(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103997609 A
(43)申请公布日 2014.08.20

(21)申请号 201410260404.4

(22)申请日 2014.06.12

(71)申请人 四川川大智胜软件股份有限公司
地址 610045 四川省成都市武侯区武科东一路七号
申请人 四川大学

(72)发明人 兰时勇 吴岳洲 吴健 黄飞虎

(74)专利代理机构 成都信博专利代理有限责任公司 51200
代理人 卓仲阳

(51)Int.Cl.
H04N 5/262(2006.01)
H04N 5/265(2006.01)
H04N 9/64(2006.01)
G06T 5/50(2006.01)
G06T 15/00(2011.01)

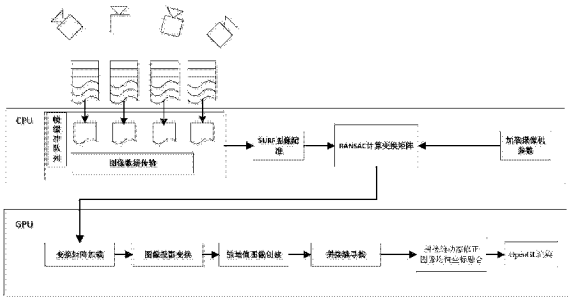
权利要求书2页 说明书8页 附图6页

(54)发明名称

基于CUDA的多视频实时全景融合拼接方法

(57)摘要

本发明公开了一种基于CUDA的多视频实时全景融合拼接方法，包括系统初始化的步骤和实时视频帧融合的步骤；所述系统初始化的步骤在CUDA架构的CPU端运行，所述实时视频帧融合的步骤在GPU端运行，且S21、S22、S23和S24基于CUDA架构的流处理模式：在GPU端创建4个并发处理的执行流，将S21、S22、S23和S24部署到相应的流处理序列中。本技术方案跟现有技术相比，有如下优点：1)实现多视频无鬼影和色彩亮度差异的实时全景视频；2)对机场场面、园区、广场等大场景超高分辨率整体监视效果显著，应用前景广泛。



CN 103997609 A

1. 一种基于 CUDA 的多视频实时全景融合拼接方法，其特征在于，包括

系统初始化的步骤，包括

S11 ：获取每路视频源的第一帧图像，执行配准操作以及柱面投影变换，求得整体的透视变换模型 ；

S12 ：根据透视变换模型对每路视频源的第一帧图像进行透视变换处理，同时求得透视变换掩模图以及相邻视频源间的重叠区掩模图 ；

S13 ：针对相邻视频源的视频图像，使用动态规划算法分别求得拼接缝上的像素点坐标 ；

S14 ：将透视变换掩模图、重叠区掩模图以及拼接缝像素点传输至 GPU 端的各个缓冲区中 ；实时视频帧融合的步骤，包括

S21 ：利用透视变换掩模图将每路视频源图像序列中的同一帧图像变换至同一坐标系下 ；

S22 ：利用重叠区掩模图求得每路视频源图像序列中的同一帧图像的重叠区能量值图 ；

S23 ：利用拼接缝像素点缓冲区中的坐标信息以及重叠区能量值图，使用拼接缝的实时动态修正算法重新计算拼接缝像素点坐标，并利用该坐标值更新拼接缝像素点缓冲区 ；

S24 ：根据上述求得的新拼接缝，基于均值坐标无缝融合算法消除各相邻图像间的拼接痕迹 ；

S25 ：将生成的全景视频帧图像传输至已绑定的 OpenGL 的像素缓冲区中，进行快速图像渲染 ；

S26 ：针对后续视频流中的每一帧视频图像重复 S21 至 S25 ；

所述系统初始化的步骤在 CUDA 架构的 CPU 端运行 ；所述实时视频帧融合的步骤在 GPU 端运行，且 S21、S22、S23 和 S24 基于 CUDA 架构的流处理模式 ：在 GPU 端创建 4 个并发处理的执行流，将 S21、S22、S23 和 S24 部署到相应的流处理序列中。

2. 如权利要求 1 所述的基于 CUDA 的多视频实时全景融合拼接方法，其特征在于，所述获取每路视频源的第一帧图像，执行配准操作以及柱面投影变换，求得整体的透视变换模型的方法是 ：对每路视频的第一帧图像，使用 SURF 算法进行图像配准，经 RANSAC 算法去除误匹配后，求得相邻两路视频图像之间的透视变换矩阵 ；利用柱面坐标转换计算得到整体的透视变换模型。

3. 如权利要求 2 所述的基于 CUDA 的多视频实时全景融合拼接方法，其特征在于，所述求得透视变化掩膜图的方法是 ：创建与透视变换矩阵维度大小一样的透视变换掩模矩阵，使其空间大小可以包含进所有变换后的像素 ；在参与变换的所有源图像的像素位置上将掩模矩阵对应位置的值置为 1，非变换像素的位置上将掩模矩阵对应位置的值置为 0，得到掩膜矩阵图，掩膜矩阵图对应的显示图即为透视变化掩膜图 ；所述求重叠区掩模图的方法是 ：将透视变化掩模图上存在实际像素位置的掩模矩阵元素的值设为 0.5，则两幅图像变换后的重叠区位置上的矩阵元素的值为 1 ；遍历整个掩模矩阵将非 1 的元素值重置为 0，得到重叠区掩膜矩阵，重叠区掩膜矩阵对应的显示图即为重叠区掩膜图。

