(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 104254868 A
(43)申请公布日 2014.12.31

(21)申请号 201380007015.4
(22)申请日 2013.08.23
(30)优先权数据
2013102854 2013.01.30 RU
(85)PCT国际申请进入国家阶段日
2014.07.28
(86)PCT国际申请的申请数据
PCT/US2013/056402 2013.08.23
(87)PCT国际申请的公布数据
WO2014/120281 EN 2014.08.07
(71)申请人 LSI 公司
地址 美国加利福尼亚
(72)发明人 D·V·帕克霍门库
I·L·马祖雷恩库
P·A·阿里塞切克 D·N·巴比恩
D·V·扎伊特塞弗
(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 刘倜
(51)Int.Cl.
G06T 1/00(2006.01)

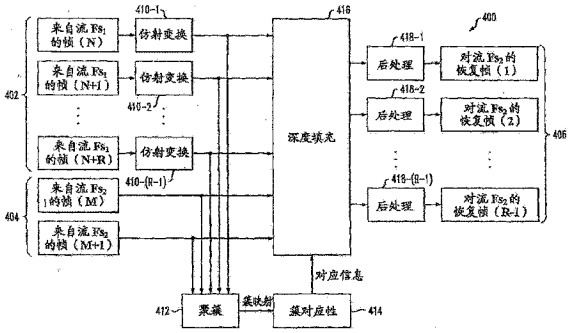
权利要求书2页 说明书14页 附图6页

(54)发明名称

使用至少一个较高帧速率图像流提高图像流帧速率的方法和装置

(57)摘要

一种图像处理系统包括图像处理器，所述图像处理器被配置为获得具有第一帧速率的第一图像流和具有第二帧速率的第二图像流，所述第二帧速率低于所述第一帧速率，基于所述第一图像流和第二图像流的现有帧恢复所述第二图像流的附加帧，以及使用附加帧来为第二图像流提供增加的帧速率。基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧包括确定用于在各自的迭代中插入第二图像流中的连续现有帧的相应的对之间的一个或多个附加帧的集合。



CN 104254868 A

1. 一种方法，包括：

获得具有第一帧速率的第一图像流和具有比所述第一帧速率低的第二帧速率的第二图像流；

基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧；以及

使用所述附加帧来为所述第二图像流提供增加的帧速率；

其中在至少一个包括耦接到存储器的处理器的处理设备中执行所述获得、恢复和使用。

2. 如权利要求 1 所述的方法，其中获得所述第一图像流和第二图像流包括从深度成像器获得作为深度图像流的、所述第一图像流和第二图像流中的至少一个图像流。

3. 如权利要求 1 所述的方法，其中获得所述第一图像流和第二图像流包括：从配置用于对给定的场景成像的相应的不同图像传感器获得所述第一图像流和第二图像流。

4. 如权利要求 1 所述的方法，其中所述第二图像流中的两个连续现有帧对应于所述第一图像流中的 R+1 个连续现有帧，从而如果所述第一图像流中的第 N 个现有帧对应于所述第二图像流中的第 M 个现有帧，则所述第一图像流中的第 (N+R) 现有帧对应于所述第二图像流中的第 (M+1) 现有帧。

5. 如权利要求 4 所述的方法，其中基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧包括：确定用于插入所述第二图像流中的第 M 现有帧和第 (M+1) 现有帧之间的 R+1 个附加帧。

6. 如权利要求 5 所述的方法，其中所确定的用于插入所述第二图像流中的第 M 现有帧和第 (M+1) 现有帧之间的 R+1 个附加帧是基于所述第一图像流中的对应的 R+1 个连续现有帧和所述第二图像流中的第 M 现有帧和第 (M+1) 现有帧确定的。

7. 如权利要求 1 所述的方法，其中基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧包括：确定用于插入所述第二图像流中的相应对的连续现有帧的之间的一个或多个附加帧的集合。

