法流程，并成功移植到名为TK1的显卡开发板上。

本文将物体追踪问题分为两类，分别是单物体和多物体追踪。本文首先分别提出了独立的两套算法流程，并在最后通过深度学习算法，将两类算法进行合并，这样的思路在知名数据集MOT15上是首创的。

关键词：物体分割；多物体追踪； 在线；实时；粒子滤波；深度学习

ABSTRACT

At present, Object Detection, Segmentation and Recognition still remain the three main problems in Computer Vision. Whether academically or commercially, Object Tracking is in vigorous development with application of Pedestrian Tracking, Face Tracking, etc. There is still great room for improvement of Object Tracking, as most of the public pipelines are not for real-time and online situations. There is still much we can do by combining deep learning and traditional tracking methods. This paper proposes several modern object tracking pipelines based on deep learning and traditional methods. These methods have been successfully transplanted to an embedded development board called TK1.

This paper treats Object Tracking as two parts, Single and Multiple Object Tracking respectively. This paper proposes two independent tracking pipelines for these two problems, then combines the two pipelines using a deep learning algorithm, which is the first of its kind to dataset MOT15 worldwide.

Keywords: Object Segmentation; Multiple Object Tracking; Online; Real-time; Particle Filtering; Deep Learning

目 录

第1章 引言 1

1.1 选题背景与意义 1

1.2 研究内容 2

1.3 主要贡献 3

1.4 论文结构 3

第2章 单物体追踪算法 5

2.1 概述 5

2.2 数据来源 5

2.3 粒子滤波 6

2.3.1 蒙特卡洛粒子滤波 6

2.3.2 直方图均衡 7

2.3.3 整体流程 7

2.4 扩展MOT 9

2.4.1 修改输出过程 9

2.4.2 非极大值抑制 9

2.4.3 整体流程 10

2.5 小结 11

第3章 多物体追踪算法 13

3.1 概述 13

3.2 数据来源 13

3.2.1 2D MOT 2015数据集 13

3.2.2 Microsoft-COCO数据集 13

3.2.3 Poselets数据集 14

3.3 主流框架 14

3.3.1 帧间追踪 15

3.3.2 卡尔曼滤波 16

3.3.3 单帧识别-RPN 17

3.3.4 单帧识别-HED 18

3.3.5 单帧识别-稠密盒 21

3.3.6 离线算法 23

3.4 新型追踪算法 24

3.4.1 深度回归网络 24

3.4.2 飞天椅子数据集 25

3.4.3 训练过程 26

3.4.4 在线实时算法流程 27

3.5 小结 28

第4章 实验结果与分析 29

4.1 实验设计 29

4.2 实验结果 29

4.3 结果分析 31

第5章 结论 32

5.1 工作总结 32

5.2 进一步工作 32

插图索引 33

表格索引 35

参考文献 37

致 谢 39

声 明 41

附录A 书面翻译 43

主要符号对照表

如果论文中使用了大量的物理量符号、标志、缩略词、专门计量单位、自定义名词和术语等，应编写成注释说明汇集表。若上述符号和缩略词使用数量不多，可以不设专门的汇集表，但必须在论文中出现时加以说明。

SOT 单物体追踪(Single Object Tracking)

MOT 多物体追踪(Multiple Objects Tracking)

CNN 卷积神经网络

FCN 全卷积神经网络

VGG 视觉几何组（Visual Geometry Group）

IOU 交除并（Intersection Over Union）

KM Khun-Munkres算法

NMS 非极大值抑制（Non-Maximum Suppression）

DenseBox 稠密盒

# 引言

## 选题背景与意义

一直以来，物体检测、物体分割和物体识别都是计算机视觉领域里经典、基础且重要的问题。在过去的数十年里，尽管有大量的针对性算法被提出，然而它们中的大多数都只能在某些特定场景里正常工作。直到最近卷积神经网络在图像处理上的崛起，物体检测、物体分割和物体识别的工作进展有了质的提高。在引入深度学习以后，算法对环境变化的敏感度在逐步下降，具体任务的表现也正稳步上扬，领域呈现蓬勃发展的态势。

物体追踪是基于物体检测、物体分割和物体识别的问题，不论在学术上还是商业上，都有着很高的研究价值。目前较为热门的物体追踪应用有行人追踪、车辆追踪以及普适物体追踪等。针对这些应用，世界上也出现了很多被广泛使用的公开数据集，并且直到现在仍每年都有新的数据集发布或进行更新。在行人追踪与车辆追踪上，有不少公司在商业上进行研究与开发，更有大量的实验室在这些方面投入了大量的精力。

物体追踪主要分为两大类任务，分别是单物体追踪（SOT）以及多物体追踪（MOT）。一直以来，两大问题在解决方法上，都有着比较少的联系。更多的时候，单物体追踪的相关方式方法与多物体追踪的思路完全不一样，并且互相的借鉴空间也比较小，这也是比较违反直觉的一个方面。

在单物体追踪任务上，一般是先对给定视频某一帧中感兴趣的物体作标记（通常为包围盒），算法在随后的每一帧中对对应物体进行定位。一个优秀的单物体追踪算法流程应当具备优秀的抗环境干扰能力，比如物体的快速移动、观察视角的变化、光照的突然变化以及其他的环境变化情况。在很多系统中，单物体追踪都是其中重要的一部分。比如，足球机器人必须时刻追踪足球的实时位置以及运动轨迹，才能作出正确的反应；车辆智能驾驶时，必须对障碍物进行追踪，通过其在不同帧中的具体位置以及视角的变化，估计障碍物的位置以及将来的运动轨迹，以避免发生碰撞引起交通事故。

多物体追踪可以被视为一个数据互联的过程，需要在视频不同的帧之间把检测结果互联起来。一般多物体追踪算法都遵循“检测-追踪”的思路框架，也就是先在每一帧中检测出所有需要追踪的物体，再在帧与帧之间对检测结果进行追踪。这其中，检测的质量将是重中之重，直接决定追踪算法的现实效果。 同时，由于检测算法往往运行时间比较长，整体追踪流程需要在追踪准确率与速度之间找到平衡点。尽管如此，目前大多数公开追踪算法流程都不能支持实时运行，甚至有很多算法都不能做到在线运行（不利用未来帧信息）。

## 研究内容

本文主要针对物体追踪的两大类——单物体追踪与多物体追踪——提出了对应的算法流程。

定义 1‑1**包围盒：**一个四元组表示一个矩形，其中表示矩形的左上角坐标，分别表示矩形的宽度与高度。一个包围盒的例子如 图 1‑1所示，其中红色矩形即是对应包围盒。



图 1‑1 红色矩形的行人包围盒

问题 1**单物体追踪：**给定包含物体的视频片段（按时间发生顺序），并在第一帧中给出所要追踪物体的包围盒，对于随后的每一帧，算法输出物体在对应帧中的包围盒。

问题 2**多物体追踪：**预先定义所要追踪的物体类型集合（如行人、车辆），给定视频片段（按时间发生顺序），算法对于随后的每一帧，输出所有需要追踪物体在对应帧中的包围盒。

## 主要贡献

单物体追踪与多物体追踪算法之间本来比较独立，互相合作的可能性比较小。在第4章中，笔者介绍了新型追踪算法——深度回归网络。这原本是在单物体追踪任务上曾取得世界最优结果的方法。笔者通过对算法流程的修改，使得深度回归网络能够在多物体追踪任务上与RPN、稠密盒等物体识别算法相派和，完成多物体追踪在线实时算法流程。在主流多物体追踪数据集MOT15的排行榜上，以及在笔者的认知范围内，这是首创的做法。

## 论文结构

本文的章节安排如下：

**第2章** 介绍我们的单物体追踪算法思路

**第3章** 介绍我们的多个多物体追踪算法流程

**第4章** 对实验结果进行阐述

**第5章** 总结本文的算法表现及研究过程的不足，提出改进思路

# 

# 单物体追踪算法

## 概述

本章主要介绍了如何解决单物体追踪任务，在公开数据集上，应用传统的粒子过滤算法进行实验。由于本文并非关注单物体追踪任务，以下算法更多是试验性质，并未引入学界最新算法，浅尝辄止。

本章首先介绍了公开的单物体追踪数据集ALOV[2]，如何在其上进行数据处理；然后介绍了基于蒙特卡洛粒子滤波的单物体追踪算法流程；接着提出了上述算法流程的扩展算法，用于尝试解决多物体追踪任务；最后总结本章工作与不足之处，提出改进思路。

## 数据来源

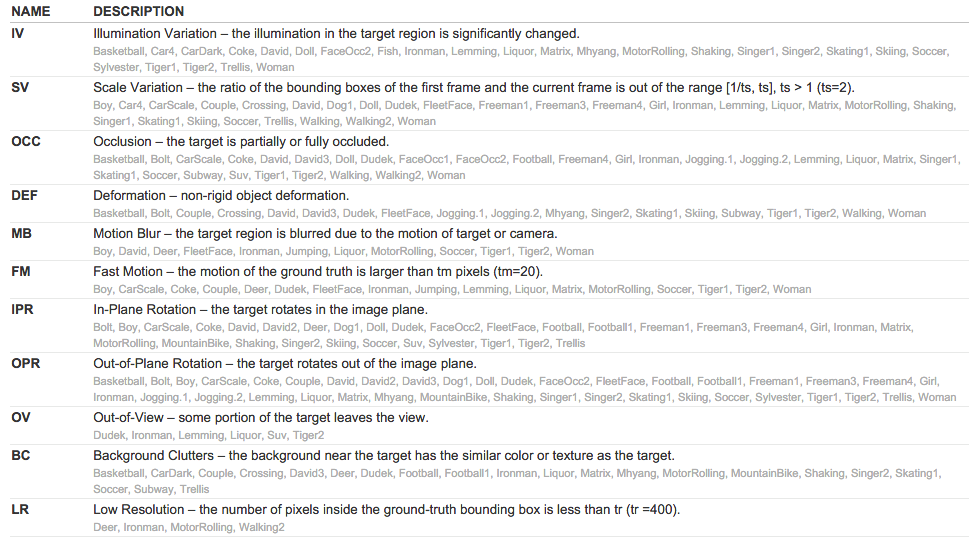


表 2‑1 视频环境变化列表[2]

本章使用的是ALOV[2]数据集，该数据集包含314段视频片段，每个片段都包含不少于50帧图像。视频片段分为14个类别，涵盖了如表 2‑1所示的11种环境变化情况，包含尺度变化、光照改变等等。这是公开数据集中环境变化最全最多的，能够更全面地评价追踪器在不同情况下的鲁棒性。

本文并不专注于单物体追踪，为了降低问题难度，本章假设所有物体在帧与帧之间长宽无变化。

## 粒子滤波

粒子滤波算法是传统物体追踪方法之一，即便在现代图像处理被卷积神经网络统治的情况下，大量单物体追踪算法仍然离不开粒子滤波的框架。

本章使用的单物体粒子滤波框架使用的是蒙特卡洛粒子滤波与直方图均衡作图像相似度计算的方法。

### 蒙特卡洛粒子滤波

定义 2‑1 **蒙特卡洛粒子：**二元组，其中为包围盒，粒子的坐标与包围盒中心一致，为粒子的权重，取值为[0,1]的实数，权重越大表示该粒子对应包围盒所选定的区域为追踪物体的可能性越大。

