基于滑动扫描框的高速物体的图像实时跟踪算法

郑远力 胡志坤

(中南大学物理与电子学院 长沙 410012)

摘 要 TLD(Tracking-Learning-Detection)算法是近期广受关注的单目标长期跟踪算法。该算法由跟踪器、检测器、学习器协同工作,解决了目前大部分跟踪算法在目标丢失后不能重新识别目标的问题。但是由于检测器的计算量很大,该算法的实时性较差。针对这个问题,提出了一种动态生成检测扫描框的方法。输入的图片先采用跟踪器的前后向金字塔光流法加以计算,估计出目标的大概位置。然后在该位置区域生成滑动扫描框来检测。该方法可以有效缩小检测区域,减少检测器的计算量。将改进后的算法与原始算法以及Camshift、CT(Compress Tracking)算法进行了比较实验。结果表明,对于实时摄像头监控,改进的算法比原始算法具备更快的跟踪速度和更高的跟踪准确率。对于固定的图像序列,改进的算法的精度和速度都超过Camshift、CT算法。

关键词 TLD,实时性,动态滑动扫描框

中图法分类号 TP391.4 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.10.058

Real-time Tracking Algorithm for Fast Target Based on Dynamical Scanning Boxes

ZHENG Yuan-li HU Zhi-kun

(School of Physics and Electronics, Central South University, Changsha 410012, China)

Abstract TLD algorithm is a long-term tracking algorithm for single target. And it has drawn wide attention recently. It can recognize target even the target that has been lost. However, its real-time performance is not good because of a large number of scanning boxes. We proposed a method which can generate scanning boxes dynamically. This method can reduce the calculation time efficiently and thus make TLD suit real-time situation. Experiment were conducted to compare the performance of the improved algorithm, original algorithm, Camshift and CT(Compress Tracking) algorithms. The experiment results show that when they are applied to real-time camera, the improved algorithm has faster tracking speed and higher accuracy. When they are applied on picture sequences, the speed and accuracy of the improved algorithm are better than other algorithms.

Keywords TLD, Real-time, Dynamical scanning boxes

1 研究背景

视频跟踪是计算机视觉最重要的应用之一。2011年,Zdenek Kalal[1]等人提出了TLD算法,将跟踪分成了3个子任务:跟踪器、学习器、检测器。它可以实现对目标的长期在线跟踪。学习器[2]设置了同时检测检测器正负结果的P-N experts,可以找出错误的正负图块作为新的目标和背景的特征,生成样本添加到Object Model中。跟踪器采用Pyramidal Lucas-Kanade[3]估计运动轨迹,再用Forward-Backward Error[4]计算每个轨迹的可靠性,最后再抛弃可靠性偏低的50%的跟踪点,得到最终的估计轨迹。检测器采用了3个分类器在当前帧中确定目标的位置,同时纠正跟踪器的错误。Integrator从跟踪器和检测器处取来各自输出的Bounding Box,取两者的加权整合作为最后的Bounding Box显示到图像上。

但是,由于TLD算法的跟踪器采用金字塔光流法,检测器对每一帧图片都要产生大量的子窗口进行全局扫描,加之学习器采集样本的方式,使得TLD的计算量巨大,运行速度非常缓慢。由于帧与帧之间间隔的时间太久,来不及学习目标的新的变化,因此很容易丢失一些高速运动、形状快速变化的目标,比如快速运动的汽车、摩托车。所以它在实时跟踪时准确率低,这使得它不适用于实时性要求高的场合。

针对这个问题,很多人提出了相应的改进算法。江博[5]用卡尔曼滤波方法来辅助TLD中的金字塔光流法跟踪器。高帆[6]等人用MIL跟踪器[10]替代TLD算法中的金字塔光流法跟踪器,减少了跟踪器部分的计算量,提高了目标被遮挡情况下的跟踪的鲁棒性。

