计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(1):146−155 [doi: 10.15888/j.cnki.csa.008889]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 CUDA 加速的图像配准算法①

牛　彤, 刘立东, 武忆涵

(长安大学 信息工程学院, 西安 710064)
通信作者: 牛　彤, E-mail: 2216788619@qq.com

摘　要: 针对传统图像拼接算法速度较慢, 难以满足获取大分辨率全景图像的实时性要求, 本文提出一种基于 CUDA 的快速鲁棒特征 (speeded-up-robust features, SURF) 图像配准算法, 从 GPU 线程执行模型、编程模型和内存模型等方面, 对传统 SURF 算法特征点的检测和描述进行 CUDA 并行优化; 基于 FLANN 和 RANSAC 算法, 采用双向匹配策略进行特征匹配, 提高配准精度. 结果表明, 相对串行算法, 本文并行算法对不同分辨率的图像均可实现 10 倍以上的加速比, 而且配准精度较传统配准算法提高 17%, 精度最优可高达 96%. 基于 CUDA 加速的 SURF 算法可广泛应用于安防监控领域, 实现全景图像的实时配准.

关键词: 快速鲁棒特征; 统一计算设备架构; 并行加速; 快速最近邻搜索算法; RANSAC; 双向匹配; 图像配准

引用格式: 牛彤,刘立东,武忆涵.基于 CUDA 加速的图像配准算法.计算机系统应用,2023,32(1):146–155. http://www.c-s-a.org.cn/1003-3254/8889.html

Image Registration Algorithm Based on CUDA Acceleration

NIU Tong, LIU Li-Dong, WU Yi-Han

(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: Traditional image stitching algorithms are slow and fail to meet the requirements of obtaining large-resolution panoramic images in real time. To solve these problems, this study proposes an image registration algorithm based on CUDA's speeded-up-robust features (SURF) and carries out CUDA parallel optimization on the detection and description of feature points of traditional SURF algorithms in terms of GPU thread execution model, programming model, and memory model. In addition, based on FLANN and RANSAC algorithms, the study adopts a bidirectional matching strategy to match features and improve registration accuracy. The experimental results show that compared with serial algorithms, the proposed parallel algorithm can achieve an acceleration ratio of more than 10 times for images with different resolutions, and the registration accuracy is 17% higher than that of traditional registration algorithms, with an optimal accuracy of as high as 96%. Therefore, the SURF algorithm based on CUDA acceleration can be widely used in the field of security monitoring to realize the real-time registration of panoramic images.

Key words: speeded-up-robust features (SURF); computing unified device architecture (CUDA); parallel acceleration; fast library for approximate nearest neighbors (FLANN); RANSAC; bidirectional matching; image registration

1　引言

全景视频拼接技术通过图像配准、图像拼接、畸变校正、图像融合等步骤, 可快速获得宽视角、高精度、小畸变、无重影、实时性好的全景图像, 被广泛应用于智能交通、遥感探测、医学影像、计算机视觉、模式识别等领域.

图像配准作为全景视频拼接的基础, 它将传统摄像头拍摄的两个或多个图像的重合区域进行特征点相

① 基金项目: 陕西省自然科学基金 (2020JM-220)
收稿时间: 2022-04-28; 修改时间: 2022-06-01; 采用时间: 2022-06-29; csa 在线出版时间: 2022-08-26
CNKI 网络首发时间: 2022-11-15

似度配准. 经典的匹配算法有 SIFT 和 SURF (speeded-up-robust features), 之后相继出现的 ORB、BRISK 等加速优化算法以牺牲匹配准确率为代价来满足实时性要求. SURF 算法通过盒子滤波器近似 SIFT 算法中二阶高斯函数的方式对源积分图像进行卷积运算, 极大地提高了算法速度[1]. 因此, 进一步对 SURF 算法进行加速优化研究, 使其在保证高精度特征提取的同时达到实时性要求, 对全景视频监控领域具有深远意义和应用前景.

近年来, 随着图像处理器 (graphic processing unit, GPU) 的发展, 很多学者采用 GPU 并行架构对算法进行优化加速, 很大程度上提高了全景图像拼接效率. 王艳梅等[2] 基于 OpenCL 平台, 从数据传输、内存访问、负载均衡等方面优化 SURF 算法, 具有较好的可移植性. 刘金硕等[3] 利用 GPU 的内存模型和线程映射实现了 SURF算法的统一计算设备架构 (computing unified device architecture, CUDA) 加速. 徐晶等[4] 结合 SURF 算法和 SVM 算法进行特征提取和分类, 检测速度提升了 5–9 倍. 郭景等[5] 基于 OpenCL 平台提出了一种特征点检测的并行实现方法, 对旋转、缩放和模糊等处理都具有较好的自适应性. 卢嘉铭等[6] 基于 CUDA 平台进行图像拼接和融合, 实现了 4K 分辨率全景图像显示输出. 周亮君等[7] 基于 GPU 并行加速使得 SURF 算法较普通 PC 架构速度提高了约 5 倍.

本文基于 CUDA 架构并行化原理, 对 SURF 算法中计算 Hessian 响应值、非极大值抑制、确定特征点主方向、生成特征描述符等步骤进行 GPU 加速处理, 并基于 FLANN 和 RANSAC 算法进行图像配准, 采用双向匹配策略提高图像配准精度.

本文第 2 节介绍了传统 SURF 特征检测算法原理; 第 3 节阐述了基于 CUDA 的 SURF 优化算法实现过程和采用双向匹配策略的图像配准步骤; 第 4 节分析了实验结果; 第 5 节得出结论并提出展望.

2 SURF 特征匹配算法

SURF 算法是基于 SIFT 的加速鲁棒特征检测算法, 其速度比 SIFT 算法快 10 倍[8]. 主要包括特征点的检测和描述两大步骤.