4. 如权利要求 1 所述的基于 CUDA 的多视频实时全景融合拼接方法，其特征在于，所述使用动态规划算法求得拼接缝上的像素点坐标，包括

S41 :计算重叠区的像素能量值矩阵 En ;

S42 :构建方向矩阵 Dir 以及能量和矩阵 Cum ;

S43 :方向矩阵 Dir 中搜索拼接缝坐标路径。

5. 如权利要求 1 所述的基于 CUDA 的多视频实时全景融合拼接方法,其特征在于,所述使用拼接缝的实时动态修正算法重新计算拼接缝像素点坐标,包括

并行寻找拼接缝位置修正起始点的步骤 :在当前重叠区像素能量值图中,根据加载的前一帧图像的拼接缝像素点坐标进行比对寻找拼接缝变化的起始点位置,比对规则如下 :

如果在上一次拼接缝的像素点位置附近的同一行两个像素邻域内仍旧是最小值,则在这一次拼接缝的像素路径中,该像素点位置不需要修改,否则即为修改的起始点坐标 ;

GPU 端搜索新的拼接缝修改路径 :在 GPU 端根据拼接缝位置变化起始点和贪婪搜索算法重新搜索新的拼接缝位置点。

6. 如权利要求 1 所述的基于 CUDA 的多视频实时全景融合拼接方法,其特征在于,所述基于均值坐标无缝融合算法消除各相邻图像间的拼接痕迹,包括以拼接缝上像素点的色彩强度计算权值,去依次校正待融合图像的重叠区中的每一个像素的色彩强度校正值。

基于 CUDA 的多视频实时全景融合拼接方法

技术领域

[0001] 本发明涉及图形图像处理以及计算机视觉领域，特别是一种基于 CUDA 的多视频实时全景融合拼接方法。

背景技术

[0002] 全景视频拼接技术的目的是通过散落在场景中的多个不同点位的相机捕捉的实时视频信息拼接融合成全景实时视频。有些视频影像采集设备比如鱼眼摄像机，虽然可以包含几乎 360 度全景的场景信息，但是图像的分辨率有限，纹理细节不够丰富；有些高清网络摄像机虽然可以拍摄分辨率高达上千万像素的图像，但是受摄像机视角的限制，容纳的视景较小。全景视频拼接的目的正是为了同时满足全景视角和高清场景信息这两个方面的需求。

[0003] 该项技术的基础是图像拼接，主要流程包括：视频帧图像的预处理、图像配准、图像变换以及图像融合。考虑到固定摄像机的视频拼接帧率问题，现阶段使用较多的全景视频拼接技术方案是计算出固定位置的多个摄像机变换模型，然后将采集到的每一帧视频图像进行变换后，再使用加权融合或渐入渐出融合输出全景视频。

[0004] 现阶段的全景视频拼接系统迫切需要一种可以解决融合图像中的图像错位和鬼影的快速视频拼接算法。在此基础之上，为了进一步解决融合速度慢的问题，应充分利用 CUDA(Compute Unified Device Architecture，统一计算设备架构）架构的并行计算能力，使得生成的全景视频帧率较高且稳定。

发明内容

[0005] 本发明的目的是提供一种基于 CUDA 的多视频实时全景融合拼接方法，旨在解决传统算法中，相邻视频帧图像间重叠区域拼接融合后出现的鬼影和图像错位现象，最终生成无鬼影和色彩亮度差异的实时全景视频。该方法充分结合 CPU 和 GPU 各自的优点，构建两者协同工作的编程模型；利用相邻视频源间重叠区域的特征自动配准求解变换映射矩阵、各路视频帧图像透视变换至同一坐标系中、计算寻找拼接融合缝、针对视频帧图像进行融合缝的动态修正。最终实现超高分辨率大视角的大场景实时整体视频。

[0006] 实现本发明目的的技术方案如下：一种基于 CUDA 的多视频实时全景融合拼接方法，包括系统初始化的步骤，包括

[0007] S11：获取每路视频源的第一帧图像，执行配准操作以及柱面投影变换，求得整体的透视变换模型；

[0008] S12：根据透视变换模型对每路视频源的第一帧图像进行透视变换处理，同时求得透视变换掩模图以及相邻视频源间的重叠区掩模图；

[0009] S13：针对相邻视频源的视频图像，使用动态规划算法分别求得拼接缝上的像素点坐标；

[0010] S14：将透视变换掩模图、重叠区掩模图以及拼接缝像素点传输至 GPU 端的各个缓

冲区中；实时视频帧融合的步骤，包括

[0011] S21：利用透视变换掩模图将每路视频源图像序列中的同一帧图像变换至同一坐标系下；

[0012] S22：利用重叠区掩模图求得每路视频源图像序列中的同一帧图像的重叠区能量值图；

[0013] S23：利用拼接缝像素点缓冲区中的坐标信息以及重叠区能量值图，使用拼接缝的实时动态修正算法重新计算拼接缝像素点坐标，并利用该坐标值更新拼接缝像素点缓冲区；