8. 如权利要求 7 所述的方法，其中对于给定的迭代，确定用于插入第二图像流中的所述连续现有帧中的对应的一对之间的一个或多个附加帧的特定集合，所述确定是基于第一图像流中多个对应的现有帧和第二图像流中的所述对应对的连续现有帧的。

9. 如权利要求 1 所述的方法，其中基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧包括：

对所述现有帧中相应的帧应用聚簇操作，以产生相应的簇映射；以及

产生指示所述簇映射的簇之间的对应性的簇对应信息。

10. 如权利要求 9 所述的方法，还包括在对其应用聚簇操作之前，至少对所述第一图像流和第二图像流中的至少一个图像流的现有帧的子集应用仿射变换。

11. 如权利要求 9 所述的方法，其中对所述现有帧中给定的现有帧应用的聚簇操作包括基于统计区域合并的聚簇操作，其中，将所述给定的现有帧分到多个簇中，所述多个簇中的每一个对应不同统计区域。

12. 如权利要求 11 所述的方法，其中所述基于统计区域合并的聚簇操作根据以下式，对于所述给定的现有帧的任意两个统计区域 R_1 和 R_2，使用规定的合并谓词实现回归合并：

$$P(R_1,R_2)=\begin{cases}\text{真，如果}|R_1-R_2|\le\sqrt{b^2(R_1)+b^2(R_2)}\\ \text{假，}\quad\text{其它情况}\end{cases},$$

其中 $|R_1 - R_2|$ 表示区域 R_1 中的像素的数目和区域 R_2 中的像素的数目之间的差的大小，并且 $b(R_i)$ 是区域 R_i 中像素的数目的函数并且是在所述给定的帧中的像素的最大可能值，使得如果 $P(R_1, R_2)$ ＝真，则将区域 R_1 及 R_2 合并到单一的簇中。

13. 如权利要求 9 所述的方法，还包括使用簇对应信息执行深度填充操作，其中，与恢复所述附加帧关联地，将与第一图像流和第二图像流的一个或多个现有帧中的一个或多个簇相关联的深度数据，添加到所述一个或多个所述附加帧中对应的簇。

14. 如权利要求 1 所述的方法，其中基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧包括：

识别所述第一图像流的一个或多个现有帧的部分和所述第二图像流的一个或多个现有帧的部分之间的对应性；以及

使用来自所述识别的部分的图像信息形成所述附加帧中的至少一个。

15. 一种其中嵌入有计算机程序代码的计算机可读存储介质，其中所述计算机程序代码当在处理设备中执行时，使得所述处理设备执行如权利要求 1 所述的方法。

16. 一种装置，包括：

至少一个处理设备，其包括耦接到存储器的处理器；

其中所述至少一个处理设备被配置来获得具有第一帧速率的第一图像流和具有比所述第一帧速率低的第二帧速率的第二图像流，基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧；以及使用所述附加帧来为所述第二图像流提供增加的帧速率。

17. 如权利要求 16 所述的装置，其中所述处理设备包括图像处理器，所述图像处理器包括：

聚簇模块，配置为对所述现有帧中相应的现有帧应用聚簇操作，以产生相应的簇映射；以及

簇对应模块，配置为产生指示所述簇映射的簇之间的对应性的簇对应信息。

18. 如权利要求 17 所述的装置，其中所述图像处理器还包括深度填充模块，所述深度填充模块被配置为使用所述簇对应信息执行深度填充操作，其中，与恢复所述附加帧关联地，将与第一图像流和第二图像流的一个或多个现有帧中的一个或多个簇相关联的深度数据，添加到一个或多个所述附加帧中相应的簇。

