Deep Sort

关于 Deep Sort 的一些理解

猫弟
计算机视觉/摄影师/音乐爱好者

关注他

最近学习了多目标跟踪算法--Deepsort，打个总结。

我认为是目前工程上效果还不错的跟踪算法。首先，虽然算法类似于 two stages的结构，没法完成端到端的训练，但是可以让使用者更好的针对跟踪效果分别对检测器或者跟踪器做优化。其次，基本可以达到real time的程度，速度可以根据ReID的尺度进行调节，而且精度也表现不错，可作为比赛的baseline使用。最终，deepsort借鉴了sort的思想，加以完善，易于初学者学习与使用。

网上各种博客解释较多，但大多数都仅对论文翻译或代码的复制粘贴，对算法的理解帮助较小。下面给出我的学习轨迹和自己的思考，望各位针对不足予以指正。

论文地址：https://arxiv.org/pdf/1703.07402.pdf

代码地址：https://github.com/nwojke/deep_sort

推荐博客①：

https://blog.csdn.net/sgfmby1994/article/details/98517210
blog.csdn.net

博主用具体实例对代码流程进行讲解，且print了代码中很多变量的具体形式，对于理解很有帮助。

推荐博客②：

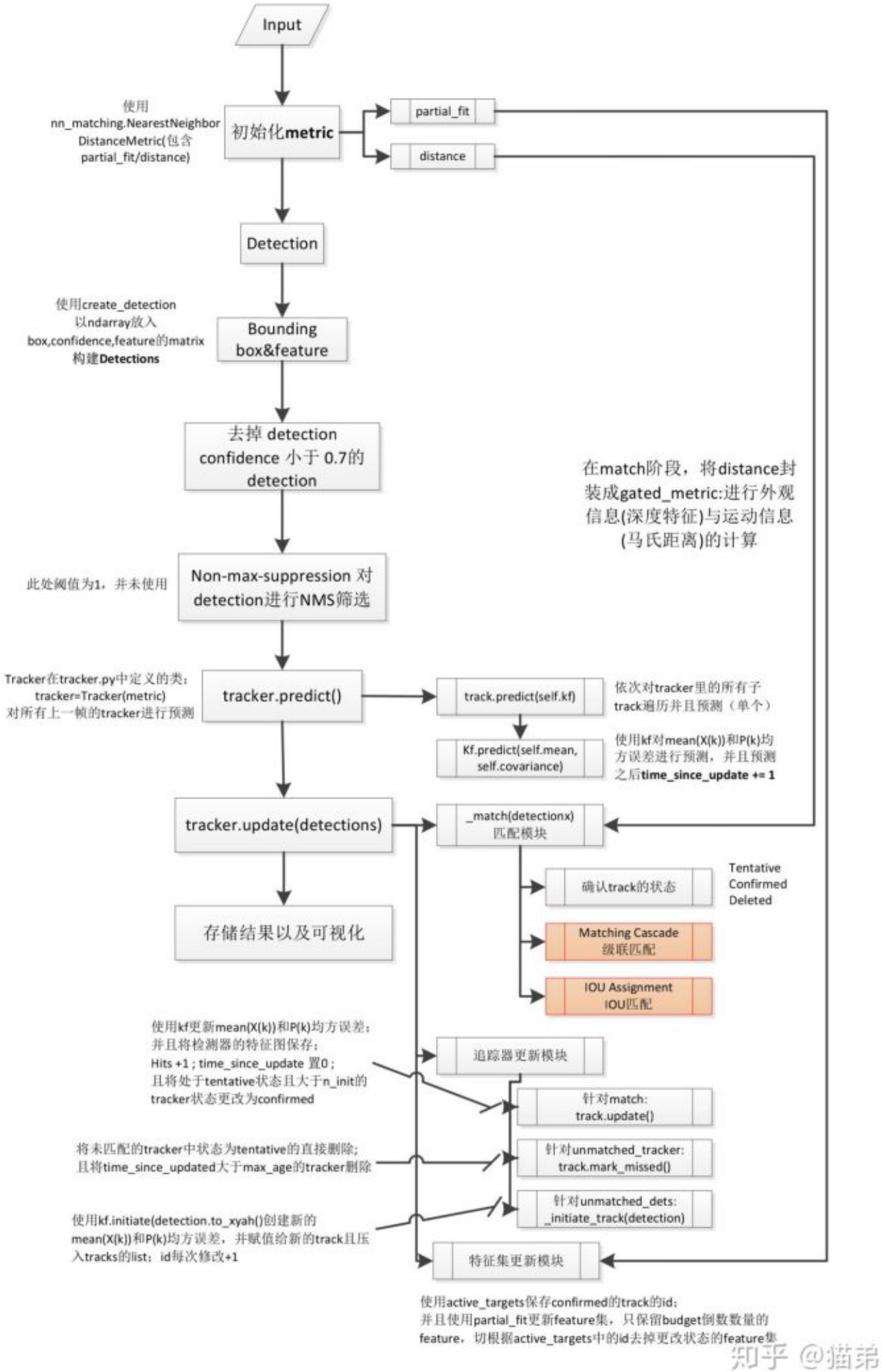
https://blog.csdn.net/zjc910997316/article/details/83721573
blog.csdn.net