他们都是对跟踪器的部分进行了改进优化,而没有对检测器进行改进。实际上,检测器部分的计算量是最大的,耗费的时间也是最多的。

因此,本文对检测器部分进行了改进,提出了一种动态生

到稿日期:2014-05-18 返修日期:2014-07-26 本文受湖南省自然科学基金委员会与株洲市政府自然科学联合基金(13JJ9038),湖南省科技计划(2013GK3005)资助。

郑远力(1991—),男,硕士生,主要研究方向为图像处理、嵌入式系统,E-mail:799858969@qq.com;胡志坤(1976—),男,博士,教授,主要研究方向为复杂系统、设备状态监测与故障诊断系统。

成扫描框的方法。该方法可以缩小检测器的检测区域，从而减小整体算法的计算量，提高实时性，能适合一些实时监控场合，比如车辆监控。为区别原始 TLD 算法，本文称改进的 TLD 算法为实时跟踪算法。

2 跟踪器

在上一帧的目标框中先用网格均匀撒点的方法采样 10 * 10=100 个特征点。然后用金字塔 Lucas-Kanade 光流法[8]进行跟踪。

以第 1 帧和第 2 帧的计算为例。第 1 帧上的采样特征点的坐标为$(x_{1,a1}, y_{1,a1})$，a1 代表在第 1 帧上面的采样点的序号。经过光流法后，得到第 2 帧中该物体的特征点坐标为$(x_{2,b2}, y_{2,b2})$。

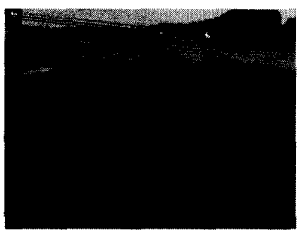

图 1 光流法示意图

然后对第 2 帧的估计点$(x_{2,b2}, y_{2,b2})$使用反向金字塔光流法计算其在第 1 帧上对应的点$(x_{1,c1}, y_{1,c1})$，然后比较所有$(x_{1,a1}, y_{1,a1})$与$(x_{1,c1}, y_{1,c1})$的几何距离 Forward-Backward Error[4]，抛弃其中大于中位值的 50%的数据，用剩下的点估计出目标在当前帧的位置。如果最终跟踪的点中有轨迹误差大于 10 的点，则直接认为跟踪失败。

假设保留下来的点的数量是 m，在第 1 帧上的点$(x_{1,d1}, y_{1,d1})$对应的在第 2 帧上面的是$(x_{2,d2}, y_{2,d2})$，则第 2 帧相对第 1 帧矩形框的中心位置横坐标变化为：

$$O_x=\frac{1}{m}\sum_{i=k_2}^{k_m}(x_{2,d2}-x_{1,d1}) \tag{1}$$

中心位置的纵坐标变换为：

$$O_y=\frac{1}{m}\sum_{i=k_2}^{k_m}(y_{2,d2}-y_{1,d1}) \tag{2}$$

假设第 1 帧的目标框在横坐标上的范围是$(x_{1,L}, x_{1,R})$，在纵坐标上的范围是$(y_{1,D}, y_{1,U})$，则第 1 帧的目标框的中心位置为：

$$p_{o1,x,y}=(\frac{x_{1,L}, x_{1,R}}{2}, \frac{y_{1,D}, y_{1,U}}{2}) \tag{3}$$

所以第 2 帧的目标框估计的中心位置为：

$$p_{o2,x,y}=(p_{o1,x}+O_x, p_{o1,y}+O_y) \tag{4}$$

计算第 2 帧中的$(x_{2,d2}, y_{2,d2})$两两之间的距离得到矩阵 D_2，计算第 1 帧中的$(x_{1,d1}, y_{1,d1})$两两之间的距离得到矩阵 D_1。D_2 中的每一个元素除以 D_1 中的对应的元素，得到行矩阵 $D_{2,1}$，对该矩阵取平均得到缩放倍数 S。

