一种新颖的基于颜色信息的粒子滤波器跟踪算法

李培华

(黑龙江大学计算机科学与技术学院 哈尔滨 150080)

摘 要 传统的基于直方图的粒子滤波器算法常常需要在准确表达颜色分布和计算效率之间做出妥协,从而影响跟踪算法的性能甚至导致跟踪算法失败.针对这一问题,文中提出一种新颖的基于颜色信息的粒子滤波器跟踪算法.该算法采用自适应剖分颜色空间的概率模型,能够用较少的子空间准确地表达目标的颜色分布.文中进一步提出一种推广的积分图像,通过在该积分图像上进行数组索引操作得到每一个子空间的像素数目、均值向量和协方差矩阵,从而能够快速地计算出颜色模型.然而在 CPU 上计算积分图像十分耗时,为此文中提出一种基于 GPU 的并行算法快速计算积分图像.该并行算法在显卡的 GPU 上创建 3 个线程网格,分别顺序执行 3 个 Kernel 函数,依次完成创建原始积分图像以及对它的行和列执行前缀求和算法的任务.同传统的基于直方图的粒子滤波器算法相比,新算法每帧平均跟踪时间显著减少,同时跟踪准确性和鲁棒性都有较大提高.

关键词 目标跟踪;粒子滤波器;颜色模型;积分图像;并行算法

中图法分类号 TP391 **DOI 号**: 10.3724/SP.J.1016.2009.02454

A Novel Color Based Particle Filter Algorithm for Object Tracking

LI Pei-Hua

(*College of Computer Science and Technology, Heilongjiang University, Harbin* 150080)

Abstract The traditional histogram based particle filter often has to compromise between accurate representation of color distribution and computational efficiency, which affects the performance of the tracking algorithm or even results in tracking failures. To address this problem, the paper presents a novel color based particle filter algorithm for object tracking. The proposed algorithm utilizes a model based on adaptive partition of color space, which can represent accurately the color distribution of the object with smaller number of subspaces. The paper proposes extended integral images, by which the pixel number, mean vector and covariance matrix of each subspace can be obtained in simple array read operations that results in fast computation of the color model. The construction of the proposed integral images on CPU is, however, time-consuming, thus this paper proposes a GPU based parallel algorithm for fast computation of the integral images. The parallel algorithm consists of three thread grids respectively executing three Kernel functions with GPU on the video card, which sequentially builds the raw integral images, performs prefix sum with respect to rows and then with respect to columns of the original integral images. Compared to the traditional histogram based particle filter algorithm, the proposed one has much shorter tracking time, and in the meantime, attains improved tracking accuracy and robustness.

Keywords object tracking; particle filter; color model; integral images; parallel algorithm

收稿日期:2007-09-03;最终修改稿收到日期:2009-07-27.本课题得到国家自然科学基金(60673110,60973080)、黑龙江省普通高等学校新世纪优秀人才计划(1153-NCET-002)和人事部留学归国人员择优项目资助.**李培华**,男,1971 年生,博士,教授,主要研究领域包括图像处理和模式识别等. E-mail: peihualijia@vip.sina.com.

1 引　言

基于颜色的粒子滤波器(particle filter)跟踪近年来得到了越来越多的研究者的关注，在视频监控、智能人机交互、医学图像处理、视频编码以及智能机器人等许多研究领域得到了广泛的应用. 这种持续增长的研究兴趣是因为在复杂的环境中目标跟踪一般是非线性非高斯问题，而粒子滤波器能够提供一种统一的理论框架解决这类问题；另一方面是因为颜色信息是一种最常见而且重要的信息，它具有平移、旋转不变性和对遮挡及姿态变化不敏感的优点.

粒子滤波器跟踪的基本思想是将目标跟踪问题抽象为系统的状态估计问题，其中系统状态的后验概率密度迭代地通过一组具有权重的离散采样(称为粒子)来近似[1]. 由于粒子滤波器的本质是蒙特卡洛(Monte Carlo)模拟，为了准确地表达后验概率密度需要几百甚至上千个粒子. 在跟踪过程的每一帧图像中，需要对每一个粒子的状态进行观测，因此粒子滤波器跟踪的计算代价通常很大[2]. 为了使算法能够达到实时，研究者们一般采用粗略地表达目标颜色信息的方法来减少每个粒子的计算代价，同时采用尽可能少的粒子数. Nummiaro 等[3]在 RGB 颜色空间中用 $8\times8\times8$ 的加权直方图对目标进行观测，采用粒子数为 200 的粒子滤波器进行目标跟踪. Perez 等[4]在 HSV 颜色空间中用 H、S 通道的均匀 10×10 联合直方图加上 V 通道的均匀 10bin 直方图建模颜色特征，用 100 个粒子进行跟踪. 自适应剖分颜色空间的模型一般来说比均匀剖分颜色空间的模型能够更准确地刻画目标的颜色分布. Jacquot 等[5]提出了一种自适应的均匀直方图模型，他们利用 Akaike 模型选择理论自动地确定直方图的区间个数，并结合粒子滤波器进行跟踪. 他们的实验表明自适应直方图模型比传统的均匀剖分整个颜色空间的直方图模型能够得到更好的跟踪结果. Town[6]使用 K-均值聚类算法分析目标的颜色分布，在聚类分析的基础上自动地确定目标直方图区间以及区间个数，并将该颜色模型作为粒子滤波器的观测模型进行目标跟踪.

Viola 和 Jones[7]提出了一种积分图像的概念，利用积分图像仅仅使用数组索引和加法运算能够快速计算出图像中任意矩形区域的均值和方差. 在这一思想的启发下，Yang 等[8]选取多个矩形作为特征，利用积分图像分别计算这些矩形区域中每一个颜色通道中的均值来表达目标的颜色信息，其中特征的相似性用欧式距离来度量. 文献[9]同样选取一组矩形特征表达目标的表观模型，使用费舍尔分析在线地选择具有最大分辨能力的前景和背景特征，在粒子滤波器框架下进行跟踪.

在粒子滤波器跟踪算法中，从图像中得到目标的观测信息，即观测概率密度计算，是最耗时的过程，是粒子滤波器能否实时实现的关键[2]. 因此在准确表达目标的颜色分布和快速算法之间需要做出妥协：如果要更准确地表达目标的颜色分布，例如在传统直方图中采用较多的颜色区间数，那么对每个粒子计算观测概率所需的时间将会大大增加，从而影响算法的实时性；反之，如果粗略地表达目标的颜色分布，算法的跟踪速度会提高，然而由于无法准确地刻画目标的颜色密度，很容易导致跟踪误差增大甚至跟踪失败.