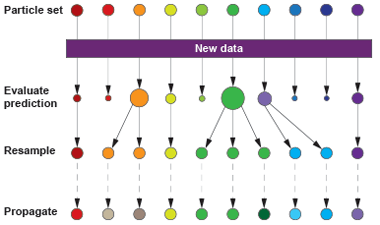


图 2‑1 蒙特卡洛粒子滤波算法框架图

如图 2‑1所示，蒙特卡洛粒子滤波算法框架主要分为4个部分：

1. **粒子初始化：**在全图中随机放置蒙特卡洛粒子，其中包围盒的长度、宽度与第一帧给定物体的包围盒相同。
2. **粒子评价：**对蒙特卡洛粒子集合中的每一个粒子，取其包围盒对应图像区域的子图，并通过某种算法计算评分，更新。通常还会在这一步取权重最大的粒子作为目标包围盒。
3. **重采样：**根据蒙特卡洛粒子集合中粒子的进行归一化后，以此为概率重新初始化粒子。初始化时对粒子进行一定范围的随机抖动。
4. **传播：**对粒子进行维护工作，并开始下一轮粒子滤波。

蒙特卡洛粒子滤波算法具有思路简单清晰、变种方式多样且容易的特点，天然支持并行算法以达到高效的运行速度，理论上不依赖任何先验知识。但它的缺点也是明显的——评价算法直接决定最终结果的质量，往往成为算法的瓶颈，粒子的数量也会影响结果精度。

### 直方图均衡

上述蒙特卡洛粒子滤波框架的粒子评价过程，有许多算法可供选择，本质上只要能够对粒子对应子图与目标物体之间计算相似度，就能满足要求。

目前效果最好的方法包括使用卷积神经网络生成子图的向量表示，然后与目标图的向量表示进行相似度计算（如余弦相似度）。本文由于关注点以及工作量的关系，并未实现此类算法，而使用的是传统的直方图均衡算法[3]。直方图均衡算法是图像处理领域的经典算法，一直被广泛使用，限于篇幅不在此不再赘述其算法原理。

本章算法使用直方图均衡算法对粒子对应子图计算其直方图，并将其理解为子图的向量表示，最后通过与目标图的直方图向量做余弦相似度计算得到粒子的评分。其余步骤与上小节所述相同。

### 整体流程

本章算法所有部分均为在线算法，没有提前训练的过程。整体算法流程如图 2‑2所示。

对ALOV数据集的每段视频，取第一帧图像及标注信息作为算法输入，进行初始化过程。随后的逐帧图像送入算法，得到对应的包围盒，再与标注信息之间进行比较。

粒子评价过程中，图像输入为的3通道全彩色图片，对应一个粒子所表示的子图。生成过程为，先用粒子表示的包围盒在原图中进行抠图，再缩放到分辨率为的图片。包围盒的各参数均采用范围[0,1]的实数，各参数表示与原图间的相对大小。输出为范围[0,1]的实数，表示粒子评分。

实验中，每帧取权重最大的粒子作为结果输出。

对于单物体追踪的实验，由于不是重点，并没有进行完全的实验，这里举出两个标志性的结果供参考。

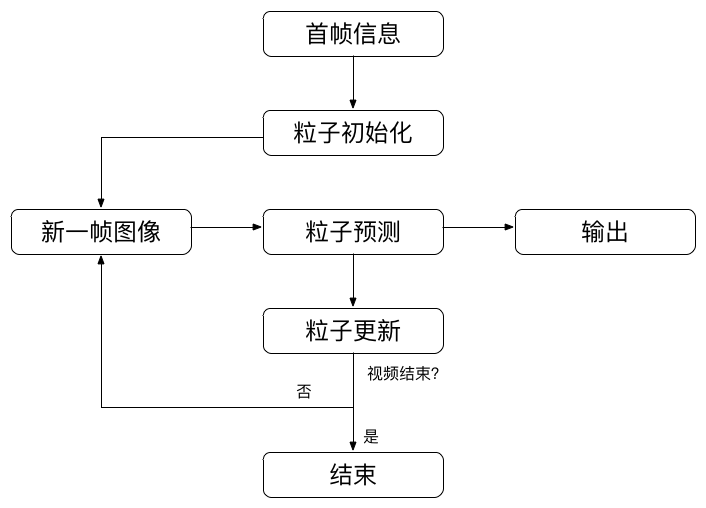


图 2‑2 基于蒙特卡洛粒子滤波的物体追踪算法流程图

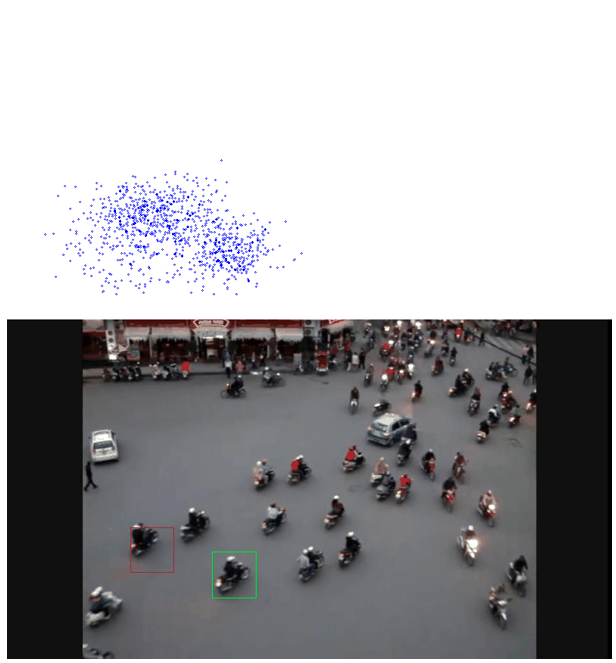
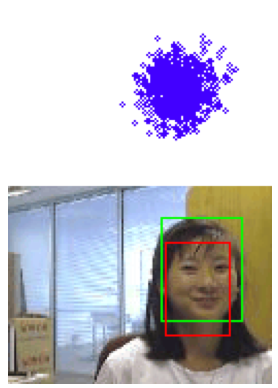


图 2‑3 单物体追踪测试集例子

如图 2‑3所示，上图蓝色点为蒙特卡洛粒子的中心，下图为食品中的某一帧，帧上绿色框是包围盒标注值，红色框是算法给出的预测包围盒。左图是单一人脸，可以看出这种情况下包围盒还是比较准确的；右图是长相类似的骑着机动车的人，由于图中有太多类似物体，在视频的开始片段中，算法还能给出正确的包围盒，但随着时间推移，包围盒的位置开始发生偏离，但所代表的子图仍然是一个标准的骑着机动车的人。这也指出了该类方法的问题——对于图片中大量出现的同类相似物体，该类算法没有很好的方式去避免包围盒的偏离。

## 扩展MOT

上述基于蒙特卡洛粒子滤波的算法框架仅仅是为单物体追踪所设计的，为了让这个框架能够工作于多物体追踪，我进行了如下改动与实验。

### 修改输出过程

在2.3.1 中提到的蒙特卡洛粒子滤波算法框架中，输出的算法是取评分权重最大的粒子，这决定了输出的包围盒只可能有一个。

多物体追踪需要对多个物体输出包围盒。要实现这一点，我把取评分权重最大的粒子改为取评分达到一定阈值的粒子作为输出。

但这会引入新的问题——同一个物体内部往往存在多个粒子超过阈值，使得输出结果中存在多个包围盒对应同一个物体的情况。显然这样的情况是我们不愿意看到的，我们将在下一节讨论解决这种问题的方法。

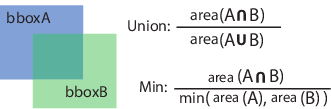


图 2‑4 IOU定义公式

### 非极大值抑制

定义 2‑2 **交除并（IOU）：**如图 2‑4 所示，包围盒bboxA与包围盒bboxB的IOU定义为图中Union等式，即：。

非极大值抑制（NMS）的适用场景为，在多个包围盒对应同一个物体时，可以合并多余的包围盒，得到最能代表物体的包围盒。如图 2‑5 所示，左图为多达5个包围盒，用红框表示，右图为合并后的包围盒。

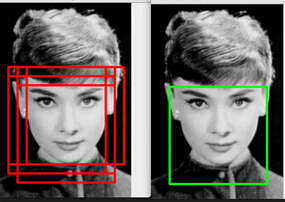


图 2‑5 NMS功能示例

NMS有非常多种类的设计与实现，只要目的不变（合并多余包围盒）即可。本文使用的NMS都是同一种，算法思路为：维护答案包围盒集合S；逐一枚举包围盒，与S中每个包围盒计算IOU值；如果存在一个IOU值小于一个预先设定的阈值，则丢弃本个包围盒，否则将当前包围盒加入S中。

### 整体流程

为了体现IOU阈值对结果的影响，我们使用上述从SOT扩展到MOT的算法，禁用NMS功能，设定IOU阈值为0.8，把权重超过阈值的所有蒙特卡洛粒子所代表的包围盒以红色矩形的形式画在图像上。如图 2‑6 所示，左图为视频序列第一帧，右图为经过轻微旋转并缩小的物体，他们符合要求的粒子数目差异非常大。

可以看出，左图由于就是第一帧图像，符合要求的包围盒数量远远优于右图。然而，实际上两图的图片内容几乎完全一样。使用其他图像实际测试时也发现，使用阈值过滤蒙特卡洛粒子以实现多物体追踪的思路，只是理论上可行，实际上由于物体变化过程中，蒙特卡洛粒子的权重绝对值变化相当剧烈，虽然取最大值的思路对于单物体追踪而言并无变化，但对于多物体追踪来说却是致命的打击。需要注明，这里没有使用权重归一化的原因是，需要重点说明物体抖动对粒子权重的影响程度，而不是针对这一个例。



图 2‑6 扩展MOT粒子权重比较

在测试过程中发现，该方法要想达到比较精确的多物体包围盒，蒙特卡洛粒子数量与需要追踪物体数量的之比大约是1:1000。这样的数量要求无疑对计算性能的要求非常高。同等的计算量条件下，有太多基于深度学习的算法可供选择，而效果远比蒙特卡洛粒子滤波要好。

包含上述两点原因在内，是蒙特卡洛粒子滤波目前不能成为多物体追踪的主流算法的原因。

## 小结

本章重点介绍单物体追踪任务的解决方法，并提出了从经典单物体追踪算法蒙特卡洛粒子滤波算法扩展到多物体追踪任务的算法流程。由于论文关注点不在单物体追踪任务上，本章节提到的算法更多是浅尝辄止，进行定性的分析，并没有进行大批量的量化测试。

单物体追踪任务由来已久，拥有大量的公开数据，也有大量的文献可供查阅。但现有算法大多是基于传统方法的，这些方法对于不同的场景并不具有足够强的适应能力。但很多传统方法仍能与现代深度学习算法相结合，给追踪算法的表现带来新的提高，蒙特卡洛粒子滤波就是一个显著的例子。最后，虽然个别单物体追踪算法能够被扩展到多物体追踪任务上来，但由于这些算法设计时并没有考虑多物体，它们的大多数并不能达到多物体追踪主流算法的效果。