第 2 帧的目标框的估计宽度为：

$$w_2=s*(x_{1,R}-x_{1,L}) \tag{5}$$

第 2 帧的目标框的估计高度为：

$$h_2=s(y_{1,U}-y_{1,D}) \tag{6}$$

通过上述步骤，就可以估计预测出第二帧的目标的位置。将预测到的位置图像块归一化到 15×15，计算这个图像块与在线模型的保守相似度，如果保守相似度小于阈值，则认为跟踪无效。如果有效，则输出预测位置 TBB。

3 检测器

检测器计算当前图像的积分图，并对图像进行高斯模糊。然后遍历每个窗口，对每个窗口进行检测。对于每一帧图片，都要产生大量的扫描框进行上述的步骤。这里占用了大量的计算资源。而其中绝大多数子窗口并不包含目标的图像内容，因此造成了计算资源的极大浪费。

但实际情况是，在一般情况下跟踪器能跟踪出大致的位置，只需要检测器对它进行更精确的检测。而且，一般来说，目标大小一般都只占整张图片很小的一个比例，如果对每帧图片都进行全局扫描，会浪费大量的计算资源，降低运行速度。

为此，本文采用在跟踪器估计的目标框附近区域产生检测框的方式，来缩小 TLD 的目标检测区域，提高效率。

以第 1 帧为例，设整个视野的宽度为 w_l，高度为 h_l。第 1 帧目标窗口尺寸的宽度为 w_{bb}，高度为 h_{bb}，将其依次放大 λ^{-5}、λ^{-4}、λ^{-3}、λ^{-2}、λ^{-1}、λ^{0}、λ^{1}、λ^{2}、λ^{3}、λ^{4}、λ^{5}倍，形成共 11 种不同大小的矩形框。除第一个矩形框外，每个矩形框的大小是前一个矩形框的 λ 倍。则扫描过程的描述如下。

第 i 种矩形框的宽度计算公式为

$$w_i=w_{bb}*\lambda^i, i=-5,-4,\cdots,5 \tag{7}$$

第 i 种矩形框的高度计算公式为

$$h_i=h_{bb}*\lambda^i, i=-5,-4,\cdots,5 \tag{8}$$

如果将第 1 帧目标的窗口尺寸的长宽定义为 1，则各个扫描框的大小如图 2 所示。

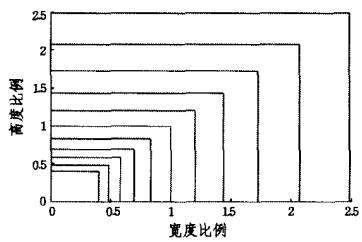

图 2 各个扫描框的相对大小

然后每一个矩形框逐行滑动，滑动的行步长是该矩形框宽度的 a 倍，每个矩形框滑动的列步长是该矩形框高度的 a 倍。

则第 i 种矩形框扫描的行步长计算公式为：

$$s_{w,i}=a*w_i \tag{9}$$

第 i 种矩形框扫描的列步长计算公式为：

$$s_{h,i}=a*h_i \tag{10}$$

所以，第 i 种矩形框在行上的扫描个数与该帧图片的宽度 w_l 有关，其计算公式为

$$n_{line,i}=\frac{w_l}{s_{w,i}} \tag{11}$$

每个矩形框依据行列步长逐行滑过整个图片。

第 i 个矩形框在第 j 个位置的左上角 x 坐标为

$$x_{tl,i,j}=1+s_{w,i}*[(j-1)\%n_{line,i}] \tag{12}$$

第 i 个矩形框在第 j 个位置的左上角的 y 坐标为

$$y_{tl,i,j}=1+s_{h,i}*[(j-1)\%n_{line,i}] \tag{13}$$

第 i 个矩形框在第 j 个位置的右下角的 x 坐标为

$$x_{br,i,j}=x_{tl,i,j}+w_i \tag{14}$$

第 i 个矩形框在第 j 个位置的右下角的 y 坐标为

$$y_{br,i,j}=y_{tl,i,j}+h_i \tag{15}$$

其中，j 的范围与该帧图片的高度 h_l 和宽度 w_l 有关，其计算公式为

$$1 \leqslant j \leqslant \frac{(h_l - h_i) * w_l + (w_l - w_i)}{s_{w,i}} \tag{16}$$

