基于 CUDA 的数字化放射图像重建算法*

朱 爽, 常晋义
(常熟理工学院 计算机科学与工程学院, 江苏 常熟 215500)

摘 要: 为了提高重建图像的速度及质量,利用 CUDA(compute unified device architecture)架构下 GPU(graphic processing unit)的多核并行运算能力,将光线投射的几何变换、场景遍历和渲染三个步骤在可编程图像硬件中实现,降低模拟所需的时间;利用 3D 纹理、光线程基元的同步遍历机制及不透明度提前终止,在不影响成像质量的前提下,减少生成最终模拟效果所需的时间。实验结果表明,该算法不仅可以提高重建的速度,而且成像质量较好。

关键词: 统一计算设备架构; 数字化放射图像重建; 并行技术
中图分类号: TP391.41 **文献标志码**: A **文章编号**: 1001-3695(2014)05-1577-04
doi:10.3969/j.issn.1001-3695.2014.05.072

Digital reconstruction radiograph rendering algorithm based on CUDA

ZHU Shi, CHANG Jin-yi
(*School of Computer Science & Engineering, Changshu Institute of Technology, Jiangsu Changshu* 215500, *China*)

Abstract: In order to improve the speed and quality of reconstruction image, making use of multi-core parallel of GPU based on CUDA, this paper actualized three steps of ray-casting which was geometric transform, scene traversal, rendering through programmable graphics hardware, it could reduce the simulation time required. By 3D texture, synchronous traversing mechanism of ray thread element and opacity early termination, it could reduce the time required for the final simulation results without affecting the imaging quality. The experimental results show that, the algorithm can complete reconstruction of digital radiography image of high speed and high quality.

Key words: CUDA(compute unified device architecture); digital reconstructed radiography(DRR); parallel technology

数字化放射图像重建(DRR)是一种通过模拟 X 光线的衰减和曝光过程,产生类似于 X 光片成像效果的直接体绘制技术。其主要原理是依据 X 光线在不同介质中传播时由于光电吸收和散射产生的能量损耗不同而成像。DRR 以体绘制中的光线投射法为基础,根据不同器官组织对 X 光线的吸收和散射的不同,设置相应的 X 光线衰减系数作为绘制过程中的传输函数。直接体绘制技术按照不同的重采样方法分为图像空间扫描法(如 ray-casting 法)、物体空间扫描法(如 cell projection 和 splatting)。光线投射法(ray-casting)能够产生高质量的图像,但绘制速度较慢,交互性不够;滚雪球法(splatting)利用预积分足迹表获得交互的绘制速度,但不适合 DRR。为了改善体绘制的绘制速度和交互性,提出了不少的加速方法,如空体元跳跃、不透明度光线提前终止、错切变换等多种改进算法。DDR 的算法中,石教英等人[1]提出的错切和滚雪球结合的加速算法,在平行投影条件下加速效果明显,其光线投射算法绘制速度提高到原来的 5.6 倍。LaRose[2]提出了基于图像绘制的 Transgraph 法,牺牲了精度,产生的图像走样现象严重。Wang 等人[3]提出的基于圆柱调和映射快速 DRR 技术,没有达到实时或交互绘制的要求。Krüger 等人[4]提出了利用 GPU 来加速体绘制,受制于硬件,加速效果有限。陈为[5]提出了硬件加速的 DRR 重建算法,具有较好的交互性,不过提升速度受到硬件条件的约束。

随着可编程图形硬件的发展,特别是 CUDA 架构的推出,为三维数据场数据的大规模并行计算提供了条件。本文利用 CUDA 下 GPU 的多核运算能力,将光线投射的几何变换、场景遍历和渲染在硬件中实现,利用 3D 纹理及不透明度光线提前终止,减少生成最终模拟效果所需的时间,改善成像效果。

1 算法基础

1.1 光学模型

DRR 是模拟 X 光线衰减的一个过程,X 光线根据穿过的组织不同,被吸收的量也不同,所以采用光线吸收模型进行模拟,有如下公式:

$$I(s) = I_0 \exp\left(-\int_0^s \tau(x)\,\mathrm{d}x\right) \tag{1}$$

其中:I_0 为 X 光线进入三维数据场时($s=0$)的光线强度,$\tau(x)$ 是光线强度的衰减系数,定义了沿光线投射方向 x 处的光线吸收率。若将 0 到 S 区间等分为 n 个子区间,每个子区间取一个采样点 x_i 得到 $\tau(x_i)$(其中$(i-1)\times\Delta x \leqslant x_i \leqslant i\times(\Delta x)$),令 $x_i = i\times\Delta x$,则式(1)的离散形式如(2)所示。

$$I(s) = I_0 \exp\left(-\sum_{i=0}^{n}\tau(i\times\Delta x)\times\Delta x\right) \tag{2}$$