2.1 特征点检测

2.1.1 计算积分图像

定义灰度图像I在任意像素点$X(x,y)$处的积分图像$I_{\sum}(x)$为: 输入图像I中原点和X构成的矩形区域内所有像素点的灰度值之和, 如式 (1) 所示, 则如图 1 所示, 任意矩形区域 $ABCD$ 的积分图像为每个顶点区域积分图像的简单求和, 如式 (2)[9]:

$$I_{\sum}(X)=\sum_{i=0}^{x}\sum_{j=0}^{y}I(i,j) \tag{1}$$

$$I_{\sum}ABCD=I_{\sum}A+I_{\sum}D-I_{\sum}B-I_{\sum}C \tag{2}$$

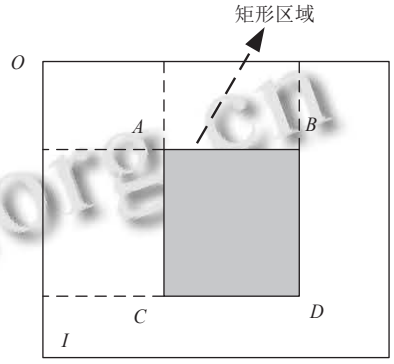


图 1 积分图像

2.1.2 构建 Hessian 矩阵

图像$I_{\sum}$上像素点$X(x,y)$在尺度σ处的 Hessian 矩阵被定义为:

$$H(x,\sigma)=\begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \tag{3}$$

其中, $L_{xx}(X,\sigma)$是图像$I_{\sum}$与高斯二阶偏导数$\frac{\partial^2}{\partial x^2}g(\sigma)$的卷积, $L_{xy}(X,\sigma)$和$L_{yy}(X,\sigma)$类似.

SURF 算法利用不同权重大小的盒子滤波器进行二维高斯滤波, 从而近似二阶高斯偏导数 (如图 2 所示), 该过程可利用积分图像降低计算成本[10].

图 3 中 9×9 盒子滤波器是尺度$\sigma=1.2$的高斯近似, 代表最高空间分辨率, 分别表示为D_{xx}、D_{yy}、D_{xy}, 则 Hessian 矩阵判别式 (DOH 值) 被定义为:

$$det(H)=D_{xx}D_{xy}-\left(\omega D_{xy}\right)^2 \tag{4}$$

为提高计算效率引入权重ω, 其值常取 0.9.

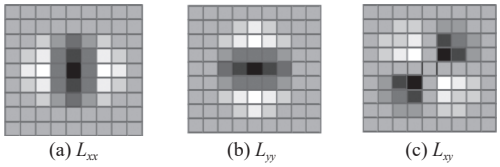
2.1.3 构建尺度空间

SURF 算法通过增加盒子滤波器的大小来构建尺度空间. SURF 尺度空间被分为若干层 (*octave*), 图 3 中 9×9 盒子滤波器为 SURF 尺度空间第一层, 用来计算最小尺度 ($\sigma_0=1.2$) 的 Hessian 响应值, 每层采样间隔 (*interval*) 增加一倍, 相邻层盒子滤波器大小按 6 个像素步长递增, 由式 (5) 便可得到如图 4 所示的 SURF 尺度空间模板大小[11].

$$L = 3 \times 2^{octave} \times interval + 1 \tag{5}$$

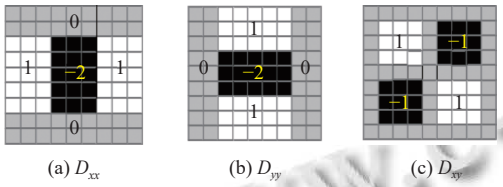
不同层对应尺度为:

$$s = \sigma_0 \times \frac{L}{9} = 1.2 \times \frac{L}{9} \tag{6}$$



(a) L_{xx} (b) L_{yy} (c) L_{xy}

图 2 高斯二阶导数模板



(a) D_{xx} (b) D_{yy} (c) D_{xy}

图 3 高斯二阶导数的近似 (盒子滤波器)

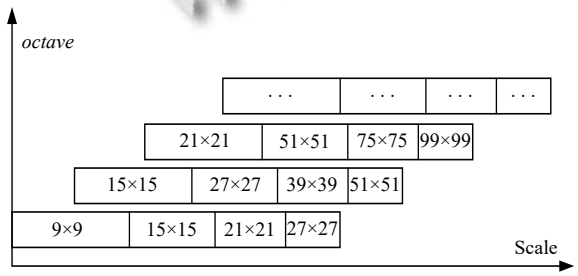


图 4 SURF 尺度空间中模板尺寸大小

2.1.4 特征点检测

SURF 算法采用 3×3×3 邻域非极大值抑制来检测特征点, 如图 5 所示, 将由式 (4) 计算得到的每个像素点 DOH 值, 与其 3×3×3 邻域中的 26 个点进行比较, 若该点的模最大, 则初步定为极值点. 然后去除不稳定和能量较弱的兴趣点, 将 $det(H)$ 值大于阈值的点精选出来作为最终的特征点[12].

2.2 特征点描述

2.2.1 确定特征点主方向

为保证图像旋转不变性, SURF 算法通过计算特征点邻域内 Haar 小波响应来确定每个特征点的主方向[13]. 如图 6 所示, 以特征点为中心, 利用尺寸为 $4s$ 的 Haar 小波模板, 计算半径为 $6s$ 的圆形邻域内 x 和 y 方向上的小波响应 d_x、d_y; 并统计沿逆时针方向滑动的 $\pi/3$ 扇形滑窗内所有特征点的水平和垂直响应之和, 得到一个局部方向向量 $(m_\omega, \theta_\omega)$. 定义所有窗口中最长的向量为特征点的主方向, 如式 (9) 所示[14]:

$$m_\omega = \sum_\omega d_x + \sum_\omega d_y \tag{7}$$

$$\theta_\omega = \arctan \frac{\sum_\omega d_x}{\sum_\omega d_y} \tag{8}$$

$$\theta = \theta_\omega \left| \max \{m_\omega\} \right. \tag{9}$$

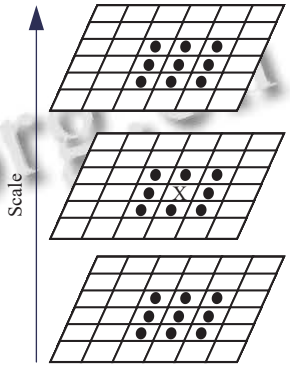


图 5 3×3×3 邻域非极大值抑制

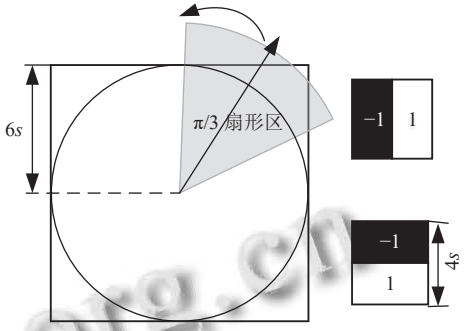


图 6 SURF 算法主方向确定

2.2.2 生成特征点描述符

如图 7 所示, 以特征点为中心, 特征点主方向为 x 轴, 构建一个边长为 $20s$ 的方形区域, 并将其等分成 4×4 个边长为 $5s$ 的正方形子区域, 计算每个子区域中模板尺寸为 $2s$ 的 Haar 小波响应, 得到 4 维特征描述子向量:

$$V = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y| \right) \tag{10}$$

其中, d_x、d_y 为 x 和 y 方向的 Haar 小波响应[12].

如此, 16 个子区域可得到该特征点的 16×4=64 维描述符. 对 SIFT 算法的 128 维描述符进行降维, 大大提高了特征检测速度.

3 SURF 并行优化

CUDA 是 NVIDIA 公司推出的并行计算平台, 兼

容 CPU 逻辑处理能力和 GPU 并行计算能力, 可实现 CPU+GPU 异构并行优化[15]. GPU 通过 CUDA 流进行异步数据传输, 为不同 CUDA 流分配不同任务, 可实现数据并行和任务并行, 大大缩短程序执行时间[16]. CUDA 有 6 种内存模型: 全局内存、本地内存、常量内存、共享内存、纹理内存和寄存器内存, 从左到右内存的读写速度递增[17].

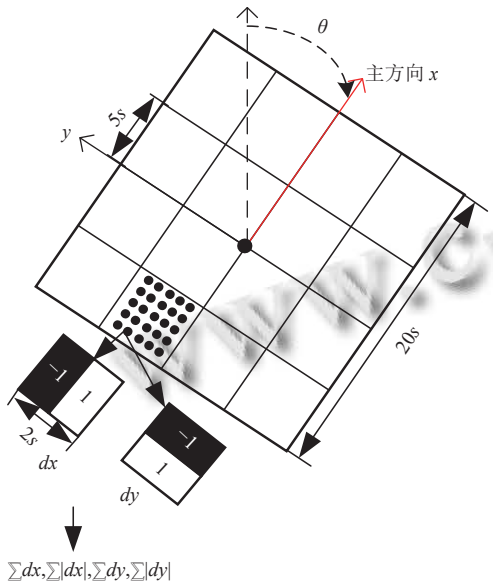


图 7 生成特征描述符

基于上述 SURF 算法关键技术的研究, 为提高算法速度, 本文基于 CUDA 并行架构编程模型, 通过将一些固定参数设置为常量内存、将各个步骤间相互传递使用的参数设置为共享内存、将积分图像设置为纹理存储结构, 从而加快参数传输和内存访问; 并创建计算 Hessian 响应、非极大值抑制、确定特征点主方向、计算特征描述符等 Kernel 函数, 实现数据和任务并行加速处理, 其并行加速优化流程如图 8 所示.

3.1 特征点检测

3.1.1 基于前缀加法的积分图像计算

前缀加法即通过分段计算数组元素之和, 应用到图像处理中, 即将图像行和列分成多个段, 构造多层扫描数据, 计算并更新从行或列起始位置到该段末的像素和, 行与行的计算并列, 列与列的计算并列, 从而有效减少计算冗余[17]. 具体算法流程如算法 1.

$$I_{\sum}(x,y)=I_{\sum}(x-1,y)+i(x,y) \tag{11}$$

$$I'_{\sum}(x,y)=I_{\sum}(x,y-1)+I_{\sum}(x,y) \tag{12}$$

算法 1. 基于前缀加法的积分图像并行计算流程

1) 输入源图像, 将其转化成灰度图像并绑定为纹理内存.
2) 从左向右遍历源图像行像素点, 计算所遍历行像素点灰度和, 则积分图像该位置像素为式 (11).
3) 从上向下遍历积分图像列像素点, 计算所遍历列像素点灰度和, 则积分图像当前遍历位置像素为式 (12).
4) 将积分图像绑定为纹理内存.

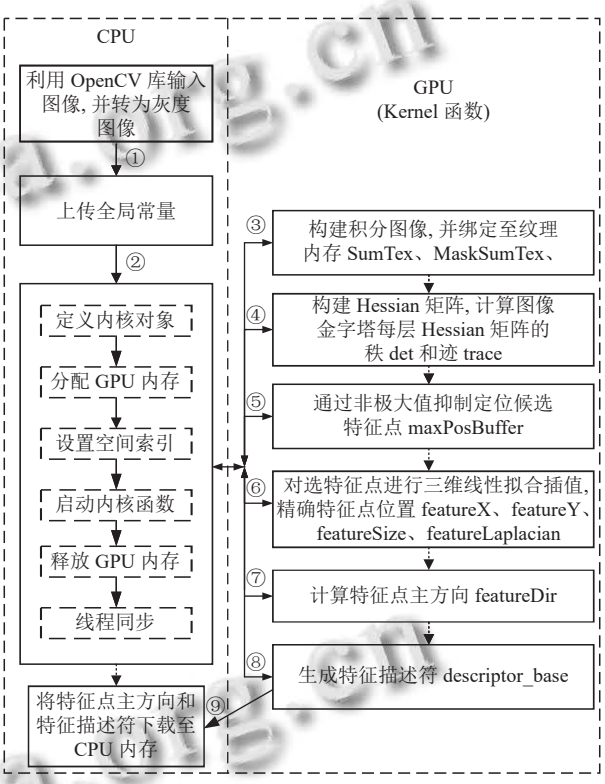


图 8 SURF 算法并行加速优化流程

3.1.2 计算 Hessian 响应值

SURF 算法在构建尺度空间、计算 Hessian 矩阵以及计算特征点描述符等步骤中均可利用积分图像简化算法步骤. 因此, 本文算法将源图和积分图像绑定为纹理内存, 从而通过坐标较快地获取 texture 存储器的数据, 加快 Kernel 函数数据传输.