博主的代码流程图比较清晰。我借鉴了作者Matching Cascade与IOU Assignment 部分的流程图，并对主要代码架构部分的流程图进行了完善。

下面是我对deepsort的一些理解：

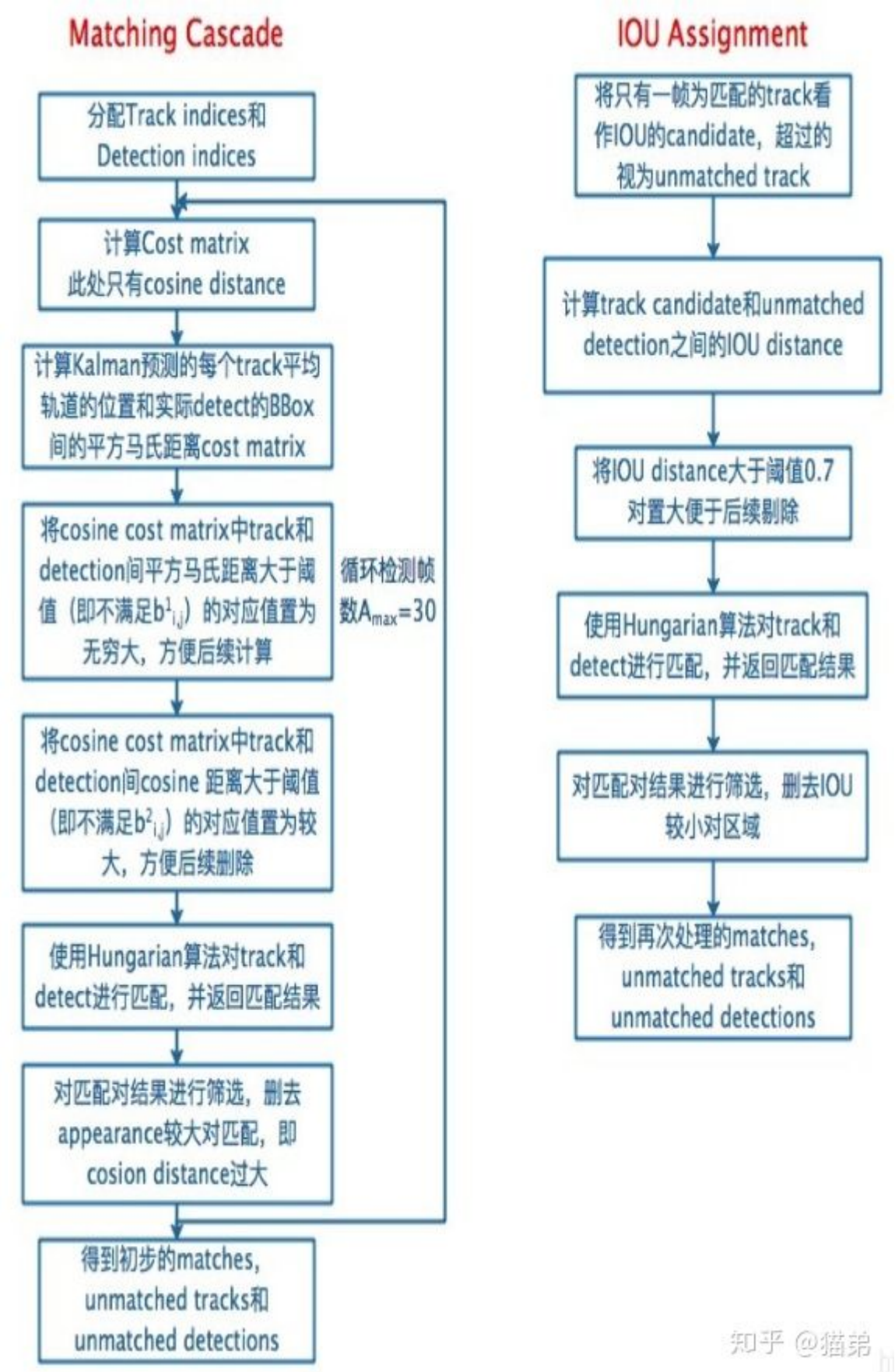
代码部分:

