基于CUDA的快速全景环带图像展开

凌 瞳,赵 昕,侯 哲,汪凯巍,白 剑
(浙江大学 现代光学与仪器国家重点实验室,浙江 杭州 310027)

摘 要:全景环形镜头将以透镜为中心的360°视场内的景物投影到环形区域内,具有大视场和无穷远的景深。由于全景环形镜头所拍摄到的图像具有较大的失真,为了方便人眼观看,需要利用坐标变换,将全景环形镜头获得的环带图像展开为条幅状图像。图像变换过程需要大量处理的数据,尤其对于高分辨率图像而言,采用传统的基于中央处理器(CPU)的坐标转换算法耗时较长,图像停顿和滞后严重,无法实现实时显示。为解决此问题,文中提出采用最新出现的基于通用图形处理器(GPGPU)的并行处理技术实现图形的展开,大规模并行计算的使用缩短了图形展开的时间,从而实现了大视场、高分辨率、实时监控显示。

关键词:全景环形镜头;CUDA;二维平面展开
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2011)11-0023-04

Fast Panoramic Annular Image Stretching Based on CUDA

LING Tong, ZHAO Xin, HOU Zhe, WANG Kai-wei, BAI Jian
(State Key Laboratory of Modern Optics and Instrument, Zhejiang University, Hangzhou 310027, China)

Abstract: Panoramic Annular Lens (PAL) can form annular image circling the optical axis with field range of 360 degree and infinite depth of field. The annular image captured by PAL must be stretched into a rectangular image in order to be observed and measured conveniently. For a high resolution image, the traditional method of stretching process conducted by CPU will cost too much time. Propose a novel method using the latest GPU technology - CUDA. With this method, it can achieve a processing speed 10 times faster than traditional processing method by CPU, which makes the real-time high resolution panoramic detection possible.

Key words: panoramic annular lens (PAL); CUDA; two-dimensional plane stretching

0 引 言

安全监控、工业及医学内窥、航空航天、机器视觉等领域需要具有大视场、高分辨和实时的成像系统。传统大视场成像系统遵循中心投影法,为了获取周围360°的影像需要连续拍摄不同角度的市场并完成拼接,难以做到实时显示,而且系统复杂,图像处理数据量也很大。全景环形镜头PAL(Panoramic Annular Lens)可以有效地解决这些问题[1~3]。它利用FCP投射法将周围一圈柱形区域内的景物投射到一个二维平面圆环区域,视角为360°,景深为100mm至无穷远,可以实现高效的全视角成像。然而,由于PAL成像所得的图像是将周围360度区域的圆柱形区域投影到环形区域中,存在较大的图像畸变,不适于人眼直接观察,需要将其展开为条幅状图像才能便于观察[4~7]。图像的展开通常通过坐标变换加以实现,而坐标变换需要处理大量的数据。当使用高分辨率的感光元件时,由于每帧图像都有大量数据需要处理,传统基于CPU处理的转换便无法实时完成,造成图像的停顿和延迟,难以达到实时展开和监控的要求。

最近出现的基于通用图形处理器(GPGPU)的大规模并行运算为图像信息的处理提供了新的途径,基于此已被大量用于医学图像处理、油藏建模、神经建模、电磁场模拟、生物序列匹配、金融风险建模等领域[8~12]。文中提出基于通用图形处理器的并行展开算法,用于加速全景环带图像的展开,实验中收到了很好的加速效果,能够实现高分辨率的图像展开。

1 全景环形镜头(PAL)原理

全景环形透镜成像的原理如图1所示。其中2和3为反射面,1和4为折射面,L为转向透镜,使PAL成像在CCD传感器上。能够成像的是小于α角范围内的图像。PAL通过平面圆柱投影法把围绕光学系统光轴360°范围的圆柱视场投影到二维平面的一个环形区域[6],用二维的平面来表示圆柱面。能够成像部分是α角的两条边绕轴旋转360°后得到的区域,像面上

收稿日期:2011-04-27;**修回日期**:2011-08-03
作者简介:凌 瞳(1989-),男,江苏泰州人,博士研究生,主要从事光学检测方面的研究;汪凯巍,副教授,博士,主要研究工作为精密测试技术。