尺度空间金字塔不同层计算相互独立, 可构建不同尺度的 Hessian 矩阵; 同层不同像素点的 Hessian 响应值计算也相互独立. 因此, 本文基于图像级和像素级同时并行优化, 设计如下算法流程如算法 2.

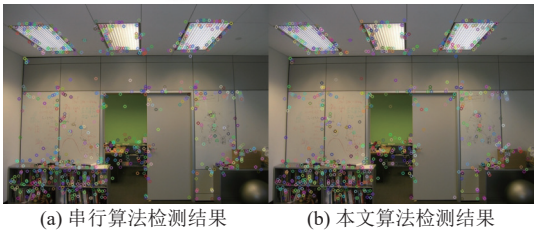
算法 2. 计算 Hessian 矩阵响应值算法

1) 在 CPU 端将源图及相关参数上传至 GPU 全局内存.

2) 创建不同方向具有不同权重大小的盒子滤波器模板, 并将其存储于常量内存.
3) 为 Kernel 函数分配与图像大小相同的索引指数, 即为每个像素点分配一个内核线程, 用于并行计算不同像素点在x方向、y方向及xy方向的高斯二阶偏导数, 进而确定每个像素的 Hessian 响应值.
4) 为金字塔每层图像分配内存空间, 进行不同层 Hessian 响应值的并行计算.

3.1.3 非极大值抑制

特征点检测实质就是进行 3×3×3 邻域非极大值抑制, 然后通过三维线性插值法确定亚像素级特征点位置[18]. 该过程是在尺度空间像素级上进行的, 故而同理, 进行像素级和图像级的并行加速优化, 设计如算法 3 所示流程. 为进一步提高特征点检测效率, 本文算法利用积分图像对边界效应引起的边界空白区域像素进行预处理, 简化特征点检测步骤; 通过原子操作, 确保在多个并行线程间共享内存的读写保护. 特征点检测结果如图 9, 图 9(a) 为文献 [9] 串行算法检测结果, 图 9(b) 为本文算法检测结果.



(a) 串行算法检测结果 (b) 本文算法检测结果

图 9 特征点检测结果

算法 3. 非极大值抑制算法流程

1) 创建 Haar 小波滤波器模板并设置为常量内存.
2) 创建掩码积分图并绑定为纹理内存.
3) 创建 Kernel 函数, 为每个像素分配索引空间, 遍历中间层图像的所有像素, 设置阈值剔除对照度低的点.
4) 当前点与其周围 26 个点比较, 进行非极大值抑制.
5) 通过三维二次曲线拟合对候选特征点插值, 并将该候选特征点坐标、Laplace 迹以及索引号存入向量组.

3.2 特征点描述

3.2.1 寻找特征点主方向

寻找特征点主方向时, 各特征点主方向计算相互独立, 计算圆形域内各像素 Haar 小波响应及扇形域内响应值累加求和同样相互独立. 因此, 并行化处理时, 为每个特征点分配一个线程块, 该特征点每个邻域像素点被分配在线程块的每个线程中, 其算法流程如算法 4.

$$angle = \arctan\frac{d_x}{d_y} \tag{13}$$

算法 4. 寻找特征点主方向算法流程

1) 使用大小为 4s 的小波在半径为 6s 的圆中对x和y方向梯度进行采样, 利用积分图像计算x、y方向的 Haar 小波响应值d_x、d_y.
2) 使用σ=2s的高斯加权函数对d_x、d_y进行高斯加权, 通过式 (13) 计算向量的方向角 *angle*, 并将其分别存储在共享内存 s_X、s_Y、s_angle 中.
3) 用 60°的滑动窗口以 0.2 弧度步长进行旋转, 每个线程并行计算每个窗口内采样点的 Haar 小波响应值之和 sumx, sumy.
4) 通过式 (7) 和式 (8) 计算特征向量的长度m_ω和方向θ_ω, 得到方向矢量(m_ω,θ_ω), 取最长矢量对应方向为主方向.

3.2.2 生成特征点描述符

计算主方向时, 使用4s×4s的正窗计算半径为6s 圆域内的 Haar 响应, 因此可借助积分图像简化运算; 而计算描述符时, 使用2s×2s的倾斜窗计算边长为 21s 倾斜方形域内的 Haar 特征, 如图 10 所示, 因此不能利用积分图像快速计算. 点(i, j)旋转前对应积分图像的位置(x,y)为式 (14)[19]:

$$\begin{cases} x = x_0 - j\times scale\times\sin\theta + i\times scale\times\cos\theta \\ y = y_0 + j\times scale\times\cos\theta + i\times scale\times\sin\theta \end{cases} \tag{14}$$

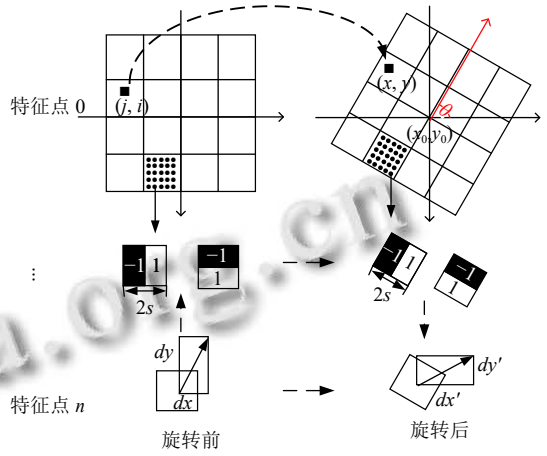


图 10 Haar 小波响应计算示意图

为防止误匹配, 本文引入高斯权重, 并给远离特征点的点赋予较小权值. 生成的描述符如图 11 所示, 图 11(a) 为文献 [9] 串行算法生成描述符结果, 图 11(b) 为本文算法生成描述符结果. 其算法流程如算法 5.

$$\begin{cases} d'_x = \omega\left(-d_x\times\sin\theta + d_y\times\cos\theta\right) \\ d'_y = \omega\left(d_x\times\cos\theta + d_y\times\sin\theta\right) \end{cases} \tag{15}$$

算法 5. 生成特征点描述符算法流程

1) 定义采样间隔和尺度大小, 通过点旋转公式 (式 (13)) 把点旋转到主方向上, 并通过最近邻插值计算“倾斜窗”所有像素.

