doi:10.11959/j.issn.1000-436x.2015259

像素点特征加权的尺度自适应跟踪算法

罗会兰[1]，杜芳芳[1]，孔繁胜[2]
(1. 江西理工大学 信息工程学院，江西 赣州 341000；2. 浙江大学 计算机科学技术学院，浙江 杭州 310027)

摘 要：针对目标运动过程中的姿态变化、旋转、干扰以及缩放等情况，提出了结合像素点特征加权的尺度自适应跟踪算法。首先利用目标区域中每个像素点的颜色特征和位置特征，建立目标模型；其次用目标的平均权值图估算尺度变化系数，以实现目标尺度的自适应；最后构建一个更新模型，对跟踪过程中的目标模型和背景模型进行更新。实验表明，提出的算法充分利用目标区域内各像素点间的差异，可以做到快速、有效的跟踪，且具有较强的顽健性。

关键词：目标跟踪；尺度自适应；更新模型；像素点特征加权
中图分类号：TP391 **文献标识码**：A

Pixel feature-weighted scale-adaptive object tracking algorithm

LUO Hui-lan[1], DU Fang-fang[1], KONG Fan-sheng[2]
(1. School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China;
2. School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

Abstract: An effective object tracking method using weighted pixel features was proposed to deal with all kinds of complicated tracking situations, such as target movement, rotation, background interference and scaling and so on. First, the color feature and location information of the pixels in the target area were used to build the object model. Then the average weight image was used to estimate the scale variation coefficient. The aim was to adapt to the scale changes of the target. Finally, an update model was proposed, which was able to renew the object model and background model. The experimental results show that the proposed algorithm could make full use of the differences between pixels in the target area, so it can track more quickly and more effectively with strong robustness.

Key words: object tracking; scale-adaptive; model updating; pixel weighted-feature

1 引言

目标跟踪方法可以归为 2 类，决策型的跟踪和匹配型的跟踪[1]。决策型的跟踪是把跟踪作为一个分类问题，构建分类器对候选目标做出判断。Zhang 等[2]提出一种简单顽健的压缩跟踪(CT, compressive tracking)算法，先用随机投影矩阵对图像特征降维，再用朴素贝叶斯分类器进行分类，最后得到目标的位置。Zhang 等[3]提出的在线选择显著特征的实时跟踪 (ODFS, object tracking via online discriminative feature selection)算法，通过在线对特征的筛选，得到分类能力强的特征，进而得到顽健性比较强的分类器，对目标进行跟踪时具有较好的跟踪效果。决策型的跟踪算法通常需要提取分类特征并构建分类器，因此需要一定的计算量作为代价，同时也增加了跟踪过程所消耗的时间。

匹配型跟踪是对目标进行建模，把与模版匹配度最高的区域作为跟踪目标的位置。Mean-shift[4]跟踪算法是应用最广泛的匹配型目标跟踪算法之一[4~8]，它是一种基于核概率密度估计的无参数算法，不但计算量比较小，而且用核函数直方图对目标建模可以较好地应对边缘遮挡、目标旋转和姿态

收稿日期：2014-12-16；**修回日期**：2015-04-06
基金项目：国家自然科学基金资助项目(61105042, 61462035)；江西省教育厅科技基金资助项目(GJJ13421)
Foundation Items: The National Natural Science Foundation of China (61105042, 61462035); The Science and Technology Foundation of Education Department of Jiangxi Province (GJJ13421)

变化等。基于其诸多优点，Mean-shift 跟踪算法在过去的几年中得到了很好的发展。在目标跟踪中，检测到的目标区域或多或少都会包含一定的背景信息，当需要跟踪的目标与背景相似度比较高时，目标定位的精确度就会下降，Mean-shift 算法的跟踪效果就会变差。Comaniciu[5]在 Mean-Shift 算法的基础上提出了一个表示背景特征的模型 (BWH, background-weighted histogram)，试图通过减少目标模型和目标候选模型中同时具有的显著背景特征，以降低背景对目标的干扰，但是该算法并不能有效地减少背景特征，跟踪效果与传统的 Mean-shift 也没有太大的差异。Ning 等[6]基于 Mean-shift 和 BWH[5]算法提出了修正的背景加权直方图(CBWH, mean-shift tracking with corrected background-weighted histogram)算法，用归一化的颜色加权直方图对跟踪目标建模，并通过减少目标模型中与背景相似特征的权值，来减小背景的干扰，但是该算法在建立目标模型时，只对目标区域内各像素点按位置特征进行加权，并没有考虑颜色特征的显著性差异。为了实现自适应的跟踪效果，Ning 等[7]提出了一种基于 Mean-shift 的尺寸和方向自适应的跟踪 (SOAMST, scale and orientation adaptive mean shift tracking) 算法，用目标区域中所有像素点的权值的总和来表示目标在目标候选区域的权重面积，并用权值图的大小变化来表示目标区域的尺度变化，该算法虽然可以较准确地描述目标运动过程中尺度和方向的变化，但是在描述目标模型时没有充分利用目标区域内的显著特征，在目标快速运动以及受到背景干扰时，容易丢失目标，同时在实现自适应时计算过程非常复杂，致使跟踪消耗的时间变长。Zhang 等[8]提出时空上下文学习的快速跟踪 (FTSTC, fast tracking via spatio-temporal context learning)算法，用外观模型表示目标，对图像的强度加权，利用贝叶斯框架对目标和目标候选区域按时空上下文关系进行建模，在图片区域中寻找匹配程度最佳的位置作为跟踪结果，并用连续 2 帧目标区域和置信图来预测下一帧的目标尺度，以实现目标区域的尺度自适应。受目标模型的限制，FTSTC 算法在目标长时间被遮挡以及快速运动时很容易丢失目标，在实现尺度自适应时，FTSTC 算法也没有对尺度范围进行限制，在跟踪过程中表示目标的窗口大小可能超出图片范围，导致跟踪失败。

