**基于深度回归网络在100FPS中学习跟踪**

David Held, Sebastian Thrun, Silvio Savarese

Department of Computer Science

Stanford University

{davheld,thrun,ssilvio}@cs.stanford.edu

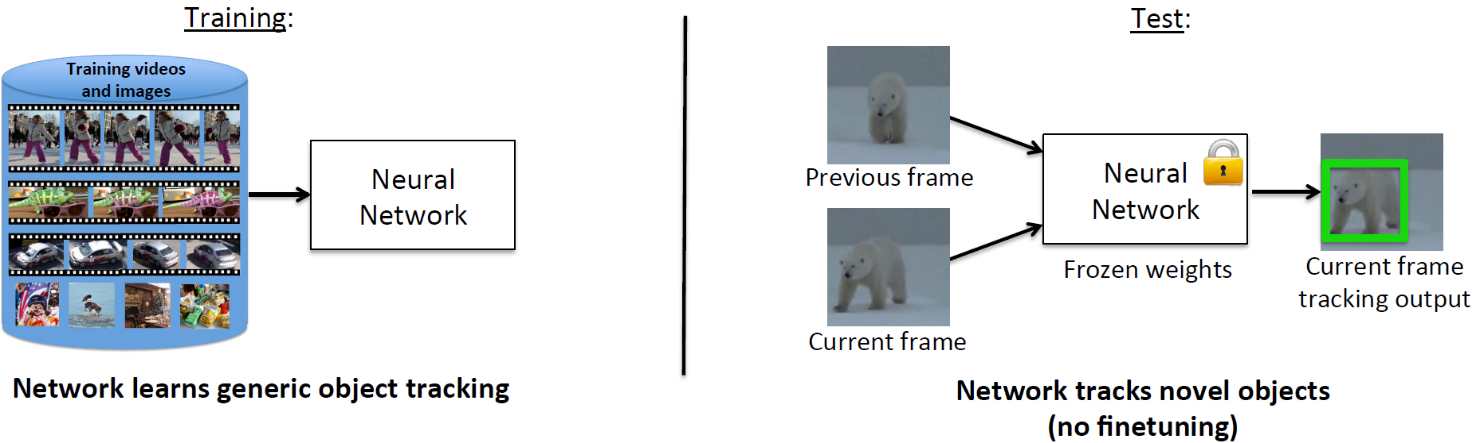
**摘 要**：机器学习技术经常被用在计算机上，因为他们有能力利用大量的培训数据提高性能。不幸的是，大多数通用的对象跟踪器仍然是在线开始训练，并没有从很多可以随时用于离线训练的视频中获益。我们建议一种神经网络离线训练方法，这种方法能在100fps、测试时间内跟踪新的对象。我们的跟踪器显着快于以前使用神经网络进行跟踪的方法，运行速度非常慢，对于实时应用来说不实用。我们的跟踪器使用一个简单的，不需要在线训练的前馈网络。跟踪器学习对象运动和表现之间的一般关系，并且可以用来追踪没有在训练集中出现的新对象。我们在一个标准的跟踪基准上测试我们的网络来演示我们的跟踪器的最先进的表现。进一步，随着我们为离线训练集添加更多的视频，跟踪器的表现也会提高。据我们所知，我们的跟踪器[[1]](#footnote-1)是第一个在100fps中学习跟踪通用对象的神经网络跟踪器。

**关键词**：跟踪；深度学习；神经网络；机器学习。

**1 介绍**

考虑到在视频的一帧中标记了一些感兴趣的对象，“单目标跟踪”是在随后的视频帧中将该对象定位，尽管是对象有运动，视点变化，照明变化或其他变化。单目标跟踪是许多系统的重要组成部分。 对于跟踪人的应用，机器人必须跟踪这个人在环境中的移动。对于自主驾驶，机器人必须跟踪动态障碍物，来预估他们估算哪里移动，并预测他们将来如何移动。

通用对象跟踪器（此跟踪器不专门针对特定类型的物体）完全从头开始在线训练（即在测试期间）[15,3,36,19]，而不进行离线训练。 这样的跟踪器性能不佳，因为他们无法利用大量的很容易获取的视频来改善他们的性能。离线训练视频能被用来教授跟踪器处理旋转，视点改变，亮度改变和其他复杂的挑战。