其中:$\tau(i\times\Delta x) = m_i\rho$,$m_i$ 为质量衰减系数,ρ 为物体密度。在 DRR 中,光线强度的衰减与光线穿过物体组织的组成有关,本文采用 CAI 提出的分段线性函数来模拟各种人体器官组织对

收稿日期: 2013-07-04; **修回日期**: 2013-08-16 **基金项目**: 江苏省自然科学基金资助项目(BK2012209)

作者简介: 朱爽(1979-),男,江苏常熟人,讲师,硕士研究生,主要研究方向为计算机软件与理论、三维数据可视化(zhus@cslg.cn);常晋义(1955-),男,教授,主要研究方向为空间决策支持系统、数据库安全技术、软件工程、GIS.

光线吸收的特性。对于每个部分的 m_i,用一个线性百分比组合来表示不同器官组织所占比例,定义为

$$m_i \rho = \sum_{j=0}^{n} m_{ij} \rho_j \tag{3}$$

其中:m_{ij}是对应密度为 ρ_j 的物体的质量衰减系数。

1.2 光线投射

DRR 采用经典的光线投射算法实现。光线投射法是基于图像序列的直接体绘制算法。从图像的每个像素沿着固定的方向(通常是视线方向)发射一条光线,光线穿越整个图像序列,并在这个过程中对图像序列进行采样获取颜色信息,同时依据光线吸收模型将颜色值进行累加,直至光线穿越整个图像序列,最后得到的颜色值就是渲染图像的颜色。将所有像素点的颜色拼接起来,就得到了一幅完整的可视化图像。

1.3 CUDA 架构

CUDA 架构是一种将 GPU 作为数据并行计算设备的软硬件体系,其编程模型的核心是 grid、block、thread 三层结构,以线程网格的形式组织,每个 grid 由若干个线程块(block)组成,每个 block 又由若干个线程(thread)组成。程序以 block 为执行单位,各个 block 并行执行,相互之间无法通信,同一个 block 中的线程通过共享存储器来相互协作。

从光线投射算法中每个像素出发,沿着固定方向发射的光线之间的计算过程是相互独立的,计算方式是相同的,具有高度的并行性。根据其特点,整个屏幕看成一个 grid,根据区域特点屏幕划分为若干区域,每个区域看成一个 block,每个像素看成一个 thread,由此将光线投射法利用 CUDA 的三层架构进行实现,通过 CUDA 强大的并行运算能力提高绘制速度。

2 算法描述

2.1 算法流程

算法主要分成两个组成部分,一个是 CPU 部分(host 部分),另一个是 GPU 部分(device 部分)。

CPU 上的算法步骤如下:a)读取三维数据场 raw 文件为一个个 unsigned char 阵列,存储在 device memory 中;b)OpenGL 环境的初始化,建立 PBO(OpengGL 像素缓存对象),PBO 与 GPU 显存之间的绑定,设置模型变换矩阵,设置视线方向;c)根据数据场大小,计算 CUDA 所需的 block 及 thread 的大小。

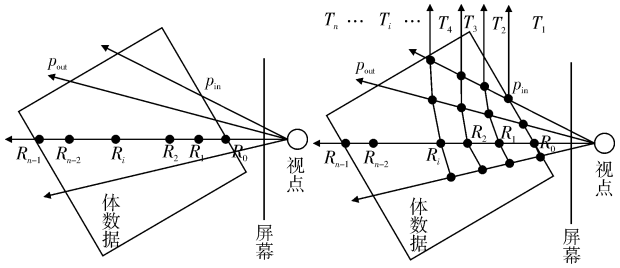
GPU 上的算法流程如下:a)CUDA 环境的初始化,建立 CUDA 的 3D array 及 3D texture,device memory 中的三维数据场数据读入 3D array,设置纹理参数,并将 3D textrue 和 3D array 进行绑定;b)设定前置参数,根据线程索引计算对应的屏幕像素位置、视点坐标;c)采用包围盒法,根据射线方向,分别求出光线进入数据体和离开数据体的交点坐标 p_{in} 和 p_{out} 以及与相机的距离;d)由前往后,沿射线方向依次按照预设的采样间距进行采样,得到采样点的颜色和不透明度,按照由前往后的方式进行累加合成,并利用不透明度提前截止,若不透明度值接近 1.0,停止计算,否则继续采样下一个点,直至到达离开点为止;e)将最终累加的颜色和不透明度值写入 PBO。