本文提出了一种新颖的算法来解决这一问题. 我们采用一种自适应剖分颜色空间的模型，该模型能更准确地表达目标的颜色分布. 在 Viola 和 Jones 的积分图像的基础上，我们提出了一种推广的积分图像，利用该积分图像通过简单的数组索引能够快速地计算颜色模型. 在 CPU 上构造积分图像是一个十分耗时的过程，为此我们提出了一种基于 NVIDIA G80 GPU 的并行算法，该算法能够快速地将积分图像计算出来. 这样我们就得到了一种新颖的基于颜色信息的粒子滤波器算法，该算法在跟踪时间和跟踪性能上都要优于传统的基于直方图的算法.

2 基于颜色的粒子滤波器算法

2.1 自适应颜色模型

本文使用文献[10]提出的颜色模型，该模型在聚类分析的基础上自适应地划分直方图区间，能够使用数量很少的区间准确地表达目标的颜色概率密度. 下面对这一模型进行简要介绍.

首先应用 Comaniciu 等提出的算法[11]对目标的颜色分布进行聚类分析，该算法能够自动地确定聚类数 d. 在聚类分析的基础上，根据如下的方法对目标的颜色空间进行剖分. 令 $\bar{\boldsymbol{z}}_u$ 和 $\boldsymbol{S}_u$ 分别是聚类 $C_u(u=1,2,\cdots,d)$ 包含的像素 $\boldsymbol{z}_u$ 的均值向量和协方差矩阵. 将矩阵 $\boldsymbol{S}_u$ 进行分解 $\boldsymbol{S}_u=\boldsymbol{V}_u\boldsymbol{\Sigma}_u\boldsymbol{V}_u^{\mathrm{T}}$，其中 $\boldsymbol{\Sigma}_u$ 是按降序排列的 $\boldsymbol{S}_u$ 的特征值 $\lambda_{u,i}(i=1,2,3)$ 组成的对角矩阵，$\boldsymbol{V}_u$ 是相应的归一化的正交向量组成的矩

阵.对颜色空间进行标准正交变换

$$\boldsymbol{w}_u = \boldsymbol{V}_u^{\mathrm{T}}(\boldsymbol{z}_u - \bar{\boldsymbol{z}}_u) \tag{1}$$

这样 $\boldsymbol{w}_u$ 的不同分量 $w_{u,i}$ 彼此正交,其标准差是 $\sqrt{\lambda_{u,i}}$.因此在变换空间中聚类 C_u 可以用一个中心在坐标原点 3 条边分别平行于 3 个坐标轴的三维矩形来表示,该三维矩形的边长分别是 $4\sqrt{\lambda_{u,i}}$, $i=1,2,3$.根据等式(1)进行如下的反变换

$$\boldsymbol{z}_u = \boldsymbol{V}_u \boldsymbol{w}_u + \bar{\boldsymbol{z}}_u \tag{2}$$

能够在原来的颜色空间中确定表征聚类 C_u 的三维矩形.

通过这种自适应剖分,整个颜色空间按照目标颜色分布被分成了 d 个颜色子空间,没有颜色分布的子空间将不予考虑.每一个子空间对应一个颜色聚类,相应地对应直方图的一个区间.对于每一个这样的直方图区间,我们的思想是不但考虑落入子空间的像素的个数,而且进一步用高斯分布建模每一个子空间的颜色分布.给定由 N 个像素组成的目标的参考图像 $\boldsymbol{D}(x,y)$,目标的模型表达为

$$\boldsymbol{p}=[p_1,\cdots,p_u,\cdots,p_d]^{\mathrm{T}},\quad p_u = b_u G(\boldsymbol{\mu}_u,\boldsymbol{R}_u) \tag{3}$$

其中,$b_u = n_u/N$, n_u 是落入第 u 个颜色子空间(直方图区间)的像素的数目,$G(\boldsymbol{\mu}_u,\boldsymbol{R}_u)$ 表示均值向量为 $\boldsymbol{\mu}_u$、协方差矩阵为 $\boldsymbol{R}_u$ 的高斯分布.注意到式(3)与文献[10]有所不同:由于我们不需要 Mean Shift 移动,同时为了能够快速计算 n_u,我们未考虑每个像素的加权距离.

考虑一个由 N' 个像素组成的候选图像区域的颜色分布 $\boldsymbol{p}'=[p'_1,\cdots,p'_d]^{\mathrm{T}}$,其中 $p'_u = b'_u G(\boldsymbol{\mu}'_u,\boldsymbol{R}'_u)$.两个分量分布 p_u 和 p'_u 之间的相似性根据 Bhattacharrya 距离来度量

$$\rho(p_u,p'_u)=\left(\frac{2n_u n'_u |\boldsymbol{R}_u|^{\frac{1}{2}}|\boldsymbol{R}'_u|^{\frac{1}{2}}}{NN'|\boldsymbol{R}_u+\boldsymbol{R}'_u|}\right)^{\frac{1}{2}}\times \exp\left(-\frac{1}{4}(\boldsymbol{\mu}_u-\boldsymbol{\mu}'_u)^{\mathrm{T}}(\boldsymbol{R}_u+\boldsymbol{R}'_u)^{-1}(\boldsymbol{\mu}_u-\boldsymbol{\mu}'_u)\right) \tag{4}$$

因此 $\boldsymbol{p}$ 和 $\boldsymbol{p}'$ 之间的相似性定义如下

$$\rho(\boldsymbol{p},\boldsymbol{p}') = \sum_{u=1}^{d}\rho(p_u,p'_u) \tag{5}$$

2.2 构造推广的积分图像快速计算颜色模型

基于 Viola-Jones 的积分图像[7],我们在文献[12]中提出了"积分直方图图像"的概念,能够快速计算图像中任意矩形区域的直方图.注意到 Porikli 等提出了类似的方法并分析了算法的计算复杂性[13].将这一思想进一步推广,我们可以构造相应的积分图像,基于这些积分图像能够快速计算图像中任意矩形区域的每一个聚类的均值向量和协方差矩阵,因而能够快速计算颜色模型(3).