本文中称所有滑动点位置记录的总和为扫描窗口库。每帧的扫描框依据跟踪器估计的目标框 TBB 动态生成。将扫描窗口库中的各个框的位置与估计的目标框 TBB 进行重合率检测。设扫描窗口库中的各个框的左上角坐标为 (x_{bl}, y_{bl})，右下角坐标为 (x_{br}, y_{br})；TBB 的左上角坐标为 (x_{Bl}, y_{Bl}) 右下角坐标为 (x_{Br}, y_{Br})，则两个框如图 3 所示。

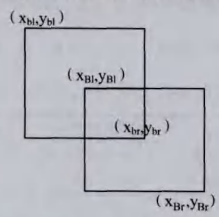

图 3　扫描窗口库中的框与 TBB 重合

重合率 $overlap$ 的计算式为两个框的重合部分面积除以未重合部分面积之和。

记录所有满足 $overlap > \gamma$ 的 i 和 j 作为在 TBB 附近滑动的扫描框的标签。图 4 中的矩形框即为动态生成的滑动扫描框。

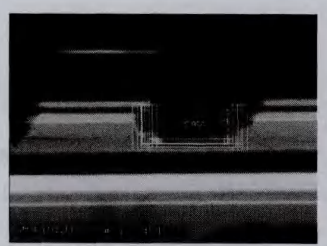

图 4　动态滑动扫描框

遍历这些筛选出来的扫描框，对每个框中的区域进行检测。先经过方差分类器检测，利用积分图计算每个待检测窗口的方差，若方差大于阈值，则认为其含有前景目标。然后采用集成分类器，先计算该图像片的特征值，再计算该特征值对应的后验概率累加值，若集合分类器的后验概率平均值大于阈值，则认为含有目标。

选出检测出目标的 100 个窗口，窗口是按照后验概率的大小降序排列后选择的前 100 个。

对选择后的窗口采用最近邻分类器进行检测。将窗口归一化到 15×15 大小。计算该窗口图像与在线模型的相关相似度和保守相似度。若相似度大于阈值，则认为含有目标。

4　学习器

在跟踪学习检测算法框架中，跟踪器和检测器两部分是并行独立的，因此之间的数据交互和自身的模型更新都是通过学习器实现的，如上所述，这部分被称为 P-N 学习器。由于图像中目标与背景并不是相互独立的，因此不仅目标含有有价值的信息，而且背景的信息也具有一定的价值。P-N 学习器中含有两个检错器 P-expert、N-expert，分别检测检测器中分类器输出中目标特征和背景特征的错误。将错误作为新的正负训练样本，用于更新检测器。

P-expert 的任务是找出检测器中被标示为背景特征的目标特征，即错误的负图块。P-expert 是从时间序列的角度查找错误的。它用跟踪器估计出当前帧中目标的位置，与检测器得出的正图块位置进行比较，如果跟踪器中有但检测器中没有，则该图块就是 P-expert 要找的错误。P-expert 会依据其生成新的正训练样本，添加到 Object Model 中，同时会更新检测器中的分类器。

N-expert 的工作方式与 P-expert 相似，只是 N-expert 要找被标示为目标特征的背景特征，即错误的正图块。它会从位置空间的角度，将跟踪器与检测器的结果作比较，找到错误的正图块，生成新的负训练样本，更新 Object Model。P-N 学习器框架图如图 5 所示。