在目标区域中，每一个像素点在区分背景及目标时所起的作用差别很大，就像人在识别快速运动的物体时总是根据一些非常显著的特征。选择合适的特征来表达目标，是跟踪算法要重点解决的一个关键问题[9]。本文提出一种基于像素点特征加权的方法来利用显著特征提高跟踪效果，旨在目标模型中表达不同像素点在目标识别中的不同作用，尽可能让具有显著特征的像素点在目标模型中具有更重要的作用。本文将目标区域中每个像素点的颜色特征和位置特征作为显著特征，对颜色较显著的点赋予较大的权值，对离中心点较近的点赋予较大的权值，得到目标区域中每个像素点的加权系数，从而构建一个新的目标模型。这种思想可以适用于其他特征，或者适用于融合更多种特征。但考虑到如果采用太多种特征表示目标模型，虽然可以更精确地表示目标模型，但是会增加跟踪过程中的计算量，从而导致跟踪耗时变长。只考虑像素点的颜色特征和位置特征既能保证一定的跟踪效果，又能保证跟踪效率。目标跟踪是一个持续的过程，目标在运动的过程中保持着动态特性，采用固定的目标尺度和目标模型不能准确地描述目标的实时变化，从而降低了跟踪结果的有效性[10]。为了实现跟踪尺度的自适应，全面准确地描述目标特征，本文提出用连续两帧中目标区域像素点的平均权值图之比作为尺度变化系数，结合目标初始平均权值图进行修正，并利用之前各帧的目标尺度对下一帧的目标尺度进行预测，在保证跟踪过程简单、高效的同时，实现一定程度的尺度自适应。为了适应背景的变化，在跟踪的过程中，本文提出结合各帧间的时间上下文关系，对目标模型进行实时更新，并按照背景变化的强度，有选择地更新背景模型，来减少背景对目标的干扰，以实现跟踪过程的稳定性和顽健性。

2　SAOTPFW 算法

本文提出一种利用目标区域各像素点特征显著程度和位置信息构建目标模型的方法，算法的主要思想是：利用目标区域中各像素点的颜色特征和位置特征，对每个像素点进行分析。在颜色特征方面，对目标区域中比较显著的特征赋予较大的权值；在位置特征方面，对于目标区域中心点距离较近的像素点赋予较大的权值，综合这 2 个权值，结合抑制背景相似特征系数，构建目标模型。为了实现目标尺度的自适应，本文提出用连续 2 帧的目标

平均权值图之比来表示目标区域的尺度变化，结合初始帧进行修正，并利用之前帧的目标尺度来估算下一帧目标尺度，以完成目标尺度的自适应。最后提出一个判定当前帧与上一帧中背景相似程度的系数，并结合各帧间的时间上下文关系，根据背景变化的强度对模型进行实时更新。本文提出的算法命名为 SAOTPFW(scale-adaptive object tracking based on pixels feature-weighted)。

2.1 特征表示

对目标区域中的所有像素点进行统计，求得目标区域中所有像素点颜色特征的均值，然后将每个像素点的颜色特征与均值进行比较，得到一个特征差，对得到的特征差进行排序，具有较大的特征差的像素点赋予较大的权值，较小的特征差赋予较小的权值。

假设目标区域有 n 个像素点，第 i 个像素点的颜色特征可以量化为

$$c_i' = \frac{|c_{\text{mean}} - c_i|}{\sum_{i=1}^{n}|c_{\text{mean}} - c_i|} \tag{1}$$

其中，c_i' 是第 i 个像素点颜色特征的量化值，c_i 是第 i 个像素点的颜色特征，$c_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} c_i$ 是目标区域中所有像素点颜色特征的均值。本文中的颜色特征采用像素点的 RGB 特征，并用16×16×16 的三维空间对 3 个颜色通道进行量化。

对目标区域中所有的像素点按照与中心点的位置关系进行统计，离中心点较近的点赋予较大的权值，则第 i 个像素点的位置特征可以量化为

$$d_i' = \frac{|d_{\max} - d_i|}{\sum_{i=1}^{n}|d_{\max} - d_i|} \tag{2}$$

其中，d_i' 是第 i 个像素点位置特征的量化值，d_i 是第 i 个像素点到目标区域中心点的距离，$d_{\max}$ 是目标区域中距离目标中心距离最远的像素点到目标区域中心点的距离。

指数函数常用于相似性度量中,比较符合人类视觉对显著性感知的衰减属性。综合式(1)和式(2)，结合指数函数的性质，目标区域中第 i 个像素点的加权系数 s_i 定义为

$$s_i = \mathrm{e}^{c_i'+d_i'} \tag{3}$$

将目标区域中具有第 u 个特征值的 m 个像素点集合在一起，假设这 m 个像素点在目标区域中的集合为 Ω_m，则可以得到特征 u 的加权系数，如式(4)所示

$$s_u = \sum_{r=1}^{m} s_r \ , \quad r \in \Omega_m \tag{4}$$

2.2 模型建立

2.2.1 目标候选模型

假设坐标中心为 y 的目标候选区域中有 n_h 个像素点，候选目标模型中每个像素的位置用 $\{x_i\}_{i=1,\cdots,n_h}$ 表示。$p(y)=\{p_u(y)\}_{u=1,\cdots,m}$ 表示中心为 y 的目标候选模型，其中第 u 个特征的分布概率计算为

$$p_u(y) = C_h\sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta\left[b(x_i)-u\right] \tag{5}$$

其中，$k(\cdot)$ 为核函数，h 是核函数的带宽，δ是 Kronecker 函数，$b(x_i)$表示 x_i 像素点对应的颜色特征值的量化值。C_h 为归一化系数，计算方法为

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|\right)^2} \tag{6}$$

通过统计目标候选模型中目标周围的背景信息，以类似方法计算出背景特征分布概率，如式(7)所示

$$\{o_u\}_{u=1,\cdots,m} \tag{7}$$

其中，令 o^*为 $\{o_u\}_{u=1,\cdots,m}$ 中的最小非零值，定义一个目标模型与目标候选模型的转换系数，如式(8)所示，用以减少目标中与背景相似的显著特征的权值

$$\left\{v_u = \min\left(\frac{o^*}{o_u},1\right)\right\}_{u=1,\cdots,m} \tag{8}$$

2.2.2 目标模型

假设目标区域中有 n 个像素点，每个像素点都以目标区域的中心点为坐标的原点，目标区域一般表示为 $\{x_i^*\}_{i=1,\cdots,n}$。目标模型表示为 $q=\{q_u\}_{u=1,\cdots,m}$ 其中，特征 u 的分布模型为

$$q_u = C\sum_{i=1}^{n} k\left(\left\|x_i^*\right\|^2\right)\delta\left[b(x_i^*)-u\right] \tag{9}$$

其中，归一化系数 C 的计算方法如式(10)所示。$k(\cdot)$

为核函数，δ是 Kronecker 函数，$b\left(x_i^*\right)$表示x_i^*像素点对应的颜色特征值的量化值。

$$C=\frac{1}{\sum_{i=1}^{n}k\left(\left\|x_i^*\right\|^2\right)} \tag{10}$$

目标模型中某个特征 u 所包含像素点离目标区域中心点越近，与目标颜色特征的均值差别越大，直观上认为这种特征抗干扰的能力越强，在目标定位时所发挥的作用也就越大，对其赋予较大的权值，可以突出它对跟踪过程的影响。同时，为了降低目标中与背景相似的显著特征的影响，结合显著特征加权系数s_u (如式(4)所示)与抑制背景相似特征系数 v_u(如式(8)所示)，得到目标模型的特征加权系数 $f_u=s_u v_u$。如果一种特征在背景中非常显著，则 v_u 值会增加，导致 f_u 值下降，起到抑制背景相似特征的作用。