给定原始的三通道彩色图像 $\boldsymbol{D}(x,y)=(D_i(x,y),i=1,2,3)$,其中 i 是通道索引.通过一次扫描原始图像,我们能够构造出积分图像 $I_{b_u}(x,y)$, $I_{\mu_{u,i}}(x,y)$ 和 $I_{\sigma_{u,i,j}}(x,y)$, $u=1,2,\cdots,d$, $i=1,2,3$, $j\le i$,用于计算每个子空间的像素数目、均值向量和协方差矩阵.考察积分图像 $I_{b_u}(x,y)$,该积分图像在位置 (x,y) 的值等于原图像中 (x,y) 左上的子图像中落入到第 u 个直方图区间的像素的个数,即

$$I_{b_u}(x,y)=\sum_{x'\le x,y'\le y}\delta_u(x',y') \tag{6}$$

其中 $\delta_u(x',y')=1$,如果 $\boldsymbol{D}(x',y')$ 属于第 u 个直方图区间;否则 $\delta_u(x',y')=0$.令 $I_{b_u}(x,0)=0$, $I_{b_u}(0,y)=0$,通过下列方程可以构造出积分直方图图像 $I_{b_u}(x,y)$

$$\begin{aligned} I_{b_u}(x,y) &= I_{b_u}(x-1,y)+I_{b_u}(x,y),\\ I_{b_u}(x,y) &= I_{b_u}(x,y-1)+\delta_u(x,y) \end{aligned} \tag{7}$$

令图像中某一矩形区域的左上角的坐标是 (x,y),长和宽分别为 w 和 h,该矩形区域的直方图可以通过 $4d$ 次数组索引和 $3d$ 次加法运算计算出来:

$$\begin{aligned} n_u = {} & I_{b_u}(x+w,y+h)-I_{b_u}(x+w,y)-\\ & I_{b_u}(x,y+h)+I_{b_u}(x,y),\ u=1,2,\cdots,d \end{aligned} \tag{8}$$

类似地,我们可以通过定义如下的积分图像计算均值向量和协方差矩阵

$$\begin{aligned} I_{\mu_{u,i}}(x,y) &= \sum_{x'\le x,y'\le y}D_i(x',y')\delta_u(x',y'),\\ I_{\sigma_{u,i,j}}(x,y) &= \sum_{x'\le x,y'\le y}D_i(x',y')D_j(x',y')\delta_u(x',y') \end{aligned} \tag{9}$$

通过类似于式(7)的方程,每个直方图区间的均值向量和协方差矩阵的分量可以通过数组索引快速计算出来:

$$\begin{aligned} \mu_{u,i} = {} & (I_{\mu_{u,i}}(x+w,y+h)-I_{\mu_{u,i}}(x+w,y)-\\ & I_{\mu_{u,i}}(x,y+h)+I_{\mu_{u,i}}(x,y))/n_u,\\ \sigma_{u,i,j} = {} & (I_{\sigma_{u,i,j}}(x+w,y+h)-I_{\sigma_{u,i,j}}(x+w,y))-\\ & I_{\sigma_{u,i,j}}(x,y+h)+I_{\sigma_{u,i,j}}(x,y))/n_u-\mu_{u,i}\mu_{u,j} \end{aligned} \tag{10}$$

其中 $u=1,2,\cdots,d$, $i=1,2,3$, $j\le i$.因此我们得到第 u 个直方图区间的均值向量和协方差矩阵如下

$$\boldsymbol{\mu}_u=\begin{bmatrix}\mu_{u,1}\\ \mu_{u,2}\\ \mu_{u,3}\end{bmatrix},\quad \boldsymbol{R}_u=\begin{bmatrix}\sigma_{u,1,1} & \sigma_{u,1,2} & \sigma_{u,1,3}\\ \sigma_{u,1,2} & \sigma_{u,2,2} & \sigma_{u,2,3}\\ \sigma_{u,1,3} & \sigma_{u,2,3} & \sigma_{u,3,3}\end{bmatrix} \tag{11}$$

当积分图像 $I_{b_u}(x,y)$，$I_{\mu_{u,i}}(x,y)$ 和 $I_{\sigma_{u,i,j}}(x,y)$ 计算完之后，我们可以通过式(8)和式(10)将颜色模型(3)快速计算出来. 表 1 给出了对于 256×256、256×128 和 128×128 的图像用 CPU 计算积分图像所用的时间，其中聚类数 d 分别是 4 和 8. 由表中可见在 CPU 上计算积分图像是一个十分耗时的过程，会影响跟踪任务的实时性，对此我们提出了基于 NVIDIA G80 GPU 的并行算法计算积分图像.

2.3　基于 GPU 的并行算法快速计算积分图像

NVIDIA 公司推出的 G80 系列 GPU 的硬件体系结构和软件架构(CUDA)与以前的系列相比发生了根本的变化(http://www.nvidia.com/object/cuda_home.html)，是真正意义上的桌面并行计算设备. G80 GPU 是由一组多处理器组成的，每个多处理器具有单指令、多数据体系结构(SIMD). 从软件层面上来说，G80 GPU 能够并发执行数量很大的多个线程，这些线程在不同的数据上执行相同的被称为“Kernel”的函数. 执行同一 Kernel 函数的线程按照线程网格的方式组织：线程网格由连续编号的线程块组成，而每个线程块又包含连续编号的一系列线程. 同一线程块中的线程可以访问 16K 的片上共享高速缓存，每 32 个线程组成一个 Warp，每 half-Warp 的 16 个线程在物理上同时得到执行.

我们的积分图像计算基于并行处理算法中经典的前缀求和问题(prefix sum)[14]，即给定一组有序元素$[a_0,a_1,\cdots,a_{m-1}]$，按顺序计算这些元素的部分和$[0,(a_0+a_1),\cdots,(a_0+a_1+\cdots+a_{m-1})]$. 这一算法构造一个 $\log_2 m$ 层的平衡树，包括从根节点向叶节点的向上扫描过程和相反的向下扫描过程. 在向上扫描阶段，将每一层两个相邻叶节点相加并将值存在后一个叶节点中；而在第 2 阶段，在每一层将两个相邻节点值互换，然后将二者的和存在后一个叶节点中. 这一算法共需要 $m/2$ 个线程，计算复杂性是 $O(n)$. 值得注意的是，G80 的片上高速共享缓存组成 16 个内存池(memory bank)，两个相邻的 32-bits 字被分配到两个相邻的内存池中. 因此将待处理的数据载入共享内存后，每 16 个字应填充一个冗余字. 否则，同一 half-warp 的线程会访问同一内存池，造成共享内存访问冲突从而使并发线程串行化，从而大大地降低运行效率[14].