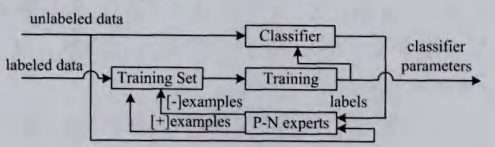

图 5　P-N 学习器框架图

5　实验

本文将实时跟踪算法与 TLD 以及 Camshift、CT 算法进行了比较。

5.1　实时跟踪算法与 TLD 的比较

为保证实验的公正，实验中使用的 car 图像序列和 motor 图像序列是 TLD 算法原作者的测试图像序列；使用的 TLD 的程序是直接从原作者的个人主页上下载的，这样可以保证程序中的参数都是原作者调好的最优结果，可让 TLD 发挥出最好的性能。在实验计算机上，实时跟踪算法的程序与 TLD 的程序都对图像序列进行跟踪，比较两者的运行速度与准确率。在两个图像序列上的实验过程的截图如图 6 和图 7 所示，实验结果记录在表 1 中。

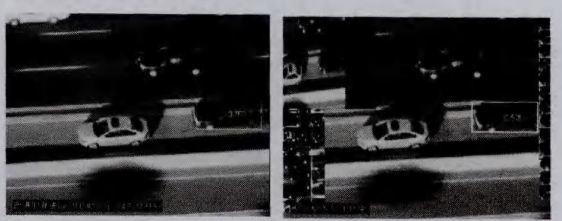

图 6　检测 car 图像序列(左为实时算法，右为 TLD)

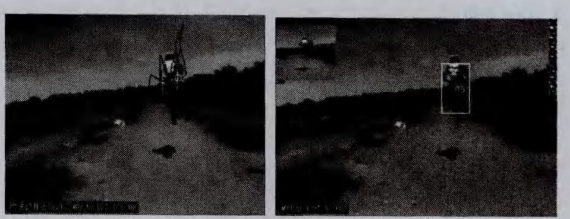

图 7　检测 motor 图像序列(左为实时算法，右为 TLD)

表 1　实时跟踪算法与 TLD 在图像序列上的比较

	实时跟踪算法	TLD
car 平均跟踪速度	15.6248 帧/秒	5.2545 帧/秒
car 序列准确率	97.49%	97.49%
motor 平均跟踪速度	12.2532 帧/秒	7.2459 帧/秒
motor 序列准确率	95.92%	95.92%

以上实验表明，对于固定的图像序列，实时跟踪算法能在较保证准确率不变的情况下大幅度提高对图像跟踪的速度，

其跟踪速度远远超过 TLD。实时跟踪算法能满足现实生活中对已经存储好的大量的图像序列跟踪的要求。

5.2 在实时摄像头上的跟踪比较

为了保证实时跟踪算法和 TLD 实时跟踪的景象是相同的，将 car 图像序列合成 avi 视频，在另外一台计算机 b 上播放，作为实时景象，实验计算机的摄像头实时拍摄计算机 b 上的播放视频的区域，来保证在两个算法分别运行时，摄像头采集到的实时图像是相同的。实时跟踪算法和 TLD 算法都运行在实验计算机上，实时采集摄像头的图像，进行跟踪。实验计算机 a 和 b 的硬件条件均为 Pentium(R) Dual-Core CPU T4200 @ 2.00GHz，2.93GB 的内存，显卡为 Intel GMA 4500M，摄像头的参数为 YUY2_320×240，屏幕大小为 14 英寸，分辨率为 1366×768。

第一组实验：在计算机 b 上播放 car 视频，实时跟踪算法运行在计算机 a 上实时采集摄像头图像进行跟踪检测，记录平均帧速、抓拍到的帧数、成功的帧数。第二组实验：在计算机 b 上播放同样的 car 视频，TLD 运行在实验计算机 a 上实时采集摄像头图像进行跟踪检测，记录平均帧速、抓拍到的帧数、成功的帧数。以上两组实验视频播放长度相同，程序停止时间相同。依次轮换各进行 4 次，对实验结果取平均值。实验过程截图如图 8 所示，实验结果如表 2 所列。