[0014] S24：根据上述求得的新拼接缝，基于均值坐标无缝融合算法消除各相邻图像间的拼接痕迹；

[0015] S25：将生成的全景视频帧图像传输至已绑定的 OpenGL 的像素缓冲区中，进行快速图像渲染；

[0016] S26：针对后续视频流中的每一帧视频图像重复 S21 至 S25；

[0017] 所述系统初始化的步骤在 CUDA 架构的 CPU 端运行，所述实时视频帧融合的步骤在 GPU 端运行，且 S21、S22、S23 和 S24 基于 CUDA 架构的流处理模式：在 GPU 端创建 4 个并发处理的执行流，将 S21、S22、S23 和 S24 部署到相应的流处理序列中。

[0018] 进一步地，所述获取每路视频源的第一帧图像，执行配准操作以及柱面投影变换，求得整体的透视变换模型的方法是：对每路视频的第一帧图像，使用 SURF 算法进行图像配准，经 RANSAC 算法去除误匹配后，求得相邻两路视频图像之间的透视变换矩阵；利用柱面坐标转换计算得到整体的透视变换模型。

[0019] 更进一步地，所述求得透视变化掩膜图的方法是：创建与透视变换矩阵维度大小一样的透视变换掩模矩阵，使其空间大小可以包含进所有变换后的像素；在参与变换的所有源图像的像素位置上将掩模矩阵对应位置的值置为 1，非变换像素的位置上将掩模矩阵对应位置的值置为 0，得到掩膜矩阵图，掩膜矩阵图对应的显示图即为透视变化掩膜图；所述求重叠区掩模图的方法是：将透视变化掩模图上存在实际像素位置的掩模矩阵元素的值设为 0.5，则两幅图像变换后的重叠区位置上的矩阵元素的值为 1；遍历整个掩模矩阵将非 1 的元素值重置为 0，得到重叠区掩膜矩阵，重叠区掩膜矩阵对应的显示图即为重叠区掩膜图。

[0020] 进一步地，所述使用动态规划算法求得拼接缝上的像素点坐标，包括

[0021] S41：计算重叠区的像素能量值矩阵 En；

[0022] S42：构建方向矩阵 Dir 以及能量和矩阵 Cum；

[0023] S43：方向矩阵 Dir 中搜索拼接缝坐标路径。

[0024] 进一步地，所述使用拼接缝的实时动态修正算法重新计算拼接缝像素点坐标，包括并行寻找拼接缝位置修正起始点的步骤：在当前重叠区像素能量值图中，根据加载的前一帧图像的拼接缝像素点坐标进行比对寻找拼接缝变化的起始点位置，比对规则如下：如果在上一次拼接缝的像素点位置附近的同一行两个像素邻域内仍旧是最小值，则在这一次拼接缝的像素路径中，该像素点位置不需要修改，否则即为修改的起始点坐标；

[0025] GPU 端搜索新的拼接缝修改路径：在 GPU 端根据拼接缝位置变化起始点和贪婪搜索算法重新搜索新的拼接缝位置点。

[0026] 进一步地，所述基于均值坐标无缝融合算法消除各相邻图像间的拼接痕迹，包括以拼接缝上像素点的色彩强度计算权值，去依次校正待融合图像的重叠区中的每一个像素的色彩强度校正值。

[0027] 本技术方案跟现有技术相比，有如下优点：

[0028] 1) 实现多视频无鬼影和色彩亮度差异的实时全景视频

[0029] 2) 对机场场面、园区、广场等大场景超高分辨率整体监视效果显著，应用前景广泛。

附图说明

[0030] 图 1 是系统流程图；

[0031] 图 2 是摄像机采集源图；

[0032] 图 3 是掩模矩阵中像素变换位置的值为 255 的掩模显示图。；

[0033] 图 4 是图像拼接缝划分示意图；

[0034] 图 5 是未加改进融合缝处理效果截图（图中框标明有明显痕迹的拼接缝）；

[0035] 图 6 是改进的均值坐标融合去缝效果截图（图中框内原拼接缝痕迹得到有效改进）；

[0036] 图 7 是 4 台 300 万像素高清摄像机视频拼接的效果截图。

[0037] 图 8 是 CUDA 并发流处理模式示意图。

具体实施方式

[0038] 下面结合附图说明本发明的实施过程，以四台摄像机 (Camera1，Camera2，Camera3，Camera4) 输出的四路视频源为例。

[0039] 如图 1 所示，本发明的实现过程主要分为系统初始化和实时视频帧融合两个阶段：

[0040] 1、系统初始化阶段：

[0041] (1) 获取各路视频的第一帧图像，执行配准操作以及柱面投影变换以求得整体的透视变换模型；

[0042] (2) 根据透视变换模型对每路视频的第一帧图像进行透视变换处理，同时求得透视变换掩模图以及相邻视频源 (Camera1-2，Camera2-3，Camera3-4) 间的重叠区掩模图；

[0043] (3) 针对相邻的两台摄像机的视频图像使用动态规划算法分别求得拼接缝上的像素点坐标；

[0044] (4) 将初始化阶段中求得的透视变换掩模图、重叠区掩模图以及拼接缝像素点传输至 GPU 端的各个缓冲区中。

[0045] 2、实时视频帧融合阶段：

[0046] (1) 利用透视变换掩模图将各路摄像机的每一帧图像变换至同一坐标系下；

[0047] (2) 利用重叠区掩模图求得每一帧图像的重叠区能量值图；

[0048] (3) 利用拼接缝像素点缓冲区中的坐标信息以及重叠区能量值图，使用拼接缝的实时动态修正算法重新计算拼接缝像素点坐标，并利用该坐标值更新拼接缝像素点缓冲区；