每一个同心圆是与轴成同一角度的点的轨迹。环形透镜产生的环形像的高度对应于侧向视场的范围。2β 角所代表的区域为盲区。

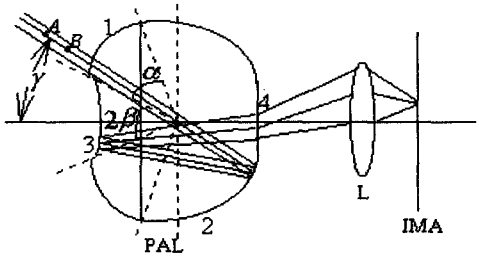

图1 全景环形镜头成像原理

CUDA (Compute Unified Device Architecture)是 NVIDIA 推出的运算平台,是一种通用并行计算架构,该架构使得 GPU 能够解决复杂的计算问题[10]。在 CUDA 平台上,可以通过并行处理,加快图像的转换速度,真正实现高分辨率、实时、无死角全景观察。

2 全景环带图像展开算法

全景环带图像若要展开为矩形图像,可将环带图像从一处断开后经过拉伸和弯曲操作变换为矩形图像,在这个过程中可以发现,原环带图像的扇形部分变换为了矩形图像的矩形部分,如图2所示。再加以仔细分析,图2中环带图像以圆形为中心一个方向的径向像素与矩形图像的一列纵向像素相对应。由此,如果将两幅图像表示为像素矩阵的形式,则矩形图像可以看做是全景环带图像经过一种线性变换得到的[2]。

为推导这样一种线性变换,建立如图3所示直角坐标系。

设 $P(x,y)$ 为矩形图像中的任意像素,$P'(x',y')$ 为相对应的环形区域中的像素,并设 $P'(\rho,\theta)$,其中 ρ 为 P' 点离开环形区域中心点 $O(x_c,y_c)$ 的距离称为径向长度,θ 为 OP' 与 x 轴的夹角称为径向角,则有

$$\begin{cases} x' = x_c + \rho\cos\theta \\ y' = y_c + \rho\sin\theta \end{cases} \tag{1}$$

根据等最大尺度原则,环带图像中与中心点 O 保持某一相同径向长度或径向角的考察点,分别与矩形图像中的某一行或某一列相对应,即 ρ、θ 每增加 $\Delta\rho$ 和 $\Delta\theta$,相对应的矩形区域中的 x、y 分别增加 Δx 和 Δy,则有

$$\begin{cases} \Delta\rho = \Delta y \\ \Delta\theta = \Delta x \end{cases} \tag{2}$$

式中,$\Delta\rho$、Δx、$\Delta\theta$、Δy 均以像素为单位。

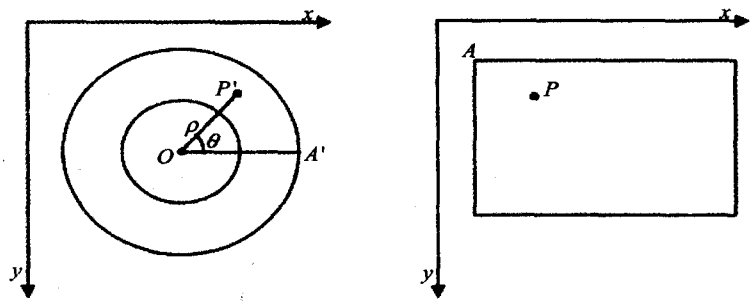

图3 环形图像坐标系和矩形图像坐标系

设环带图像展开的起始点位置 $A'(\rho_0,\theta_0)$ 与矩形图像的点 $A(x_0,y_0)$ 相对应,则有

$$\begin{cases} \Delta x = x - x_0 \\ \Delta y = y - y_0 \end{cases} \tag{3}$$

根据式(3),有

$$\begin{cases} \rho = \rho_0 + \Delta\rho = \rho_0 + \Delta y = \rho_0 + (y - y_0) \\ \theta = \theta_0 + \Delta\theta = \theta_0 + \Delta x = \theta_0 + (x - x_0) \end{cases} \tag{4}$$

将式(4)代入式(1),得到

$$\begin{cases} x' = x_c + [\rho_0 + (y - y_0)] \cdot \cos[\theta_0 + (x - x_0)] \\ y' = y_c + [\rho_0 + (y - y_0)] \cdot \sin[\theta_0 + (x - x_0)] \end{cases} \tag{5}$$

至此,(5)式便完成了矩形图像中的点 $P(x,y)$ 和对应的环带图像中的点 $P'(x',y')$ 之间的坐标变换,令两者的颜色值相等 $g(x,y)=f(x',y')$,即可实现环带图像的展开过程[4-6]。