**图1 使用边界框标签的视频和图像的集合（但没有类信息），我们训练神经网络来跟踪通用对象。 在测试的时候，网络能够跟踪新的对象而不需要任何调整。 通过避免微调，我们的网络能够以100fps的速度跟踪**

在计算机视觉的许多其他领域，如图像分类，对象检测，分割或活动识别，机器学习允许视觉算法来从离线数据训练并了解世界[5,23,13,25,9,28]。在每种情况下，算法的性能随着训练图像集的迭代而提高。 这样的模型是从神经网络在大规模的数据中学习复杂功能的能力中获益。

在这项工作中，我们表明了学习通过观看世界上运动物体的离线视频来即时跟踪通用对象是可能的。 为了实现这个目标，我们介绍*GOTURN*，Generic Object Tracking Using Regression Networks使用回归网络跟踪通用对象。 我们训练神经网络以完全离线的方式进行跟踪。在测试的时候，当追踪新物体时，网络权重被冻结，不需要在线调整（如图1所示）。 通过培训过程中，跟踪器学习了以快速，稳健和准确的方式

2 Held, Thrun, Savarese

跟踪新的对象。

虽然使用神经网络跟踪的一些初步工作已经完成，但这些努力制造出了在实际使用中很慢的神经网络跟踪器。相比之下，以我们最好的结论来看，我们的跟踪器能够以100fps的速度跟踪物体，成为迄今为止最快的神经网络跟踪器。我们的实时速度由两个因素决定。首先，大多数以前的神经网络跟踪器是在线训练的[26,27,34,37,35,30,39,7,24];然而，训练神经网络是一个缓慢的过程，会导致跟踪缓慢。相比之下，我们的跟踪器是经过离线训练来学习外观和动作之间的一般关系，所以不需要在线训练。其次，大多数追踪器采取分类方法，分类许多图像补丁找到目标对象[26,27,37,30,39,24,33]。比较来说，我们的跟踪器使用基于回归的方法，只需要一个前馈通过网络直接回归到目标对象的位置。离线训练和一次回归的结合导致了一个比起之前方法显著的速度提升，并且允许我们以实时速度追踪目标。

GOTURN是第一个能以100 fps运行的通用对象神经网络跟踪器。 我们使用一个标准的跟踪基准来证明我们的跟踪器优于最先进的跟踪器。 我们的追踪器从一个标记过的训练视频和图像训练而来，但我们不需要任何级别的标签或关于正在跟踪的对象的类型的信息。 GOTURN成立一个新的跟踪框架，其中外观和动作的关系是以一种通用的离线方式学习的。我们的代码和其他经验可以在以下网址找到：

<http://davheld.github.io/GOTURN/GOTURN.html>

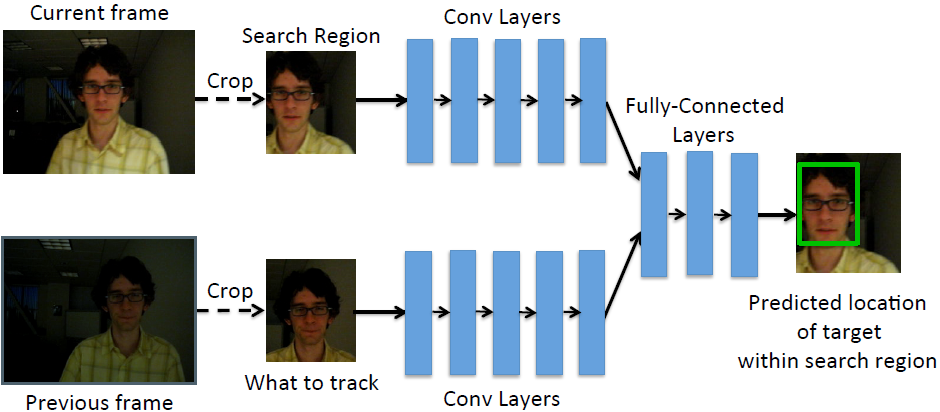
**2 相关的工作**

**在线跟踪培训。**用于通用对象跟踪的跟踪器通常是从视频的第一帧开始完全在线训练[15,3,36,19]。 一个典型的跟踪器将在目标对象的补丁附近采样，这被认为是“前景”[3]。还有一些距目标物体较远的区域也被采样，这些被认为是“背景”。然后这些补丁被用来训练一个前景背景分类器，这个分类器被用来给目标对象下一帧估测的新位置的补丁评分[36,19]。不幸的是，由于这些跟踪器完全在线培训，他们不能从很多可以随时用于离线训练的可能可以提升性能的视频中获益。

