

一种基于 CUDA 的快速宽视频拼接的方法

刘有科¹,高 珏²,谭 松¹,许华虎³

(1. 上海大学 计算机工程与科学学院,上海 200444;

2. 上海大学 计算中心,上海 200444;

3. 上海上大海润信息系统有限公司,上海 200444)

摘 要:在视频拼接过程中,需要进行大量的图像处理计算,高复杂度的拼接算法很难满足实时性要求。因此,为了实现多视频源快速拼接,不仅要求拼接算法具有较强的鲁棒性,同时还需要具有较低的复杂度。文中提出了一种新的快速宽视频拼接方法,此方法首先利用 SURF 算子提取图像特征点并进行匹配,接着使用多分辨率融合算法进行全景图融合,然后利用基于路径的方法结合光流实现低复杂度的视频插帧,同时整个宽视频拼接过程使用 CUDA 进行并行加速,最后用一个双目拼接系统对文中提出的方法进行可行性验证。

关键词:SURF;多分辨率融合;视频插帧;CUDA;视频拼接

中图分类号:TP301

文献标识码:A

文章编号:1673-629X(2015)01-0015-04

doi:10.3969/j.issn.1673-629X.2015.01.004

A Fast Wide Video Stitching Method Based on CUDA

LIU You-ke¹,GAO Jue²,TAN Song¹,XU Hua-hu³

(1. College of Computer Engineering and Science, Shanghai University, Shanghai 200444, China;

2. Computing Center, Shanghai University, Shanghai 200444, China;

3. Shang Da Hai Run Information System Co., Ltd., Shanghai 200444, China)

Abstract:In the process of video stitching,high complexity algorithm can't use to real-time video stitching since most image process contains mass of computing. In order to achieve real-time stitching,the stitching algorithm must have strong robustness and lower complexity. In this paper,propose a new method about fasting-video stitching. First,it extracts and matches the image feature points by using SURF,and blends the panorama image by using multi-resolution fusion algorithm. Then achieve a low complex video interpolation by combining the base-path with optical flow. Meanwhile it accelerates the whole process by using CUDA. At last,a binocular stitching system is introduced to verify the whole process's feasibility.

Key words:SURF;multi-resolution fusion;video interpolation;CUDA;video stitching

0 引 言

随着现代多媒体技术的发展,全景视频的应用越发广泛。目前主要有两种方式来得到全景视频:第一种是利用硬件进行拼接,直接利用硬件的高效性与相应的算法来拼接视频^[1];另一种则是主要利用软件,同时使用相关加速硬件进行计算加速^[2-3]。此外,利用视频插帧以减少全景拼接的复杂度也越来越受学者重视^[4-5]。在全景拼接中的特征提取与匹配方面,Herbert Bay等^[6]从SIFT^[7]算子中出发,提出了一种加速鲁棒算子,即SURF算子。由于SURF算子精确度高,

对尺度、旋转都具有不变性,对光照变化也具有部分不变性,因此广泛应用于快速特征提取。在视频插帧方面,传统的方法是利用光流法来实现,但由于光流法得到的中间图像不够平滑,因此Dhruv Mahajan等^[5]提出了一种基于路径的视频插帧法,实现了关键帧之间的平滑过渡;接着复旦大学颜波教授等对此方法结合光流进行改进在保证插帧精度的同时降低了算法的复杂度^[8-9]。此外,随着现代GPGPU计算的发展,利用GPGPU进行计算加速成为快速计算的第一选择,因此,NVIDIA公司提出了CUDA模型,利用GPGPU对

收稿日期:2014-01-20

修回日期:2014-04-22

网络出版时间:2014-11-17

基金项目:国家重大科技专项课题(2009ZX04001-111)

作者简介:刘有科(1989-),男,硕士,研究方向为多媒体技术;高 珏,副教授,研究方向为多媒体、Internet技术和嵌入式应用等;许华虎,教授,博士生导师,CCF会员,研究方向为人机交互、图像处理、多媒体网络技术等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20141117.2205.015.html>

复杂计算进行并行加速^[10-13]。

文中简要介绍了全景图拼接的关键技术,即 SURF 算子。重点描述结合光流的基于路径的视频插帧法,并提出了视频拼接方法。利用双目拼接系统对提出的方法进行相关验证并进行了总结。