基于像素点特征加权的目标模型q_u'定义如下

$$q_u'=C_f f_u\sum_{i=1}^{n}k\left(\left\|x_i^*\right\|^2\right)\delta\left[b\left(x_i^*\right)-u\right] \tag{11}$$

归一化系数

$$C_f=\frac{1}{\sum_{i=1}^{n}k\left(\left\|x_i^*\right\|^2\right)\sum_{i=1}^{m}f_u\delta\left[b\left(x_i^*\right)-u\right]} \tag{12}$$

其中，$k(\cdot)$为核函数，是一个单调递减函数，δ是 Kronecker 函数，$b\left(x_i^*\right)$表示x_i^*像素点对应的颜色特征值的量化值。

2.2.3　跟踪模型

当前的目标中心位置y到新的中心位置y_1的计算过程为

$$y_1=\frac{\sum_{i=1}^{n_h}x_i w_i g\left(\left\|\frac{y-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h}w_i g\left(\left\|\frac{y-x_i}{h}\right\|^2\right)} \tag{13}$$

式(13)中的加权系数 w_i 定义为

$$w_i=\sum_{u=1}^{m}\delta\left[b\left(x_i\right)-u\right]\sqrt{\frac{q_{u'}'}{p_{u'}(y)}} \tag{14}$$

在式(13)中，$g(x)=-k'(x)$。为了简化计算，选 Epanechnikov 核函数[5]作为函数 $k(x)$，即 $g(x)=-k'(x)=1$，则式(13)可以化简为

$$y_1=\frac{\sum_{i=1}^{n_h}x_i w_i}{\sum_{i=1}^{n_h}w_i} \tag{15}$$

根据式(15)可以找到新一帧中与目标对象最相似的区域。

2.3　像素点特征加权对跟踪的影响

假定目标候选区域中某像素点x_i的显著特征量化值为 u'，则有 $\delta\left[b\left(x_i^*\right)-u'\right]=1$，$u\neq u'$时，有 $\delta\left[b\left(x_i^*\right)-u'\right]=0$。

因此式(14)所示的权值公式可以化简为

$$w_i'=\sqrt{\frac{q_{u'}'}{p_{u'}(y)}} \tag{16}$$

将式(11)中的目标模型 q_u' 和式(5)中的候选目标模型 p_u 代入式(16)，则有

$$w_i'=\sqrt{\frac{C_{\mathrm{f}}f_{u'}\sum_{i=1}^{n}k\left(\left\|x_i^*\right\|^2\right)\delta\left[b\left(x_i^*\right)-u'\right]}{C_h\sum_{i=1}^{n_h}k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta\left[b\left(x_i\right)-u'\right]}}$$

上式的分子分母同时乘以式(10)所示的归一化因子 C，则有

$$w_i'=\sqrt{\frac{C}{C}\frac{C_{\mathrm{f}}f_{u'}\sum_{i=1}^{n}k\left(\left\|x_i^*\right\|^2\right)\delta\left[b\left(x_i^*\right)-u'\right]}{C_h\sum_{i=1}^{n_h}k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta\left[b\left(x_i\right)-u'\right]}}$$

$$=\sqrt{\frac{C_{\mathrm{f}}f_{u'}}{C}}\sqrt{\frac{q_{u'}}{p_{u'}(y)}}$$

因为 C_{f}(如式(12)所示)和 C(如式(10)所示)均为规一化系数常量，对跟踪算法没有影响，所以可以忽略上式中的$\sqrt{\frac{C_{\mathrm{f}}}{C}}$，则上式可以化简为

$$w_i'=\sqrt{f_{u'}}\sqrt{\frac{q_{u'}}{p_{u'}(y)}} \tag{17}$$

背景加权系数降低了目标模型中与背景相似特征的权值，像素点特征加权系数增大了目标模型中显著特征的权值，所以结合两者的加权系数 $f_{u'}$ 可以进一步增大目标模型中显著特征的差别，从而使显

著性较强的特征在跟踪过程中起到更重要的作用。

2.4 尺度自适应

目标在运动时，由于观测角度和距离的改变，观测到的目标尺度也经常发生变化。目标区域中各像素点的权值总和构成一幅权值图，它能反应出目标区域的尺度变化[7]，图像序列中第 t 帧目标的权值图计算如下

$$M_t = \sum_{i=1}^{n} w_i' \tag{18}$$

其中，w_i' 如式(17)所示，n 为图像序列第 t 帧中目标区域内像素点的个数。基于像素点加权的目标模型计算得到的权值图，包含了目标区域所有像素点的颜色信息和位置信息，可以更精确地反应目标区域的尺度变化。将权值图平均分布在这 n 个像素点，得到权值图的均值，称为平均权值图。

$$\overline{M_t} = \frac{1}{n} M_t \tag{19}$$

2 帧间的平均权值图之比表示这 2 帧之间的目标区域的尺度变化的系数，则第 t 帧与第 t−1 帧的尺度变化系数可以表示为 $\lambda_t = \dfrac{\overline{M_t}}{\overline{M_{t-1}}}$。$\lambda_t$ 反应第 t 帧与上一帧目标区域面积的比例关系。

为了避免跟踪过程中的误差逐级放大，同时约束尺度变化的范围，结合初始目标的平均权值图对尺度变化系数进行修正，则有

$$\lambda_t' = \alpha \frac{\overline{M_t}}{\overline{M_1}} + (1-\alpha)\frac{\overline{M_t}}{\overline{M_{t-1}}} \tag{20}$$

其中，α表示修正程度，是一个常数，且 $0 \leqslant \alpha < 1$，当α=0 时，表示不对当前帧的尺度变化系数进行修正。

在视觉跟踪中，相邻帧之间的目标区域不会发生突然变化，根据当前帧和之前帧的目标区域可以预测下一帧目标区域。为了使预测的目标区域更接近真实值，先利用前面各帧中目标区域的平均值对下一帧目标区域进行估算，再结合当前帧的尺度变化进行修正，从而得到较准确的目标区域。用 S_i 表示第 i 帧目标区域的面积，则第 t+1 帧目标区域的面积可表示为

$$S_{t+1} = \frac{1}{t}\sum_{i=1}^{t} S_i + (\lambda_t' - 1) S_t \tag{21}$$

式(21)简单明确地描述了目标区域的尺度变化，也让预测的目标区域更加接近目标的真实尺度。

2.5 模型更新

在跟踪的过程中，目标对象经常面临着干扰、遮挡、姿态变化等挑战，只有实时更新目标模型与背景模型才能更精确地表示目标特征。目标跟踪是一个连续的过程，相邻帧之间有着较强的时空关系[8]，其目标和背景也存在一定的联系。