一些研究人员也试图使用神经网络以传统的在线训练框架进行跟踪[26,27,34,37,35,30,39,7,24,16]，显示了最先进的成果[30,7,21]。 不幸的是，神经网络的训练非常缓慢，并且如果需要在线训练，就会导致跟踪器在测试时也非常的慢。 这种从0.8 fps [26]到15 fps [37]的最高性能的神经网络跟踪器在GPU上只能以1 fps运行[30,7,21]。因此，这些跟踪器不适用于大多数实际应用。 因为我们跟踪器是以一种离线通用的方式进行训练的，不会有在线培训需求，使我们能够在100帧/秒跟踪。

**基于模型的跟踪器**。 一个单独的跟踪器类是基于模型的跟踪器，旨在跟踪特定类别的对象[12,1,11]。 比如，如果只对跟踪行人感兴趣，那么可以训练一个行人探测器。 在测试期间，这些检测可以使用时间信息连接在一起。 这些追踪器是离线训练的，但是它们是有限制的，因为他们只能跟踪特定类别的对象。 我们的跟踪器已经过通用方法离线训练，并且可用于在测试时间跟踪新类别的对象。

**其他神经网络跟踪框架。** 一个相关的研究领域是补丁匹配[14,38]，最近在[33]中用于追踪，以4 fps运行。 在这样的方法中，很多候选补丁都通过了网络，选择匹配得分最高的补丁作为跟踪输出。 相比之下，我们的网络只通过网络传递两个图像，网络直接回归到目标对象的边界框位置。得以避免给很多候选补丁评分的需求，我们能够以100fps的速度追踪。



**图2.我们的跟踪网络架构。 我们从当前帧的和前一帧的目标向网络输入一个搜索区域。 网络学习到比较这些裁剪以找到当前图像中的目标对象**

之前已经尝试使用神经网络在不同的其他方式跟踪，包括视觉注意力模型[4,29]。但是，这些方法在复杂的跟踪数据集的评估时不是其他最先进的跟踪器的对手。

**3 方法**

**3.1 方法概观**

在较高的层面上，我们将视频的帧提供给神经网络，网络能依次输出每帧中被跟踪物体的位置。 我们用视频序列和图像完全离线训练跟踪器。 通过我们的离线训练进程，我们的跟踪器学习外观和运动之间的一般关系，这可以在测试时用来跟踪新的对象而不需要在线训练。

**3.2输入/输出格式**

**跟踪什么。**如果视频中有多个对象，则为网络必须接收哪一个对象正在被跟踪的信息。为了达到这个目的，我们把目标对象的图像输入到网络中。 我们裁剪并缩放前一帧，以目标对象为中心，如图所示在图2中。这个输入允许我们的网络跟踪它没有见过的新物体; 网络将跟踪该作物中输入的任何对象。我们填充该作物以允许网络接收一些关于目标物体的周围环境的上下文信息。

更详细地说，假设在帧t-1，我们的跟踪器之前预测该目标位于以c =（,）为中心的宽度为w，高度为h的边界框中。 在t时刻，我们以（cx; cy）为中心的第t - 1帧作裁剪，其宽度和高度分别为k1 w和k1 h。这个裁剪告诉网络正在哪一个是正在被追踪的对象。 k1的值决定了网络接收多少前一帧的上下文信息。