1 SURF 特征算子

Herbert Bay 等从 SIFT 中得到启发而提出 SURF 算子。SURF 算子对尺度和旋转具有不变性,对光照变化具有部分不变性,此外,特征点提取的速度也是非常之快,因此广泛应用于对快速特征提取与匹配。

SURF 算子从整数图像出发,给定一个输入图像 I 和一个点 (x, y) , 那么这个整数图像 I_{Σ} 就是从原点到这个点的所有值的总和,那么利用整数图像进行三次操作就能够快速计算一个区域内的值,若要计算某一区域如 $ABCD$ 中的值,则利用整数图像的性质能得到:

$$\sum_{ABCD} = \sum_A + \sum_D - \sum_B - \sum_C$$

此外, SURF 算子使用 Hessian 矩阵进行计算, Hessian 矩阵具有如下特性。对函数 $f(x, y)$, 若 f 的二阶导数都存在, 则 f 的 Hessian 矩阵为:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2$$

若在点 (x_0, y_0) 行列式 $H > 0$, 则当 $\frac{\partial^2 f}{\partial x^2} > 0$ 时, 点 (x_0, y_0) 为局部极小点; $\frac{\partial^2 f}{\partial x^2} < 0$ 时, (x_0, y_0) 为局部极大值点。

因此, 对应源图像 I 的一个点 $X = (x, y)$, 它在尺度 σ 下的 Hessian 矩阵可以定义为:

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

其中, $L_{xx}(X, \sigma)$, $L_{xy}(X, \sigma)$, $L_{yy}(X, \sigma)$ 分别是高斯二阶偏导和图像 I 在 X 点处的卷积。Herbert Bay 等^[6]从 David G Lowe 用 DoG 近似 LoG 中得到启发, 为了加快卷积的速度, 采用了盒子型滤波器 (box filter) 对上面的滤波器进行近似, 则 Hessian 的行列式可近似为:

$$\det(H_{\text{approx}}) = D_{xx} * D_{yy} - (wD_{xy})^2$$

最后在圆形区域计算 Harr 小波响应, 以 $\pi/3$ 的滑动窗口, 计算窗口内响应总和得到主方向。沿着主方向构建一个方形区域并将方形区域划分为 $4 * 4$ 的子区域, 对每一个子区域又进行 $5 * 5$ 的样本点划分, 计算 d_x 、 d_y 、 $|d_x|$ 、 $|d_y|$ 就得到了 32 维的特征。

2 结合光流的基于路径的视频插帧法

基于路径的插帧法非常直观, 其思想是在前后两幅图像的对应像素点之间存在着一条路径, 当找到这条路径后, 对前一图像的某一像素, 只要沿着这条路径移动, 在需要插帧的地方计算对应的像素值就行, 整个过程就是简单的拷贝, 所以研究的重点就是找到这条路径, 而当找到这条路径以后, 由于像素间的计算与拷贝非常快, 因此插帧的过程非常快。

此外, 光流表达了图像的变化, 它包含运动目标的信息, 可以用来确定观察者相对目标的运动情况, 光流有三个要素: 运动、带光学特征的部分、成像投影。光流法的核心就是求解出运动目标的光流, 也就是速度。根据视觉感知原理, 客观物体在空间上一般是相对连续运动的, 在运动过程中, 投射到传感器平面上的图像实际上也是连续变化的。

因为在路径计算阶段, 路径集合从各个方向都有可能, 因此计算路径的复杂度相对较高, 这样直接导致了后期路径的筛选阶段和平滑代价都很高。既然光流描述了像素的移动, 那么在基于路径的视频插帧中结合光流^[8], 在路径构建阶段就可以减少路径集的大小, 从而更有效地进行后期的路径筛选。假设像素点对应路径 $w(u_A, v_A, u_B, v_B)$ 有四个参数, u_A, v_A 表示在图像 A 中点 p 移动到跳转点 P_A 的移动向量, 而 u_B, v_B 表示点 p 在图像 B 中移动到跳转点 P_B 的移动向量。根据光流初始化路径集合, 首先计算图像 A 和 B 之间的光流, 将 p 对应的光流记为 (u_{of}, v_{of}) 。如果 $u_{of} \neq 0$, 则路径集合定义如下:

$$\{(u_A, v_A, u_B, v_B) \mid u_A \in [-\lceil |u_{of}| \rceil, \lceil |u_{of}| \rceil], v_A =$$

$$\frac{v_{of}}{u_{of}} * u_A, u_B \in [-\lceil |u_{of}| \rceil, \lceil |u_{of}| \rceil], v_B = \frac{v_{of}}{u_{of}} * u_B\}$$

式中, u_A, v_A, u_B, v_B 均为整数。

如果 $u_{of} = 0$, 则路径就简化为:

$$\{(u_A, v_A, u_B, v_B) \mid u_A = 0, v_A \in [-\lceil |v_{of}| \rceil, \lceil |v_{of}| \rceil], u_B = 0, v_B \in [-\lceil |v_{of}| \rceil, \lceil |v_{of}| \rceil]\}$$

同样 v_A, v_B 也均为整数。

很显然, 通过使用光流, 很大程度地降低了路径集大小, 同时保证了跳转点的任意性。

此外, 在原始的基于路径的视频插帧中, 对每个像素计算最佳路径的方法是对最小化能量方程, 而能量方程又由匹配对应项和平滑项两部分组成。既然邻近像素点在移动方向具有相似性, 因此对邻近像素点就可以使用相似的光流, 平滑的过程就可以近似为最小化像素点的光流值和它对应路径的差异, 记为 $V(\text{len}_p, \text{len}_{of})$, 则

$$V(\text{len}_p, \text{len}_{of}) = \min(|\text{len}_p - \text{len}_{of}|)$$

其中, len_p 表示路径长度:

$$\text{len}_p = \sqrt{u_A^2 + v_A^2} + \sqrt{u_B^2 + v_B^2}$$

而 len_{of} 表示对应的光流:

$$\text{len}_{\text{of}} = \sqrt{u_{\text{of}}^2 + v_{\text{of}}^2}$$

为了得到更好的效果,可以对 $V(\text{len}_p, \text{len}_{\text{of}})$ 做幂次处理:

$$V(p, \omega) = (V(\text{len}_p, \text{len}_{\text{of}}))^{\beta}$$

其中, β 可以任意取整数值。

因此,全局能量方程改进为:

$$E = \sum_p C(p, \omega) + \lambda \sum_p V(p, \omega)$$

相对于原始的基于路径的视频插帧法,这项改进可以更简单地保证路径的平滑,注意到平滑代价仅仅

由路径的长度有关而与跳转点无关,而在路径选择的改进中,减少了路径集的大小,这就间接加快了最小化能量方程的过程。邻近的像素点依旧可以随时作为跳转点进行路径计算。

3 快速多视频源拼接

在多视频源实时拼接过程中,可利用 OpenCV 从视频源获取视频图像,在图像处理并生成全景图阶段,利用 CUDA 进行特征提取与匹配和多分辨率融合^[14]得到精确全景图,然后利用视频插帧技术,在相邻全景图之间生成中间图像,实现全景图之间的平滑过渡,最后再利用 OpenCV 保存拼接后的宽视频。其整体结构如图 1 所示。