2) 以特征点为中心, 沿主方向构建20s×20s方形邻域.

3) 在线程块内分配5×5的线程子区域, 使用2s×2s的模板计算窗口内x、y方向的 Haar 小波响应d_x、d_y, 并利用式 (15) 对d_x、d_y进行高斯加权处理.

4) 对 4×4 个边长为5s的方形区域分别累加求和, 得到每个特征点的4×4×4 维描述符, 并对其标准化处理.

(a) 串行算法生成描述符结果 (b) 本文算法生成描述符结果

图 11 生成特征描述符

3.3 特征点匹配

SURF 算法用欧氏距离表征描述符的相似程度, 进而完成匹配[20]. 设特征点 p 和 q 的描述符为Des_p和Des_q, 则其欧氏距离为式 (16), 其中 i 为描述符的维度. 若某图像特征点最近邻距离d_1与次近邻距离d_2的比值小于匹配阈值 (式 (17)), 则表示此特征点与待匹配图像中欧式距离最近的特征点匹配[20]. 本文首先通过 FLANN 算法进行最近邻近似匹配, 提高匹配效率; 然后采用 RANSAC 算法进行精匹配, 并采用双向匹配策略在传统算法上进一步优化, 提高算法精度, 其流程如图 12 所示.

$$d=\sqrt{\sum i\left(Des_p(i)-Des_q(i)\right)^2} \tag{16}$$

$$\frac{d_1}{d_2}<r=0.8 \tag{17}$$

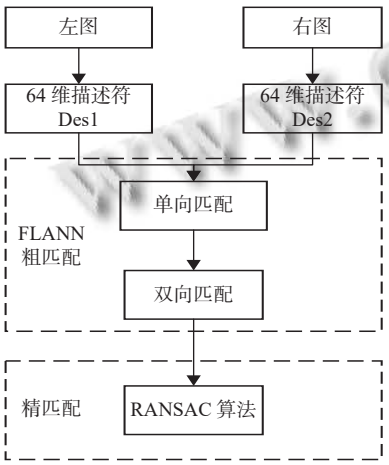


图 12 特征点配准流程

FLANN 算法即快速最近邻搜索, 其算法复杂度低, 匹配结果如图 13 所示, 由图 13 可知 FLANN 算法存在误匹配对, 需要进一步的筛选[21]. 本文通过 RANSAC (随机采样一致性) 算法进行提纯. RANSAC 算法, 是用迭代的方法通过重复随机取样来计算两幅图像的最优单应性矩阵 H, 利用矩阵变换模型 (式 (18)) 剔除误匹配点, 从而完成图像配准[22]. RANSAC 算法具有很好的旋转不变性, 精细化匹配效果如图 14 所示, 图中很好地剔除了误匹配点.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}=\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{18}$$

其中, h_{11}、h_{12}、h_{21}、h_{22}为缩放及旋转因子, h_{13}、h_{23}为平移因子, h_{31}、h_{32}为仿射因子, x、y分别为当前图像的横纵坐标, x'、y'为变换后图像的坐标.

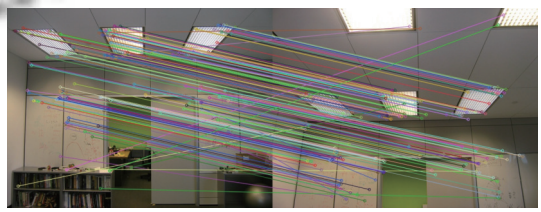


图 13 FLANN 算法单向匹配结果

图 14 FLANN 双向匹配+RANSAC 算法匹配结果

4 实验分析

4.1 实验方案

已有基于 GPU 的 SURF 加速算法有基于 OpenCL 和 CUDA 架构两个版本, 两者运行效率相当, OpenCL 架构可移植性好, 但其内存等需要程序员自己管理, 而 CUDA 架构由开发环境进行管理, 这无疑降低了编程的难度. 因此, 本文采用更高一级别的 CUDA 架构实现 SURF 特征检测算法加速, 利用 7 张分辨率不同的图像设计两组实验, 从而验证本文算法的实时性和准确性. 结果表明, 本文算法与传统 SURF 算法具有相同的匹配精度, 而对于特征提取速度有显著提升. 其主要实验环境平台为: 软件集成开发环境: Visual Studio 2015; 开源库: CUDA 9.1, OpenCV 3.4.13; 硬件运行平台: Intel(R) Core(TM) i5-8500 CPU @ 3.00 GHz; 显卡: NVIDIA GeForce GT 730; 操作系统: Windows 10 64 位.

4.2 SURF 算法优化方案

4.2.1 内存访问优化

CUDA 内存模型中全局内存读写速度最慢, 因此, 文献 [7] 算法中引入共享内存, 将计算 Hessian 响应、非极大值抑制、三线性插值等过程关联提高算法速度. 本文在此基础上引入常量内存、纹理访问和原子操作进一步优化, 具体方案如下.

(1) 使用常量内存替换全局内存, 限制只读访问机制, 减少内存带宽, 提升算法性能.

(2) 在访问原图、积分图、掩码积分图等二维数据时, GPU 内具有高速处理的纹理缓存, 因此, 本文算法在函数体外声明纹理参照系, 将这些二维数据绑定为纹理内存, 使用纹理操作在 Kernel 函数中通过坐标较快的获取 texture 存储器数.

(3) 使用原子操作, 确保在多个并行线程间共享内存的读写保护, 从而高效得到正确特征点序列.

4.2.2 程序设计优化

在程序编写中, 对现有 GPU 加速算法主要采取以下优化方案.

(1) 尽可能将多个缓存数据打包合并一次传输, 减少数据传输时的时间开销.

(2) 减少函数调用和循环语句, 必要时在循环语句前使用“#pragma unroll”指令控制循环次数, 减少不必要的循环.

4.2.3 匹配精度优化

无论是 SURF 欧式距离匹配算法还是其他匹配算法均存在一定误匹配, 因此, 本文在传统 FLANN 和 RANSAC 匹配算法上进行匹配度优化. 本文将 FLANN 粗匹配和 RANSAC 精匹配结合剔除误匹配点, 并引入双向匹配策略进一步提高配准精度, 最后通过计算配准精度验证算法性能.