**找哪里**。为了找到当前帧的目标对象，跟踪器应该知道这个对象之前是在什么位置。因为对象往往会很光滑的移动过空间，所以对象之前的位置会提供一个好的，关于网络应该在哪里找到对象。我们实现这个，是在基于对象前一帧的位置的当前帧选择一个搜索区域。我们用这个搜索区域来裁剪当前帧，并将这个裁剪输入我们的网络，正如图2所示。网络的目标是在搜索区域回归到目标对象的位置。

更详细地说，当前帧t的裁剪以c’ =（, ）为中心，其中c’是目标对象的预期平均位置。 我们设置c’ = c，相当于一个恒定位置的运动模型，虽然更复杂运动模型也可以使用。 当前帧的裁剪有一个宽度高度分别为k2 w和k2 h，其中w和h分别为前一帧的预测边界框中的宽度和高度，k2定义我们的搜索目标对象的半径。 在实践中，我们使用k1 = k2 = 2。

只要目标对象没有被遮挡，也不会移动太快，目标将定位于这个区域。 对于快速移动的对象可以增加搜索区域的大小，但代价是增加网络的复杂性。 另外，要处理长期的遮挡或大动作，我们的跟踪器可以结合另一种方法，比如一个在线训练过的物体探测器，如TLD框架[19]，或者一个视觉注意力模型[4,29,2]; 我们留下这个作为未来的工作。

**网络输出。** 网络相对于搜索区域输出对象在当前帧的坐标，。 网络的输出包括边界框的左上角和右下角的坐标。

**3.3 网络架构**

对于单目标跟踪，我们定义了一个新的图像比较跟踪架构，如图2所示（请注意，相关的“两帧”架构）也被用于其他任务[20,10]）。 在这个模型中，我们将目标对象以及搜索区域都输入进一系列卷积层。该这些卷积层的输出是捕捉了图像高层次表象的特征集。

这些卷积层的输出，被提供给一些数字完全连接的层。 完全连接层的作用是比较从目标对象到当前帧中的特征来目标对象已移动到了哪里。 在这些帧之间，对象可能经历过翻转，旋转，照明改变，遮挡或变形。该由完全连接层学习到的功能因此是一个复杂的特征对照功能，这是当输出被追踪对象相关运动时，通过许多例子学习来变得更强大以应对多变的因素的功能。

更详细地说，我们的模型中的卷积层是从CaffeNet结构的五卷积层的第一层开始[17,23]。 我们将这些卷积层的输出串联在一起（即池5的特征）输出到单个矢量中。这个向量被输入到3个完全连接的层，每层有4096个节点。 最后，我们将最后一个完全连接层连接到一个包含4个节点代表输出边界框的输出层。 我们把输出提高10倍，使用我们的验证集（与我们所有的超参数一样）来选择。 网络超参数取自CaffeNet的默认值，以及在每个完全连接的层之间，我们如CaffeNet 里一样非线性的使用压缩和的ReLU。我们的神经网络是使用Caffe实现的[17]。

**3.4 跟踪**

在测试期间，我们从第一帧开始用一个实际边界框来初始化跟踪器，就像单一目标跟踪的标准做法一样。 在随后的每一个帧t，我们将帧t-1和帧t的裁剪输入到网络中（如3.2节所述）来预测对象在帧t中的位置。 我们继续重新裁剪，并将帧对提供给我们的网络剩余的视频，我们的网络将贯穿整个视频序列跟踪目标对象的移动。

**4 训练**

我们用视频和静止图像的组合来训练我们的网络。 培训程序如下所述。 在这两种情况下，我们都用预测的边界框和实际边界框之间的L1损失来训练网络。

**4.1从视频和图像训练**

我们的训练集由一系列视频组成，其中包含一部分帧在每个视频都标有一些对象的位置。 对于训练集中的每对连续帧，我们按3.2节所述裁剪。在训练期间，我们将这一对帧送入网络并尝试来预测对象如何从第一帧移动到第二帧（如图3所示）。 我们也用我们的运动模型来增加这些训练的例子，如4.2节所述。