假设前一帧的背景模型为式(7)和式(8)所示的 $\{o_u\}_{u=1,\cdots,m}$ 和 $\{v_u\}_{u=1,\cdots,m}$，当前帧的背景模型为 $\{o_u'\}_{u=1,\cdots,m}$ 和 $\{v_u'\}_{u=1,\cdots,m}$，那么，当前帧与上一帧的背景相似系数定义为 $\rho = \sum_{u=1}^{m} \sqrt{o_u o_u'}$。

若ρ大于预先设定的阈值ε_2，说明背景没有发生太大的变化，则不需要更新背景模型。若ρ小于设定的阈值ε_2，则说明背景发生很大改变，需要对模型进行更新。

当判定背景需要更新时，考虑当前帧与上一帧的时间上下文关系，分别用 $\{o_u''\}_{u=1,\cdots,m}$ 和 $\{v_u''\}_{u=1,\cdots,m}$ 更新 $\{o_u\}_{u=1,\cdots,m}$ 和 $\{v_u\}_{u=1,\cdots,m}$。其中，$\{o_u''\}_{u=1,\cdots,m}$ 和 $\{v_u''\}_{u=1,\cdots,m}$ 的计算过程为

$$\left\{o_u'' = \beta o_u + (1-\beta) o_u'\right\}_{u=1,\cdots,m} \tag{22}$$

$$\left\{v_u'' = \beta v_u + (1-\beta) v_u'\right\}_{u=1,\cdots,m} \tag{23}$$

其中，$0 \leqslant \beta < 1$，为一个常数，β的值越小表示之前背景信息在新的背景模型中所占的比重越小，背景模型更新的越彻底。

背景模型更新之后，再利用式(11)重新计算目标模型 q_u'，即完成了目标模型的更新。

2.6 算法流程

本文提出了一种基于像素点特征加权的跟踪算法(SAOTPFW)。在目标表示方面，把颜色特征和位置特征相结合，构建一个新的加权系数；在尺度自适应方面，利用平均权值图计算相邻 2 帧之间目标区域的尺度比例系数，结合初始帧进行修正，并预测出下一帧的目标尺度；在模型更新方面，结合各帧之间的时间上下文关系，根据当前帧与上一帧的背景相似系数，有选择性地对背景模型进行更新，并依据背景模型和预测的目标尺度对目标模型进行实时更新。其步骤如下。

SAOTPFW 算法

输入：图像序列以及第一帧图像的目标区域面

积 S_1 和位置 y_0。

输出：图像序列每一帧中目标区域的面积和位置。

Step1　提取目标特征，用式(4)计算 s_u，用式(8)计算 v_u;

Step2　用式(11)计算出当前帧的目标模型 $\{q_u^{'}\}_{u=1,\cdots,m}$;

Step3　令迭代次数 k =0;

Step4　用式(5)计算出当前帧的候选模型 $\{p_u(y)\}_{u=1,\cdots,m}$;

Step5　用式(15)计算目标候选区域中新的目标位置 y_1;

令 d=|y_1−y_0|，k=k+1，用 y_1 更新 y_0;

如果 $d<\varepsilon_1$ 或 $k\geqslant N$

则停止迭代，输出跟踪结果，进入下一帧;

计算当前帧的 $\{o_u'\}_{u=1,\cdots,m}$,$\{v_u'\}_{u=1,\cdots,m}$;

如果 $\rho<\varepsilon_2$

则更新背景模型;

转至 Step1;

3　实验结果及分析

3.1　实验数据集

实验用 6 组具有挑战性图片序列并对其跟踪，其中，torus 序列来自文献[11]，lemming 序列来自文献[12]，skating2 序列来自文献[13]，panda 序列来自文献[14]，bird2 序列来自文献[15]，skiing 序列来自文献[16]。

3.2　实验环境及参数设置

本次实验的硬件环境为 Intel Core i7 2.67 GHz，内存 4 GB，显卡 128 MB，操作系统 Windows 7 32 bit，仿真软件为 Matlab 2014a。

在实验中，当前帧候选目标的搜索范围是上一帧目标区域往上下左右扩大 7 个像素点的范围，这是基于运动连续的假设，也是目前基于 Mean-shift 跟踪算法常采用的方法。Mean-shift 收敛阈值为 ε_1=0.15，最大迭代次数为 N=15，用 RGB 颜色作为颜色特征，并将其量化为 16×16×16 的三维特征空间(参照文献[6]设置)。之前的背景模型在模型更新所占比例 β =0.1 (实验选此值时效果较好)，尺度变化系数的修正程度 α=0.2 (实验选此值时效果较好)，判定更新的相似系数阈值为 ε_2=0.5 (参照文献[6]设置)，本次实验预测的目标位置用一个以预测区域的中心点为形心的矩形框来表示。本次实验所选参数引自经典算法或通过实验设置，均有较广泛的适用性。

3.3　实验结果对比及分析

为了证明本文提出算法 SAOTPFW 的效果，将 SAOTPFW 与 Mean-shift[4]，CBWH[6]，CT[2]，ODFS[3]，SOAMST[7]和 FTSTC[8]等跟踪算法进行了一系列的实验比较与分析。实验中参与比较的算法代码来自于作者网站，算法参数也是依据作者提供的可执行代码进行设置。所有的实验结果是运行 6 次取平均值得到的。

3.3.1　SAOTPFW 与 6 种算法的跟踪效果

SAOTPFW 与 Mean-shift，CBWH，CT，ODFS，SOAMST 算法和 FTSTC 在 6 组图片序列上的部分跟踪结果如图 1 所示。其中，每组图片的前 2 幅为部分跟踪结果，后 2 幅为其对应的局部放大图。

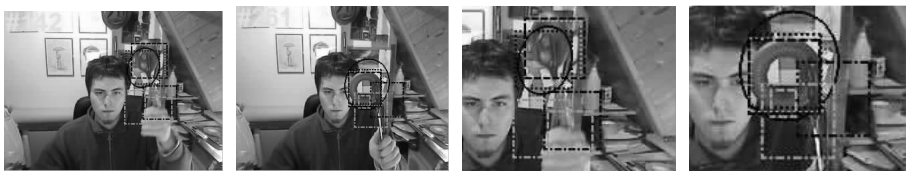
图 1(a)为在 torus 序列上的实验跟踪结果，Mean-shift 算法在第 184 帧丢失目标，CT 算法在第 136 帧丢失目标，ODFS 算法在第 138 帧丢失目标，FTSTC 算法在第 16~74 帧之间以及第 134~224 帧之间没有正确标出目标的位置。而 CBWH、SOAMST 和 SAOTPFW 算法可以长时间跟踪目标。