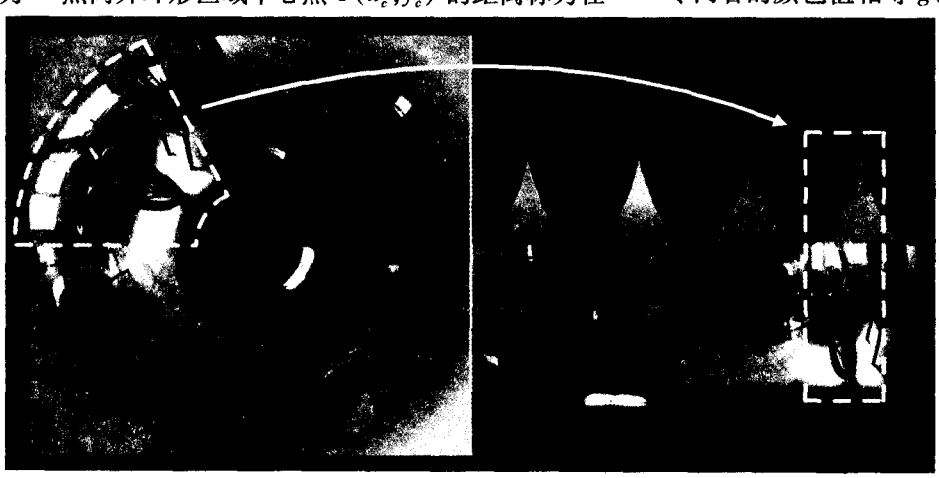

图2 全景环形镜头实拍图及其展开

3 CUDA 并行计算

CUDA 是由 NVIDIA 公司提出的并行计算架构,旨在为具有一定并行度的科学计算提供一套高效的解决方案。由于不少图像处理算法本质上都是矩阵变换,而变换后得到的矩阵其各元素之间都是相互独立的,所

以可以采用并行计算的方法加快图像处理速度。观察上面得到的(5)式,同样在全景环带图像展开中,目标矩形图像的各像素之间也没有任何联系,CUDA 技术便可帮助我们实现多个像素点的并行计算。

CUDA 是以 GPU 为载体的技术,而在日常的科学计算中经常访问的是 CPU 和内存,所以利用 CUDA 并行计算就不可避免地需要进行内存和显存之间的数据交换;另外,Kernel 函数的编写也是 CUDA 并行计算中的核心部分,实验中的 convert( )就是用来进行坐标变换和对应颜色值赋值的 Kernel 函数,其单个线程的 threadID(threadId.x, threadId.y)即代表目标矩形图像中的对应像素位置 (x, y),通过上面介绍的全景环带图像展开算法可求出该像素在原环带图像中对应像素的位置 (x', y'),在 convert( )函数中将原环带图像 d_pBmpBuf 中(x', y')位置像素的颜色值复制给目标矩形图像 d_NewBmpBuf 中(x, y)位置像素,即可求得原全景环带图像的展开图像。过程如图 4 所示。

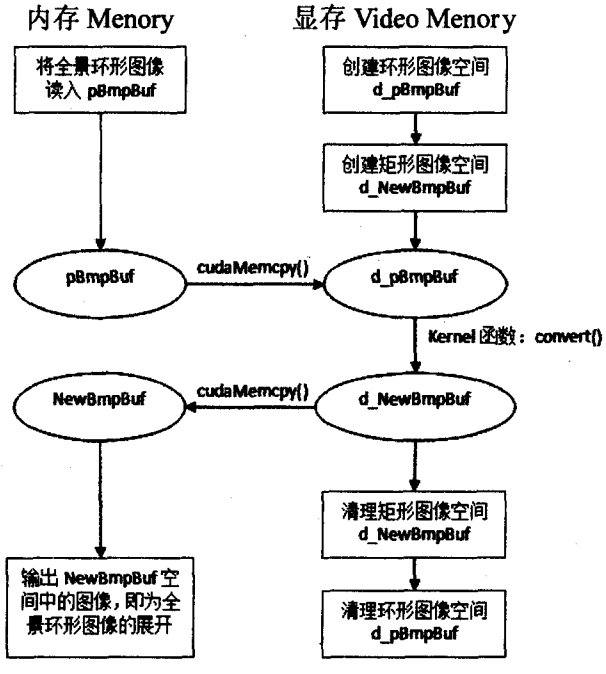

图 4　CUDA 环形图像展开过程

具体实现步骤如下:

1)将全景环带图像读入内存中,指针为 pBmpBuf。

2)定义 dim3 dimBlock(16, 16); dim3 dimGrid((N+dimBlock.x-1)/dimBlock.x, (N+dimBlock.y-1)/dimBlock.y),定义二维 Block 索引来标识线程,即线程的 threadID 是(threadIdx.x+threadIdx.y * Dx)。

3)使用 cudaMalloc 语句在显存中创建原图像空间 d_pBmpBuf 和目标图像空间 d_NewBmpBuf,空间大小均为原图像行占用字节数 * 行高,即 cudaMalloc((void * *)&d_pBmpBuf, lineByte * bmpHeight); cudaMalloc((void * *)&d_NewBmpBuf, lineByte * bmpHeight)。

4)使用 cudaMemcpy 语句将保存在内存中的图像 pBmpBuf 复制到显存的 d_pBmpBuf 空间中。

5)调用 Kernel 函数 convert( )并行处理,其单个线程的 threadID(threadId.x, threadId.y)即代表目标矩形图像中的对应像素位置 (x, y),通过上面介绍的全景环带图像展开算法可求出该像素在原环带图像中对应像素的位置 (x', y'),再将原环带图像 d_pBmpBuf 中 (x', y') 位置像素的颜色值复制给目标矩形图像 d_NewBmpBuf 中 (x, y) 位置像素。

6)使用 cudaMemcpy 语句将显存中的图像 d_NewBmpBuf 复制到内存的 NewBmpBuf 空间中。

7)调用 cudaFree 函数清理显存中 d_pBmpBuf 和 d_NewBmpBuf 空间。

8)将内存中 NewBmpBuf 空间的图像输出,即为所需要的展开图像。

4　实验结果

为了观察采用 CUDA 并行处理技术的算法相比原来使用 CPU 计算的算法有何种程度上的改进,设计了如下实验:

使用不同型号的 CPU、GPU 分别处理 500 * 500、1 000 * 1 000、1 280 * 1 280、2 000 * 2 000、3 000 * 3 000、4 000 * 4 000 六种分辨率的图像,记录处理时间并进行比较,其中,记录计算耗时为 CPU 或 GPU 的计算时间,传输时延为图像从设备到内存或从内存到显存的传输时间,总耗时为计算耗时与传输时延之和。

在 3 个不同的配置的电脑上进行了计算速度测试,3 个实验平台的具体配置如下:

实验平台 1 的配置为酷睿 2 T7100 @1.8GHz(CPU 型号),GeForce 8400M G(GPU 型号),内存 2G。

实验平台 2 的配置为酷睿 i5 750 @2.667GHz(CPU 型号),GeForce GTX260+(GPU 型号),内存 4G。

实验平台 3 的配置为酷睿 i5 760 @2.8GHz(CPU 型号),GeForce GTX480(GPU 型号),内存 4G。

测得的实验结果均表明基于 CUDA 的全景环带图像展开算法比传统的 CPU 展开算法具有更快的计算速度。下面,选取实验平台 3 的数据进行具体说明,比较结果如图 5、图 6 所示。

通过实验观察发现,在传统的采用 CPU 计算的全景环带展开算法中,计算耗时占了总耗时的大部分;而在 CUDA 并行计算中,传输时延比计算耗时高了好几个数量级。再仔细比较 CPU 与 GPU 的总耗时,可以发现在数据量较小的情况下,CPU 与 GPU 的总耗时相差不多;但当数据量增大到 4000 * 4000 时,GPU 的总耗时最小为 CPU 总耗时的 4.5%,极大地加快了图像展开的速度。又由于计算耗时很小,在处理图像流时不