我们的训练程序也可以利用每个都标有一个对象的位置的一组静止图像。 这套训练图像教导我们的网络跟踪更多元化的对象并防止和我们的培训视频中的对象过度拟合。 从图像训练我们的跟踪器，我们根据我们的运动模型采取图像的随机裁剪（见第4.2节）。 在这两个裁剪之中，目标对象已经经历了明显的翻转和规模的变化，如图4所示。我们认为这些裁剪就好像他们被采取来自视频的不同帧。 虽然这些裁剪比我们其他训练视频中找到的运动的变化少，但是这些图像仍然对我们的网络追踪多变的不同对象有作用。

**图3.培训视频的例子。网络的目标是预测在第一行视频帧被转移到下一行之后，中心所示的目标对象的位置。真值框被标记为绿色。**

**图4.训练图像的例子。网络的目标是预测在第一行裁剪视频帧被移动到如下一行的样子之后的目标对象位置。 地面真值边框被标记为绿色**

**4.2 Learning Motion Smoothness**

现实世界中的物体倾向于在空间中平稳地移动。 给一个模糊不清的图像，目标对象的位置不确定，一个跟踪器应该预测目标对象的位置在它上次被观测到的位置的附近。 这在包含多个相似对象的视频中尤为重要，例如相同类型的多个水果。 因此我们希望教我们的网络，其他的都是平等的，比起大的动作更喜欢小的动作。

为了体现运动平滑的思想，我们模拟了当前帧中边界框的中心（;）相对于在前一帧中边界框的中心（; ）的关系为

(1)

(2)

其中w和h分别是前一帧边界框的宽度和高度。 术语和是根据边界框的尺寸来捕捉它的位置改变的随机变量。在我们的训练集，我们发现物体这样以改变位置都能用平均值为0的拉普拉斯分布建模（详见附录）。这种分布对较大运动来说，较小的运动的可能性更高。

相似的，我们模型尺寸由以下公式改变γ

(3)

(4)

其中和是当前边界框的宽和高， 和是前一个边界框的宽度和高度。 术语是捕捉边界框尺寸变化的随机变量。 我们在我们的训练集中发现，由平均数为1的拉普拉斯分布建模。这样的分布使边界框有更高概率保持与前一帧的尺寸大小相同。

为了教导我们的网络更喜欢小动作，我们用从上述拉普拉斯分布中提取的随机裁剪来加强训练集如上所述（见图3和4为例）。 因为这些培训是从拉普拉斯分布中采样，小的运动比起大的运动将被更多的采样，因此我们的网络将会学习更喜欢小动作，其他都是平等的。 我们将展示这拉普拉斯剪裁程序与用于分类任务的标准相比提高了我们跟踪器的性能[23]。

拉普拉斯分布的尺度参数通过 = 1/ 5（对于边界框中心的运动）和 = 1/15（用于边界框大小的改变）来选择。 我们限制随机裁剪这样它必须包含每个维度中至少一半的目标对象。 我们也限制了尺寸的变化：（0.6,1.4），以避免过度拉伸或者收缩边界框对网络学习造成困难。

**4.3训练过程**

为了训练我们的网络，每个训练示例都是从视频或图像中交替采集的从图像。 当我们使用视频训练的例子时，我们随机选择一个视频，我们在这个视频中随机选择一对连续的帧。 然后我们根据第3.2节中描述的过程裁剪视频。另外如第4.2节所述，将当前帧的k3个随机裁剪结合起来增强数据集。 接下来，我们随机抽样一个图像，并且我们重复上述过程，其中随机裁剪创建人造“动作”（参见4.1和4.2节）。每次视频或图像被采样，新的随机裁剪即时被生产出来，这在我们的训练过程中创造多样化。 在我们的实验中，我们使用k3 = 10，我们使用50的批量大小。

我们网络中的卷积层是在ImageNet上预先训练好的[31,8]。由于我们有限的训练集大小，我们不会将这些图层调整以防止发生调整过度。 我们以学习率为1e-5来训练这个网络，并且还有其他超参数取自CaffeNet [17]的默认值。