4.3 实验结果

由算法实现过程知, 特征点检测过程性能与输入图像分辨率及其复杂度有关, 计算特征描述符与特征点数量有关, 因此, 为验证本文算法的准确度, 设计了文献 [9] 串行算法与本文优化算法对比实验, 如表 1 和表 2 所示, 分别记录了 7 组不同分辨率图像特征点检测和特征点描述过程的运行时间, 其实验结果取 10 次试验的平均值.

由表 1 可以看出, 特征检测过程运行时间随着图像分辨率的增大而增大, 其中传统串行算法耗时尤为明显; 本文算法与文献 [9] 检测特征点数相同, 由式 (19) 得检测精度可高达 99.81%. 对于分辨率为 800×600 的图像, 其复杂度较低, 检测到的特征点数目较少, 算法运行时间较短, 检测精度可达到 100%. 去掉该图, 得到如图 15 所示特征点检测时间对比图, 可见本文算法在保证检测精度的同时, 大大优化了特征检测速度.

$$\text{检测精度} = \frac{\text{正确特征点数}}{\text{正确特征点数} + \text{错误特征点数}} \tag{19}$$

表 1 SURF 算法检测特征点过程对比

分辨率	文献[9]串行算法		本文算法		精确度
	特征点数	时间 (ms)	特征点数	时间 (ms)	
640×480	829	425.99	829	34.24	0.9976
730×487	2082	508.29	2082	41.35	0.9981
779×519	3441	622.90	3441	51.32	0.9977
800×600	882	480.03	882	50.81	1.0000
1155×867	6677	837.71	6677	113.04	0.9970
2000×1329	19490	1783.70	19490	283.53	0.9816
2448×3264	54952	4410.35	54902	823.42	0.9800

表 2 特征描述过程时间对比

分辨率	特征点数	文献[9]串行算法 (ms)	本文算法 (ms)	精确度
640×480	829	140.07	6.94	0.8444
800×600	882	196.37	10.23	0.7891
730×487	2082	294.68	14.66	0.9155
779×519	3441	559.35	25.15	0.9797
1155×867	6677	1013.27	51.53	0.8544
2000×1329	19490	2977.91	145.06	0.8905
2448×3264	54952	8177.09	388.27	0.8764

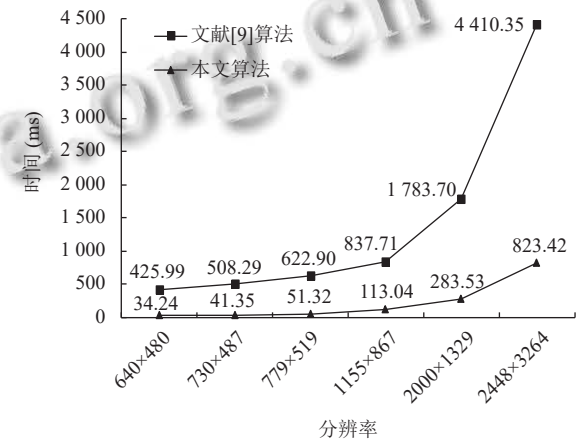


图 15 特征点检测时间对比

如表 2 所示, 记录了 6 组不同分辨率图像特征描述步骤的运行时间, 从而可得到如图 16 所示特征描述时间对比. 由此可见, 特征描述过程运行时间随着图像检测特征点数的增大而增大, 其中文献 [9] 串行算法耗时尤为明显. 根据式 (20) 计算生成特征描述符精度, 相比于文献 [9], 本文算法得到的特征描述符精度高达

97.97%, 且图 16 本文算法大大节省了特征描述用时.

$$描述精度=\frac{正确特征描述符}{正确特征描述符+错误特征描述符} \quad (20)$$

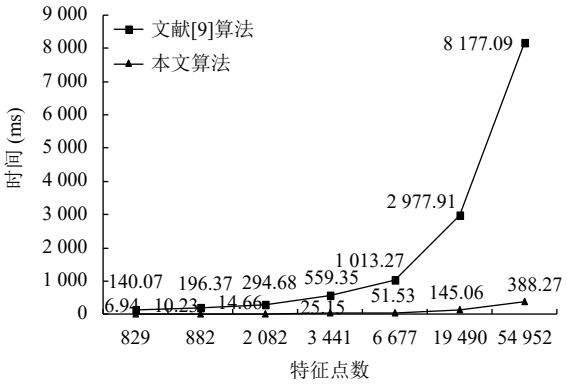


图 16 特征描述过程时间对比

基于上述实验过程, 统计了本文并行算法与文献 [9] 串行算法的总耗时和加速比 (如表 3 所示), 进而可得到其时间对比折线图 (如图 17 所示). 图 17 也证实了上述结论, 随着分辨率的增大, 传统串行算法耗时严重, 而本文算法对不同分辨率的图像均可实现平均 10 倍以上的加速比, 加速效果较好. 然而, 由于随着图像分辨率的增大以及图像复杂度的增加, 检测到的特征点数量增大, 进而算法的并行化程度增大, GPU 的并行计算能力也增强; 当分辨率与特征点数增大到一定程度时, GPU 内存竞争导致加速比逐渐下降, 故而出现如图 17 所示下降趋势.

表 3 SURF 优化前后实验结果

分辨率	文献[9]串行算法 (ms)	本文算法 (ms)	加速比
640×480	566.06	41.18	13.75
730×487	802.97	56.00	14.34
779×519	1 182.25	76.47	15.46
1155×867	1 850.98	164.57	11.25
2000×1329	4 761.61	428.59	11.11
2448×3264	12 587.44	1 211.69	10.39

为突出本文算法的先进性, 设计了本文算法与已有 CUDA 加速算法的对比实验, 如表 4 所示记录了文献 [7] 与本文算法的运行时间及加速比, 可以看出, 文献 [7] 算法检测速度至少提高了 6 倍, 而本文优化算法对不同分辨率图像均可实现 10 倍以上的加速比. 从加速比折线图 (图 18) 也可明显看出, 本文算法在提高算法效率方面具有很大优势.