# 多物体追踪算法

## 概述

本章主要介绍了在多物体追踪任务上，现代的主流算法框架，并提出了基于新型追踪算法的算法流程。

本章深度学习模型的训练使用混合数据来源，具体会在3.2 中阐述。

本章还针对新型追踪算法，提出了新颖的模型训练方法，效果出色。

## 数据来源

多物体追踪任务的公开数据集有很多，除了物体内容的区别，格式上都可以抽象成“图像+包围盒”的结构。本小节介绍了本文使用到的一部分公开数据集，并解释了如何抽象其中的数据。

### 2D MOT 2015数据集

几乎所有挑战多物体追踪任务的作者，都会接触到MOT15[1]这个著名的数据集。该数据集集合了众多公开数据集（如KITTI、ETH等）的部分行人视频片段，并进行详细的行人包围盒标注。

训练数据集中包含11个视频片段，时长从3秒到2分钟不等，最低的分辨率为­­，最高达到。

测试数据集同样包含11个视频片段，分别是训练集中对应视频的同类场景，在最大可能上减轻未出现场景对算法效率的影响。

数据集同样支持在线结果测评，共有10余项评测指标共同验证算法的有效性，在线账户可以每72小时提交一个结果用于测试集的评测。而训练集的测评可以通过开源的MATLAB工具包自行完成。

### Microsoft-COCO数据集

微软COCO[5]数据集起始与2014年，直到2016年都仍在更新，同时包含单物体和多物体，提供了丰富的标注信息，是一个集物体识别、物体分割于一身的公开数据集。

本章的模型训练过程，主要使用了微软COCO数据集中的所有单人物图片。这些图片与行人有所不同，有部分场景中只出现了人物的部分身体。

### Poselets数据集

Poselets[4]数据集主要搜集了人体各个不同部分的图片，粗略分为了150个不同类别，每个类别平均有1000张左右的图片实例。

其他公开数据集基本是完整的行人，而该数据集包含比如肩膀、头部等不同部位的图片，如图 3‑1所示为其中某些分类的平均图像。可以利用它做类似分割重叠人物的工作。



图 3‑1 Poselets分类示意图[4]

## 主流框架

目前多物体追踪任务的主流算法流程主要分为两个部分——帧内物体识别以及帧间物体追踪。在大多数算法流程中，此两个部分相对独立，可以根据不同的算法特点进行自由组合。

在帧内物体识别与帧间物体追踪两部分中，在很长一段时间内帧间物体追踪基本大同小异，而决定算法表现的，往往是帧内的物体识别算法。

### 帧间追踪

在单帧图像中识别出多个物体，得到多个包围盒。如何在相邻两帧之间的多个包围盒之间建立对应关系，就是帧间追踪所关注的问题。

在一般的视频片段中，相邻两帧时间间隔最大也在百毫秒级。由此我们可以假设，同一个物体在相邻两帧中的位置变化不会太大——或者换一种说法，同一物体在相邻两帧中的包围盒的IOU值较大（起码不为0）。

于是我们可以使用最优化算法求解这个问题，建模如下：

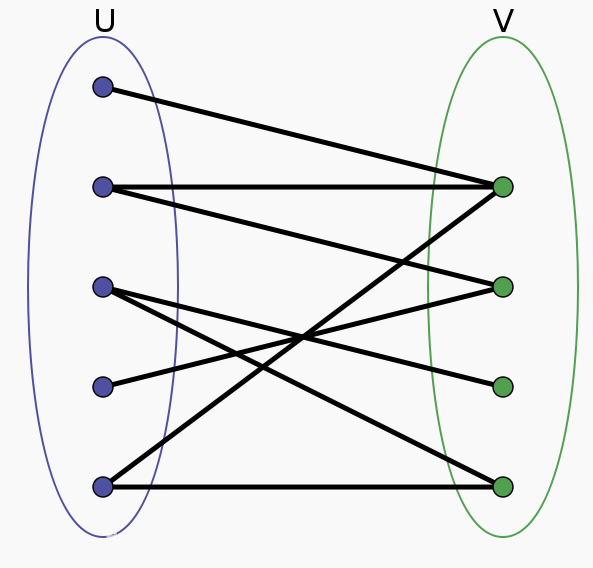


图 3‑2 二分图匹配示意图

如图 3‑2所示，集合U中各点表示前一帧的所有包围盒，集合V中各点表示当前帧的所有包围盒。对集合U中某点u以及集合V中的某点v，如果点u代表的包围盒以及点v代表的包围盒之间的IOU值大于一定阈值（暂设为0），则在点u与点v之间连接一条权值为的有权边。由于同一帧内包围盒相互独立，而相邻帧之间包围盒存在对应关系，所以集合U和集合V以及它们之间的有权边构成了二分图。

求解带权二分图的算法有很多——如最小费用最大网络流算法、Kuhn-Munkres算法[6]等。这里我们使用实现更为简单的Kuhn-Munkres算法（KM算法）。KM算法是经典计算机图论算法，它把任意的带权二分图作为输入，输出则是带有最大边权和的匹配对——即选择原图中m条带权边，使得它们的顶点间完全不重合，并且边权和最大。最开始的版本运行时间复杂度为，其中n为图中点数。后来有了改进版算法，时间复杂度降到了。与各种版本的网络流算法类似，解决二分图完美匹配的KM算法虽然理论时间复杂度高，但在实际使用时远远达不到理论上界，实际运行还是非常高效的。本章算法的实现中，使用的都是时间复杂度更低的实现方法。由于KM算法的理论比较复杂，本文限于篇幅不再展开赘述其算法原理。

利用相邻帧间所有包围盒构成带权二分图，并运用KM算法可以解得多对匹配点，对应相邻帧之间多个互相匹配的包围盒。这样的求解结果使得IOU值大的包围盒之间的对应关系更容易被保留下来，而IOU值小的包围盒对本身存在关联的概率就比较低，要被保留下来非常不容易，这也符合我们的直觉。

在得到相邻帧间互相匹配的包围盒的同时，二分图中集合U与集合V有可能各有部分未被匹配的点。集合U表示的是前一帧中的包围盒，其中的点未被匹配说明在当前帧中，这个包围盒所代表的物体没有被检测到，很有可能已经消失在视野范围内了；而集合V表示的事当前帧中的包围盒，其中未被匹配的点表示一个新近被检测到的物体，很可能是当前帧才刚刚出现的新物体。根据这一特性，我们可以对得到匹配的包围盒对进行维护，对已经消失的包围盒进行删除，对新出现的包围盒进行插入，完成帧间追踪的过程。

### 卡尔曼滤波

在MOT 15数据集上，笔者参照了其中一篇在线实时算法[11]的实现过程中的卡尔曼滤波部分。

卡尔曼滤波的主要用途是，对某个测量目标，在拥有历史测量值以及当前测量值的前提下，更准确地得出该测量目标的当前真实测量值。具体算法原理比较复杂，在此不再赘述。

在本文提出的算法流程中，卡尔曼滤波用于在完成包围盒帧间追踪步骤完成后，精细地调节包围盒的参数。实际上，我们对包围盒建模如(3‑1)：

|  |  |  |
| --- | --- | --- |
|  |  | (3‑) |

包围盒模型中，前四项分别是左上角坐标、面积以及长宽比，后三项分别是左上角坐标以及面积的增量值（可以为负数）。这里我们假设包围盒长宽比不会发生变化。

有了包围盒的建模，再结合卡尔曼滤波对包围盒进行后处理即可。此举可以避免帧间包围盒的突然变化，相当于进行了一次平滑处理。

### 单帧识别-RPN

RPN[7]方法中文全称是“区域提议网络”，有另一个流传更为广泛的名字——“Faster R-CNN”。这个方法由“Fast R-CNN”与全卷积神经网络（FCN）之间结合得来，是最为广泛使用的物体检测算法框架之一。

RPN与“Fast R-CNN”的目标都是更快速地获得区域的提议——也就是物体的包围盒。它们都先利用某种显著性目标检测方法（如Selective Search方法）对原图进行处理，得到多个初始区域候选。RPN原文中提到了一种叫平移不变性（Translation-Invariant Anchors）的特性，本质上就是预先设定多种不同长宽比、不同缩放尺度的滑动窗口。比如原文使用了3种不同的缩放尺度以及3种不同的长宽比（1:1、1:2和2:1），组合得到了k=9个锚点。

RPN是FCN的一种，模型先通过卷积得到关于原图的特征图，再把特征图映射到一个低维向量（如256维向量），然后增加两个卷积层旁支，一个分支预测k种锚点对应子图的分类，另一个分支本质上对k种锚点分别完成了包围盒回归，也就是对锚点对应的子图，预测更“紧凑”的物体包围盒。将上述两类预测进行组合，再配合2.4.2 提到的非极大值抑制算法合并重复的包围盒，得到最终结果。整体流程如图 3‑3所示。

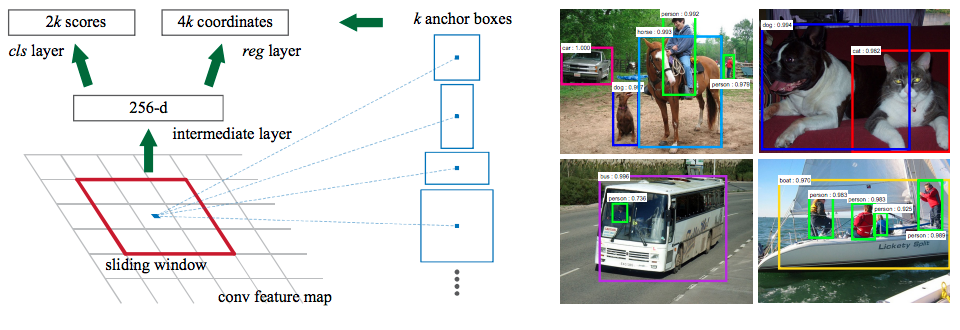


图 3‑3 RPN模型结构及输出结果示例[7]

RPN可以被端到端地训练，作者也通过项目开源平台Github以Caffe项目的形式对实验进行了开源。由于笔者在其他任务上尝试过RPN模型，最后只能达到基本效果，其中假性正例的情况比较严重。 虽然算法流程中的帧内识别部分可以被完整地替换成基于RPN实现的方法，但笔者的最后实现中并没有使用RPN。这里只是作为一种实际可行性思路介绍RPN。

### 单帧识别-HED

HED[8]英文全称是“Holistically-nested Edge Detection”，也就是“整体嵌套边缘检测”，提出的是一套学习框架。HED主要解决精确语义分割的任务，也就是使用卷积神经网络进行轮廓线的计算，而且该模型的设计能够适应不同尺度的物体。如图 3‑4所示即为模型各分支输出。

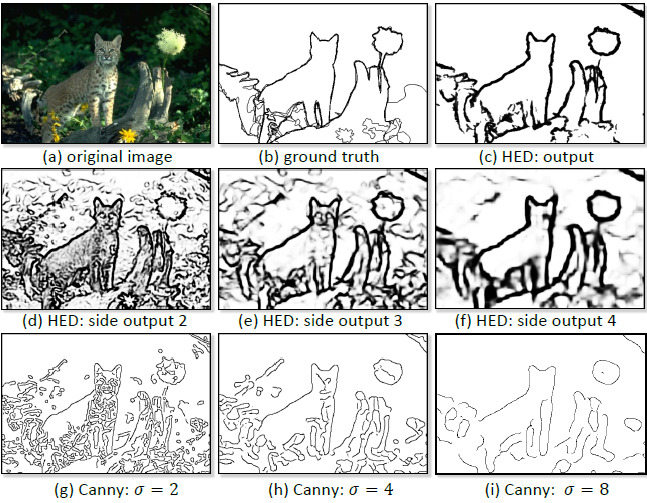


图 3‑4 HED分支输出示例[8]