我们提出的积分图像算法首先在 CPU 上完成数据的预处理，即通过直方图的 Lookup 表得到新的四通道图像 $\boldsymbol{D}'(x,y)=[u,D_1(x,y),D_2(x,y),D_3(x,y)]^{\mathrm{T}}$，其中 u 表示该像素属于第 u 个子空间，并将数据从 CPU 内存拷贝到 GPU 的全局内存中. 在 GPU 上的算法创建 3 个线程网格，分别执行 3 个 Kernel 函数. 其中第 1 个网格的输入数据是 $\boldsymbol{D}'(x,y)$，包括 h 个线程块，每个线程块包括 w 个线程，每个线程处理一个像素，其中 w 和 h 分别是原始图像的宽和高，为了计算方便，取 w 和 h 是 2 的整数次幂. 第 1 个网格的 Kernel 函数计算如下的原始(Raw)积分图像

$$\begin{aligned}&Z_{b_u}=\delta_u(x,y),\\&Z_{\mu_{u,i}}=D_i(x,y)\delta_u(x,y),\\&Z_{\sigma_{u,i,j}}=D_i(x,y)D_j(x,y)\delta_u(x,y),\\&u=1,2,\cdots,d,\ i=1,2,3,\ j\le i\end{aligned}\tag{12}$$

第 2 个网格的输入数据是原始(Raw)积分图像，包括 $10dh$ 个线程块，每个线程块包括 $w/2$ 个线程，每个线程完成两个相邻叶节点的处理. 其 Kernel 函数首先将原始积分图像的每一行数据读入片上高速缓存，然后对该行数据执行前缀求和算法从而得到中间结果积分图像. 第 3 个网格的输入数据是第 2 个网格计算的输出，包括 $10dw$ 个线程块，每个线程块包含 $h/2$ 个线程，其 Kernel 函数将中间结果图像的每一列数据读入片上高速缓存，然后对该列数据执行前缀求和算法. 最后我们得到了存储在 GPU 的全局内存中的积分图像 $I_{b_u}(x,y)$，$I_{\mu_{u,i}}(x,y)$ 和 $I_{\sigma_{u,i,j}}(x,y)$.

表 1 给出了在 GPU 上和 CPU 上计算积分图像所需要的时间. 从表中可见，在 $d=4$ 时在 GPU 上所需的时间约为 CPU 上所需时间的 1/3.7，在 $d=8$ 时在 GPU 上所需的时间约为 CPU 上所需时间的 1/3.

表 1　使用 GPU 和 CPU 计算积分图像的时间比较(单位：ms)

		聚类数 $d=4$		聚类数 $d=8$	
		GPU 计算时间	CPU 计算时间	GPU 计算时间	CPU 计算时间
图像尺寸	256×256	10.0	37.4	20.0	60.3
	256×128	5.0	18.8	10.0	30.3
	128×128	2.3	9.5	4.5	15.3

2.4　跟踪算法描述

同大多数论文相同[3-4,6,8-9]，目标的形状用矩形描述，该矩形允许在图像平面内平移、其长度和宽度允许以相同的尺度发生变化. 目标的运动模型(先验概率密度)$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ 建模为随机行走，其中 $\boldsymbol{x}_k$ 是目标在 k 时刻的状态. 目标的观测模型表达为

$$\pi(\boldsymbol{y}_k|\boldsymbol{x}_k)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1-\rho(\boldsymbol{p},\boldsymbol{p}')}{2\sigma^2}\right)\tag{13}$$

我们提出的跟踪算法简略地描述如下，关于粒子滤波器跟踪算法的详细描述可参阅文献[2].

1. 初始化

检测目标，构造目标的颜色分布 $\boldsymbol{p}$，令 $k=1$.

2. 采样和更新阶段

根据第 2.3 节介绍的方法构造积分图像 $I_{b_u}(x,y)$，$I_{\mu_{u,i}}(x,y)$，$I_{\sigma_{u,i,j}}(x,y)$.

For $i=1,2,\cdots$：

(a) 对每一个粒子 $\boldsymbol{x}_k^{(i)}$，根据先验密度进行预测，得到新的状态 $\hat{\boldsymbol{x}}_k^{(i)}$.

(b) 根据式(8)、(10)和式(11)计算候选目标的颜色模型，根据式(5)和(13)计算观测概率密度.

End

对粒子的权重进行归一化处理.

3. 输出和重采样

将系统后验密度的零阶矩(均值)输出作为跟踪结果，对粒子进行重新采样获得 $i.i.d$ 分布的粒子.

4. $k=k+1$，go to 步 2

3 实 验

程序在 Visual C++ 2005 编程环境中，使用主频为 3.2GHz 内存为 1.5GB 的台式机 HP Compaq dx6128MT 调试通过，并行算法在配备 Geforce 8800 GTS GPU 的影驰显示卡上调试通过. 程序使用 RGB 颜色空间，初始化是在第 1 帧中用手工完成的，真值(Ground truth)也是用手工标注得到的. 我们使用所提出的算法(简称新算法，粒子数：500)在 3 个图像序列中进行了实验，并与传统的基于直方图($8\times8\times8$)的粒子滤波器算法(简称传统算法，粒子数 200)进行了比较.

论文采用跟踪结果的 x 坐标误差、y 坐标误差和非重叠区域比来度量跟踪结果的准确程度. 令 M 和 T 分别为跟踪结果矩形和真值所包围的像素组成的集合，非重叠区域比 r 定义如下

$$r=\frac{M\oplus T}{|M|+|T|}=1-\frac{2|M\cap T|}{|M|+|T|} \tag{14}$$

其中$\oplus$表示两个集合的对称差，$\cap$ 表示两个集合的并集，$|\cdot|$ 表示集合包含的元素个数. r 的值介于 0 和 1 之间，r 越小跟踪结果越准确. 如果跟踪结果完全准确，即 $M=T$，则 $r=0$；如果跟踪结果 M 与 T 无交集，则 $r=1$. 图 1 表示了非重叠区域比的 3 种情况：$0<r<1$，$r=0$，$r=1$.