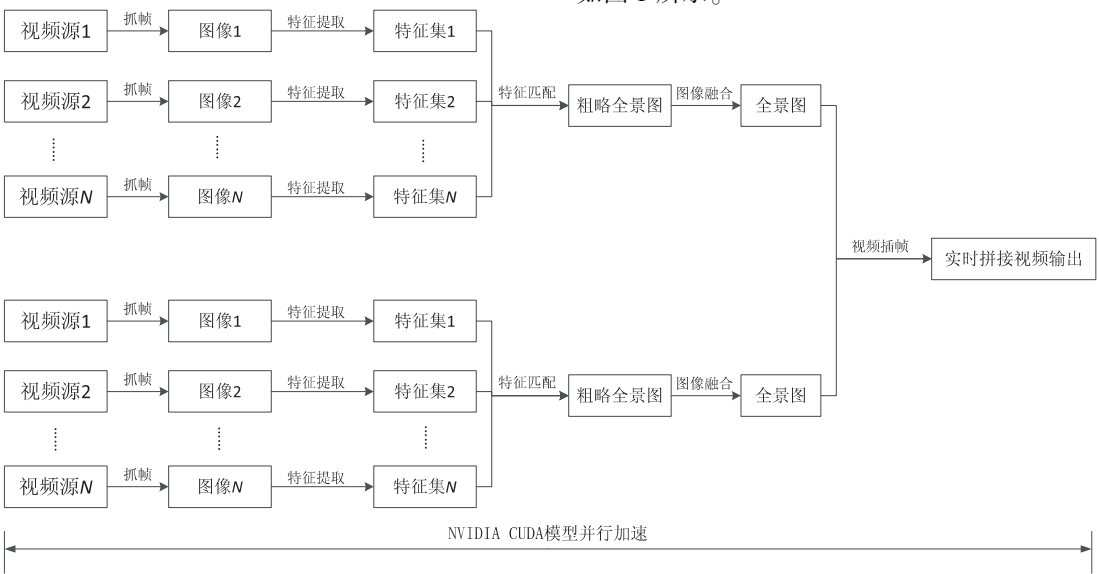


图 1 快速视频拼接结构

第一步:在多个视频源中同步提取各帧作为后面处理的原始材料,这里视频源可以是录制的视频,也可以从摄像头中实时抓取。

第二步:利用 SURF 算子进行特征提取与匹配,生成粗略全景图后再利用多分辨率融合^[9]算法进行图像融合得到了精确的全景图。在此阶段,可分别建立特征提取内核、匹配内核和融合内核,由 CUDA 模型中的内核调用,实现这几个步骤的并行进行。

第三步:在视频插帧阶段,利用两个全景图像矩阵,先利用光流计算前一个图像中的像素在后一个图像中的偏移,然后计算像素的移动路径。这个过程中可以分别建立光流内核函数和路径内核函数,对这两步进行并行计算,完成以后建立拷贝内核函数,沿着得到的路径进行像素拷贝,完成视频插帧。

最后对拼接的全景图和通过视频插帧得到的全景图进行输出与保存。

伪代码如下:

```
While( true )
```

```
{
    Catching frames from videos or cameras //抓取帧
    Extracting featherns kernel//抽取特征内核
    Matching featherns kernel//特征匹配内核
    Multi-resolution kernel//融合内核
    Video interpolation kernel //全景图间的插帧内核
    Outputting panorama video //输出宽视频
}
```

4 实验分析

文中实现一个双目拼接系统沿着提出的快速视频拼接方法的可行性。

硬件环境:主机(联想 Y470):处理器为 Intel Core i5-2410,主频 2.3 GHz,双核心/四线程,内存为 4 G DDR3;显卡(内置于主机):NVIDIA GeForce GT 550 M,显存:2 G DDR3,流处理器数:96;其他:2 个普通网络摄像头。

软件开发环境:Visual Studio 2010 专业版;软件开发包:NVIDIA CUDA Toolkit 5.5,OpenCV 2.4.7。

整个系统的实现为:首先利用 OpenCV 分别从两个摄像头中分别抓取一帧,然后利用 SURF 算子分别进行特征提取与匹配,如图 2 所示;再利用融合算法得到拼接全景图,如图 3 所示;最后利用插值实现全景图之间的平滑过渡,整个过程中利用 CUDA 在各个内核函数进行并行加速,实现快速视频拼接。

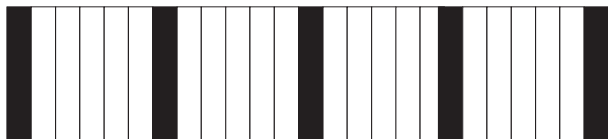


图 2 特征提取与匹配



图 3 全景生成

当利用视频插帧时,输出的视频结构如图 4 所示。





其中:  表示通过拼接算法得到的全景帧
 表示通过视频插帧算法得到的全景帧

图 4 视频帧结构

表 1 为双目系统的性能对比,分别为纯全景拼接输出、拼接与插帧(CPU)、拼接与插帧(CUDA)。

表 1 双目拼接性能对比

环境条件	每秒拼接全景数	相邻帧视频插帧数	每秒帧输出数 fps
纯拼接	0.3	0	0.3
拼接+插帧	0.3	2	2.3
拼接+插帧(CUDA)	3	5	13