HED本质上是全卷积神经网络与深度监督网络的结合，模型前半部分使用在ImageNet上预先训练的VGG模型进行初始化。同时，模型包含五个不同层级的子结构，每个子结构均预测与原图大小相同的热力图，表示最终的轮廓结果，与全卷积神经网络思路相似；而每个层级均利用深度监督网络接收监督信号。 如所示，5个层级越往后，模型预测输出越抽象，越接近真实轮廓线；而越往前的层级，则越接近传统算法得到的边界信息，还能保留更多的局部细节。正是这种明显的层级特性，体现出HED与DSN相辅相成的特点。作者在文章中表示，这也可能导致算法在高召回率的情况下，准确率会有所下滑。

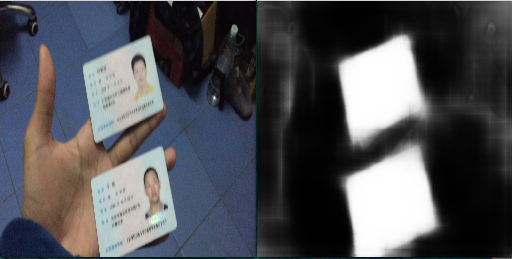


图 3‑6 卡片测试集输出结果示例

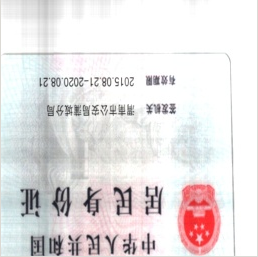


图 3‑5 卡片训练数据示例

原文利用HED进行轮廓线的计算，而由于轮廓线的有效信息比较稀疏，不利于实现物体检测，这里笔者对HED进行了修改。为了提高输出结果的稠密度，我们把监督信号从轮廓线改成了实心的热力图。

然而在实验过程中，笔者发现改成实心热力图的版本始终得不到较好的结果。在原文中，作者提到损失函数的设计要参考全卷积神经网络对像素级分割做的思考，也就是如果标注值并不是轮廓线，则有响应的部分数量会远超轮廓线的部分。如果这时不改变损失函数，则模型很可能会过拟合为没有轮廓线，引入均衡化过的损失函数可以解决这个问题。笔者考虑上述情况后，结果仍然没有明显的改善。最后经过朋友建议，取消了前四层级的监督信息，效果变得很好，但至今仍然不清楚其中的原因。

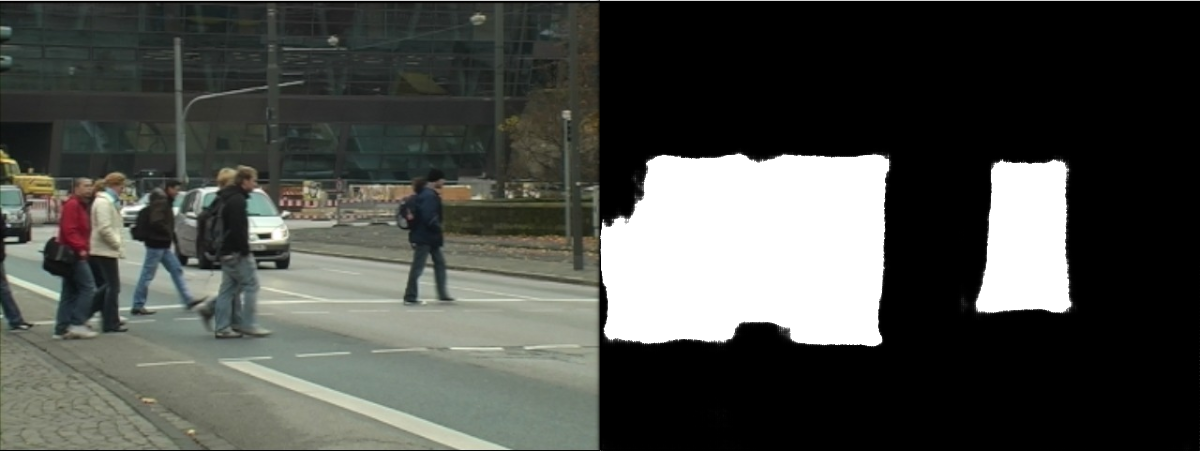


图 3‑8 HED行人测试集输出结果示例



图 3‑7 行人训练数据示例

由于卡片具有形状规则、不易重叠等特点，我们在身份证上做了一次定性的实验。训练时输入数据如图 3‑5所示：左图是一张背面身份证，右图是对应的实心包围盒热力图。如图 3‑6所示则是在测试图片上，训练完毕模型的表现。可以看到，尽管HED原本设计初衷是为了解决边缘检测的问题，但经过修改仍能在实心的热力图上良好地工作。而且即便训练数据中全是单张身份证，但在实际工作时，对图中的多张卡片仍然能正常工作。

有了卡片数据集上的成功经验，笔者还在行人数据集MOT15上进行了实验。实验时同样使用端到端训练，输入数据如图 3‑7所示，左图是一张包含若干行人的普通街景照片，右图是包含对应每个行人实心包围盒的热力图。使用上述训练完毕的模型，在测试集上的预测结果如图 3‑8所示。可见算法对不同类型的任务具有较强鲁棒性。

然而从图 3‑8也可以看出，当物体倾向于发生重叠时，上述方法并不能有效地区分不同的物体。这在现实世界的物体追踪时，无疑会导致最终算法流于实验室玩具的级别，而不能走向实用。笔者想到的补救方案为，对粘连在一起的联通快，使用人头检测分割不同的行人。这里可以使用Poselets数据集进行人头HED的训练。但这种方法仍是治标不治本。

我们将在下一节中介绍此算法的改进版本，提出上述问题的解决方案。

### 单帧识别-稠密盒

稠密盒[9]是HED的改进版本，具体译文请参照附录A 书面翻译**。**

稠密盒对HED的改进在于，它使用了两个旁路，一个与HED一致，另一个输出4通道的与原图大小一致的图，每个像素点的4个值表示该像素点与所属包围盒四条边界之间的距离。除了输出后处理时的nms以外，整体算法框架都符合全卷积神经网络的结构。所以稠密盒也具有能够端到端学习的特点，可以在生成热力图的同时生成包围盒，而且也能处理遮挡的情况——HED则完全不能有遮挡。

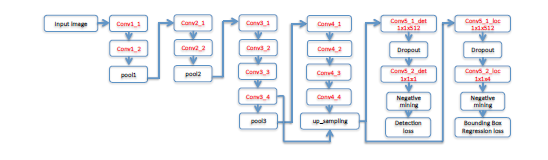


图 3‑9 稠密盒模型结构[9]

与HED相似，模型如图 3‑9所示，模型的前半部分使用VGG模型初始化，然后经过一连串的卷积层后，通过两个分支，一个输出1通道的置信度热力图，另一个输出4通道的包围盒回归。



图 3‑10 行人测试集输出结果示例

原文还提到了训练时的一系列优化方式：比如标注值的热力图不要直接是实心矩形，而应该是一个实心圆；比如损失函数应该把包围盒回归部分的四通道输出图中，不属于物体包围盒的部分直接忽略掉，否则会使损失函数被无关紧要的区域所统治；其中笔者认为最重要的一点是，原文提到了负例样本挖掘技术，也就是在训练过程中，应该尽量使得预测结果与标注值差距最大的一批样本占据更大的比例，论文实际训练时的配置为，普通样本与损失函数值最大的负例样本间数量比例为1:1。

由于发现此论文时毕业设计工作已经接近尾声，笔者在tensorflow[10]开源框架上实现的时候，基本遵照HED的训练过程进行稠密盒的训练，而并没有使用稠密盒原文提到的负例样本挖掘技术。

在行人数据集MOT15上，笔者训练的稠密盒模型在部分场景下的预测结果如图 3‑10所示，这里使用了上文提到的NMS算法，IOU阈值设为0.2。可以看出，在遮挡比较严重的区域，包围盒仍然比较正确地被预测；而同样的图，在HED上则是一片互相融合的热力区域。

### 离线算法

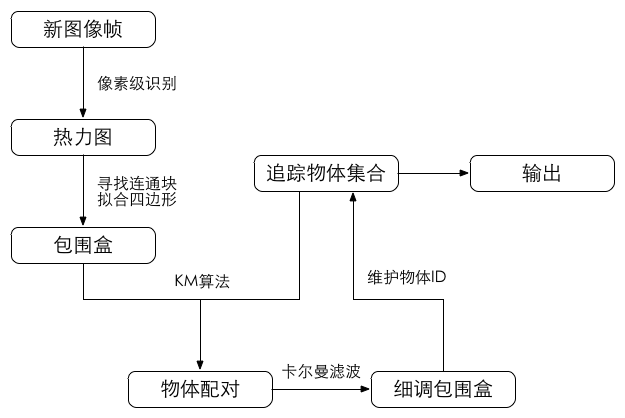


图 3‑11 多物体离线追踪算法流程图

如图 3‑11所示为使用HED的算法流程，使用稠密盒的的算法流程与其大同小异，在此不再赘述。

流程对于新的图像帧，先使用HED模型，预测出所追踪物体的热力图；而后通过广度优先搜索算法，每个被激活的像素点与欧几里得距离小于2的像素点可以相连，得到若干个物体的包围盒；而后使用3.3.1 提到的帧间追踪算法进行包围盒的配对，其中也使用到了3.3.2 提到的卡尔曼滤波算法进行包围盒的细调；最后把当前维护的物体集合输出。

对于使用稠密盒的算法流程，则把上述流程中广度优先搜索算法寻找包围盒的步骤直接省略，而套用NMS算法去除重复包围盒即可，其余步骤保持一致。

由于识别算法HED或稠密盒都使用了VGG模型的部分结构，所以在速度上，此算法流程即便有显卡设备的加速，也仍然不能在1080P级别的高分辨率上做到实时输出，所以这套算法流程只能作为离线算法使用。我们将在下一节中提出我们的在线算法流程。

## 新型追踪算法

### 深度回归网络

深度回归网络[12]首次提出是在2016年4月，笔者认为，它的提出是具有里程碑意义的。在它的出现以前，绝大多数多物体追踪算法在帧间追踪时都是使用的传统方法，包括上文提到的所有算法流程。然而深度回归网络的出现，使得帧间追踪的任务也能被神经网络化，虽然它的提出本身是为了提高单物体追踪的精度，而并不是针对多物体追踪所设计的。

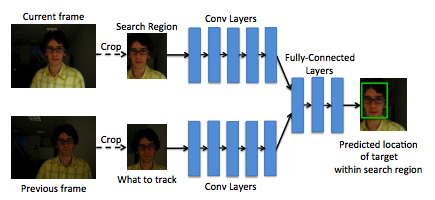


图 3‑12 深度回归网络模型结构图[12]