至此,GPU 上工作完成,CPU 再通过 OpenGL 环境下的 API 直接将 PBO 中的数据在屏幕上显示。

2.2 算法关键

2.2.1 光线程基元

为了改变传统光线投射法由于采用逐条光线遍历体数据,不利于硬件加速的弊端,提出光线程基元的概念。设体数据的包围盒为 box,如图 1 所示,从视点 E 出发的任一光线,与 box 通常有两个交点,记为 p_{in} 与 p_{out}。对光线采样时,采用沿着光线从前至后,等间距采样方式进行,采样间距是 p_{step},则一条光线上总的采样数 $n = (p_{out} - p_{in})/p_{step}$。定义 $R_i(i = 0, 1, \cdots, n-1)$ 为某条光线 ray 上的光线程基元,则某个像素的光亮度由这个像素对应的光线上的所有光线程基元共同决定。采用光线程基元可以利用 CUDA 架构下的每个 thread 模拟一条光线,在进行光线遍历时,若干个 thread 同步并行运行,提高遍历的效率。



(a)传统光线投射法光线遍历机制 (b)改进的光线投射法光线遍历机制

图 1 传统和改进的光线投射法光线遍历机制

2.2.2 纹理的使用

为了加快在 GPU 上的绘制,充分利用硬件的加速,采用纹理存储器来存储读入的数据。通过将三维数据场看成若干个二维数据场的连续组合,而每个二维数据场就是一张 2D 的灰阶图,则整个数据场可以利用一个三维 CUDA 数组存储,即 3D array(x, y, z),其存储的是三维数据场中第 z 层,平面上第 x 行、第 y 列上某点的灰度值。利用 cudaBindTextureToArray 函数绑定该 3D array 至 3D texture 内存。所用的纹理缓存的大小由需要存储的三维数据场的大小及存储每位灰度所需的存储空间的乘积决定,即 sizeof(3D texture) = width × height × depth × 1 Byte。其中 width、height、depth 分别是数据场的宽、高、深。DRR 需要存储的都是灰度值,故只需要 1 Byte,8 bit 来存储 256 阶灰度值。为了保证纹理绑定时纹理的坐标与纹理的尺寸无关,采用归一化方式对纹理进行寻址,将三个维度上的 x、y、z 分别映射在[0,1]范围,并对纹理的滤波采用三线性插值模式,保证了成像更加平滑自然。在光线遍历时,根据光线程基元中点的空间坐标获取相应的纹理信息,并进行相应累积。具体实现步骤如下:

a)根据数据场大小声明 CUDA 数组,分配空间,采用 cudaMalloc3DArray(d_volumeArray, channelDesc, volumeSize)。其中 volumeArray 是一个 CUDA array,大小为 volumeSize,存储数据的类型为 channelDesc。

b)声明纹理参照系,指定读取的纹理数据类型、维度及是否归一化操作。采用 texture〈volumeType, 3, cudaReadModeNormalizedFloat〉,设置纹理坐标为归一化的浮点型。

c)设置运行时纹理参照系属性,设定纹理的寻址方式等,对步骤 b)声明的 textrue 类型的变量 tex 相应属性进行设置,将 normalized 置为 true,即进行归一化;filterMode 置为 cudaFilterModeLinear,即线性插值方式。

d)纹理绑定,根据设置的纹理参照系,将数据与纹理绑定。cudaBindTextureToArray(tex,d_volumeArray,channelDesc)。

e)纹理拾取,利用 tex3D(tex, pos.x, pos.y, pos.z)获取所对应纹理坐标的纹理值,并由相应的累积函数累积。

2.2.3　前置参数的设置

在进行绘制前需要进行一些基本参数的设置。其中最重要的是根据线程索引计算其对应的光线在屏幕上成像的坐标,即对应的像素点位置。线程的索引 index 通过式(4)进行计算。

$$index = (blockindex) \times (blocksize) + (threadindex) \tag{4}$$

由式(4)求得的 index 有 x、y 两个分量,为了将其转换至规范的[-1,1]范围,通过式(5)(6)进行转换。

$$u = (index.x/imageW) \times 2 - 1 \tag{5}$$

$$v = (index.y/imageH) \times 2 - 1 \tag{6}$$

其中:image$W$ 与 image$H$ 代表视平面的宽度与高度。相关空间设置如图 2 所示。其中 box 为包围盒,视平面为(0,0,2),大小为 2 ×2,视点为(0,0,4)。