[0049] (4) 根据上述求得的新拼接缝，基于均值坐标无缝融合算法消除各相邻图像间的明显的认为拼接痕迹；

[0050] (5) 将生成的全景视频帧图像传输至已绑定的 OpenGL 的像素缓冲区中，进行快速的图像渲染；

[0051] (6) 针对后续摄像机视频流中的每一帧视频图像重复步骤 (1)-(5)。

[0052] 在上述两个阶段中，系统初始化阶段是运行在 CPU 端的，且只在系统首次运行时执行一次；实时视频帧融合阶段是运行在 GPU 端的，在整个系统中针对每一帧视频图像一直处于高效运行状态。在 GPU 端，依靠 CUDA 的多线程数据并行处理模式，即对图像每个像素构建一个处理线程，实现对数据处理实行并行处理。与此同时，在处理视频帧图像序列的流程中，基于 CUDA 架构的流处理模式，对处理任务实行并发执行模式，这样进一步提升了整体流程的执行效率。

[0053] 其中，初始化阶段中透视变换掩模图的求解：

[0054] 采集相邻网络摄像机的第一帧图像，使用 SURF 算法进行图像配准，经 RANSAC 算法去除误匹配之后，可以计算得到两幅图像间的投影变换矩阵。

[0055] 如图 2 中选取的是某公司外部视景图像，根据该算法计算得到的 (2) 图相对于 (1) 图的投影变换矩阵是：

[0056] $$M_{21}=\begin{bmatrix} 0.38758991 & 0.00771459 & 1093.44204682 \\ -0.26907066 & 0.83318325 & 209.16073628 \\ -0.0003175537 & -0.00001448085 & 1 \end{bmatrix}$$

[0057] (3) 图相对于 (2) 图的投影变换矩阵是：

[0058] $$M_{32}=\begin{bmatrix} 0.31518341 & 0.00143954 & 1124.91994290 \\ -0.30305187 & 0.79484374 & 264.49889858 \\ -0.000339152 & -0.0000388052 & 1 \end{bmatrix}$$

[0059] 将视频采集端的其中一台摄像机的图像设为基准图像，利用双线性插值算法根据投影变换矩阵 M_{21} 完成另一台摄像机图像的投影变换，利用上述过程就可以将两幅图像映射到同一坐标系下。对于第三台摄像机图像而言，先将其与第二台摄像机的图像进行配准得到变换矩阵 M_{32}，那么它相对基准图像的透视变换矩阵 M_3 等于 $M_{21}*M_{32}$，依次类推可以求得后续添加的多摄像机图像相对基准图像的透视变换矩阵。基于求得的针对每台谁相机的透视变换矩阵 M_1、M_2、M_3 和 M_4，再利用柱面坐标转换原理就可以计算得到整体图像的透视变换矩阵。

[0060] 创建与透视变换矩阵维度大小一样的透视变换掩模矩阵，使其空间大小可以包含进所有变换后的像素。在参与变换的所有源图像的像素位置上将掩模矩阵对应位置的值置为 1，非变换像素的位置上将掩模矩阵对应位置的值置为 0 就可以得到掩模矩阵图。图 3 所示的是掩模矩阵中像素变换位置的值为 255 的掩模显示图。

[0061] 初始化阶段中相邻视频源间的重叠区掩模图的求解：

[0062] 两台相邻的摄像机在求解透视柱面坐标变换后，将透视柱面变换掩模图上存在实际像素位置的掩模矩阵元素的值设为 0.5，这样在两幅图像变换后的重叠区位置上的矩阵元素的值为 1。遍历整个掩模矩阵将非 1 的元素值重置为 0，这样就可以得到用于取出重叠

区像素值的掩模矩阵。

[0063] 初始化阶段中拼接缝的求解：

[0064] 在初始化阶段，拼接缝的寻找主要包括 3 步：计算重叠区的像素能量值矩阵 En、构建方向矩阵 Dir 以及能量和矩阵 Cum、在方向矩阵 Dir 中搜索拼接缝坐标路径。

[0065] 1、计算重叠区的像素能量值矩阵 En

[0066] 设 I_1 和 I_2 分别是相邻两台摄像机的重叠区图像，I 是新创建的像素能量值矩阵，其宽和高的大小与 I_1 和 I_2 均相等。利用公式 (1) 可以求得像素能量值矩阵 I 中像素位置为 (i, j) 的像素能量值 Δ_{ij}。

[0067] $$\Delta_{ij} = w_1\delta_{ij}^{1} + w_2\delta_{ij}^{\nabla} \quad (1)$$

[0068] $$\delta_{ij}^{1} = \frac{abs(\mathrm{I}_{1_{ij}} - \mathrm{I}_{2_{ij}})}{\max(\mathrm{I}_{1_{ij}}, \mathrm{I}_{2_{ij}})} \quad (2)$$

[0069] $$\delta_{ij}^{\nabla} = \frac{abs(G_{1_{ij}} - G_{2_{ij}})}{\max(G_{1_{ij}}, G_{2_{ij}})} \quad (3)$$