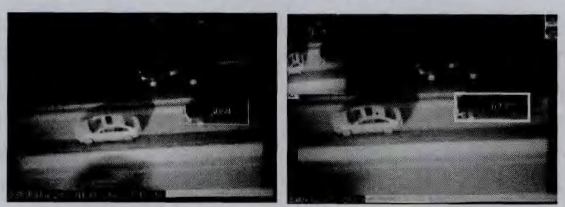

图 8 检测 car 实时视频(左为实时算法，右为 TLD)

表 2 实时跟踪算法与 TLD 在实时摄像头上的比较

	实时跟踪算法	TLD
平均跟踪速度	10.9134 帧/秒	5.1543 帧/秒
抓拍的全部帧数	147	74
成功跟踪的帧数	137	55
实时景象成功率	93.20%	74.32%

由实验数据可知，在实时跟踪目标的场合，实时跟踪算法无论在速度还是准确度上都大幅优于 TLD。因为实时跟踪算法具有速度上的优势，所以它可以更快速地抓拍图像，也就能更加准确地记录目标的每个变化。在实时场景中，目标始终在变化运动，实时跟踪算法在目标只发生微小变化时也能及时采集目标图像，对目标新特征进行处理与学习，也就保证了在实时场景跟踪目标的准确率。与此相反，TLD 因为运行速度慢，所以在实时场景跟踪时抓拍时间间隔较大，目标在两帧之间已经发生了较大的变化，而 TLD 对目标发生的新变化却不能及时学习，导致了跟踪准确率的下降。

以上实验表明，在实时跟踪场合，实时跟踪算法在跟踪速度、准确率方面都远远优于 TLD。实时跟踪算法更加适用于实时场景跟踪。

5.3 实时跟踪算法与 Camshift 算法的比较

为保证实验的公正，在本节实验中采用的图像序列是 Camshift(Continuously Adaptive Mean-SHIFT)[9] 最经典的 man 图像序列，采用 Camshift 算法最经典的 matlab 程序。实时跟踪算法程序与经典 Camshift 算法程序分别运行在 man 图像系列上，分别记录跟踪速度以及准确率。实验过程截图如图 9 所示，实验结果如表 3 所列。

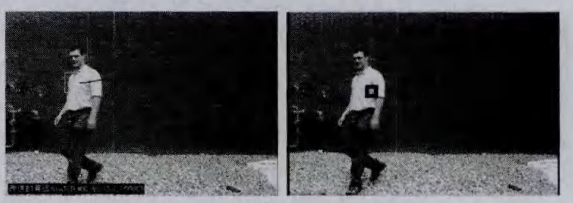

图 9 检测 man 序列(为实时算法，右为 Camshift)

表 3 实时跟踪算法与 Camshift 的比较

	实时跟踪算法	Camshift 算法
man 平均跟踪速度	13.3068 帧/秒	5.4958 帧/秒
man 序列准确率	100.00%	—

比较两个算法的运行过程与结果可知，跟踪学习检测算法能始终非常稳定地跟踪目标区域，运行流畅，速度快，准确率达到 100%。而 Camshift 算法在开始运行后，其跟踪的目标区域大小就出现错误，并且跟踪很不稳定，在目标区域周围发生严重的漂移，虽然每帧都有结果显示，但是结果都不准确。如果从指示目标大范围区域的角度来讲，其准确率是 100%；但是如果从准确跟踪目标这个角度来讲，其准确率为 0。