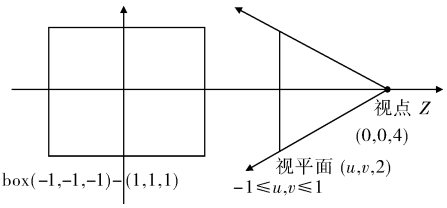


图 2　空间关系图

2.2.4　光线求交及颜色累积

对任意一条光线,为其设置两个最基本的参数,起点坐标(origin)和视线方向(direction),利用向量表示法,任意一条光线可以表示为式(7),其中 t 表示在视线方向上的位移。

$$Ray_i = eyeRay.o + eyeRay.d \times t \tag{7}$$

光线与包围盒求交采用 Owen 提出的方法,求出一条光线和包围盒的两个交点 p_{in} 和 p_{out},然后以 p_{in} 为起始点,以 p_{step} 为间距沿着视线方向步进。每前进一步,通过式(8)计算出光线上的某个采样点在空间中的坐标。

$$pos = eyeRay.o + eyeRay.d \times p_{in} + eyeRay.d \times p_{step} \tag{8}$$

其中:eyeRay.o 是光线的起始坐标;eyeRay.$d \times p_{in}$ 是光线入点的坐标;eyeRay.$d \times p_{step}$ 是光线的方向。由于该坐标是位于包围盒中的,坐标值介于(-1, -1, -1)和(1,1,1)之间,通过转换,将坐标值转换至纹理坐标[0,1]范围内。这样,根据点的坐标值提取相应纹理坐标的纹理值,即为该点的灰度值,然后进行不透明度和色彩的累加,将累加的结果直接写入 PBO 中。

2.2.5　光线的并行遍历

利用 CUDA 架构的并行特点,每个 thread 模拟一条光线,每次进行光线遍历时,若干个 thread 沿着视线方向同步逐层并行运行,如图 1(b)所示,每次遍历即处理一层 T_i。每绘制一层光线基元,相当于绘制一个恰好覆盖视平面的长方形,并将得到的不透明度和颜色积累的结果以纹理的形式附加在该长方形上。这种光线的遍历充分利用了 CUDA 三层架构的特点,不是逐条光线遍历,而是对所有的光线逐层遍历,通过硬件加速,大大提高了遍历速度。具体过程如下:

```
for  任一层光线程基元
  for 任一光线
    if 光线 in 包围盒中 and 不透明度未达到阈值
    根据 index 计算其对应的视平面坐标
        根据其光线方程计算空间中的坐标,并转换至纹理坐标,
        拾取其纹理值,并进行色彩和不透明度累积,写入 PBO
    end
  next
next
```

3　实验结果及分析

本文实验用 Intel Core2 双核 T5550 CPU,2 GB DDR Ⅱ 内存,NVIDIA GeForce 8400M GS,512 MB 显存,利用 VC ++ 6.0 和 CUDA SDK 3.1,OpenGL 3.1 库,在 Windows XP 环境下实现了上面描述的 DRR 重建算法。

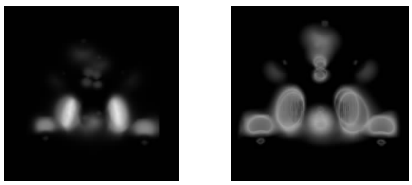
3.1　不同体数据的绘制结果

本文采用了多个体数据进行测试,有人体头部、足部及负电位高蛋白质分子三种。表 1 列出了所采用的三维数据场的具体参数信息。

表 1　实验数据场基本信息

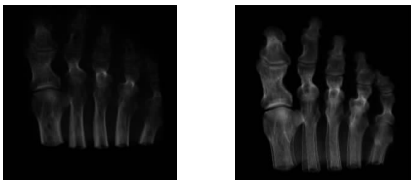
体数据	大小	大小	平均采样率
负电位高蛋白质分子	64 ×64 ×64	256 KB	64
足部	256 ×256 ×256	16 MB	256
头部	256 ×256 ×256	14.0 MB	256