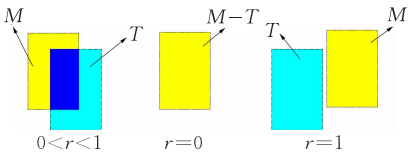


图 1 非重叠区域比的 3 种情况

3.1 跟踪实验

第 1 个实验是关于办公室环境中的人脸跟踪，图像序列是在典型的办公室环境中采集得到，共 500 帧，图像尺寸是 256×192. 由于目标和摄像机都在运动，因此目标帧间位移较大. 基于新算法的典型的跟踪结果如图 2 所示，从左至右、从上到下依次为第 1、50、90、140、200、280、360 和 490 帧的结果. 可以看到目标存在各种姿态变化，在第 200 帧附近、280 帧附近分别存在短时遮挡，在第 360 帧附近存在光照变化. 新算法和传统算法都能够在整个序列中稳定地跟踪目标. 图 3 给出了两种算法在每一帧中的 x 坐标、y 坐标误差和非重叠区域比，其中实线表示新算法，虚线表示传统算法. 表 2 给出了两种算法的平均跟踪误差和每帧平均跟踪时间. 可以看出新算法的 x 坐标误差、y 坐标误差和非重叠区域比比传统算法小. 传统算法和新算法的每帧平均跟踪时间分别为 30ms 和 24(10)ms，其中括号中的数据表示在新算法中 GPU 计算积分图像所用的时间.

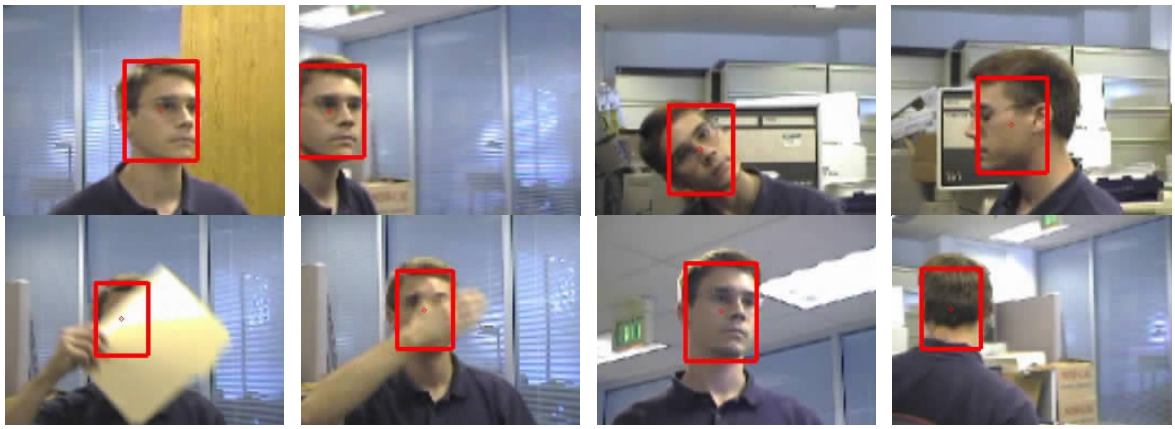
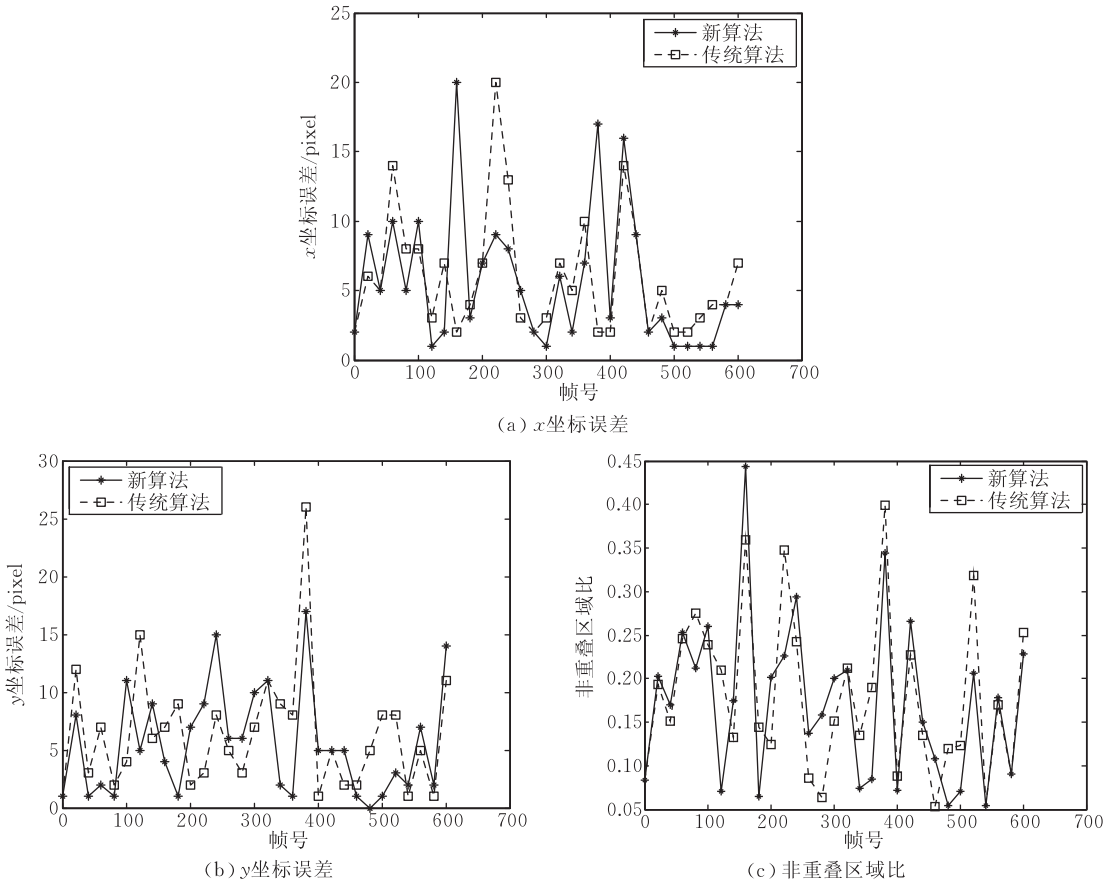


图 2 新算法对人脸的跟踪结果(从左至右、从上到下依次为第 1、50、90、140、200、280、360 和 490 帧)



(a) x坐标误差　(b) y坐标误差　(c) 非重叠区域比

图 3　人脸序列中传统算法(虚线)和新算法(实线)的跟踪误差比较

表 2　人脸序列中跟踪误差(均值±标准差)和平均跟踪时间

	x 坐标误差/pixel	y 坐标误差/pixel	非重叠区域比	平均跟踪时间*/帧(单位:ms)
传统算法	5.79±5.97	6.32±5.19	0.19±0.11	30
新算法	5.56±5.19	5.77±4.67	0.18±0.09	24(10)

注：* 表示括弧中的数据表示在平均跟踪时间中 GPU 计算积分图像需要的时间.