图 1(b)为在 lemming 序列上的实验跟踪结果，CT 算法在第 546 帧丢失目标，ODFS 算法在第 385 帧丢失目标，FTSTC 算法在第 382 帧丢失目标。而 SAOTPFW、Mean-shift、CBWH 和 SOAMST 算法可以跟踪目标至序列结束。

图 1(c)为在 skating2 序列上的实验跟踪结果，CT 算法在跟踪到第 589 帧时丢失了目标，ODFS 算法在第 81 帧丢失目标，SOAMST 算法在第 554~585 帧之间短暂丢失目标，FTSTC 算法在第 83 帧丢失目标，而 Mean-shift 算法、CBWH 算法和 SAOTPFW 算法则可以较好地跟踪目标。

图 1(d)为在 panda 序列上的实验跟踪结果，Mean-shift 算法在第 89 帧丢失目标，CBWH 算法在第 391 帧丢失目标，SOAMST 算法在 122 帧丢失目标。而 CT、ODFS、FTSTC 和 SAOTPFW 算法可以跟踪目标到序列结束。

图 1(e)为在 bird2 序列上的实验跟踪结果，Mean-shift 算法在第 94~96 帧未能标出目标的位置，之后 Mean-shift、CT 和 SOAMST 算法均在第 99 帧丢失目标。CBWH 算法、ODFS 算法、

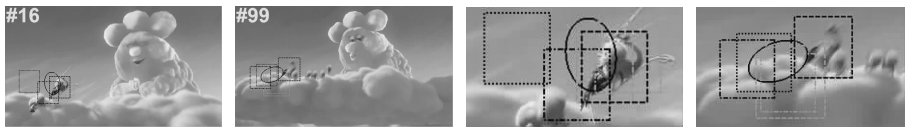


(a) torus(第 142,261 帧及其局部放大图)

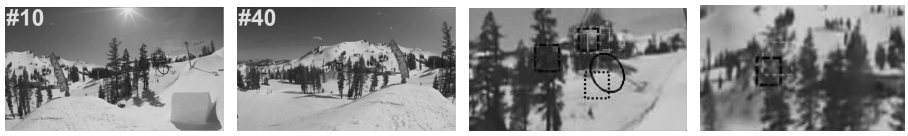
(b) lemming(第 204,506 帧及其局部放大图)

(c) skating2(第 293, 602 帧及其局部放大图)

(d) panda(第 174, 391 帧及其局部放大图)



(e) bird2(第 16, 99 帧及其局部放大图)



(f) skiing(第 10,40 帧及其局部放大图)

(Meanshift　CBWH　CT　ODFS　SOAMST　FTSTC　SAOTPFW)

图 1　7 种跟踪算法在 6 个图片序列上的部分跟踪效果比较

FTSTC 算法和 SAOTPFW 算法自始至终可以跟踪到目标。

图 1(f)为在 skiing 序列上的实验跟踪结果，Mean-shift 算法在第 10 帧丢失目标，CT 算法在第 6 帧丢失目标，ODFS 算法在第 9 帧丢失目标，SOAMST 算法在第 10 帧丢失目标，FTSTC 算法在第 15 帧丢失目标，而 CBWH 算法和 SAOTPFW 算法在跟踪过程中没有丢失目标。

图 1 的结果表明，在对这 6 组图片序列跟踪时，只有 SAOTPFW 算法和 CBWH 算法可以较长时间的跟踪目标，而且 SAOTPFW 算法的跟踪效果最好。

3.3.2　SAOTPFW 与 SOAMST 和 FTSTC 算法的跟踪尺度自适应对比

跟踪面积是指在目标跟踪过程中所用算法对目标区域面积的预测值。预测出的区域完全包含目标区域，且目标区域在跟踪面积中所占的比例越大，表示预测的目标区域越精确。本次实验所采用的 6 组数据集中，在 torus、lemming 和 skating2 序列中目标出现了较明显的缩放，用 SOAMST 算法、FTSTC 算法和 SAOTPFW 算法在这 3 组数据集上的跟踪效果（如图 1 所示）。3 种算法在这 3 组数据集上预测的目标面积对比如图 2 所示。图 2 中纵坐标表示预测面积减去真实面积，纵坐标越接近于 0，