4.4 特征匹配优化实验结果

为验证本文算法的普适性, 本文在传统 SURF 算法最近邻匹配基础上进行了精细匹配实验, 采用双向匹配策略对传统 FLANN 和 RANSAC 算法进行优化, 从而提高匹配精度.

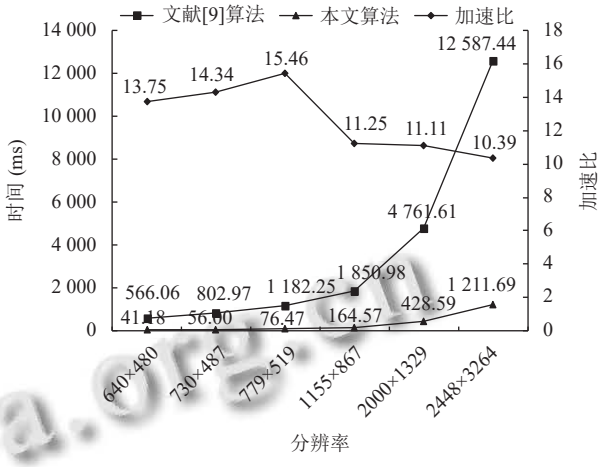


图 17 SURF 优化前后时间对比

表 4 文献 [7] 与本文算法的运行时间对比

分辨率	文献[7]并行算法		本文算法	
	时间 (ms)	加速比	时间 (ms)	加速比
640×480	79.61	7.11	41.18	13.75
730×487	89.42	8.98	56.00	14.34
779×519	124.06	9.53	76.47	15.46
1155×867	256.01	7.23	164.57	11.25
2000×1329	679.26	7.01	428.59	11.11
2448×3264	1 853.82	6.79	1 211.69	10.39

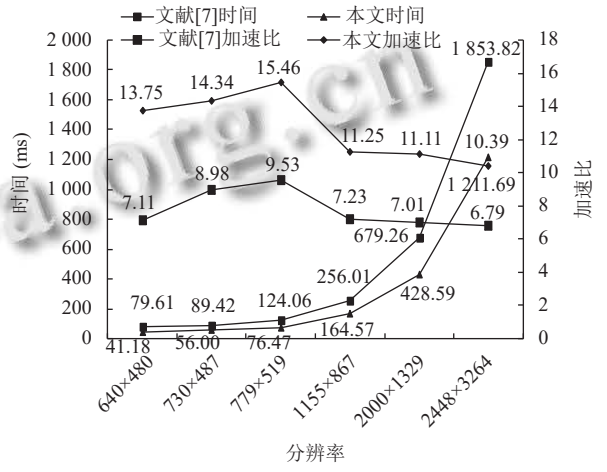


图 18 文献 [7] 与本文算法加速比

选用 4 组具有代表性的不同分辨率的图像进行匹配度分析: 从不同仰角拍摄的室内图像 office、 光线比较暗的图像 temple、 纹理比较复杂的图像 train 和水平位置拍摄的不同方位图像 intersection, 得到实验效果分别如图 19–图 22 所示, 从图中可以看出, 无论是室内图像还是户外图像, 对于不同纹理、 不同光强、不同角度的图像, 本文算法均有较好的配准效果.

(a) KNN 单向匹配+RANSAC 算法

(b) KNN 双向匹配+RANSAC 算法

图 19 分辨率为 640×480 office.jpg

(a) KNN 单向匹配+RANSAC 算法

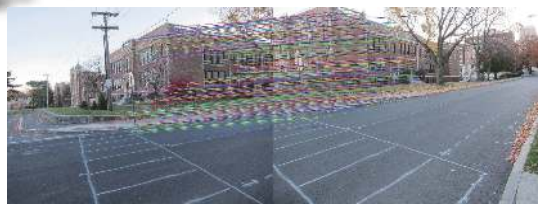
(b) KNN 双向匹配+RANSAC 算法

图 20 分辨率为 640×480 temple.jpg

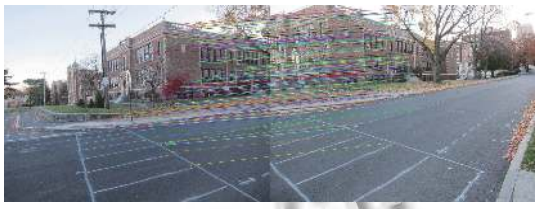
(a) KNN 单向匹配+RANSAC 算法

(b) KNN 双向匹配+RANSAC 算法

图 21 分辨率为 779×519 train.png



(a) KNN 单向匹配+RANSAC 算法



(b) KNN 双向匹配+RANSAC 算法

图 22 分辨率为 1155×867 intersection.jpg

为客观评价配准效果, 本文利用式 (21) 计算匹配精度, 得到如表 5 所示对比实验结果.

表 5 传统 SURF 算法和本文算法对比实验结果

图像分辨率	图像名称	传统FLANN+RANSAC算法			基于双向匹配的FLANN+RANSAC算法		
		内点	外点	精度	内点	外点	精度
640×480	office.jpg	143	36	0.799 1	105	10	0.911 5
730×487	temple.jpg	362	73	0.832 6	262	15	0.945 2
779×519	train.png	217	94	0.696 7	167	25	0.869 2
1155×867	intersection.jpg	341	204	0.626 1	215	61	0.777 7
2000×1329	yard.jpg	4 000	518	0.885 3	3 410	127	0.964 2
2448×3264	garden.jpg	1 855	789	0.701 7	1 443	269	0.701 7

$$\text{配准精度} = \frac{\text{正确匹配数}}{\text{正确匹配数} + \text{错误匹配数}} \tag{21}$$

图像的复杂程度、光线强弱、重叠区域大小等都会对配准精度有一定的影响. 为显著比较算法优化前后精度, 得到如图 23 所示精度对比曲线, 从图中可以看出, 本文优化算法配准精度明显提升, 最高配准精度可达到 96%, 进一步说明本文算法具有较好的鲁棒性.