图 3 ~5 分别是对上述三个数据场采用改进的基于 GPU 的光线投射法与本文算法进行绘制的结果比较。



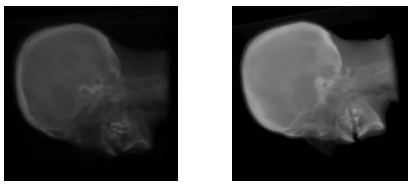
(a)基于GPU的光线投射　(b)基于CUDA的光线投射

图 3　负电位高蛋白质分子重建图像



(a)基于GPU的光线投射　(b)基于CUDA的光线投射

图 4　足部重建图像



(a)基于GPU的光线投射　(b)基于CUDA的光线投射

图 5　头部重建图像

从上述三组绘制成像效果来看,本文算法和改进的基于 GPU 的光线投射法比较,在细节表现上及成像的清晰度上来看效果更好,主要是采用了归一化线性滤波,图像的过渡更加自然平滑。

3.2　性能分析

表 2、3 分别为改进的基于 GPU 的光线投射算法[4] 和本文算法在相同条件下以相同数据场作测试后的相关数据的统计对比。

由表 2、3 可以看到,本文算法和基于 GPU 的算法在纹理生成时所需的时间均与数据场规模有关,本文算法利用 CUDA 的 3D 数组来存取纹理,基于 GPU 的算法采用的是一维数组来存取纹理,在纹理的生成及保存的时间开销上,本文算法有很大的优势,而且数据场规模越大,优势越明显。从表 1 的比对中发现,当数据场规模是 64 ×64 ×64 时,本文算法纹理生成时间是基于 GPU 算法的 0.6 倍;当数据场规模是 256 ×256 ×256 时,本文算法纹理生成时间是基于 GPU 算法的 0.2 倍。在光线的遍历上,本文引入了光线程基元,相比较基于 GPU 的光线投射算法,光线遍历时间变为原来的 1/4 左右。从上述实验数

据可以看出,本文算法在数据场规模较大的情况下,纹理生成的优势更加明显。在纹理生成及光线遍历上的优势,充分体现了图形芯片单指令多数据流水处理的特点及 CUDA 架构并行的优点,对整个绘制流程速度的提升有显著作用。整个算法的执行时间相比较于基于 GPU 的重建算法有 10% 左右的提高。

表 2 基于 GPU 的光线投射法实现过程数据 /s

体数据	纹理生成	光线遍历	总计(从数据读取到最终成像)
头部	0.062	0.044	0.812
足部	0.067	0.048	0.832
负电位高蛋白质分子	0.002	0.032	0.754

表 3 本文算法实现过程数据 /s

体数据	纹理生成	光线遍历	总计(从数据读取到最终成像)
头部	0.012	0.011	0.719
足部	0.014	0.011	0.725
负电位高蛋白质分子	0.001 2	0.009 7	0.672

4 结束语

本文利用 CUDA 架构的并行技术和可编程硬件技术,采用光线投射算法进行数字化放射图像的重建。与基于 GPU 的光线投射法比较,成像效果上本文算法在细节的展示上更能突出细节特点。由于 CUDA 架构具有并行的特点,在光线遍历时具有时间优势,在纹理的生成上,随着数据场规模的增大,并行的特点更能得到发挥,在时间开销上得到了很好的体现,整个算法实现的时间提高了 10% 左右。随着可编程硬件技术的发展,今后的工作主要集中在成像质量的提高、在交互性方面的提升以及对于成像中感兴趣部分加强显示方面的研究。

参考文献:

[1] 石教英,蔡文立. 科学计算可视化算法与系统[M]. 北京:科学出版社,1996.

[2] LaROSE D. Iterative X-ray/CT registration using accelerated volume rendering [D]. Pittsburgh: Robotics Institute, Carnegie Mellon University, 2001.

[3] WANG F, DAVIS T E, VEMURI B C. Realtime DRR generation using cylindrical harmonics[C]//Proc of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention. 2002:671-678.

[4] KRÜGER J, WESTERMANN R. Acceleration techniques for GPU-based volume rendering [C]//Proc of the 14th IEEE Visualization Conference. [S.l.]: IEEE Press, 2003:287-292.

[5] 陈为. 硬件加速的数字化放射图像重新算法[J]. 计算机辅助设计与图形学学报,2005,17(14):683-687.