深度回归网络的基本思路如图 3‑12所示。

假设在前一帧中我们已经知道所要追踪的物体的具体位置（如左下方的人像），则把该包围盒放在画面正中央，整体图像边长为包围盒边长的三倍，用这样的新包围盒对前一帧的原图进行抠图，则会得到如左下方人像的第一个输入图片，此时人脸正好位于画面正中央；对于当前帧，我们用同样的新包围盒进行抠图，得到左上方的图片，这时所要追踪的人脸虽然不在正中央，但仍然完整地在画面内部。这里有个前提假设，也就是相邻帧之间相差的时间间隔不能太大，以保证运动的物体在我们的新包围盒范围内。两帧的抠图之后都会被缩放到的分辨率。

模型把前后两帧的抠图作为输入，分别通过两个分支经过一系列的卷积层后，通过一次连接过程把两个向量表示前后连接在一起，之后通过一系列全连接层的计算，得到一个4维的输出，表示包围盒回归的结果。

深度回归网络的模型结构以及训练过程并不复杂，作者通过这样的网络模型，在VOT数据集上进行单物体追踪任务的训练，并取得了排行榜的第一名，打败了诸多经过仔细调整参数的复杂算法流程。

后文将会介绍如何把深度回归网络扩展到多物体追踪的任务上来。

### 飞天椅子数据集

飞天椅子数据集[13]提供了超过1300张不同姿态的椅子，每张椅子多个不同角度的渲染图。如图 3‑13所示即为同一张椅子不同角度的渲染图。



图 3‑13 飞天椅子数据集示例[13]

这个数据集使用1300多张椅子的3D模型，通过模型处理工具批量导出jpeg格式的图片。同时，数据集中还包含一批合成的光流数据。这批数据的优点在于，对于同一个物体，我们拥有不同角度的观测图，而不像以往只能通过一成不变的一张图片来表示一个物体，使得神经网络的训练能更有效。

下一节将介绍，如何使用飞天椅子数据集通过合理的数据增广方法，达到检测行人的效果。

### 训练过程

本节将介绍如何使用飞天椅子数据集以及MOT15行人数据集，在单行人追踪任务上训练深度回归网络。

首先使用飞天椅子数据集进行训练。

随机抽取飞天椅子数据集中同一椅子两个不同角度的图片，通过过滤白色像素点的方式得到带透明背景的椅子图片；然后经过一系列旋转、缩放、位移的操作，得到新的两张椅子图片；之后从微软COCO数据集中随机抽样一张图片作为背景图，分别把两张椅子图片通过贴图的方式贴到背景图上，得到深度回归网络的两张输入图片，如图 3‑14上图所示。



图 3‑14 训练数据及测试结果

实际测试时发现，单纯使用飞天椅子进行训练的深度回归网络在行人数据集MOT15上虽然大部分能够追踪，但只能大致定位，对于人的四肢和头部表现仍不够好，见图 3‑14下图所示。笔者猜测，这是由于椅子的形状与行人还是有很大的区别的，可能是深度回归网络没有见过行人的样子，并不知道四肢头部也是人的一部分。而且在画面中如果有多个外观相似的人出现，深度回归网络不知道哪个才是需要追踪的人，于是追踪结果为具体哪个人有一定的随机性。

为了解决上述两个问题，笔者进行了如下的实验。

首先在生成飞天椅子图片时，笔者在单张图片中引入了多张椅子，用以模拟相似追踪目标有多个的情况。经过实验发现，要追踪的椅子可以在相邻帧之间随机出现位置，而其他椅子需要在两帧中具有差不多的位置。最后生成的飞天椅子输入图片如图 3‑15所示。在上述训练数据上深度回归网络收敛以后，再使用MOT15的行人数据进行训练，直到收敛。



图 3‑15 改进训练数据

最后证明，上述实验成功在行人数据集上达到比较可观的效果。

由于深度回归网络所需的计算量非常小，进行一次完整的计算只需要数十毫秒。但它的缺点是，没有识别模型的协助，每帧的回归误差会累积起来，所以不建议全程使用深度回归网络。

### 在线实时算法流程

有了深度回归网络的帮助，我们能够在牺牲一定精确度的情况下，使得前文提到的多物体离线追踪算法流程通过修改达到实时在线的地步。

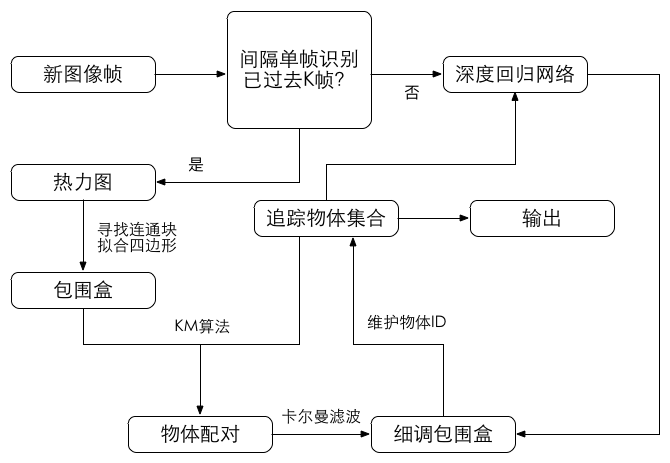


图 3‑16 在线实时多物体追踪算法流程图

在线实时算法流程如图 3‑16所示，基于稠密盒的离线算法流程基本不变，但并不是每一帧新图像都进行单帧识别，而是每隔k帧进行一次识别，校正包围盒；期间的每帧采用深度回归网络更新每个已追踪物体的位置。k的取值要适中，过大会由于深度回归网络的累积误差导致包围盒精确度下降，过小则达不到实时计算的要求。经过实验发现，k=5是运行速度与包围盒精确度间较好的平衡点。

## 小结

本章介绍了基于单帧识别以及帧间追踪的现代主流多物体追踪算法框架，其中单帧识别在算法框架中占决定性作用。

帧间追踪在本章提到的多物体追踪算法中基本一致，其中使用了KM算法求解带权二分图以完成帧间包围盒的匹配，并且使用卡尔曼滤波算法对包围盒进行平滑处理，避免帧间包围盒突然变化。

随后针对单帧物体识别方法，本章介绍了包括RPN、HED以及稠密盒等算法，并进行了相关的实验。实验证明在行人及卡片的数据集上，这些方法都可以达到我们的目的，并能良好地工作。

基于上述帧间追踪以及单帧识别的方法，本章提出了多物体追踪的离线算法流程，并取得了不错的效果。

最后引入新近发表的深度回归网络，第一次在帧间追踪过程中出现了基于神经网络的方法。由于深度回归网络计算量小，由此本章提出了离线算法的改进办法，也就是基于深度回归网络的多物体追踪的实时在线算法。

# 实验结果与分析

## 实验设计

本章主要给出关于多物体追踪的实验设计、结果及分析，因为本文对单物体追踪更多是介绍的性质，并没有投入太大的精力去优化。

在多物体追踪一章中，本文提出了离线追踪与在线实时追踪两种算法流程，以下将对这两类算法分别进行实验。

对于在线实时追踪算法，由于该算法主要是牺牲精确度以换取运行时间上的优势，所以在实验结果中，我们主要针对其在不同分辨率大小的图片上的运行时间分析，并辅以在MOT15行人测试数据集上的各项追踪指标。

对于离线算法，理论上这是本文提出的算法流程中，各项追踪指标最优的方法。以下将通过在MOT15行人测试数据集上的各项追踪指标来衡量算法优劣。

值得一提的是，本文提到的多物体追踪算法流程，都是基于实验开始前MOT15排行榜上支持实时在线计算的最优算法[11]，当时全榜排名是第10名。而截至本论文写作完成时，该算法在排行榜中的排名已经掉至第17名。由于作者并不打算继续优化算法，未来排名只可能继续下降。之所以有这么大的跌幅，是因为最近截至论文完成前的两个月内，有数个优秀的基于离线算法的提交产生，也新出现了两个思路不同的实时算法提交。本文提出的算法更多是实验性质，单个排行榜上的最优并不是唯一的目的，所以下文都是与该最优算法[11]进行比较，表明我们的算法流程确实能有效提高追踪准确度。

以下实验结果都仅使用了MOT15行人数据集中的训练集进行训练，而该比赛排行榜上大量的优秀算法还使用了比如INRIA数据集等进行训练。在不同场景的适应性上，本文提出的算法实现与现行较优算法进行比较还有不足之处。

## 实验结果

在2DMOT15数据集[1]上，我们的多物体追踪算法流程是基于当时最优的实时算法SORT[11]提出的。

SORT算法在数据集上的各项公开数据如表 4‑1所示，我们提出的离线算法测试结果如表 4‑2所示。其中各项指标如[1]中所定义，我们主要关注其中的精确率（Prcn）、召回率（Rcll）以及多物体追踪准确率（MOTA）。前两者使用广泛，这里给出MOTA的计算公式(4‑1)：

|  |  |  |
| --- | --- | --- |
|  |  | (4‑) |

公式中，FN和FP以及GT分别是第t帧中假负例、假正例以及标注值的物体数量，IDSW表示第t帧与第t-1帧中物体id错位的物体数量。

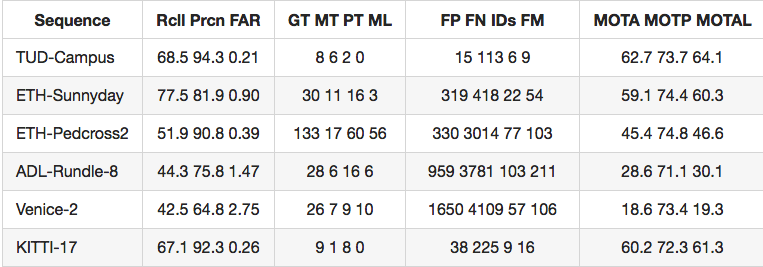


表 4‑1 SORT算法MOT15部分评测结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sequence | Rcll Prcn FAR | GT MT PT ML | FP FN IDs FM | MOTA MOTP MOTAL |
| TUD-Campus | 71.6 84.0 0.69 | 8 3 5 0 | 49 102 11 14 | 54.9 67.1 57.6 |
| ETH-Sunnyday | 86.8 88.0 0.62 | 30 19 8 3 | 219 246 52 61 | 72.2 71.9 74.9 |
| ETH-Pedcross2 | 62.4 53.4 4.07 | 133 35 67 31 | 3404 2355 252 225 | 4.0 72.5 8.0 |
| ADL-Rundle-8 | 45.2 24.2 14.64 | 28 1 25 2 | 9575 3720 376 656 | -101.5 65.3 -96.0 |
| Venice-2 | 32.6 15.2 21.70 | 26 2 13 11 | 13018 4813 208 360 | -152.6 62.2 -149.7 |
| KITTI-17 | 65.6 26.6 8.51 | 9 0 9 0 | 1234 235 44 36 | -121.5 66.4 -115.3 |