第 2 个实验是关于街道上的车辆跟踪. 图像序列是用手持式摄像机(hand-held camera) Sony DCR-PC1000E 拍摄到的，共包括 930 帧，图像尺寸是 704×576. 由于手持式摄像机的抖动，图像中目标经常发生突然的运动和显著的图像模糊，如在第 282、445、497、560、785、818 帧附近均发生了显著的运动模糊. 图 4 给出了基于新算法的典型跟踪结果.

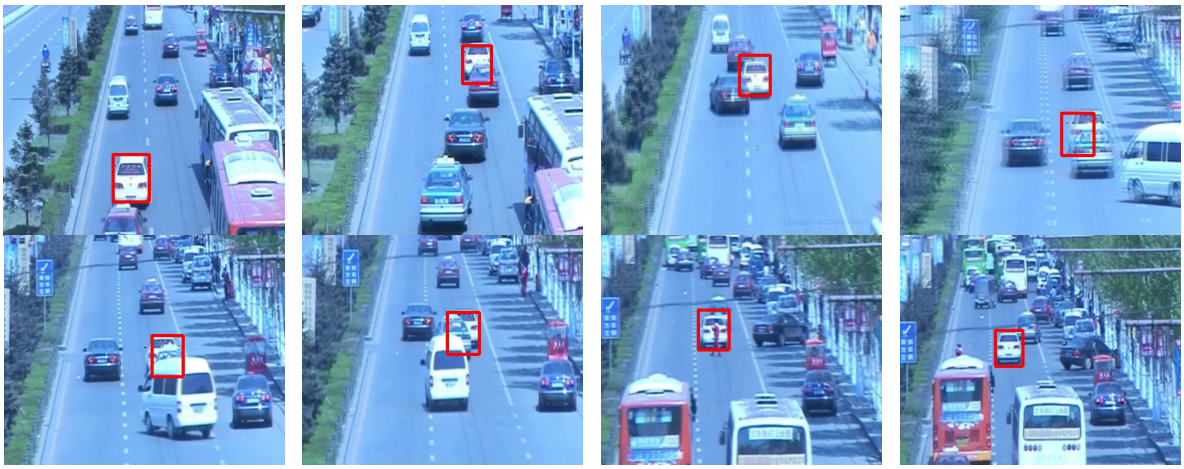
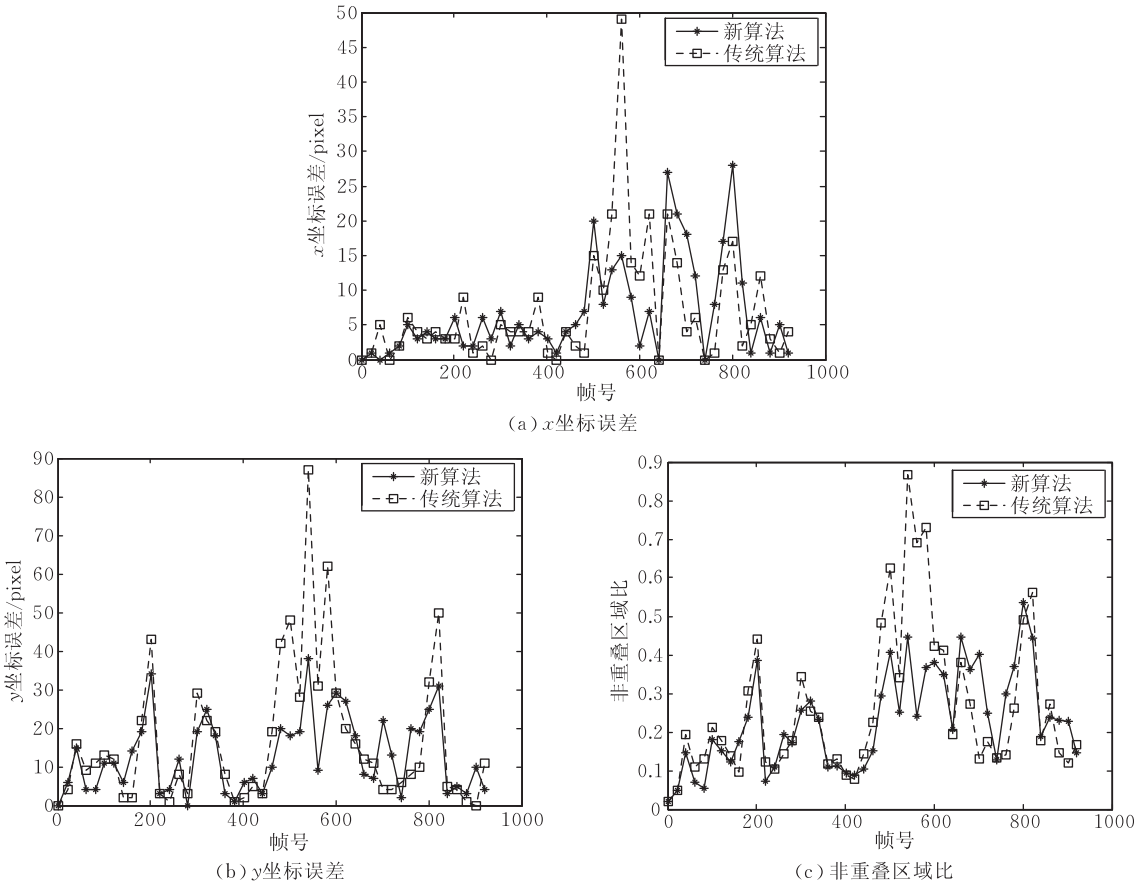


图 4　新算法对车辆的跟踪结果(从左至右、从上到下依次为第 1、160、350、520、560、610、840 和 910 帧)

从第 550 帧至第 710 帧，一辆与目标颜色相似的汽车从后面开过来，逐渐并几乎完全遮挡了目标车辆. 这辆位于目标附近的、颜色与目标相近的汽车对于目标的跟踪构成了很大的威胁. 新算法和传统算法在每一帧中的跟踪误差和平均跟踪误差分别如图 5 和表 3 所示，可见新算法的 x、y 坐标误差和非重叠区域比都要小于传统算法. 新算法和传统算法的平均跟踪时间/帧分别为 41ms 和 33(13)ms.