[0070] 在公式 (1) 中，δ_{ij}^{1} 表示重叠区图像 I_1 和 I_2 在像素位置 (i, j) 的色彩差异值，δ_{ij}^{∇} 表示 I_1 和 I_2 在像素位置 (i, j) 的几何结构差异值，其中 w_1 和 w_2 的权重值各取 0.2 和 0.8。公式 (2) 中，$\mathrm{I}_{1_{ij}}$ 和 $\mathrm{I}_{2_{ij}}$ 分别表示 I_1 和 I_2 的灰度值；公式 (3) 中 $G_{1_{ij}}$ 和 $G_{2_{ij}}$ 表示 I_1 和 I_2 的梯度值。

[0071] 依照上述计算步骤，可以利用像素能量值 Δ_{ij} 填充满整个重叠区能量值矩阵 En。

[0072] 2、构建方向矩阵 Dir 以及能量和矩阵 Cum

[0073] 构建方向矩阵 Dir 以及能量和矩阵 Cum，这两个矩阵的维度和能量值矩阵 En 的维度相等。填充矩阵 Dir 以及矩阵 Cum 中的值如下所示：

[0074] 将重叠区第一行中间像素点设为拼接缝的起始点 P(0, w/2)。按照如下规则初始化方向矩阵 Dir 和能量和矩阵 Cum：

[0075] 在第一行方向矩阵 Dir 中，将 P 点处 Dir 矩阵中的元素值置为 0，其余点处 Dir 矩阵中的元素值置为 255。在第一行能量和矩阵 Cum 中，将 P 点处 Cum 矩阵中的元素值置为 En(0, w/2)，其余点处 Cum 矩阵中的元素值置为 65536。

[0076] 从起始点 P 开始，在第 i 行能量矩阵 En 中，依次取位置为 (i-1, j-1)、(i, j-1) 和 (i+1, j-1) 的像素的能量值，标记其搜索位置值为 1、2、3；

[0077] $\mathrm{En}_{\min} = \mathrm{Min}\{\mathrm{Cum}(i-1, j-1), \mathrm{Cum}(i, j-1), \mathrm{Cum}(i+1, j-1)\}$ (4)

[0078] 标记矩阵 Dir 中 Dir(i, j) 的值为 $\mathrm{En}_{\min}$ 对应的搜索位置值

[0079] 标记矩阵 Cum 中 Cum(i, j) 的值为：

[0080] $\mathrm{Cum}(i, j) = \mathrm{En}(i, j) + \mathrm{En}_{\min}$ (5)

[0081] 3、方向矩阵 Dir 中搜索拼接缝坐标路径

[0082] 对能量矩阵 En 每一行进行步骤 2 中的操作，直至扫描结束，可得到能量和矩阵 Cum 和方向矩阵 Dir，将 Cum 中最后一行的最小值所在的位置点 Q(x, y) 设为拼接缝隙的结尾点

[0083] 在方向矩阵 Dir 中，从 Q 点开始，取出 Dir(x, y) 中的方向搜索值，根据该值确定上

一个拼接缝像素点的坐标，直至搜索至 P 点结束。P、Q 以及依次记录的像素点构成了最终的拼接缝。如图 4 所示。

[0084] 实时阶段视频帧图像的透视变换：

[0085] 根据步骤一求得的多个变换矩阵（M_1、M_2、M_3 和 M_4），将每一台摄像机采集的视频帧图像序列中的每一幅视频图像进行图像透视变换。将变换后的图像大小扩充为最终全景图的大小，非源图像的像素值置为 0。可以依次得到 4 幅变换后的图像（I_1、I_2、I_3 和 I_4），然后将这四个图像矩阵与透视变换掩膜矩阵相乘，最终就可以得到每个摄像机的视频图像序列中的同一帧图像的统一坐标系中的配准图。

[0086] 实时阶段重叠区图像的能量值图求解：

[0087] 在四台摄像机布局的图像采集系统中，Camera1 和 Camera2，Camera2 和 Camera3，Camera3 和 Camera4 之间的图像存在重叠区。以求解 Camera2 和 Camera3 之间的重叠区 $Overlap_2$ 为例，根据矩阵 I_2 与 I_3 构建矩阵 I'_2 与 I'_3，I'_2 与 I'_3 内部矩阵的值构建规则如下：存在原始像素源的像素位置上像素值置为 0.5，不存在原始像素源的像素位置上的值置为 0。I'_2 与 I'_3 的矩阵和 I_{23} 中，矩阵元素值为 1 的位置点是两幅图像的重叠区部分。

[0088] 将重叠区掩模矩阵 I_{23} 与后续 Camera2 和 Camera3 视频帧序列中的每一个图像矩阵相乘，就可以计算出每帧图像的重叠区。在 GPU 端创建与图像重叠区大小等同数量的并行线程数目，使得每一个线程可以并行同步使用公式（1）至公式（3）求解出当前相邻位置的相机的图像能量值图。

[0089] 实时阶段拼接缝的动态修正：

[0090] 拼接缝的动态修正过程是在前一帧的拼接缝像素点缓冲区上针对后续每一帧视频图像进行重新的拼接缝修正。首先系统初始化时会加载拼接缝像素点坐标存入缓冲区中，然后计算出当前视频帧图像拼接缝需要修正的像素位置的起始点坐标，再后使用贪婪算法以新的起始点进行像素能量值的大小比较以确定新的拼接缝像素点。主要包括三方面：求解出当前相邻位置的相机的图像能量值矩阵、寻找出拼接缝需要修正的起始点像素位置坐标和搜索出新的拼接缝路径。