表 4‑2 离线算法MOT15部分评测结果

而在运行时间方面，离线算法与在线实时算法、深度回归网络等在不同输入大小下的数据如表 4‑3所示。此表在MOT15数据集上运行得来，对相同分辨率取平均值作为结果。其中，在线算法每5帧进行一次稠密盒识别，取整段视频的平均运行时间作为结果。算法运行环境是16核32线程Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz处理器，3TB内存，显卡Tesla K40m。运行时均使用单张显卡单个处理器。表格中所有单位均为秒。

|  |  |  |  |
| --- | --- | --- | --- |
| Pipeline | 640x480 | 1232x384 | 1920x1088 |
| HED | 0.053899 | 0.078223 | 0.239522 |
| 稠密盒 | 0.493648 | 0.625253 | 4.112028 |
| 深度回归网络 | 0.025756 | 0.027359 | 0.105508 |
| 离线算法（稠密盒） | 0.527820 | 0.650211 | 4.139440 |
| 在线实时算法（稠密盒） | 0.124599 | 0.147591 | 0.911928 |

表 4‑3 各算法不同分辨率下单张图片运行时间

其中HED是已经优化过的算法模型，而稠密盒使用的是经过预训练VGG模型初始化的版本，所以二者运行时间存在较大区别，HED已经达到GPU上实时运行的级别。

## 结果分析

通过4.2 的实验结果可以看出，在某些场景下，本文提出的算法流程比SORT算法[11]有明显的提高，这主要归功于识别算法的提高。而在其他的一些场景下，本文提出的算法流程效果明显比较差。这可能是因为在训练过程中，由于时间紧迫，我并没有按照论文中的所有训练技巧进行训练，这可能导致在某些场景下表现不佳。

而在运行时间一项中，可以看出我们提出的多物体追踪算法在720P级别的输入分辨率下，可以做到几乎没有延时；而在1080P级别的输入分辨率下，也只有3秒左右的延时。这样的运行时间要求完全可以做到在线实时的要求。

# 结论

## 工作总结

随着机器学习特别是深度学习算法的强势崛起，多个传统研究方向都得到了新的发展方向，其中图像与语音处理更是最大的受益者。卷积神经网络的引入，使得物体追踪的新算法对不同的场景变化变得越来越鲁棒。

本文针对物体追踪这一计算机视觉中的经典研究课题，提出了包含单物体追踪及多物体追踪的一系列算法流程。 对于单物体追踪，本文提出了基于传统方法的实时在线算法流程，并且根据蒙特卡洛粒子滤波框架，给出了与现代神经网络算法相结合以提高性能的方案。由于本文重点不在此，单物体追踪并未再进行深入的探讨。本文在多物体追踪任务上给予了较多的关注，给予单帧识别以及帧间追踪的现代算法框架，本文总结并提出了帧间追踪的各个组成部分，以及单帧识别上可以用的RPN、HED及稠密盒算法，并通过实验得出这些方法的可靠性。最后针对新近出现的深度回归网络，提出了与离线算法流程的结合，使得帧间追踪不再是传统方法的专利，并由此得到多物体追踪任务上的在线实时追踪算法流程。

## 进一步工作

然而，本文提出的各算法流程并不尽善尽美。

多物体实时追踪是一个经典且难度高的研究课题，在毕业设计短短的一学期内，能仔细完成的探索与实验不多，整体工作还有很高的提升空间。以下是笔者总结的一些可能的改进方向：

* 使用更多更全面的行人数据，尝试训练出适用场景范围更大的稠密盒。
* 对于深度回归网络方面的探索还比较粗糙，可以通过更多特定领域（如行人）的实验，使得深度回归网络在累积误差方面表现得更好。
* HED和稠密盒在模型设计上都使用了VGG模型的一部分进行初始化操作。但VGG模型本身计算量非常大，对于实时算法来说不是一个明智的选择。如果有更多的时间，可以尝试对特定领域的任务，优化模型的结构，使得前半部分的运算量能减少一个量级以上，大大加速算法的运行速度。

本文提出的方法紧贴物体追踪方面的前沿学术成果，有机会希望能继续相关的研究，争取能够实现出现实生活中切实能用的应用。

插图索引

图 1‑1 红色矩形的行人包围盒 2

图 2‑1 蒙特卡洛粒子滤波算法框架图 6

图 2‑2 基于蒙特卡洛粒子滤波的物体追踪算法流程图 8

图 2‑3 单物体追踪测试集例子 8

图 2‑4 IOU定义公式 9

图 2‑5 NMS功能示例 10

图 2‑6 扩展MOT粒子权重比较 11

图 3‑1 Poselets分类示意图[4] 14

图 3‑2 二分图匹配示意图 15

图 3‑3 RPN模型结构及输出结果示例[7] 17

图 3‑4 HED分支输出示例[8] 18

图 3‑6 卡片测试集输出结果示例 19

图 3‑5 卡片训练数据示例 19

图 3‑8 HED行人测试集输出结果示例 20

图 3‑7 行人训练数据示例 20

图 3‑9 稠密盒模型结构[9] 21

图 3‑10 行人测试集输出结果示例 22

图 3‑11 多物体离线追踪算法流程图 23

图 3‑12 深度回归网络模型结构图[12] 24

图 3‑13 飞天椅子数据集示例[13] 25

图 3‑14 训练数据及测试结果 26

图 3‑15 改进训练数据 27

图 3‑16 在线实时多物体追踪算法流程图 27

表格索引

表 2‑1 视频环境变化列表[2] 5

表 4‑1 SORT算法MOT15部分评测结果 30

表 4‑2 离线算法MOT15部分评测结果 30

表 4‑3 各算法不同分辨率下单张图片运行时间 31

参考文献

1. Leal-Taixé L, Milan A, Reid I, et al. Motchallenge 2015: Towards a benchmark for multi-target tracking[J]. arXiv preprint arXiv:1504.01942, 2015.
2. A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. TPMAI, 2014.
3. Pizer S M, Amburn E P, Austin J D, et al. Adaptive histogram equalization and its variations[J]. Computer vision, graphics, and image processing, 1987, 39(3): 355-368.
4. Lubomir Bourdev, Subhransu Maji, Thomas Brox, Jitendra Malik,Detecting People Using Mutually Consistent Poselet Activations, ECCV 2010
5. Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[M]//Computer Vision–ECCV 2014. Springer International Publishing, 2014: 740-755.
6. Gupta A, Ying L. Algorithms for finding maximum matchings in bipartite graphs[M]. IBM Thomas J. Watson Research Division, 1999.
7. Ren S, He K, Girshick R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[C]//Advances in Neural Information Processing Systems. 2015: 91-99.
8. Xie S, Tu Z. Holistically-nested edge detection[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 1395-1403.
9. Huang L, Yang Y, Deng Y, et al. DenseBox: Unifying Landmark Localization with End to End Object Detection[J]. arXiv preprint arXiv:1509.04874, 2015.
10. Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015[J]. Software available from tensorflow. org.
11. Bewley A, Ge Z, Ott L, et al. Simple Online and Realtime Tracking[J]. arXiv preprint arXiv:1602.00763, 2016.
12. Held D, Thrun S, Savarese S. Learning to Track at 100 FPS with Deep Regression Networks[J]. arXiv preprint arXiv:1604.01802, 2016.
13. Fischer P, Dosovitskiy A, Ilg E, et al. Flownet: Learning optical flow with convolutional networks[J]. arXiv preprint arXiv:1504.06852, 2015.

致 谢

衷心感谢姚海龙老师在本次毕业设计及综合论文训练中对我的帮助和悉心指导。姚海龙老师认真负责的研究态度以及严谨的作风都让我受益匪浅。

感谢清华-微软未来互联网与计算兴趣团队以及北京旷视科技有限公司对本毕业论文所进行实验提供服务器及显卡支持。感谢公司同事在本文的实验及写作过程中给予的无私帮助，没有他们本文的实验进展不可能如此顺利。

最后感谢班里同学在我完成综合论文训练阶段给予我的帮助。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：\_\_\_\_\_\_\_\_\_\_\_\_\_ 日 期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 附录A 书面翻译

稠密盒：基于端到端物体识别的统一标志定位

A1 摘要

一个FCN（全卷积神经网络）要怎么在物体检测上工作？我们发明了稠密盒——一个统一的端到端FCN框架——可以直接预测单张图片中任意位置及大小物体的包围盒以及物体分类置信度。我们的贡献主要有两方面。一是我们证明了一个FCN，只要经过小心的设计以及优化，可以极其准确且高效地检测多个不同物体；二是我们证明在多任务学习时引入标志定位，稠密盒可以更好地提高物体检测精确度。我们展示了在包括MALF人脸检测以及KITTI车辆检测的公开数据集上的实验结果，结果表明我们的稠密盒系统在检测如人脸、车辆等有挑战性的物体方面是目前最先进的。

A2 引言

我们的日常生活充斥着大量的物体检测实例。驾驶时注意最近的车辆、找一个人以及定位一张熟悉的脸，这些都是物体检测的例子。物体检测是计算机视觉中最核心的问题之一。在CNN成功以前，物体检测常常会归类为基于移动窗的方法，也就是在图像中所有可能的位置及尺度中运用分类器抽取人工特征。最近，基于FCN的方法给物体检测领域带来了一场革命。这些FCN框架也遵循移动窗的特点，但他们在学习模型参数及抽取图像特征时端到端的方式显著提升了检测表现。

RCNN相对基于FCN的方法在物体检测准确率上有了进一步提高。理论上，RCNN包含两个部分。一是使用区域提议方法生成图像中所有潜在的候选包围盒；二是使用了CNN分类器为每个提议分辨出不同的物体。虽然RCNN在普适物体识别上成为了新的最先进系统，但它仍很难检测如人脸、远处的车等小物体，因为低分辨率及缺少背景信息的候选包围盒会显著降低分类准确率。更多地，RCNN流程上这两个不同部分不可以被合并地优化，为端到端训练RCNN留下了麻烦。

我们的工作关注一个问题：在物体检测上一个单级FCN能达到什么程度？为此我们提出了一个基于FCN的物体检测器——稠密盒——不需要提议生成并且可以在训练时被端到端地优化。尽管与很多已有移动窗FCN检测框架相似，稠密盒还是在小尺度及高污染的情况下被更仔细地设计。我们训练稠密盒并使用精细、困难的挖掘技术去压榨它的表现。为了使它表现更好，我们更通过多任务学习在系统中整合了标志定位。为了验证标志定位是有用的，我们在KITTI车辆检测上手工标注了一批关键点数据，并会在将来公开它们。

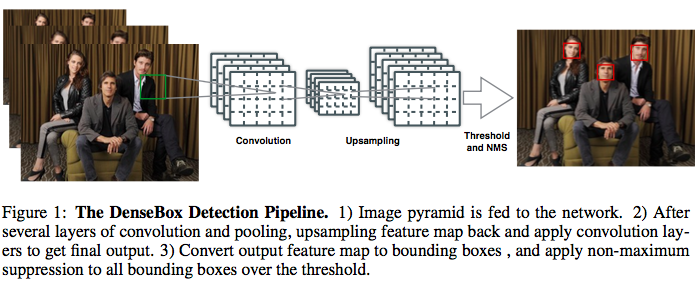
A3 稠密盒检测

图 1

一个完整的检测系统如图 1所示。一个卷积网络同时输出多个预测包围盒以及分类置信度。稠密盒中物体检测的所有部件都被建模成一个FCN（除了非极大值抑制的部分），所以区域提议生成不是必要的。在测试中，系统把一张图片（尺寸为）作为输入，并输出一张尺寸为的5通道特征图。如果在输出坐标空间中我们分别定义包围盒的左上和右下角为和，那么在输出特征图中，每一个位于的像素点i会以一个5维向量 描述一个包围盒，其中是作为一个物体的置信度分值，表明输出像素位置距离目标包围盒的边界的距离。最后每个输出图中的像素被转化为带置信度的包围盒，并对其中置信度达到阈值的包围盒应用非极大值抑制。