从表 1 可以看出,当在拼接过程中引入插帧后,实现了若干全景图的输出,其中大部分为插帧得到的中间图像;当结合 CUDA 模型进行并行加速后,全景图的输出得到了很大的提高,实现了快速视频拼接。

5 结束语

文中利用 SURF 算子和多分辨率融合算法实现全景图的拼接,同时对基于路径的视频插帧法结合光流来降低关键帧之间平滑过渡的计算复杂度。此外,对整个视频拼接过程利用 CUDA 模型对各个步骤的计算内核进行 GPGPU 加速,实现了快速视频拼接。

参考文献:

[1] 彭 勃,何 宾. FPGA 在视频拼接中的应用与实现[J]. 计算机工程与设计,2013,34(5):1635-1639.

[2] 王小强,陈临强,梁 旭. 实时全自动视频拼接方法[J]. 计算机工程,2011,37(5):291-292.

[3] 季 诚,欧阳宁,张 彤. 基于视频片段的全景图拼接[J]. 计算机技术与发展,2009,19(9):239-241.

[4] 肖永豪,余英林. 基于视频对象的自适应去帧/插帧视频处理[J]. 华南理工大学学报:自然科学版,2003,31(8):6-9.

[5] Mahajan D, Huang F C, Matusik W, et al. Moving gradients: a path-based method for plausible image interpolation [C]// Proc of ACM SIGGRAPH. [s. l.]: ACM, 2009:1-11.

[6] Bay H, Ess A, Tuytelaars T, et al. SURF: speeded up robust features [J]. Computer Vision and Image Understanding, 2008,110(3):346-359.

[7] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2):91-110.

[8] Yan Bo, Chen Yue. Low complexity image interpolation method based on path selection[J]. Journal of Visual Communication and Image Representation, 2013,24(6):661-668.

[9] 刁 茜,邝 悦,董道国. 一种近似图像插帧的快速算法[J]. 计算机应用与软件,2012,29(11):148-152.

[10] Kim J, Park E, Cui Xuenan, et al. A fast feature extraction in object recognition using parallel processing on CPU and GPU [C]//Proceedings of the 2009 IEEE international conference on system. San Antonio:IEEE, 2009:3842-3847.

[11] 刘 永,王贵锦,姚安邦,等. 基于自适应帧采样的视频拼接[J]. 清华大学学报:自然科学版,2010,50(1):108-112.

[12] 李 波,赵华成,张敏芳. CUDA 高性能计算并行编程[J]. 微型电脑应用,2009,25(9):55-57.

[13] 凌 瞳,赵 昕,侯 哲,等. 基于 CUDA 的快速全景环带图像展开[J]. 计算机技术与发展,2011,21(11):23-26.

[14] Burt P J, Adelson E H. A multiresolution spline with application to image mosaics [J]. ACM Transactions on Graphics, 1983,2(4):217-236.

一种基于CUDA的快速宽视频拼接的方法

作者：[刘有科](#)，[高珏](#)，[谭松](#)，[许华虎](#)，[LIU You-ke](#)，[GAO Jue](#)，[TAN Song](#)，[XU Hua-hu](#)

作者单位：[刘有科, 谭松, LIU You-ke, TAN Song \(上海大学 计算机工程与科学学院, 上海, 200444\)](#)，[高珏, GAO Jue \(上海大学 计算中心, 上海, 200444\)](#)，[许华虎, XU Hua-hu \(上海上大海润信息系统有限公司, 上海, 200444\)](#)

刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2015(1)

引用本文格式：[刘有科](#). [高珏](#). [谭松](#). [许华虎](#). [LIU You-ke](#). [GAO Jue](#). [TAN Song](#). [XU Hua-hu](#) [一种基于CUDA的快速宽视频拼接的方法](#)[期刊论文]-[计算机技术与发展](#) 2015(1)