[6] 徐赛花. 基于 CUDA 的三维数据场并行可视化[J]. CT 理论与应用研究,2011,20(1):47-54.

[7] CAI W. Transfer function in DRR volume rendering[C]//Proc of the 13th International Congress and Exhibition on Computer Assisted Radiology and Surgery. 1999:284-289.

[8] SHEROUSE W, NOVINS K, CHANEY E L. Computation of digitally reconstructed radiographs for use in radiotherapy treatment design [J]. International Journal of Radiation Oncology, Biology and Physics, 1990, 18(3):651-658.

[9] SCHERLH, KECK B, KOWARSCHIK M, *et al*. Fast GPU-based CT reconstruction using the common unified device architecture(CUDA) [C]//Proc of Nuclear Science Symposium and Medical Imaging Conference. Washington DC: IEEE Computer Society, 2007: 4464-4466.

[10] LU Yu-qiang, WANG Wei-ming, CHEN Shi-fu, *et al*. Accelerating algebraic reconstruction using CUDA-enabled GPU[C]//Proc of the 6th International Conference on Computer Graphics, Imaging and Visualization. Washington DC: IEEE Computer Society, 2009: 480-485.

[11] 唐泽圣. 三维数据可视化[M]. 北京:清华大学出版社,1999.

[12] 张舒,褚艳利. GPU 高性能运算之 CUDA[M]. 北京:中国水利出版社,2009.

[13] 仇德元. GPGPU 编程技术——从 GLSL、CUDA 到 OpenCL[M]. 北京:机械工业出版社, 2011.

(上接第 1573 页)生成二值描述,进行汉明匹配,使得跟踪匹配更加简单和准确。在整个跟踪过程中压缩矩阵始终保持不变,通过积分图像的思想获得图像灰度信息,并且采用汉明匹配,因而整个算法运算速度较快。由于是在线更新特征模板,所以能够快速地对目标的变化进行响应。实验结果表明,本文提出的算法具有较好的鲁棒性和实时性。

参考文献:

[1] YANG Han-xuan, SHAO Ling, ZHENG Feng, *et al*. Recent advances and trends in visual tracking: a review[J]. Neurocomputing, 2011, 74(18): 3823-3831.

[2] PORIKLI F, YILMAZ A. Video analytics for business intelligence [M]. Berlin: Springer, 2012:3-41.

[3] ZHANG Kai-hua, ZHANG Lei, YANG Ming-hsuan. Real-time compressive tracking [C]//Lecture Notes in Computer Science, vol 7574. Berlini: Springer-Verlag, 2012: 864-877.

[4] BARANIUK R, DAVENPORT M, De VORE R, *et al*. A simple proof of the restricted isometry property for random matrices [J]. Constructive Approximation, 2008, 28(3): 253-263.

[5] NG A, JORDAN M. On discriminative vs. generative classifier: a comparison of logistic regression and naive Bayes[C]//Proc of Neural Information Processing Systems Conference. 2002:841-848.

[6] HEINLY J, DUNN E, FRAHM J M. Comparative evaluation of binary features [C]//Proc of European Conference on Computer Vision. 2012:759-773.

[7] CALONDER M, LEPETIT V, STRECHA C, *et al*. BRIEF: binary robust independent elementary features [C]//Proc of European Conference on Computer Vision. 2010:778-792.

[8] VIOLA P, JONES M. Rapid object detection using a boosted cascade of simple features[C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2001:511-518.

[9] DIACONIS P, FREEDMAN D. Asymptotics of graphical projection pursuit[J]. The Annals of Statistics, 1984, 12(3): 228-235.

[10] KALAL Z, MIKOLAJCZYK K, MATAS J. Tracking-learning-detection[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2010, 34(7):1409-1422.

[11] KWON J, LEE K. Visual tracking decomposition[C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2010:1269-1276.

[12] SANTNER J, LEISTNER C, SAFFARI A, *et al*. PROST parallel robust online simple tracking[C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2010:723-730.

[13] ROSS D A, LIM J, LIN R S, *et al*. Incremental learning for robust visual tracking [J]. International Journal of Computer Vision, 2008, 77(1-3):125-141.

[14] PAVANI S K, DELGADO D, FRANGI A F. Haar-like features with optimally weighted rectangles for rapid object detection[J]. Pattern Recognition, 2010, 43(1): 160-172.