A3.1 生成标注值

把整个图像输入到网络里训练是不必要的，因为这会花费大多数计算时间在对背景的卷积操作上。一个聪明的策略是剪切含有人脸以及足够背景信息的子图进行训练。在论文中，我们在单尺度上训练网络，并使用多尺度进行校验。

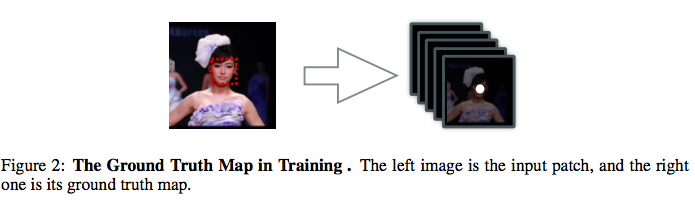
通常来说，我们提出的网络是使用类似分割的方法训练的。在训练时，子图会被剪切并缩放到，并包含位于正中间的高度大致50像素的人脸。训练时输出标注值时一个被缩放到的5通道图，向下采样系数为4。标注图第一个通道内的正例标注区域是一个半径为的填充圆，圆心在人脸包围盒的中心。半径对包围盒尺寸是正定的，输出坐标空间中相对盒子尺寸的缩放系数被设为0.3，如图 2所示。剩下的4通道被输出图像素坐标距离最近包围盒的左上、右下角的距离填满。

图 2

注意如果在单个子图中出现多张人脸，如果他们在相对于子图中心缩放范围内（在我们的设置中设为0.8到1.25），我们会保持这些脸为正例.其他脸会被视为负例。第一个通道中的像素点——表明了分类的置信度——在标注图中会被初始化为1，之后只要在正例标注区域内则设为1。我们也发现我们的标注值生成与Pinheiro等提出的分割工作比较相似。在他们的方法中，像素标注被物体在子图中的位置所决定，而在稠密盒中，像素标注被感应区域决定。具体地，如果像素满足感应区域大致包含一个物体在中心且在给定尺度的限制，则它会被标注为1。每个像素会被视为一个抽样，因为每个5通道像素描述一个包围盒。

A3.2 模型设计

我们展示在图 3的网络架构是从用于图像分类的VGG 19模型演化过来的。整个网络有16个卷积层，前12个卷积层被VGG 19模型初始化。conv4\_4的输出被送进四个的卷积层，其中前两个卷积层输出表示分类值的单通道图，另两个通过4通道图预测包围盒的相对位置。最后的卷积层就像移动窗里的全连接层一样工作。

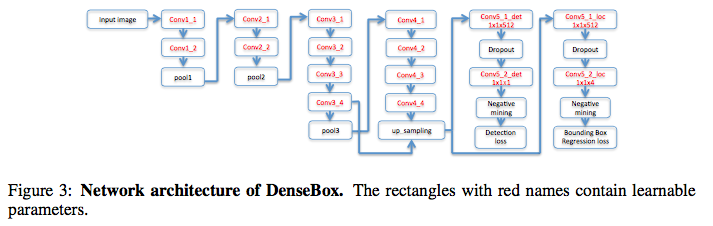
多层特征融合。最近的工作表明使用不同卷积层得到的特征可以增强如边界检测与分割的效果。旁支级别特征关注物体的本地细节以找到有判断性的外观部件，物体级别和高级别特征为了识别物体通常有更大的感应区域。更大的感应区域还未预测更准确的结果带来背景信息。在我们的实现中，我们把conv3\_4和conv4\_4的特征图连接在一起。conv\_3\_4的感应区域（或称移动窗口尺寸）是，几乎和训练中脸的尺寸一样，并且conv4\_4有一个大得多的感应区域，大概，可以利用全局纹理和上下文用于检测。注意conv4\_4的特征图尺寸是conv3\_4生成的图的一般，所以我们使用双向上采样层去把它们转换成相同的分辨率。

图 3

A3.3 多任务训练

我们使用ImageNet提前训练的VGG 19网络去初始化稠密盒。实际上，在初始化时，我们只保留前12个卷积层（从conv1\_1到conv4\_4），并且VGG 19中的其他层都被四个带有xavier初始化的新卷积层所替代。

像Fast R-CNN一样，我们的网络有两个兄妹输出分支。第一个输出（输出图中的每个像素）是一个目标物体的置信度分值。给定标注值，分类的损失函数可以被定义如下：

|  |  |  |
| --- | --- | --- |
|  |  | (‑) |

这里我们在人脸和车辆检测任务中都使用L2损失函数。我们没有尝试像hinge损失或者交叉熵损失这样其他的损失函数，这看起来是一个更正确的选择，因为我们发现简单的L2损失函数在我们的任务中表现很出色。

第二个分支输出包围盒的回归损失，标记为。它的目标是最小化预测位置偏移和目标间的L2损失，用公式表示为：

|  |  |  |
| --- | --- | --- |
|  |  | (‑) |

A3.3.1 平衡抽样

选择负例样本的过程是学习中的一个关键部分。如果在一个迷你批中简单地使用所有负例样本，由于他们统治了所有样本，将会使得对负例样本的预测有偏差。更多地，如果我们在那些位于正例与负例区域间的样本上进行损失惩罚，检测器将会降级。这里我们对每个输出像素使用一个二值遮盖来表明它是否在训练中被选中。

无视灰色区域。灰色区域是定义在正例与负例区域之间的地方。它不应该被看作正例或者负例，并且它的损失权重应该被设为0。对每个在输出坐标空间中的非正例标注像素，当且仅当存在在个像素长度之内任意带正例标注的像素，它的无视标志才被设为1。

困难负例挖掘。对在SVM中的困难负例挖掘过程的分析，通过搜索预测不好的样本而不是随机样本，我们使得学习更有效。在负例挖掘之后，预测不好的样本会有很大的几率被选中，于是在这些样本上的梯度下降学习引向了更鲁棒的预测同时还有更少的噪音。具体地，负例挖掘可以通过在线引导被有效地实施。在向前传播过程中，我们对输出像素损失(7‑1)进行倒序排序，并且对前1%分配为困难负例。在所有实验中，我们保留了所有正例标注像素（样本）并且正例与负例之比为1:1。在所有负例样本中，他们的一半是从困难负例样本中被抽样，并且剩下的一半会随机地从非困难负例中抽样。为了方便，我们对那些在迷你批中的像素（样本）设标志。

带遮盖的损失。现在我们可以为每个样本定义遮盖为一个关于上述提到的标志的函数：

|  |  |  |
| --- | --- | --- |
|  |  | (‑) |

然后如果我们合并分类(7‑1)和带遮罩包围盒回归(7‑2)损失，我们的全多任务损失可以被表示成，

|  |  |  |
| --- | --- | --- |
|  |  | (‑) |

其中θ是网络中参数的集合，并且艾弗森括号函数只有当标注值得分为正时才会被激活。明显的是对负例样本（背景）的包围盒回归损失应该被无视，因为他们并没有任何记号。分类和回归任务之间的平衡被参数所控制。在我们的实验中，我们通过除标准物体高度，也就是在标注图中50/4，对回归目标进行归一化，并且在这种归一化的所有实验中，表现良好。

其他实现细节。训练中，当一个输入批在特定尺寸下在正中间包含一个居中的物体，那它会被视为正例批。这些批的正例样本周围仅仅包含负例样本。为了在整个数据集中完全探索负例样本，我们也从训练图像中随机剪切了随机尺寸的子图，并把它们缩放到相同的尺寸并把它们送进网络。我们把这种批叫做“随机批”，并且训练时“正例批”与“随机批”之间的比例为1:1。还有，为了更好地增加我们模型的鲁棒性，我们也在送进网络前对每个批进行随机抖动。具体地，我们应用了左-右翻转、位移变换（在25像素范围内）以及尺度变形（从0.8到1.25）。

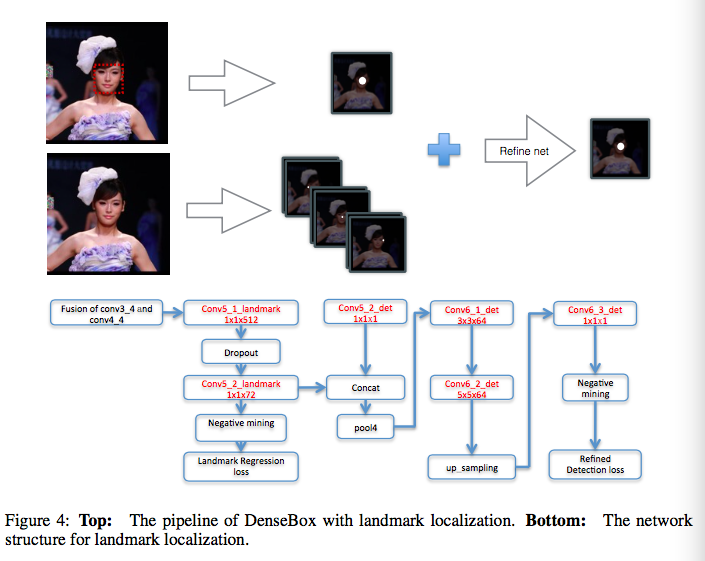
我们在训练中使用迷你批SGD并且批大小被设为10。损失以及输出梯度必须被有贡献像素的数量所缩放，使得损失和输出梯度都能在多任务学习中变得可比较。全局学习率一开始为0.001，并在每10万次迭代中被参数为10地衰减。我们保持默认动量项权重为0.9以及权重衰减系数为0.0005。

图 4

A3.4 改善标志定位

在这一部分，我们展示了仅仅通过累加几个全卷积网络结构中的网络层，标志定位就可以在稠密盒中被实现。更多地，我们可以通过融合标志热力图和人脸分值图来改善检测结果。如图 4所示，我们引入了另一个兄妹分值输出给标志定位。假设有N个标志点，则标志定位分支输出N个对应的图，上边每个像素点代表那个位置是一个标志点的置信度分值。表面上看，标注值图在这个任务上被使用和检测的标注值很相似。对一个标志实例，标志k的第i个实例，它的标注值是一个位于输出坐标空间中第k张响应图对应位置的正例标注区域。注意半径应该相对地小（比如）以避免精度的损失。与分类任务很相似，标志定位损失被定义成一个预测值与标注的L2损失，并且我们仍然应用负例挖掘且无视之前章节讨论过的区域。