首先给出完善的代码流程图（已添加子函数名称以及注释）：



deepsort代码流程图（总）

下面是推荐博客②中的级联匹配模块与IOU匹配模块的流程图：



deepsort代码流程图（分）

代码部分的细节我不再赘述，博客①中叙述足够详尽，各位看官可以移步前往。对于代码模块较为繁杂的情况，针对代码思路进行一个梳理，可以清楚get到作者设计算法的思路。

原理部分：

DeepSort用来跟踪的思路较为明朗，也是目前主要流行的跟踪思路：**detection + track** 。 所以代码也以检测结果为**输入**：bounding box、confidence、feature 。conf主要用于进行一部分的检测框的筛选；bounding box与feature（ReID）用于后面与跟踪器的match计算；首先是预测模块，会对跟踪器使用卡尔曼滤波器进行预测。作者在这里使用的是卡尔曼滤波器的匀速运动和线性观测模型（**意味着只有四个量且在初始化时会使用检测器进行恒值初始化**）。其次是更新模块，其中包括匹配，追踪器更新与特征集更新。在更新模块的部分，根本的方法还是使用**IOU来进行匈牙利算法的匹配**，只不过作者对于数据顺序有一些自己的处理：

1.**使用级联匹配算法**：针对每一个检测器都会分配一个跟踪器，每个跟踪器会设定一个time_since_update参数。如果跟踪器完成匹配并进行更新，那么参数会重置为0，否则就会+1。实际上，**级联匹配换句话说就是不同优先级的匹配。**在级联匹配中，会根据这个参数来对跟踪器**分先后顺序**，参数小的先来匹配，参数大的后匹配。也就是给上一帧最先匹配的跟踪器高的优先权，给好几帧都没匹配上的跟踪器降低优先权（慢慢放弃）。至于使用级联匹配的目的，我引用一下博客②里的解释：

> 当一个目标长时间被遮挡之后，kalman滤波预测的不确定性就会大大增加，状态空间内的可观察性就会大大降低。假如此时两个追踪器竞争同一个检测结果的匹配权，往往遮挡时间较长的那条轨迹的马氏距离更小，使得检测结果更可能和遮挡时间较长的那条轨迹相关联，这种不理想的效果往往会破坏追踪的持续性。这么理解吧，假设本来协方差矩阵是一个正态分布，那么连续的预测不更新就会导致这个正态分布的方差越来越大，那么离均值欧氏距离远的点可能和之前分布中离得较近的点获得同样的马氏距离值。所以，作者使用了级联匹配来对更加频繁出现的目标赋予优先权。**当然同样也有弊端： 可能导致一些新产生的轨迹被连接到了一些旧的轨迹上。但这种情况较少。**