以上实验表明，实时跟踪算法无论在运行速度还是跟踪准确率上都远远优于 Camshift 算法。

5.4 实时跟踪算法与实时压缩跟踪算法的比较

实时压缩跟踪(Real-time Compressive 跟踪器，CT)算法[7]是 2012 年在欧洲计算机视觉国际会议(European Conference on Computer Vision，ECCV)上提出的算法。文献[10]将 CT 算法与其它 7 个主流的目标跟踪算法(包括 TLD 算法、MILTrack[10]、OAB[11]、SemiB[12]、Frag[13]、$\ell 1$ 跟踪器[14]、Struck 算法[15])进行比较，最后得出 CT 算法无论在运行速度和准确率都超过其它算法。所以将实时跟踪算法与 CT 算法进行比较就很有价值。

为保证比较实验的公平性，在本节实验中采用文献[10]中演示的图像序列 David、Kitesurf、Sylv，其直接从作者主页上下载得到。使用的 CT 算法的 matlab 程序也是直接从其作者主页上下载的演示程序，即其所有的参数都已调到最佳，保证 CT 算法能发挥出最佳性能。实时跟踪算法与 CT 算法分别在实验计算机上运行，识别跟踪图像序列中的目标，分别记录算法的运行速度和跟踪的准确率。准确率的计算方法采用文献[7]中的准确率计算方法。实验过程截图如图 10 所示，实验结果如表 4 所列。

图 10 检测 David 图像序列(左为实时算法，右为 CT)

表 4　实时跟踪算法与 CT 算法的比较

	实时跟踪算法	CT 算法
David 平均跟踪速度	22.3194 帧/s	9.4910 帧/s
David 序列准确率	98%	89%
Kitesurf 速度	12.5791 帧/s	7.5897 帧/s
Kitesurf 准确率	41.67%	38.55%
Sylv 速度	20.1109 帧/s	9.2533 帧/s
Sylv 准确率	88.81%	60.07%

在运行过程中，跟踪学习检测算法能够准确地跟踪目标，在目标大小发生改变的情况下，跟踪的目标框大小也有相应的改变。CT 算法也可以准确地跟踪目标，但是 CT 算法在跟踪的过程中，目标框发生了一定程度的漂移，并且目标框大小不能随着目标大小的改变而改变。

以上实验表明，跟踪学习检测算法的运行速度大大超过 CT 算法；并且实时跟踪算法能准确地框出目标位置，目标框能随着目标大小的变化而变化。

结束语　针对 TLD 运行速度慢及在实时跟踪场合准确率低的问题，本文提出了基于动态滑动扫描框的检测方法，实现了实时视频跟踪算法。实时跟踪算法能对目标进行实时准确的跟踪，对安防监控、交通监控等实时监控领域有巨大意义。

虽然实时跟踪学习检测算法具有优秀的运行速度和准确率，但是在目标被遮挡的时，其检测的准确率会下降，而且它目前只能检测单个目标。针对这两个问题，将来的工作可以分为以下两步：(1)参考霍夫森林算法[16]，基于随机森林和霍夫投票来提高在目标被遮挡时检测的准确率。(2)参考多目标信息融合算法，比如 PHD 算法[17]，将其与实时跟踪学习检测算法进行融合，使其可以实现多目标的跟踪[18,19]。

参 考 文 献

[1] Kalal Z, Mikolajczyk K, Matas J. Tracking-learning-detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1409-1422

[2] Kalal Z, Matas J, Mikolajczyk K. Pn learning: Bootstrapping binary classifiers by structural constraints[C]// 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2010: 49-56

[3] Fernando W S P, Udawatta L, Pathirana P. Identification of moving obstacles with pyramidal Lucas Kanade optical flow and kmeans clustering[C]// Third International Conference on Information and Automation for Sustainability, 2007 (ICIAFS 2007). IEEE, 2007: 111-117

[4] Kalal Z, Mikolajczyk K, Matas J. Forward-backward error: Automatic detection of tracking failures[C]// 2010 20th International Conference on Pattern Recognition (ICPR). IEEE, 2010: 2756-2759