表明预测的面积越接近真实面积，尺度自适应也越准确；横坐标表示的是图片序列集的帧数。

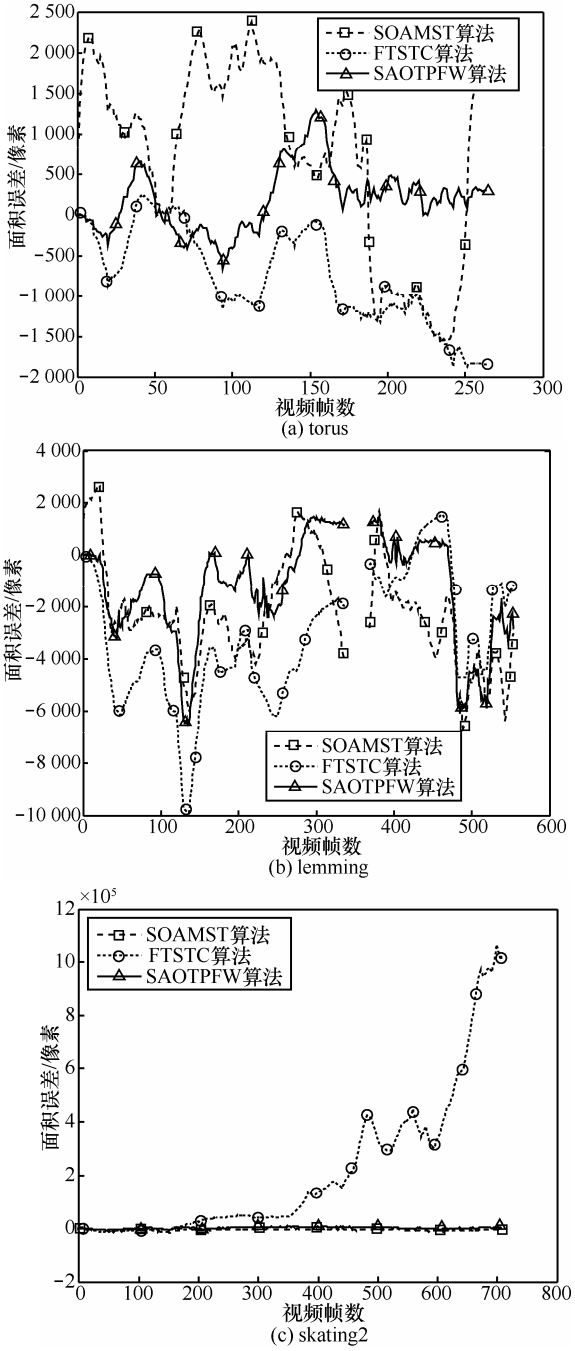


图 2　SOAMST、FTSTC 和 SAOTPFW 在 3 个图片序列上预测目标面积误差比较

在图 2(a)所示的 torus 序列中，SOAMST 算法在尺度自适应的过程中标出的目标区域包含了越来越多的背景信息，直接影响了跟踪的准确性。206~257 帧之间目标受到背景的干扰，SOAMST 算法预测的目标区域小于目标区域。在整个跟踪过程中，FTSTC 算法预测的目标区域始终小于目标大小，而 SAOTPFW 算法预测的目标区域相对于其他 2 种算法更接近目标尺度。

在图 2(b)所示的 lemming 序列中，FTSTC 算法从第 19 帧以后预测的目标区域小于目标的大小。SAOTPFW 算法预测的目标大小在 1~26 帧与真实的目标区域较接近，在 35~54 帧之间 SOAMST 算法更接近真实目标区域。在 90~159 帧之间和 469~529 帧之间，目标放大，SAOTPFW 和 SOAMST 算法预测的目标区域都小于真实目标区域，但是 SAOTPFW 算法与之更接近。

在图 2(c)所示的 skating2 序列中，随着目标的姿态变化，SAOTPFW 算法始终可以较准确地预测出目标的位置和区域，而 SOAMST 算法在第 21 帧之后逐渐缩小预测的目标区域，只能标记出目标的局部区域，甚至在 554~585 帧之间标错目标位置。FTSTC 算法在跟踪到目标时预测的区域总是小于目标的大小，在丢失目标后，预测的目标尺度逐渐增大，以至第 114 帧以后预测的目标区域超出了图片范围。

面积误差平均值是所预测的目标面积与真实的目标面积之差绝对值的平均值，其单位是像素。SOAMST、FTSTC 和 SAOTPFW 算法在对 torus、lemming 和 skating2 序列进行跟踪时，预测的目标面积的平均误差如表 1 所示，其中，标下划线的表示在该序列上的最小值。

表 1　SOAMST、FTSTC 和 SAOTPFW 算法对 3 种序列跟踪时的面积误差平均值比较（单位：像素）

算法	torus	lemming	skating2
SOAMST	1 265.5	2 645.6	6 942.4
FTSTC	795.37	3 394.8	213 451
SAOTPFW	<u>340.2</u>	<u>1 753.9</u>	<u>2 803.6</u>

由表 1 可以看出在对 torus、lemming 和 skating2 序列进行跟踪时，本文提出的 SAOTPFW 算法的面积误差平均值最小，所预测的目标面积与真实的目标面积也最接近。

3.3.3　SAOTPFW 与 6 种算法的准确率对比

SAOTPFW 算法与 Mean-shift[4]、CBWH[6]、CT[2]、ODFS[3]、SOAMST[7]和 FTSTC[8]算法在 6 组测试图片序列集上进行跟踪的跟踪偏差，即中心位置误差如图 3 所示。其中，横坐标表示各图片序列的帧数，纵坐标表示预测到的中心位置与真实的中心位置之间的差值，图中的差值以像素点为单位。

差值越小，表示预测位置与标准的中心位置越接近，目标定位的准确度就越高。

在图 3(a)所示的 torus 序列中，由于目标是无规则的随机运动，Mean-shift 算法在前 184 帧中可以跟踪到目标，在第 14~24 帧之间跟踪偏差较大，并在第 25 帧和第 57 帧出现了漂移，之后在第 57~106 帧和第 138~153 帧之间跟踪偏差比较大。CT 算法和 ODFS 算法在第 62~70 帧之间的偏移量比较大，之后 ODFS 算法在第 71 帧、103 帧、107 帧和 113 帧出现了漂移。SOAMST 算法从第 7 帧之后始终保持着一定的跟踪偏差。FTSTC 算法始终保持较大的中心位置误差，并在第 14 帧、15 帧、16 帧和 225 帧发生了漂移现象。CBWH 和 SAOTPFW 算法的跟踪过程比较稳定，目标定位也比较准确，跟踪偏差较小。

在图 3(b)所示的 lemming 序列中，第 338~368 帧出现空白是因为目标在这些帧中被遮挡。CT 算法在第 45 帧出现了漂移现象，在第 199~230 帧之间以及第 541 帧以后，跟踪偏差都比较大。ODFS 算法在第