**5 实验步骤**

**5.1 训练集**

如第4节所述，我们使用视频和静止图像的组合来训练我们的网络。我们的培训视频来自ALOV300 ++ [32]，一个314个视频序列的合集。我们删除了7个与我们的测试重叠的视频（详见附录），给我们留下了307视频用于培训。在这个数据集中，大约每个视频的第5帧都被标记了跟踪的某个对象的位置。这些视频一般很短，从几秒到几分钟不等。我们将这些视频分成251用于训练，56用于验证/超参数调整。培训一共包含了251,012个不同物体的13,082张图像，或者平均值为每个物体52帧。验证集由56个不同对象的2,795个不同的图像组成。选择我们的超参数后，我们使用我们的整个训练集模型重新训练模型（训练+验证）。删除7个重叠后视频中，训练和测试集中的视频之间没有重叠。

我们的培训程序还利用了一套用于培训的静止图像，如4.1节所述。 这些图像来自训练一套ImageNet Detection Challenge [31]，其中478,807个对象是用边界框标记过的。我们在训练时间内随机剪裁这些图像，如第4.2节所述，在两种随机裁剪之间创建明显的翻转或比例变化。 随机裁剪过程仅在以下情况下有用：被标记的对象不会覆盖整个图像; 因此，我们删除那些边界框在任一维度上至少占图像大小的66％的图片（使用我们的验证集选择）。 这给我们带来了134,821张图片中的239,283个注释。 这些图像有助于防止当教学我们的网络跟踪未出现在培训视频中的对象的时候出现的调整过度。

**5.2 测试集**

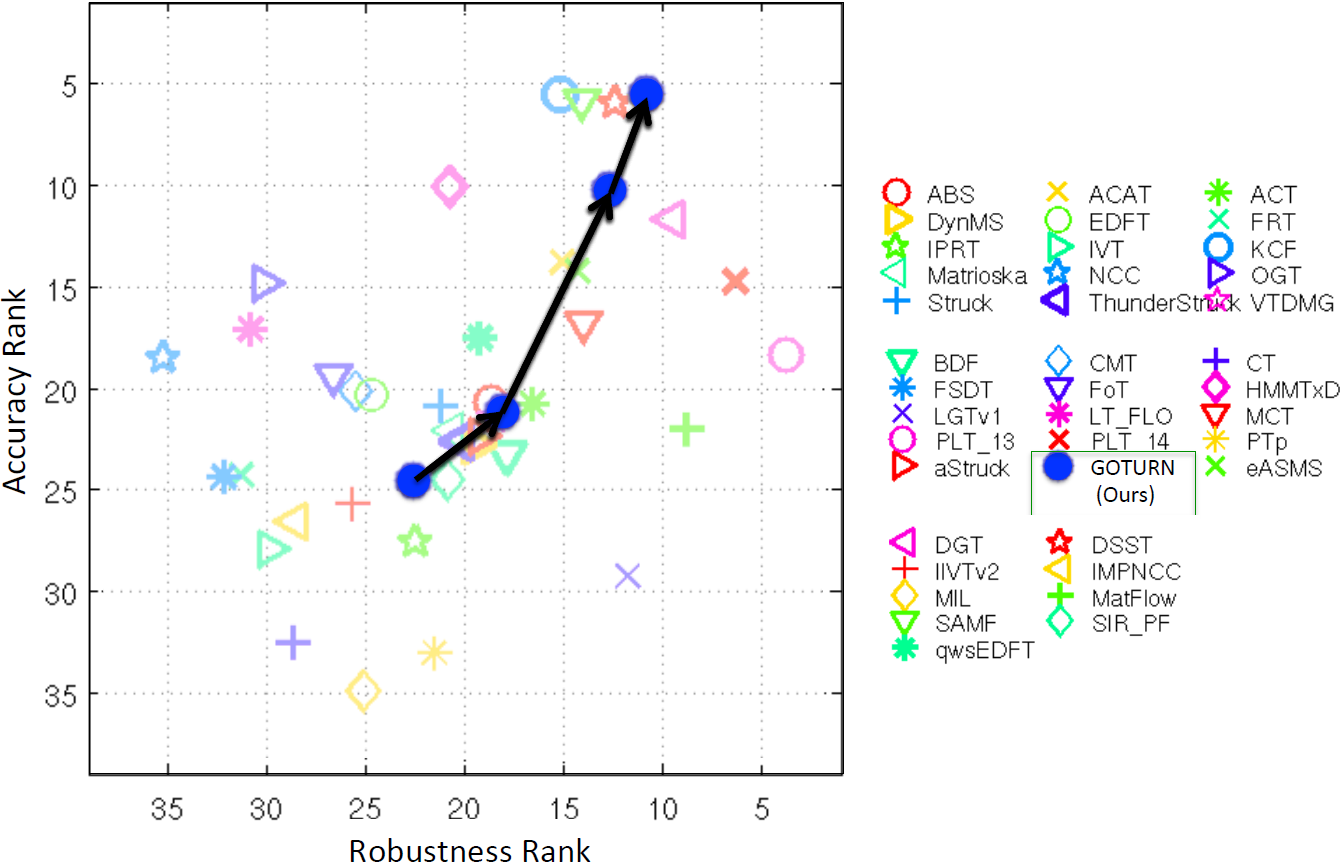
我们的测试集包含2014年VOT 2014追踪挑战赛的25个视频[22]。我们无法在VOT 2015挑战赛上测试我们的方法[21]，因为在那里在测试集和我们的训练集之间会有太多重叠。 然而，我们预计我们的方法的总体趋势仍然存在。