19. 一种图像处理系统，包括：

第一图像源，提供具有第一帧速率的第一图像流；

第二图像源，提供具有比所述第一帧速率低的第二帧速率的第二图像流；以及

图像处理器，耦接到所述第一图像源和第二图像源；

其中所述图像处理器被配置为基于所述第一图像流和第二图像流的现有帧为所述第二图像流恢复附加帧，以及使用所述附加帧为所述第二图像流提供增加的帧速率。

20. 如权利要求 19 所述的系统，其中所述第一图像源和第二图像源中的至少一个包括深度成像器。

使用至少一个较高帧速率图像流提高图像流帧速率的方法和装置

技术领域

[0001] 领域总的来说涉及图像处理，并且更具体地，涉及具有不同帧速率的多个图像流的处理。

背景技术

[0002] 图像处理在许多不同机器视觉应用中是重要的，并且这样的处理可能涉及从许多不同的图像源获得的多个不同类型的图像，可能包括二维 (2D) 图像和三维 (3D) 图像。例如，由诸如视频摄像机的图像源提供 2D 图像，以及由诸如结构光 (SL) 摄像机或飞行时 (ToF) 摄像机的深度成像器提供 3D 图像。因此，传统的图像处理技术经常需要处理来自多个不同源的图像流。然而，当不同的源以不同的帧速率产生图像时会产生问题。

[0003] 概述

[0004] 在一个实施例中，一种图像处理系统包括图像处理器，该图像处理器被配置来获得具有第一帧速率的第一图像流和具有低于所述第一帧速率的第二帧速率的第二图像流，基于所述第一和第二图像流的现有帧为第二图像流恢复附加帧，以及利用附加帧为所述第二图像流提供增加的帧速率。仅作为示例，基于所述第一和第二图像流的现有帧为第二图像流恢复附加帧可以示例性地包括：确定用于插入第二图像流中的相应对的连续现有帧之间的一个或多个附加帧的集合。

[0005] 本发明的其它实施例包括（但不限于）方法、装置、系统、处理设备、集成电路和其中嵌入有计算机程序代码的计算机可读存储介质。

附图说明

[0006] 图 1 是在一个实施例中的图像处理系统的框图。

[0007] 图 2 示出了将恢复的附加帧插入到低帧速率图像流中以增加图 1 的系统中的图像流的帧速率。

[0008] 图 3 是用于增加图 1 系统中的低帧速率图像流的帧速率的示例性处理的流程图。

[0009] 图 4 是用于增加图 1 系统中的低帧速率图像流的帧速率的另一示例性处理的流程图。

[0010] 图 5 至图 7 是示出图 1 系统中的图像处理器的簇 (cluster) 对应模块的操作的流程图。

具体实施方式

[0011] 在此将结合包括图像处理器或其它类型的处理设备的示例图像处理系统，以及用于使用至少一个较高速率的图像流提高深度图像流或其它类型图像流的帧速率的实现技术，说明本发明的实施例。然而，应理解，本发明的实施例更一般地适用于涉及处理具有不同帧速率的多个图像流的任何图像处理系统或相关的设备或技术。

[0012] 图 1 示出了本发明的一个实施例中的图像处理系统 100。图像处理系统 100 包括图像处理器 102，其通过网络 104 与多个处理设备 106 通信。图像处理器 102 从相应的图像源 107 接收至少两个不同的图像流。更具体地，图像处理器 102 从高速率图像源 107-1 接收具有第一帧速率的第一图像流 Fs_1，并从低速率图像源 107-2 接收具有比第一帧速率低的第二帧速率的第二图像流 Fs_2。可以不受限制构思图像源 107 提供从给定场景的相同或相似视点获得的图像流，使得可以在高帧速率图像流和低帧速率图像流中都呈现相似的像素簇或其它图像分段。在本文中应用到帧速率的术语“高”和“低”是相对术语，并且不应解释为需要任何特定绝对的帧速率。

[0013] 例如，高帧速率图像源 107-1 可以包括提供 2D 图像序列的视频摄像机或其它视频源，而低帧速率图像源 107-2 可以包括提供深度图像序列的深度成像器，例如 SL 摄像机或 ToF 摄像机。

[0014] 在其它实施例中，可以使用许多其它类型的图像源生成多个图像流，包括配置为产生 2D 红外图像、灰度图像、彩色图像或其它类型的 2D 图像的 2D 成像器以及 3D 成像器（例如，SL 和 ToF 摄像机），其可以任意组合。