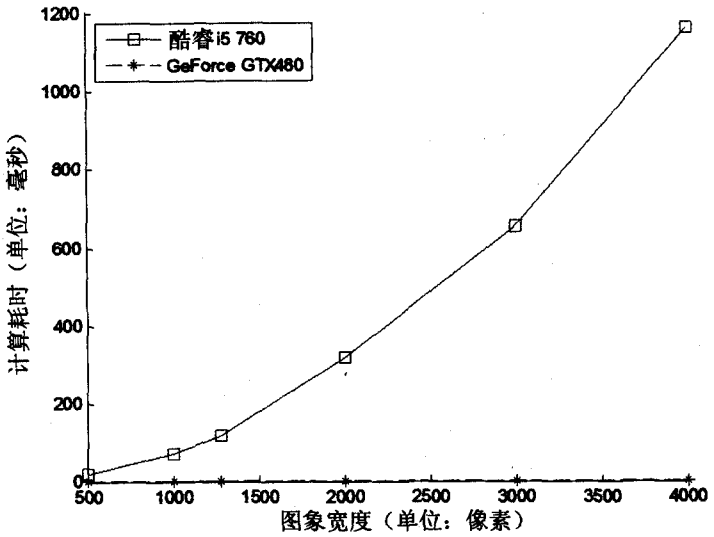

图5 CPU与GPU计算耗时比较

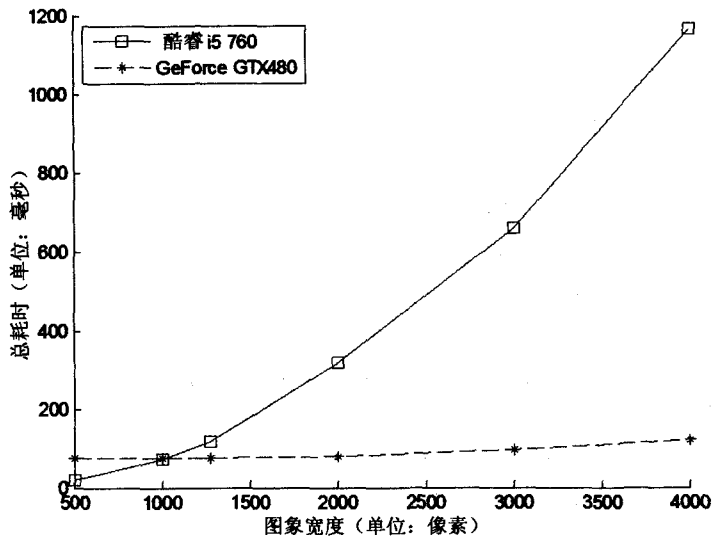

图6 CPU与GPU总耗时比较

会出现滞后现象，可以满足实时显示的需要。

5 结束语

基于CUDA的快速全景环带图像展开算法相较于传统的CPU算法在低分辨率的全景环带图像展开中差异不大，但在高分辨率的环带图像展开中可以极大地缩短计算时间，达到实时显示的要求，为全景环带图像的实时展开提供了可能。在研究的过程中也发现，使用CUDA技术的图像展开速度并不只与GPU的性能有关，还与内存的响应时间及系统总线带宽有关，其关系还有待后续深入的研究。

参考文献：

[1] Powell I. Panoramic lens[J]. Applied Opt, 1994,33(31):7356-7361.

[2] 白 剑，牛 爽，杨国光，等. 全景光学环带凝视成像技术[J]. 红外与激光工程，2006，35(3):331-335.

[3] 王道义，黄大为，邬敏贤，等. 全景环形透镜原理与特点剖析[J]. 光学技术，1998(1):10-12.

[4] 侯慧杰，白 剑，杨国光. 全景环形透镜二维平面成像展开算法研究[J]. 光子学报，2006,35(11):1686-1688.

[5] 张淑敏，陆燕慧，王保保. 基于边缘检测的全景环形图像插值复原[J]. 计算机技术与发展，2009,19(3):59-65.

[6] 肖 潇，杨国光，白 剑. 基于Bezier曲面的快速插值算法改善全景图像的分辨率[J]. 光电工程，2008,35(1):105-109.

[7] 朱方明，杨国光. 全景环形透镜环形像的线性化研究[J]. 光子学报，2001,30(5):589-593.

[8] 孟晓林，秦 安，徐 建，等. 基于CUDA的快速三维医学图像分割[J]. 中国医学物理学杂志，2010,27(2):1716-1720.

[9] Harada T, Koshizuka S, Kawaguchi Y. Smoothed particle hydrodynamics on GPUs[C]// Proceedings of Computer Graphics International. Rio de Janeiro:[s. n.],2007:63-70.

[10] 于 卓，梁晓辉，马 上，等. 基于GPU加速的binLBT压缩解压算法[J]. 北京航空航天大学学报，2010,36(3):3019-3025.

[11] 盛向治，单宝松. 基于GPGPU的多目的混音算法的研究与实现[J]. 北京大学学报(自然科学版)，2008,44(1):49-54.

[12] 温婵娟，欧嘉蔚，贾金原. GPU通用计算平台上的SPH流体模拟[J]. 计算机辅助设计与图形学学报，2010,22(3):406-411.