2014年VOT追踪挑战赛[22]是一个标准的追踪基准，这使我们能够将我们的跟踪器与各种先进的跟踪器进行比较。 跟踪器使用两个标准跟踪指标进行评估：准确性（A）和鲁棒性（R）[22,6]，范围从0到1.我们也计算精度误差（1-A），鲁棒性误差（1-R），以及总体误差1-（A + R）/ 2。

视频的每一帧都用许多属性进行注释：遮挡，照明变化，运动变化，尺寸变化和相机运动。 跟踪器的每个属性的准确性和鲁棒性也分别排名，并且排名然后跨属性平均得到最终的平均准确性和每个跟踪器的鲁棒性排名。 准确性和健壮性排名进行平均得到整体平均排名。

**6 结果**

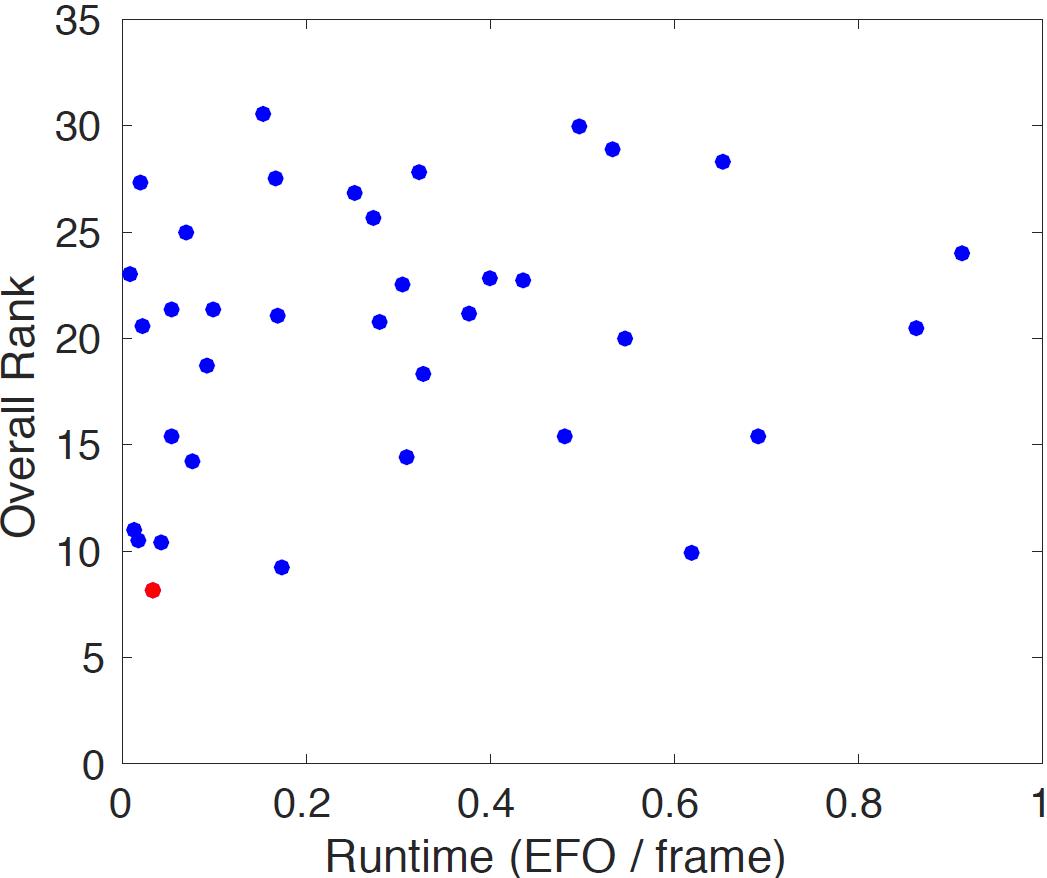
**6.1 整体表现**

我们的跟踪器的性能如图5所示，这表明我们的跟踪器具有很好的鲁棒性，在精度上接近于顶点。进一步，我们的整体排名（计算精度和鲁棒性的平均值）胜过以前在这个基准上所有的跟踪器。我们已经证明了离线训练对于提高跟踪性能的价值。此外，这些培训结果仅在307个短片中获得。图5以及附录中的分析表明，如果训练集通过标记更多视频来增大尺寸能取得更大的收获。定性结果，以及失败的案例，可以在项目页面：<http://davheld.github.io/>上找到；目前，跟踪器会因为阻塞或在训练集中过度条件对象而失败。

**图5。VOT 2014跟踪挑战的跟踪结果。我们跟踪器的性能用蓝色圆圈表示，在整体排名上比以前所有的方法都要好（平均精度和鲁棒性排名）。沿着黑线显示的点代表从14, 37, 157，和307个视频中训练，与每种情况下使用的训练图像数量是同样的。**

在一个cuDNN加速的NVIDIA GeForce GTX Titan X GPU上，我们的跟踪器以每帧6.05ms的速度运行（不包括在opencv加载每张图像用的1毫秒），或165fps。在一个GTX 680的GPU上，我们的跟踪器以每帧9.98ms的平均速度运行，或者说100fps。如果只有CPU是可用的，跟踪器运行为2.7fps。因为我们的跟踪器是能够执行所有的离线训练，在测试期间，在跟踪时只需要有一个单前馈通过网络，因此，跟踪器能以实时速度运行。