2.**添加马氏距离与余弦距离：实际上是针对运动信息与外观信息的计算。**两个名词听着较为陌生，而实际上换句话解释，马氏距离就是“加强版的欧氏距离”。它实际上是规避了欧氏距离中对于数据特征方差不同的风险，在计算中添加了协方差矩阵，**其目的就是进行方差归一化，从而使所谓的“距离”更加符合数据特征以及实际意义。**马氏距离是对于差异度的衡量中，的一种**距离度量**方式，而不同于马氏距离，余弦距离则是一种**相似度度量**方式。前者是针对于**位置**进行区分，而后者则是针对于**方向**。换句话说，我们使用余弦距离的时候，可以用来衡量不同个体在维度之间的差异，而一个个体中，维度与维度的差异我们却不好判断，此时我们可以使用马氏距离进行弥补，从而在整体上可以达到一个相对于全面的差异性衡量。而我们之所以要进行差异性衡量，**根本目的也是想比较检测器与跟踪器的相似程度**，优化度量方式，也可以更好地完成匹配。

3.**添加深度学习特征**：这一部分也就是**ReID**的模块，也是deepsort的亮点之一。deepsort在对于sort的改进中加入了一个深度学习的特征提取网络，网络结构部分各位看官可以移步论文。作者将所有confirmed的**追踪器**（其中一个状态）每次完成匹配对应的detection的**feature map存储进一个list。**存储的数量作者使用budget超参数（100帧）进行限制（我认为如果实时性效果不好的话，可以调低这个参数加快速度）。从而我们在每次匹配之后都会更新这个feature map的list，比如去除掉一些已经出镜头的目标的特征集，保留最新的特征将老的特征pop掉等等。这个**特征集在进行余弦距离计算的时候将会发挥作用。**实际上，在当前帧，会计算第i个物体跟踪的所有Feature向量和第j个物体检测之间的最小余弦距离。

4.**IOU与匈牙利算法匹配：**这个方法是在sort中被提出的。又是比较陌生的名词，我接着“换句话说”。实际上匈牙利算法可以理解成“**尽量多**”的一种思路，比如说A检测器可以和a，c跟踪器完

成匹配（与a匹配置信度更高），但是B检测器只能和a跟踪器完成匹配。那在算法中，就会让A与c完成匹配，B与a完成匹配，而降低对于置信度的考虑。所以算法的根本目的并不是在于匹配的准不准，而是在于尽量多的匹配上，这也就是在**deepsort中作者添加级联匹配与马氏距离与余弦距离的根本目的**，因为仅仅使用匈牙利算法进行匹配特别容易造成ID switch，就是一个检测框id不停地进行更换，缺乏准确性与鲁棒性。那什么是匹配的置信度高呢，其实在这里，作者使用的是IOU进行衡量，计算检测器与跟踪器的IOU，将这个作为置信度的高低（比较粗糙）。

还有一些超参数，我也已经在代码流程的部分明确讲解了，这些超参数或进行阈值的作用，或进行循环次数的限定，不断帮助算法完成最优匹配与追踪器更新。之后，将未匹配的追踪器delete，将未匹配的检测器初始化，将**匹配的追踪器使用对应的检测器进行赋值，作为输出，进入下次循环。**

以上是我对DeepSort的一些理解与思考，如果有不正确或者不严谨的地方，欢迎大家予以指正。并且对所有我参考的博客的作者予以感谢。且欢迎各路大神带我这个小白一起打比赛&交流。笔芯！

其余参考博客：

1.blog.csdn.net/HaoBBNuan...

2.blog.csdn.net/cdknight_...

3.blog.csdn.net/lzhf1122/...

编辑于 2019-09-02