最后的输出改善分支，得到分类分值图以及标志定位图作为输入，目标是改善分类结果。一个正确的方案应该是使用高层空间模型去学习标志置信度以及包围盒分值的限制，以进一步增强检测的表现。Tompson等提出的一个类MRF模型使用修改过的卷积（SoftPlus卷积）以及非负输出去连接每一个部位的空间信息分布。不管怎样，他们的模型也包含了和层级，使得模型非常难以训练。在我们的实现中，我们使用带ReLU激活的卷积操作去近似空间模型。如果我们把改善检测损失记为，几乎与之前提到但预测图是来自改善分支的分类损失中的一样，完整的损失变成了，

|  |  |  |
| --- | --- | --- |
|  |  | (‑) |

其中和控制三个任务之间的平衡。在我们的实验中，他们被分别分配为1和0.5。

A3.5 对比

稠密和的闪光点在于它把物体检测建构为一个回归问题，并且提供了一种端到端的识别框架。例如YOLO和Faster R-CNN的几个最近的工作都把区域提议生成和分类绑定在一起。这里我们比较了稠密和和其他相关检测系统，指出其中的关键相似点和不同点。

传统基于神经网络的人脸检测器。那些使用在最近取得突破性结果的图片分类卷积神经网络之前的神经网络的人脸检测系统，使用的是基于神经网络的人脸检测器。在人脸检测中使用神经网络已经有了很长的历史，可以追溯到1990年代的早期工作。Rowley等训练只在特定尺寸的人脸上可以被激活的基于神经网络的检测器，并且在带移动窗的图片金字塔中应用检测器。我们的稠密和与他们在检测流程上非常相似，除了我们使用了现代卷积神经网络作为检测器。因此稠密和可以被同样地称为“现代基于神经网络的检测器”。

OverFeat。被Sermanet等设计的OverFeat可能是在深度卷积神经网络在图片分类上成功应用后，第一个训练卷积神经网络去同时完成分类和定位的工作。它也在测试时应用全卷积神经网络，一个等价但更加有效得多的实现移动窗检测的方式。不管怎样，它仍然在训练时分离了分类和定位，并且需要复杂的后处理去处理检测结果。我们的方法和OverFeat非常相似但是一个多任务合并训练的端到端检测网络。

深度稠密人脸检测器（DDFD）。被Farfade等提出的DDFD是一个基于卷积神经网络的人脸检测系统。它声称在人脸检测任务上有超过R-CNN的表现，因为R-CNN中的区域生成可能会丢失一些人脸区域。尽管DDFD是一个完整的检测流程，但DDFD并不是一个端到端框架因为它把分类可能性预测和包围盒定位分离成了两个任务和两个层级。我们的稠密盒可以为了检测直接优化，并且可以通过引入标志信息被轻易地提高。

Faster R-CNN。Faster R-CNN仍然使用区域提议去在一张图片中寻找物体。不像它的前任变数，在Faster R-CNN中的区域提议是被区域提议网络（RPN）在第二级中与分类共享卷积特征计算生成的。RPN与我们的方法稠密盒共享了许多相似点。不管怎样，RPN需要提前定义锚点但我们的方法不需要。RPN是在多尺度物体上被训练的，但本论文展示的稠密盒是在经过抖动增广的单尺度上训练的，这意味着我们的方法需要在多尺度上验证特征。更多地，RPN与稠密盒的训练方式是非常不同的。

多重盒。多重盒训练了一个卷积神经网络去生成提议而不是通过选择搜索。多重盒和稠密盒都被训练来在一张图片中预测包围盒，它们以不同的方式生成包围盒。多重盒方法生成了800个具有非平移不变形的锚点，而我们的稠密盒像RPN一样直接输出平移不变的包围盒。因为输出图的向下采样系数为4，稠密盒会稠密地每4个像素点生成一个带分值的包围盒。

YOLO。Redmon等提出了一个统一物体检测流程，叫YOLO。稠密盒和YOLO都可以被从图片端到端地训练，但在输出层的模型设计上有所不同。YOLO系统把大小为的图片作为输入，并输出的网格，每张图片上只有49个包围盒。我们的稠密盒使用向上采样层去保持相对高分辨率的输出，在我们的模型中有一个缩放系数为4的向下采样。这让我们的网络能够检测非常小的物体以及高度重合的物体，这是YOLO不可能能够处理的。

A4 实验

在这一节中，我们阐述了在多属性标注人脸（MALF）数据集以及KITTI车辆检测任务上稠密盒的表现。不管有没有标志标注的帮助，我们都在那些任务上校验了我们的方法，显示带标志定位的多任务学习能显著地提高表现。我们比较了当前世界最好的系统与我们的方法，显示我们的方法在物体检测任务上实现了有竞争力的结果。注意到我们并没有直接比较我们的稠密盒与原始R-CNN的表现，但是我们强调了声称使用R-CNN或者其他已经被与R-CNN比较过的方法的方法的表现。

A4.1 MALF识别任务

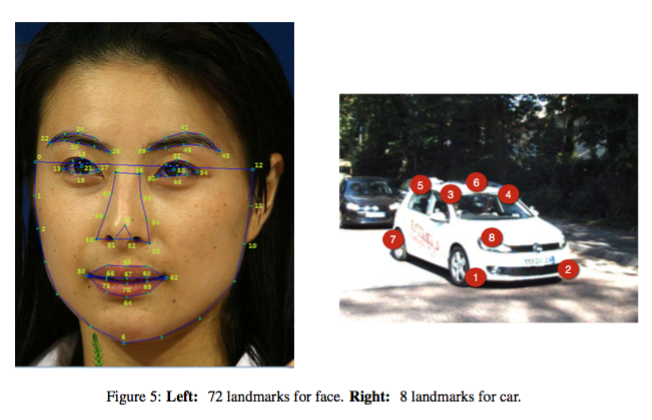
MALF检测测试数据集包含从互联网中搜集过来的5000张图片。不像被广泛使用的FDDB人脸检测数据集——这是从新闻图片和海报正面中搜集过来的额——MALF中的人脸图片有更大的变化度，使得它比FDDB更接近真实世界的应用。

图 5

训练与测试。我们在包含81024张有72个标志点的人脸31337张从互联网搜集到的图片上，训练了在第三节中描述过的两个模型，见图 5。一个模型至使用了包围盒信息，而另一个模型使用了包围盒以及标志信息，以进行比较。他们都被在ImageNet上提前训练好的VGG 19模型所初始化。在训练中人脸被大致缩放到高度为50像素，而缩放抖动方位为[0.8,1.25]，与在3.3节中描述的一样。在测试时，我们首先选择性地向下采样图片以对于每张图片最长的图片边不会超过800像素点。然后我们在每张图片上以多个尺度测试我们的模型。测试尺度以的步长从开始到结束。这种设置使得我们的模型能够检测高度从20像素点到400像素点的人脸。在人脸检测上的非极大值抑制IOU阈值被设为0.5。在这种配置下，在MALF数据集上一张英伟达K40显卡处理一张图片需要几秒时间。结果。我们描述了在MALF数据集上三种版本的稠密盒的结果。“无标志点稠密盒”显示了训练中没有标志点的稠密盒。“稠密盒标志点”是引入标志定位的模型，并且“稠密盒组装”是10个有从不同批迭代得来的标志点的稠密盒的结果。如图 6所示，标志定位给出了一个在人脸检测上显著的表现提高。我们也注意到使用不同批迭代训练的模型仍然有很高的变化度，因为另外的显著提高已经在模型组装中被看见了。然后我们比较了在MALF上我们最好的模型与其他世界最好的方法。令人意外的，我们的模型实现了最好的表现，平均召回率在87.26%，几乎比DDFD表现好了10%，说明我们在人脸检测任务上比R-CNN有更好的表现。

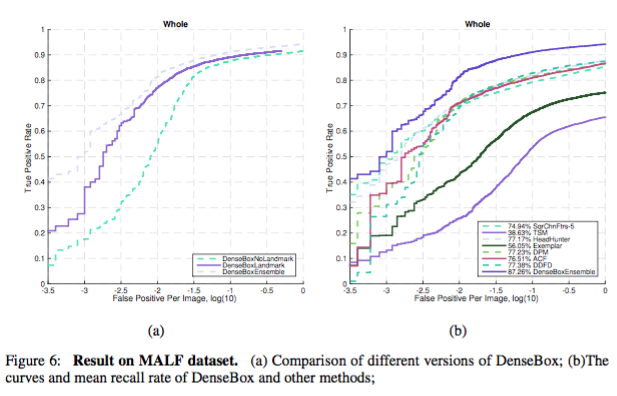
A4.2 KITTI车辆检测任务

图 6

KITTI物体检测数据集有7481张训练图片以及7518张测试图片。在训练中物体总数达到51867个，其中车辆仅仅占了28742个。KITTI车辆检测任务的主要难点在很大量的汽车是小尺寸的（高度小于40像素点）并且被遮挡的。为了克服这个难点，前人工作需要仔细的旁路及遮盖建模。

训练与测试。与人脸检测任务一样，我们在KITTI物体检测训练集上训练了两个网络（一个没有标志点，另一个有标志点）。因为KITTI没有为车辆提供标志点，我们选择性地为大车辆（高度大于50像素点）标注了8个标志点。车辆的标志点如图 5所示，我们最后标注了7790辆汽车，大概占车辆总数的27%。测试程序与在人脸检测中一样，除了我们并没有对车辆图片进行向下采样。KITTI车辆检测任务的评价方式与普适物体检测并不相同。KITTI的真正例包围盒需要有70%的重叠面积，而其他像人脸检测的任务只需要50%的重叠面积。这个严格的边界条件要求更高的车辆定位准确度。在KITTI上，我们设非极大值抑制IOU阈值为0.75。

结果。表 1显示了稠密盒以及其他方法的结果。我们可以看见部分标注标志信息（27%）仍然可以提高检测表现。平均上，有标志定位的模型轻微地比没有标志点的模型在平均精确度上提高了0.9%。在人脸检测上，表现的提高并没有那么好。原因可能是标志点信息并不充分。不充分性体现在总量（车辆上27%然而人脸是100%）和质量（车辆上8个标志点而人脸上是74）上。与其他方法相比，稠密盒仍然实现了有竞争力的结果。稠密盒以较大的差距打败了如Regionlets和spCov等传统检测系统。我们在经济适用车辆上的平均精确度为85.74%，比使用了在ImageNet上提前训练的GoogLeNet的R-CNN框架的DeepInsight轻微地好。我们的模型直到一个标题为“NIPS ID 331”匿名提交前已经4个月排名第一了，在训练和测试中使用了立体信息。最近一个叫“DJML”的方法超越了其他所有方法。

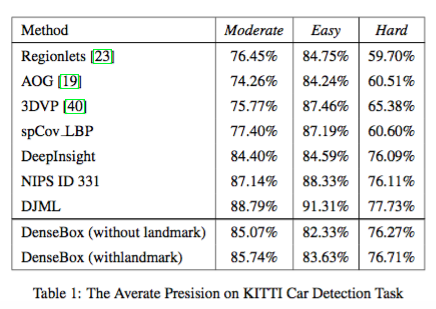
A5 总结

表 1

我们已经展示了稠密盒——一个针对检测的统一端到端检测流程。它的表现可以通过纳入标志信息被轻易地加速。我们也分析了我们的方法以及其他相关的物体检测系统，强调了其中稠密盒的不同以及贡献。稠密盒在人脸检测和车辆检测任务上都展现了令人印象深刻的表现，表明它对那些区域生成可能失败的情况的高适应性。稠密盒主要的问题是速度。论文中展示的原始的稠密盒需要几秒去处理一张图片。但是这已经在我们的后一个版本中被处理。我们会展示另一篇描述一个在KITTI和人脸检测的实时检测系统论文，叫稠密盒2。