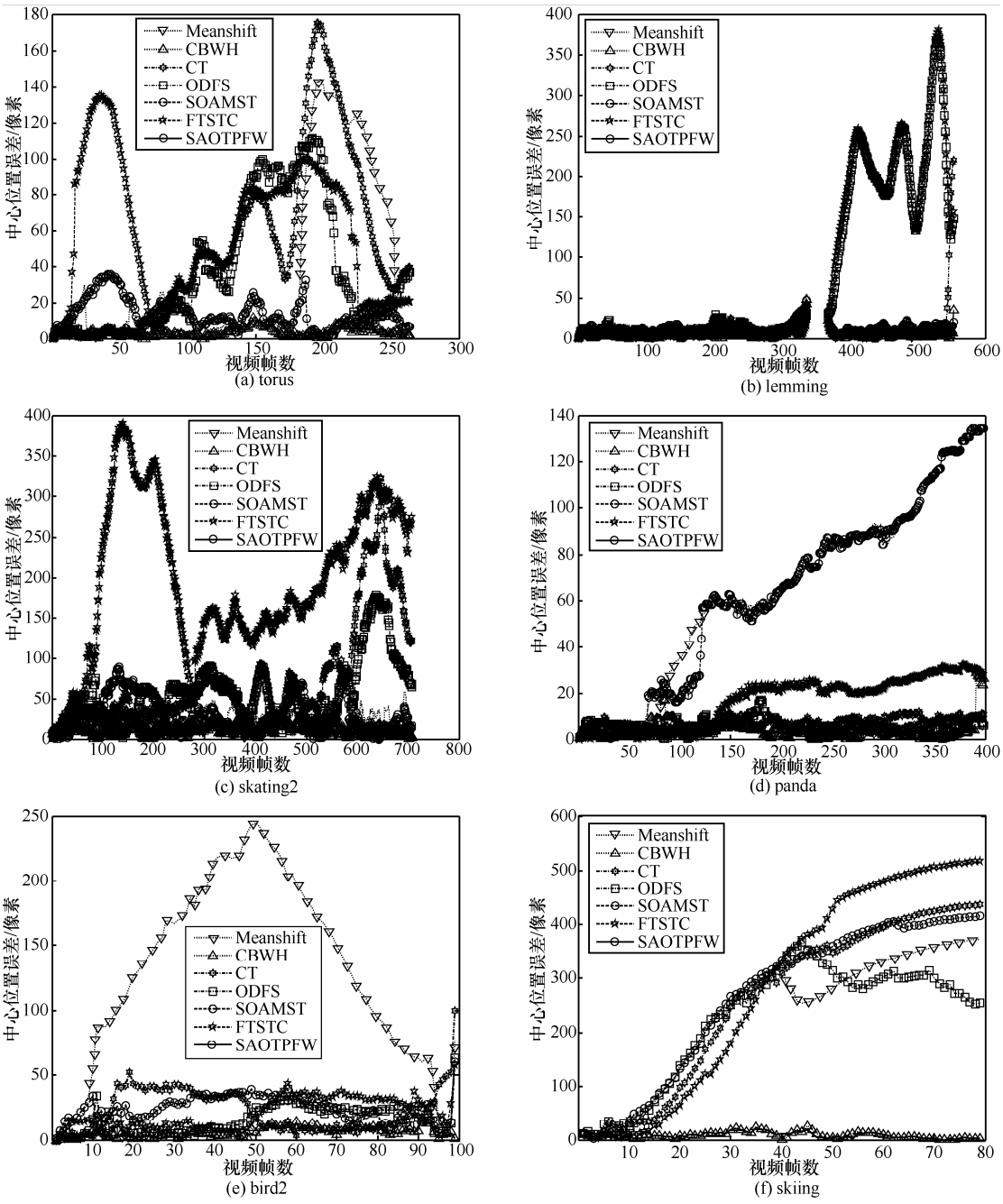


图 3　7 种跟踪算法在 6 个图片序列上的跟踪准确率比较

45 帧和第 217 帧出现漂移，在第 199~217 帧中的跟踪偏差比较大。在序列的第 297~375 帧和第 434~460 帧之间目标部分遮挡，测试的各种跟踪算法都出现了跟踪偏差，但是 CT 算法的偏差最大，SAOTPFW 算法较小。CBWH 算法、Mean-shift 算法、SOAMST 和 SAOTPFW 算法在跟踪的过程中跟踪偏差比较小。

在图 3(c)所示的 skating2 序列中，面对目标的旋转、缩放和遮挡，Mean-shift 算法在第 225~291 帧之间跟踪偏差比较大，并在第 292 帧和第 691 帧出现了跟踪漂移现象。CT 算法从第 2 帧开始就出现了较大的跟踪偏差，ODFS 算法在第 23 帧和 60 帧出现了漂移现象，之后也保持较大的跟踪偏差。SOAMST 算法在第 108 帧和 173 帧出现了漂移现象。FTSTC 算法在可以跟踪到目标时，保持较小的跟踪误差。在第 107~180 帧之间目标被部分遮挡，各跟踪算法都出现比较大的跟踪误差。SAOTPFW 和 CBWH 算法的跟踪过程比较稳定，但在第 457 帧也发生了跟踪漂移。

在图 3(d)所示的 panda 序列中，Mean-shift 算法在第 69 帧发生跟踪漂移，之后保持较大的跟踪偏差。CBWH 算法在第 391 帧发生漂移，CT 算法第 96 帧出现了漂移，ODFS 算法在第 7 帧、127 帧和 172 帧发生漂移。SOAMST 算法在第 69 帧和 122 帧出现漂移，并保持较大的跟踪偏差。FTSTC 算法在第 133 帧以后的跟踪偏差都比较大。只有 SAOTPFW 算法在整个跟踪过程比较稳定，始终保持较小的跟踪偏差。

在图 3(e)所示的 bird2 序列中，目标在运动的过程中，出现部分遮挡和旋转等现象。Mean-shift 算法在第 2 帧就出现了较大的跟踪偏差，并在第 94~96 帧没有跟踪到目标。CT 算法在第 15 帧出现了漂移，ODFS 算法在第 10 帧、16 帧、17 帧和 99 帧有漂移现象，SOAMST 算法在跟踪过程中始终保持一定的跟踪偏差，FTSTC 算法在第 11 帧、12 帧、16 帧和 17 帧发生漂移现象，SAOTPFW 算法和 CBWH 在对该序列跟踪时，跟踪过程比较稳定，跟踪效果也明显好于其他算法。

在图 3(f)所示的 skiing 序列中，由于目标运动较快，且有姿态变化，Mean-shift 算法在第 5 帧出现了漂移现象，随后一直保持比较大的跟踪偏差，CT 算法在第 5 帧和第 6 帧出现了漂移现象，ODFS 算法在第 2 帧、5 帧、6 帧和 7 帧中出现了跟踪漂移，SOAMST 算法从第 3 帧开始有较大的现偏差。FTSTC 算法在跟踪到目标时偏差较小，但是在 43 帧以后该算法预测的目标区域不在图片内。只有 CBWH 算法和 SAOTPFW 算法的跟踪性能比较稳定，但在第 40 帧、42 帧和 45 帧，这 2 种算法也出现了漂移现象。

中心位置误差平均值是所预测的目标中心位置与标准的目标中心位置之差的平均值，其单位是像素。Mean-shift、CBWH、CT、ODFS、SOAMS、FTSTCT 和 SAOTPFW 算法在 6 组图像序列集上的跟踪目标中心位置误差平均值如表 2 所示，其中，标下划线的表示在该序列上的最小值。

从表 2 可以看到，在 torus 序列中，SAOTPFW 算法的中心误差平均值仅次于 CBWH 算法，只比其高了 0.273 4 个像素点。在 lemming、panda、bird2、skiing 和 skating2 序列中，SAOTPFW 算法的中心位置误差平均值最小，远小于其他算法，跟踪更加精确。

3.3.4　SAOTPFW 与 6 种算法的跟踪耗时对比

跟踪耗时是指跟踪算法对测试的图片序列跟踪所消耗的时间，单位是秒。跟踪耗时越小，表明跟踪算法效率越高。Mean-shift、CBWH、CT、ODFS、SOAMST、FTSTC 和 SAOTPFW 算法在 6 组测试图片序列上的跟踪耗时如表 3 所示，其中，下划线表示在相应序列上的最小跟踪耗时。