[0091] 1、并行求解能量值矩阵：

[0092] 2、并行寻找拼接缝位置修正起始点：

[0093] 在当前重叠区像素能量值图中，根据加载的前一帧图像的拼接缝像素点坐标进行比对寻找拼接缝变化的起始点位置，比对规则如下：如果在上一次拼接缝的像素点位置附近的同一行两个像素邻域内仍旧是最小值，则在这一次拼接缝的像素路径中，该像素点位置不需要修改，否则即为修改的起始点坐标。

[0094] 3、GPU 端搜索新的拼接缝修改路径

[0095] 在 GPU 端根据拼接缝位置变化起始点和贪婪搜索算法重新搜索新的拼接缝位置点。GPU 端的贪婪搜索算法设计可以分成两个函数：方向标记 kernel 函数和寻路 kernel 函数。

[0096] 1)、方向标记函数：

[0097] CUDA 的 Kernel 核函数在显卡芯片上创建与重叠区像素数目相等的线程，使得创建的每一个线程并行地访问重叠区中对应位置像素的能量值。设当前执行线程所在的

二维线程层次模型中的位置是 (i, j)，则其对应索引的重叠区能量值数据区的像素位置是第 i 行中的第 j 列。在该核函数内依次顺序取出三个像素位置的能量值 :E(i-1, j+1)、E(i, j+1)、E(i+1, j+1)，选取三者中最小的能量值 E_{min}，若 E_{min} 取值应为 E(i-1, j+1)，记该点方向标记值 P(i, j) 为 1，若取为 E(i, j+1)，P(i, j) 为 2，若取为 E(i+1, j+1)，P(i, j) 为 3。将每个像素的标记位置存储在方向标记缓冲区中，该数据区保存的是对应位置像素的下一个搜索点。

[0098] 2)、路径寻找函数

[0099] 路径寻找函数是根据当前给定的拼接缝修正起始点坐标，在计算出的方向标记缓冲区中向下搜索标记路径。以拼接缝修正起始点坐标 (i, j) 为例，取出方向标记缓冲区中 (i, j) 对应的方向标记值 P(i, j)，如果该值是 1，则将下一个拼接缝像素点位置标记为 (i-1, j+1) ；如果该值是 2，则将下一个拼接缝像素点位置标记为 (i, j+1) ；如果该值是 3，则将下一个拼接缝像素点位置标记为 (i+1, j+1)。从起始位置搜索至重叠区底端，得到最终的修正后的拼接缝像素点坐标。

[0100] 通过上述计算可以得到新的修正之后的拼接缝像素位置点坐标，然后该拼接缝的像素位置信息更新之前的拼接缝像素参考点，作为下一帧拼接缝寻找的参考值。

[0101] 实时阶段拼接缝的隐藏 :

[0102] 通过实时阶段视频帧图像的透视变换可以有效的消除因为配准误差以及重叠区运动目标产生的鬼影现象以及图像错位现象，但是解决了上述问题的同时，在融合图像中会产生明显的拼接缝。如图 5 所示。为了解决该问题，发明中使用了基于均值坐标融合的拼接缝消除算法。该算法中以拼接缝上像素点的色彩强度计算权值 (计算公式参见公式 5)，去依次校正待融合图像的重叠区中的每一个像素的色彩强度校正值 (计算公式参见公式 7)。修正后的图像如图 6 所示。

[0103] $$\omega_i(q)=\frac{1/\|p_i-q\|}{\sum_{j=1}^{n}1/\|p_j-q\|} \qquad (6)$$

[0104] $$M(p)=\sum_{i}^{n}\omega_i(p)m_i \qquad m_i=I_{pi}-I_{qi} \qquad (7)$$

[0105] 公式 (6) 中，P_i 是基准图像的拼接缝上的像素点的灰度值，q 是待融合图像的重叠区中的任意一个像素。公式 (7) 中，M(p) 是当前像素的色彩修正值，I_{pi} 和 I_{qi} 分别是两幅图像拼接缝的像素点。

[0106] 图 7 是 4 台 300 万像素高清摄像机视频，经过本发明的拼接方法处理后的效果截图。

[0107] 在 CUDA 并行加速的设计中，由于位于同一行的待修正像素所用到的修正参考拼接缝像素是一致的，因此在 CUDA 线程维度设计中，将每一行的像素设计在一个 block 中，这样 block 内部的所有线程 (每个像素) 均可以共享用于计算权值的像素的位置信息以及像素值。然后，所有的位于重叠区的像素并行计算色彩修正权值，并在实际的色彩修正值上进行修正。

[0108] 对于实时处理阶段中的步骤 (1) ～步骤 (4) 处理任务中涉及的视频帧图像序列的处理流程，基于 CUDA 架构的流处理模式，在 GPU 端创建 4 个并发处理的执行流 (Stream)，将