我们在图6用我们的跟踪器的速度和等级与其他38个提交到VOT 2014 Tracking Challenge的跟踪器进行比较 [ 22 ]，使用第5.2节所述的总体排名分。我们展示了在EFO单元的跟踪器的运行时间（等效过滤操作），标准化了跟踪器在[ 22 ]上面测试的硬件类型。图6演示了与其他38个基线相比，我们的追踪速度最快，而且在整体排名上超越的所有其他方法（以精度和鲁棒性排名平均数计算）。注意一些其他的跟踪器，如ThunderStruck [ 22 ]，还使用了GPU。为了更详细的分析速度作为准确性和鲁棒性的函数，见附录。



**图6。我们的跟踪器（红）与VOT 2014 Tracking Challenge的38种基线方法（蓝）相比的整体排名和运行时间图像。每个蓝色圆点代表的性能（最好看颜色）。精度和鲁棒性指标如附录所示**

由于我们的两个方面，我们的跟踪器能够实时跟踪对象模型：首先，我们学习通用的跟踪模型，所以没有在线培训需要。 神经网络的在线培训往往非常缓慢，可以预防实时性能。 在线训练的神经网络跟踪器范围从0.8fps [26]至15 fps [37]，最高性能追踪器以1 fps运行一个GPU [30,7,21]。 其次，大多数追踪者评估的样本数量很少并选择得分最高的作为跟踪输出[26,27,37,30,39,24,33]。采样方法的准确度受样品数量的限制，但是增加样本数量也增加了计算复杂度。另一方面，我们的跟踪器直接回退到输出边界框，所以GOTURN可以实现准确的跟踪，无需额外的计算成本，实现它以100 fps的速度跟踪物体。

1. Our tracker is available at http://davheld.github.io/GOTURN/GOTURN.html [↑](#footnote-ref-1)