从表 3 可以看出，Mean-shift 和 CBWH 算法的耗时比较短，这是因为这 2 种算法没有做到自适应目标

表 2　　**各算法在测试图片序列上的中心位置误差平均值比较(单位:像素)**

算法	torus	lemming	skating2	panda	bird2	skiing
Mean-shift	36.241 2	5.467 6	25.109 5	64.902 8	131.793 4	233.907 1
CBWH	3.402 9	3.332 6	14.138 0	4.504 0	6.223 4	8.724 2
CT	49.418 9	5.379 2	74.584 7	7.160 9	22.099 7	259.105 1
ODFS	36.658 4	28.030 5	55.446 3	5.057 5	15.208 6	223.992 7
SOAMST	14.112 1	5.406 8	25.689 9	63.401 6	24.766 9	265.284 3
FTSTC	59.638 0	27.978 6	197.911 3	16.521 3	20.376 3	282.115 7
SAOTPFW	3.676 3	3.027 5	13.995 7	2.925 2	5.747 6	8.367 2

表 3　　各算法的跟踪耗时比较(单位: s)

算法	torus	lemming	skating2	panda	bird2	skiing
Mean-shift	8.172 5	30.128 8	39.034 7	10.535 4	4.573 3	2.660 1
CBWH	8.403 5	28.865 7	37.009 7	10.606 8	4.744 6	2.788 1
CT	13.989 0	40.770 5	44.603 2	25.645 8	7.146 9	5.429 2
ODFS	16.834 7	46.863 5	55.530 5	25.372 6	8.175 1	6.382 9
SOAMST	64.000 4	240.202 6	211.815 6	46.292 7	38.985 0	8.238 3
FTSTC	9.292 7	24.495 3	33.637 7	13.685 0	5.154 9	3.165 3
SAOTPFW	8.781 9	34.714 5	45.053 0	10.757 9	5.341 8	3.235 2

尺度变化。FTSTC 算法在时空模型的学习和目标的检测都通过快速傅里叶变换来实现，所以学习和检测的速度都非常快，SAOTPFW 算法的跟踪速度次于这 3 种算法。CT 和 ODFS 算法虽然也没有实现自适应，但是耗时比 SAOTPFW 算法长。SOAMST 算法的跟踪耗时最长，几乎是 SAOTPFW 算法的 7 倍。在能同时实现目标定位和尺度自适应情况下，SAOTPFW 算法的跟踪速度也较占优势，由此可见，SAOTPFW 算法基本上实现了目标跟踪的高效性。

4　结束语

本文提出的基于像素点特征加权的跟踪算法，先利用目标区域中的颜色信息和位置信息构建目标模型，然后根据初始帧和相邻 2 帧间平均权值图以及之前帧的目标尺度实现尺度自适应，再依据背景变化程度判断模型是否更新。实验结果表明 SAOTPFW 算法在应对遮挡、姿态变化、缩放等方面有较好的性能，在目标跟踪的过程中跟踪位置误差平均值较小，所需的跟踪耗时也相对较短。由此可见，SAOTPFW 算法在保证跟踪准确的同时，减少了跟踪过程中花费的时间，这对实时目标跟踪的应用有非常重要的意义。

参考文献：

[1] ZHANG K, SONG H. Real-time visual tracking via online weighted multiple instance learning[J]. Pattern Recognition, 2013, 46(1): 397-411.

[2] ZHANG K, ZHANG L, YANG M H. Real-time compressive tracking[A]. Computer Vision–ECCV 2012[C]. Springer Berlin Heidelberg, 2012. 864-877.

[3] ZHANG K, ZHANG L, YANG M H. Real-time object tracking via online discriminative feature selection[J]. IEEE Transactions on Image Processing, 2013, 22: 4664-4677.

[4] COMANICIU D, RAMESH V, MEER P. Real-time tracking of non-rigid objects using mean shift[A]. Proc of IEEE Conference on Computer Vision and Pattern Recognition[C]. 2000.142-149.

[5] COMANICIU D, RAMESH V, MEER P. Kernel-based object tracking[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(5): 564-577.

[6] NING J, ZHANG L, ZHANG D, *et al*. Robust mean-shift tracking with corrected background-weighted histogram[J]. IET Computer Vision, 2012, 6(1): 62-69.

[7] NING J, ZHANG L, ZHANG D, *et al.* Scale and orientation adaptive mean shift tracking[J]. Computer Vision, IET, 2012, 6(1): 52-61.

[8] ZHANG K, ZHANG L, YANG M H, *et al*. Fast tracking via spatio-temporal context learning[J]. arXiv preprint arXiv:1311.1939, 2013.

[9] 赵凌, 冯镔, 邱锦波. 基于自适应分块外观模型的视觉跟踪[J]. 通信学报, 2011, 32(10): 166-173.
ZHAO L, FENG B, QIU J B. Fragment-based visual tracking with adaptive appearance model [J]. Journal on Communications, 2011, 32(10): 166-173.

[10] 覃光成, 尹浩, 陈强, 等. 面向价值的战场信息处理与分发优化算法[J]. 通信学报, 2011, 32(3): 60-68.
QIN G C, YIN H, CHEN Q, *et al*. Value-oriented optimal algorithm for battlefield information processing and disseminating [J]. Journal on Communications, 2011, 32(3): 60-68.

[11] CEHOVIN L, KRISTAN M, LEONARDIS A. An adaptive coupled-layer visual model for robust visual tracking[A]. Computer Vision (ICCV), 2011 International Conference on[C]. 2011. 1363-1370.

[12] SANTNEr J, LEISTNER C, SAFFARI A, *et al*. Prost: parallel robust online simple tracking[A]. 2010 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)[C]. 2010. 723-730.

[13] KWON J, LEE K M. Visual tracking decomposition[A]. 2010 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)[C]. 2010. 1269-1276.

[14] KALAL Z, MIKOLAJCZYK K, MATAS J. Tracking learning detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1409-1422.

[15] WANG S, LU H, YANG F, *et al*. Superpixel tracking[A]. 2011 IEEE International Conference on Computer Vision (ICCV)[C]. 2011. 1323-1330.

[16] GODEC M, ROTH P M, BISCHOF H. Hough-based tracking of non-rigid objects[J]. Computer Vision and Image Understanding, 2013, 117(10): 1245-1256.

作者简介：

罗会兰（1974-），女，江西上高人，博士后，江西理工大学教授、硕士生导师，主要研究方向为机器学习、模式识别。

杜芳芳（1988-），女，河南武陟人，江西理工大学硕士生，主要研究方向为模式识别。

孔繁胜（1946-），男，浙江宁波人，浙江大学教授、博士生导师，主要研究方向为人工智能与知识发现。