步骤 (1) 至步骤 (4) 部署到相应的流处理序列中。帧图像序列在执行流管道中并发处理，具体并发处理流程如图 8 所示，最终满足视频拼接融合实时性要求。

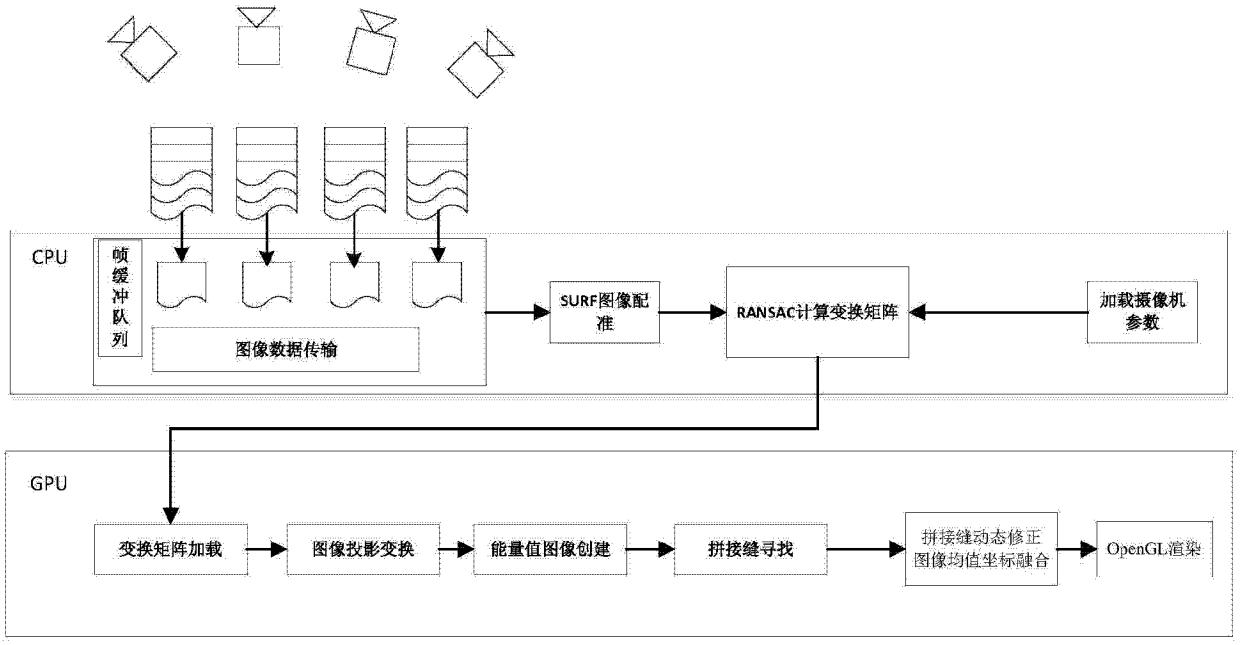

CPU
帧缓冲队列
图像数据传输
SURF图像配准
RANSAC计算变换矩阵
加载摄像机参数
GPU
变换矩阵加载
图像投影变换
能量值图像创建
拼接缝寻找
拼接缝动态修正
图像均值坐标融合
OpenGL渲染