(a) x坐标误差　(b) y坐标误差　(c) 非重叠区域比

图 5　车辆序列中传统算法(虚线)和新算法(实线)的跟踪误差比较

表 3　车辆序列中跟踪误差(均值±标准差)和平均跟踪时间

	x 坐标误差/pixel	y 坐标误差/pixel	非重叠区域比	平均跟踪时间*/帧(单位:ms)
传统算法	7.63±10.58	16.11±15.43	0.26±0.18	41
新算法	7.01±7.96	13.24±9.64	0.23±0.12	33(13)

注：* 表示括弧中的数据是在平均跟踪时间中 GPU 计算积分图像需要的时间.

第 3 个实验是关于马路上的行人跟踪. 图像序列也是用手持式摄像机拍摄到的，共包括 350 帧，图像尺寸是 704×576. 在该场景中背景非常复杂，行人摩肩接踵，存在非常严重的遮挡，光照变化也很大. 即使如此，新算法能够在整个序列中稳定地跟踪目标——穿红色衣服的行人(图中方框所示). 基于新算法的典型跟踪结果如图 6 所示. 由于背景非常复杂，需要在相应的软件中将图像放大(zoom in)才能更清晰地看清整个场景和目标. 传统算法在第 24 帧附近就无法跟踪目标而发散；在第 180 帧我们将传统算法重新初始化，然而该算法在连续跟踪 103 帧后又发散了. 因此图 7 和表 4 只给出了新算法的跟踪误差. 新算法的每帧平均跟踪时间是 22(4)ms，其中括号中的数据表示在新算法中 GPU 计算积分图像所用的时间.

表 4　行人序列中跟踪误差(均值±标准差)和平均跟踪时间

	x 坐标误差/pixel	y 坐标误差/pixel	非重叠区域比	平均跟踪时间*/帧(单位:ms)
传统算法**	—	—	—	—
新算法	6.82±5.31	10.57±8.00	0.31±0.19	22(4)

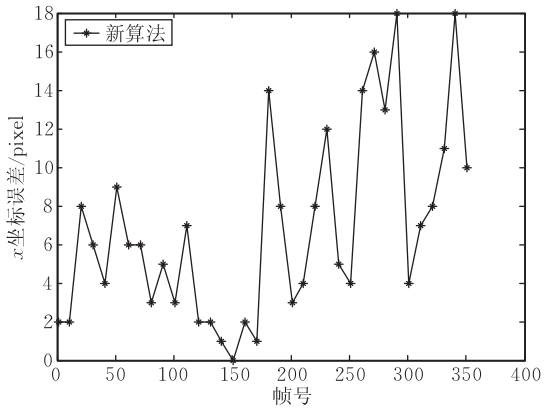
注：* 表示括弧中的数据是在平均跟踪时间中 GPU 计算积分图像需要的时间.

** 表示传统算法不能对目标进行跟踪，解释请见论文.

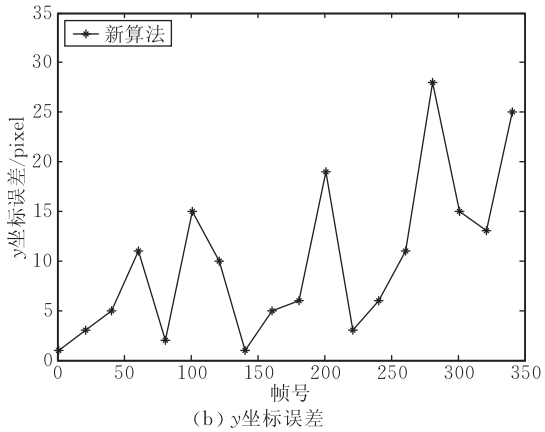
3.2　性能分析

就跟踪准确性而言，从人脸跟踪和车辆跟踪的实验可知，新的跟踪算法无论 x 坐标误差、y 坐标误差还是非重叠区域比都比传统算法小. 实验结果表

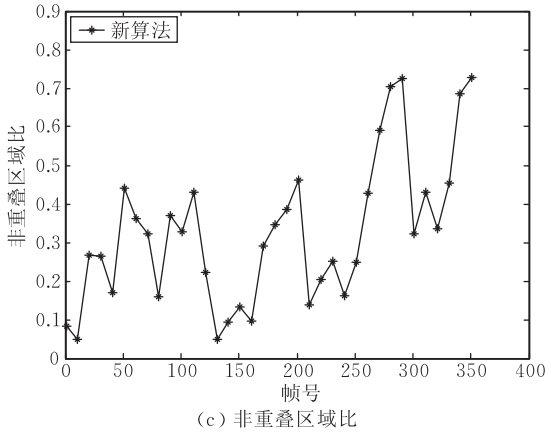
图 6　新算法对行人序列的跟踪结果(从左至右、从上到下依次为第 1、50、150、195、210、240、300 和 325 帧)



(a) x坐标误差



(b) y坐标误差



(c) 非重叠区域比

图 7　行人序列中新算法跟踪误差

明新的跟踪算法在跟踪准确性方面优于传统算法,这主要是由于论文中采用的新的颜色模型能更好地描述目标的颜色分布.就跟踪时间而言,新算法的跟踪时间显著地少于传统算法:人脸跟踪实验中比传统算法少 6ms,车辆跟踪实验中比传统算法少 8ms.这是由于论文提出的基于 GPU 的并行算法能快速地计算出颜色模型,从而在粒子滤波器算法中最耗时的观测概率密度(见式(5)、式(13))能够快速计算出来.