[5] 江博. 基于 Kalman 的 TLD 目标跟踪算法研究[D]. 西安：西安科技大学，2013
Jiang Bo. The research of TLD target tracking algorithm based on Kalman[D]. Xi'an: Xi'an University of Science and Technology, 2013

[6] 高帆，吴国平，刑晨，等. TLD 目标跟踪算法研究[J]. 电视技术，2013, 37(11): 70-74, 202
Gao fan, Wu Guo-ping, Xing Chen, et al. TLD target tracking algorithm[J]. Video Engineering, 2013, 37(11): 70-74, 202

[7] Zhang K, Zhang L, Yang M H. Real-time compressive tracking [M]// Computer Vision-ECCV 2012. Springer Berlin Heidelberg, 2012: 864-877

[8] Bradski G, Kaehler A. Learning OpenCV: Computer vision with the OpenCVlibrary[M]. O'Reilly Media, Inc., 2008

[9] 孙凯，刘士荣. 多目标跟踪的改进 Camshift/卡尔曼滤波组合算法[J]. 信息与控制，2009, 38(1): 9-14
Sun Kai, Liu Shi-rong. The optimism of multi-target tracking algorithm the combination of Camshift and Kalman[J]. Information and Control, 2009, 38(1): 9-14

[10] Babenko B, Yang M H, Belongie S. Robust object tracking with online multiple instance learning[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(8): 1619-1632

[11] Grabner H, Grabner M, Bischof H. Real-Time Tracking via Online Boosting[J]. BMVC, 2006, 1(5): 6

[12] Grabner H, Leistner C, Bischof H. Semi-supervised on-line boosting for robust tracking[M]// Computer Vision-ECCV 2008. Springer Berlin Heidelberg, 2008: 234-247

[13] Adam A, Rivlin E, Shimshoni I. Robust fragments-based tracking using the integral histogram[C]// 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 2006: 798-805

[14] Mei X, Ling H. Robust visual tracking and vehicle classification via sparse representation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(11): 2259-2272

[15] Hare S, Saffari A, Torr P H S. Struck: Structured output tracking with kernels[C]// 2011 IEEE International Conference on Computer Vision (ICCV). IEEE, 2011: 263-270

[16] Gall J, Lempitsky V. Class-specific hough forests for object detection[M]// Decision Forests for Computer Vision and Medical Image Analysis. Springer London, 2013: 143-157

[17] Clark D E, Bell J. Multi-target state estimation and track continuity for the particle PHD filter[J]. IEEE Transactions on Aerospace and Electronic Systems, 2007, 43(4): 1441-1453

[18] Zhou Xiao-long, Li Y F, He Bing-wei, et al. GM-PHD-Based Multi-Target Visual Tracking Using Entropy Distribution and Game Theory[J]. IEEE Transactions on Industrial Informatics, 2014, 10(2): 1064-1076

[19] Zhou Xiao-long, Li Y F, He Bing-wei. Game-Theoretical Occlusion Handling for Multi-Target Visual Tracking[J]. Pattern Recognition, 2013, 46(10): 2670-2684

(上接第 250 页)

[18] Louazani A, Sekhri L, Kechar B. A time Petri net model for wormhole attack detection in wireless sensor networks[C]// Proceedings of the 2013 International Conference on Smart Communications in Network Technologies (SaCoNeT). Paris, France, 2013: 1-6

[19] Adjir N, de Saqui-Sannes P, Rahmouni K M. Conformance Testing of Preemptive Real-Time Systems[J]. International Journal of Embedded and Real-Time Communication Systems, 2013, 4(4): 1-26

[20] Liu Y, Sun J, Dong J S. Pat 3: An extensible architecture for building multi-domain model checkers[C]// 2011 IEEE 22nd International Symposium on Proceedings of the Software Reliability Engineering (ISSRE). Hiroshima, Japan, 2011: 190-199