图 1

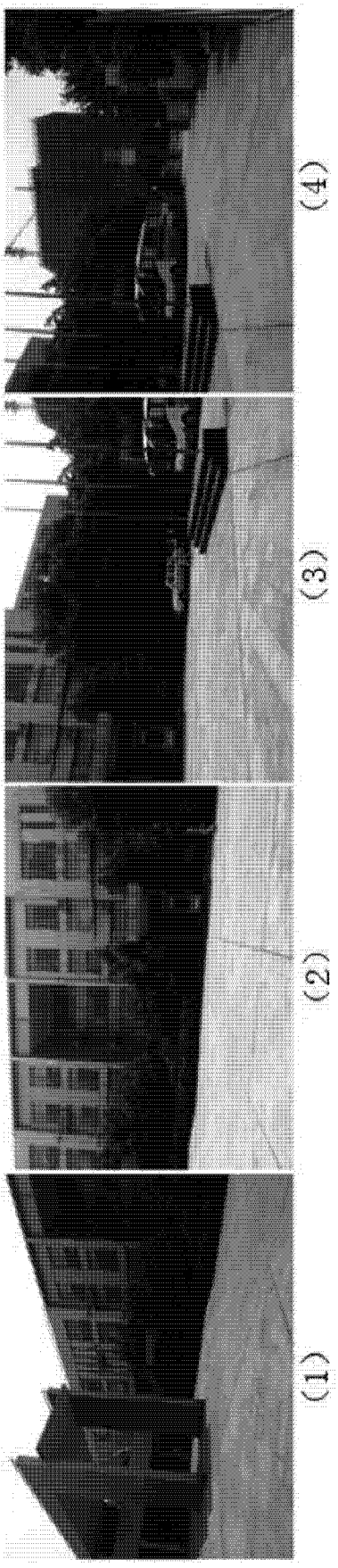


图 2

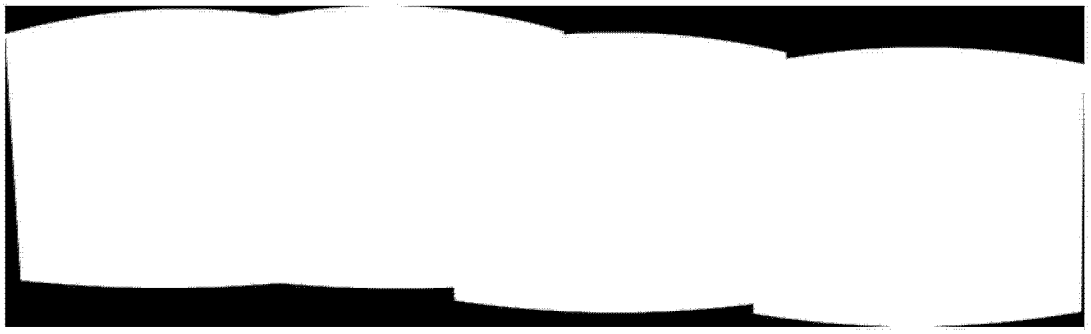


图 3

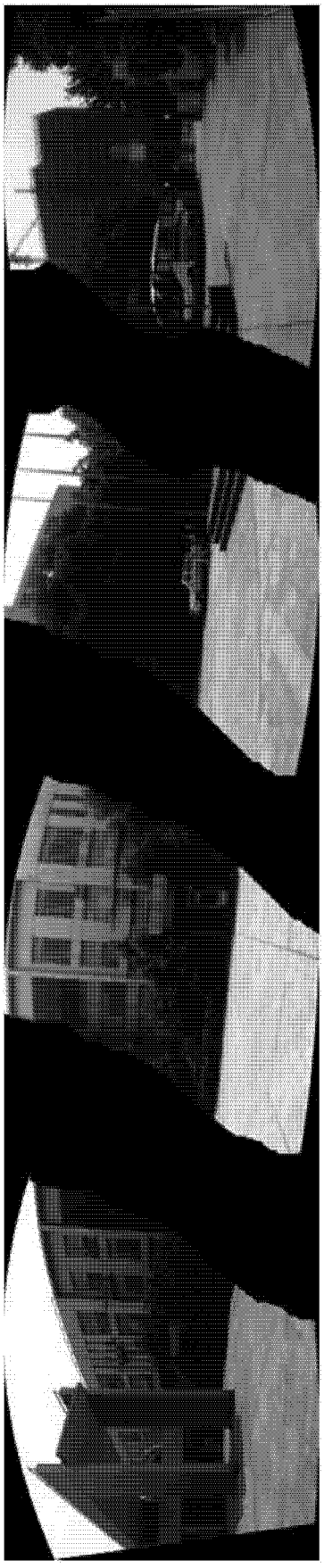


图 4

图 5

图 6

图 7

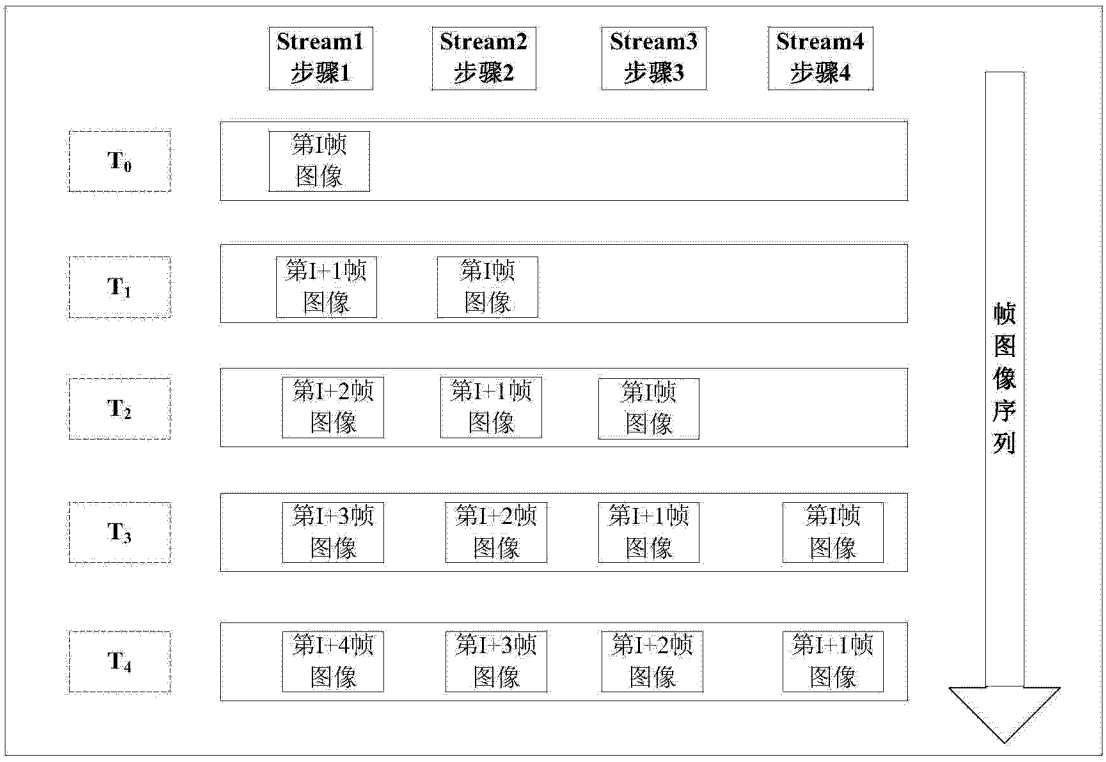

Stream1
步骤1
Stream2
步骤2
Stream3
步骤3
Stream4
步骤4
T0
第I帧
图像
T1
第I+1帧
图像
第I帧
图像
T2
第I+2帧
图像
第I+1帧
图像
第I帧
图像
T3
第I+3帧
图像
第I+2帧
图像
第I+1帧
图像
第I帧
图像
T4
第I+4帧
图像
第I+3帧
图像
第I+2帧
图像
第I+1帧
图像
帧图像序列

图 8