行人跟踪实验主要测试算法的鲁棒性.在行人序列中严重的遮挡和剧烈的光照变化使目标跟踪变得非常困难.在这种情况下,传统的基于颜色直方图的粒子滤波器算法完全失效,无法对目标进行跟踪;而论文中提出的新算法尽管跟踪误差有所增加,仍然能够在整个序列中稳定地跟踪目标.这说明新算法在鲁棒性方面优于传统算法.

在实验中传统算法的粒子数取为 200，如果进一步增加粒子数，例如增加到与新算法的粒子数相同，即 500，传统算法在人脸实验和车辆实验中跟踪准确性有所改善，但仍然无法跟踪行人目标. 与此同时传统算法的跟踪时间大大增加，人脸跟踪和车辆跟踪时间分别是 76ms 和 103ms，远远多于新算法的跟踪时间(分别是 24ms 和 33ms).

4 结 论

粒子滤波器能够以统一的理论框架处理非线性非高斯问题，因而成为目标跟踪领域一种强有力的跟踪算法. 然而由于粒子滤波器的蒙特卡洛性质，它一般要求大量的粒子才能准确地表达目标的概率分布，因此算法的计算代价很大. 为了改善计算效率，在基于颜色的粒子滤波器跟踪中研究者不得不以粗略地表达目标的颜色分布和/或减少粒子数目为代价，因此会影响算法的跟踪性能甚至使跟踪失败.

本文提出的算法在解决这一问题方面前进了一步. 我们的算法采用了一种自适应剖分颜色空间的模型，能够准确地表达目标的颜色分布. 为了能够快速地计算颜色模型，提出了一种推广的积分图像，利用该积分图像我们能够通过简单的数组索引操作快速地计算出每一个子空间的像素数目、高斯分布和协方差矩阵. 然而构造积分图像本身是一个耗时的过程，为此提出了一种并行算法，该算法能够在 NVIDIA G80 GPU 上快速地计算出积分图像. 我们提出的新的目标跟踪算法不但计算速度快，而且跟踪性能也得到了较大的提高，通过同传统的基于直方图的粒子滤波器算法的比较验证了我们的结论.

由于粒子滤波器算法估计目标状态的后验分布，能够通过大量的粒子保持对目标状态的多种假设，因此对光照变化、短时遮挡等具有较强的处理能力. 即使如此，剧烈的光照变化或长时间的遮挡仍然可能使跟踪失败. 我们将来的工作是进行多传感器或多信息融合，除了颜色信息，进一步考虑其他信息如声音信息、目标的结构知识等，以便能够进行更为鲁棒的目标跟踪.

致 谢 感谢审稿专家中肯的审稿意见!

参 考 文 献

[1] Doucet A, Godsill S, Andrieu C. On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing, 2000, 10(3): 197-208

[2] Li P, Zhang T, Pece A E. Visual contour tracking based on particle filters. Image and Vision Computing, 2003, 21(1): 111-123

[3] Nummiaro K, Koller-Meier E B, Van Gool L. An adaptive color-based particle filter. Image and Vision Computing, 2003, 21(1): 100-110

[4] Perez P, Hue C, Vermaak J, Gangnet M. Color-based probabilistic tracking//Proceedings of the European Conference on Computer Vision. Copenhagen, Denmark, 2002: 661-675

[5] Jacquot A, Sturm P, Ruch O. Adaptive tracking of non-rigid objects based on color histograms and automatic parameter selection//Proceedings of the IEEE Workshop on Motion and Video Computing, Breckenridge, USA, 2005: 103-109

[6] Town C. Multi-sensory and multi-modal fusion for sentient computing. International Journal of Computer Vision, 2007, 71(2): 235-253

[7] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Kauai Marriott, Hawaii, USA, 2001: 511-518

[8] Yang C, Duraiswami R, Davis L S. Fast multiple object tracking via a hierarchical particle filter//Proceedings of the IEEE Conference on Computer Vision. Beijing, China, 2005: 212-219

[9] Wang J, Chen X, Gao W. Online selecting discriminative tracking features using particle filter//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. San Diego, CA, USA, 2005: 1037-1042

[10] Li Pei-Hua. An improved Mean Shift algorithm for object tracking. Acta Automatic Sincia, 2007, 33(4): 347-354(in Chinese)
(李培华. 一种改进的 Mean Shift 跟踪算法. 自动化学报, 2007, 33(4): 347-354)

[11] Comaniciu D, Meer P. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(5): 603-619

[12] Wang H, Li P, Zhang T. Proposal of novel histogram features for face detection//Proceedings of the International Conference on Advances in Pattern Recognition. Bath, United Kingdom, 2005: 334-343

[13] Porikli F. Integral histogram: A fast way to extract histograms in Cartesian spaces//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. San Diego, CA, USA, 2005: 829-863

[14] Blelloch G E. Prefix sums and their applications//Reif J H ed. Synthesis of Parallel Algorithms. San Francisco, CA, USA: Morgan Kaufmann, 1993: 35-60

LI Pei-Hua, born in 1971, Ph. D., professor. His research interests include image processing and pattern recognition etc.

Background

The research of the paper is supported by National Natural Science Foundation of China under grant Nos. 60673110 and 60973080.

It is well known that color is common and highly useful information in object tracking, due to its invariance to translation and rotation, and robustness to pose variation and partial occlusion. Traditional color histogram based particle filter suffers from contradiction between accurate color density modeling and efficient computation. Concretely speaking, if the authors try to represent color distribution more accurately, for example, with a larger number of bins in histogram, then computational cost increases considerably; otherwise, very coarse description of color density may decrease performance of particle filter tracker and even lead to tracking failures.

The paper focuses on addressing the problem by proposing a novel algorithm. The authors propose to use a new color model based on clustering, which automatically partitions a color space into several sub-spaces, and considers both the number of pixels falling into each sub-space and color distribution in each sub-space with Gaussian. In order for fast computation of the color model, we present extended integral images, upon which we can compute in simple array read operations the pixel number, the mean vector and the covariance matrix of each sub-space. The computation of integral images on CPU is time-consuming, thus a parallel algorithm is proposed based on NVIDIA G80 GPU for very quick construction of the integral images. Experiments show that the proposed algorithm is superior to the traditional histogram based particle filter in both average tracking time and performance.