5 结论与展望

本文提出了一种基于 CUDA 架构的并行 SURF 算法, 通过对计算 Hessian 响应、非极大值抑制、计算

特征点主方向及生成特征描述符等步骤的并行加速优化, 从而提升算法性能. 本文给出了详细的算法流程, 并以此提出一种基于 FLANN 和 RANSAC 算法的双向配准方法. 实验结果表明: 本文算法对不同分辨率图像的特征提取均能实现 10 倍以上的加速比, 且配准精度最高可达到 96%, 具有较好的实时性和鲁棒性, 对获取全景视频图像具有深远意义. 后续工作基于本文 CUDA 加速的配准算法, 对视频图像序列进行拼接, 并通过融合处理实时生成大尺度无重影的全景图像, 为全景视频监控提供有力的技术支撑.

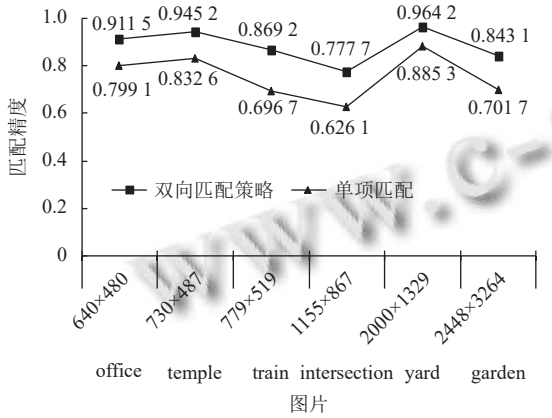


图 23 本文算法和传统单项匹配算法精度对比

参考文献

1 Zhang HJ, Hu Q. Fast image matching based-on improved SURF algorithm. Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC). Ningbo: IEEE, 2011. 1460–1463. [doi: 10.1109/ICECC.2011.6066546]

2 王艳梅, 史晓华, 于湛麟. SURF 算法在通用 GPU 和 OpenCL 的优化与实现. 电子测试, 2013, (23): 38–42.

3 刘金硕, 曾秋梅, 邹斌, 等. 快速鲁棒特征算法的 CUDA 加速优化. 计算机科学, 2014, 41(4): 24–27, 43. [doi: 10.3969/j.issn.1002-137X.2014.04.005]

4 徐晶, 曾苗祥, 许炜. 基于 GPU 的图片特征提取与检测. 计算机科学, 2014, 41(7): 157–161. [doi: 10.11896/j.issn.1002-137X.2014.07.032]

5 郭景, 陈贤富. 基于 OpenCL 的加速鲁棒特征算法并行实现. 中国科学技术大学学报, 2017, 47(10): 808–816. [doi: 10.3969/j.issn.0253-2778.2017.10.002]

6 卢嘉铭, 朱哲. 基于 GPU 加速的实时 4K 全景视频拼接. 计算机科学, 2017, 44(8): 18–21, 26. [doi: 10.11896/j.issn.1002-137X.2017.08.003]

7 周亮君, 肖世德, 李晟尧, 等. 基于 SURF 与 GPU 加速数字图像处理. 传感器与微系统, 2022, 41(3): 98–100. [doi: 10.13873/J.1000-9787(2022)03-0098-03]

8 谢雨来, 李醒飞, 吕津玮, 等. 基于 SURF 算法的水下图像实时配准方法. 计算机辅助设计与图形学学报, 2010, 22(12): 2215–2220.

9 Brown M, Lowe DG. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 2007, 74(1): 59–73. [doi: 10.1007/s11263-006-0002-3]

10 Bay H, Ess A, Tuytelaars T, *et al*. Speeded-up robust features (SURF). Computer Vision and Image Understanding, 2008, 110(3): 346–359. [doi: 10.1016/j.cviu.2007.09.014]

11 Li XG, Ren C, Zhang TX, *et al*. Unmanned aerial vehicle image matching based on improved RANSAC algorithm and SURF algorithm. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Guilin: ICGBD, 2020. 67–70.

12 Na Y, Liao MM, Jung C. Super-speed up robust features image geometrical registration algorithm. IET Image Processing, 2016, 10(11): 848–864. [doi: 10.1049/iet-ipr.2015.0528]

13 陈雪松, 陈秀芳, 毕波, 等. 基于改进 SURF 的图像匹配算法. 计算机系统应用, 2020, 29(12): 222–227. [doi: 10.15888/j.cnki.csa.007727]

14 杨志芳, 颜磊. 基于改进 SURF 算法的图像拼接技术研究. 武汉工程大学学报, 2021, 43(2): 223–226, 231. [doi: 10.19843/j.cnki.CN42-1779/TQ.202012012]

15 彭景维, 童基均. 基于 GPU_CPU 异构并行加速的人头检测方法. 计算机系统应用, 2017, 26(11): 95–100. [doi: 10.15888/j.cnki.csa.006079]

16 莫维, 唐清善, 黄涛. 一种多方向感知的高实时性视频融合算法. 计算机与现代化, 2021, (10): 81–87. [doi: 10.3969/j.issn.1006-2475.2021.10.013]

17 Nakov O, Mihaylova E, Lazarova M, *et al*. Parallel image stitching based on multithreaded processing on GPU. Proceedings of the 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC). Mon Tresor: IEEE, 2018. 1–5. [doi: 10.1109/ICONIC.2018.8601253]

18 Ding P, Wang F, Gu DY, *et al*. Research on optimization of SURF algorithm based on embedded CUDA platform. Proceedings of the 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). Tianjin: IEEE, 2018. 1351–1355. [doi: 10.1109/CYBER.2018.8688294]

19 覃方涛, 房斌. CUDA 并行技术与数字图像几何变换. 计算机系统应用, 2010, 19(10): 168–172, 116. [doi: 10.3969/j.issn.1003-3254.2010.10.035]

20 陈天华, 王福龙, 张彬彬. 基于改进 ORB 和对称匹配的图像特征点匹配. 计算机系统应用, 2016, 25(5): 147–152.

21 原伟杰, 文中华, 彭擎宇. 一种基于 SURF、FLANN 和 RANSAC 算法的图像拼接技术. 计算机与数字工程, 2022, 50(1): 169–173, 185. [doi: 10.3969/j.issn.1672-9722.2022.01.032]

22 张东, 余朝刚. 基于特征点的图像拼接方法. 计算机系统应用, 2016, 25(3): 107–112.

(校对责编: 牛欣悦)