[0015] 示例性地，图像源 107 可以包括各自的图像源，每一个产生单独的图像流。可以单独地安装传感器并将其布置为相互分开，或者，传感器可以包括一体的传感器的不同的部分，其中所述一体的传感器具有用于产生第一图像流的第一组传感器和用于产生第二图像流的第二组传感器。因此，可以从配置为对给定场景成像的相应的不同图像传感器获得第一和第二图像流。此处，通常假定这样不同的图像传感器基本上捕获相同的场景，但是以不同的帧速率捕获。

[0016] 尽管由分开的图像源 107-1 和图像源 107-2 提供图 1 的实施例中的第一图像流 Fs_1 和第二图像流 Fs_2，但是在其它实施例中，可以仅仅使用单一的成像器或其它类型的图像源来提供不同帧速率的多个图像流。因此，例如，第一和第二图像源 107-1 和 107-2 可以代表单一的多流成像器的不同部分，例如，如上所示的不同的传感器。作为此处广泛使用的术语，给定的图像源可以包括存储之前捕获的图像流以供图像处理器 102 或代表图像处理器 102 的取回的存储器和其它存储设备。又例如，图像源可以包括提供一个或多个图像流到图像处理器 102 用于处理的服务器。

[0017] 另外，在图 1 中仅处理两个输入图像流以提高其中一个图像流的帧速率，但是，在其它实施例中，可以不止使用第一和第二图像流。例如，可以处理每一个都具有不同帧速率的三个或更多个图像流，以恢复附加帧，来提高至少一个图像流的帧速率。

[0018] 在本实施例中，图像处理器 102 被配置为基于第一图像流 Fs_1 和第二图像流 Fs_2 的现有帧为第二图像流 Fs_2 恢复附加帧，并被配置为使用所述附加帧为第二图像流 Fs_2 提供增加的帧速率。例如，图像处理器 102 可以增加第二图像流的帧速率，直到其基本上等于第一图像流的帧速率，使得第一和第二图像流可以更容易地由一个或多个目标设备处理。

[0019] 更具体地，图像处理器 102 可以被配置为确定一个或多个附加帧的集合以用于插入在第二图像流中相应对的连续现有帧之间。每一这样的确定可以认为是将要结合图 3 和图 4 的流程图描述的示例性处理过程之一的迭代。对于给定的这样的迭代，基于第一图像流中的多个对应的现有帧和第二图像流中的连续现有帧对中的对应的连续现有帧对，确定一个或多个附加帧的特定集合，以用于插入在第二图像流中的连续现有帧对中对应的对之

间。

[0020] 在图 2 中示出了该技术，其中图 2 示出了第一图像流 Fs_1 的现有帧和第二图像流 Fs_2 的现有帧之间的对应的一个例子。在这一例子中，假定第二图像流是包括深度图像流的深度流。由于第一图像流具有比第二图像流高的帧速率，所以第一图像流在图中示出的时间段上包括比第二图像流多的帧。图中标为未知的第二图像流的帧是由图像处理器 102 为第二图像流 Fs_2 恢复的以为第二图像流 Fs_2 提供增加的帧速率的附加帧的例子。

[0021] 第二图像流 Fs_2 中的两个连续现有帧对应于第一图像流 Fs_1 中的 R+1 个连续现有帧，从而第一图像流 Fs_1 中的第 N 现有帧对应于第二图像流中的第 M 现有帧，并且第一图像流中的第 (N+R) 现有帧对应于第二图像流中的第 (M+1) 现有帧。如此，对于在该例子中的第一图像流中的每 R+1 个连续现有帧，在第二图像序列中仅有两个连续现有帧。

[0022] 第一图像流 Fs_1 和第二图像流 Fs_2 中对应的帧可以是由各自的图像传感器在基本上相同的时刻捕获的帧。然而，在本上下文中术语“对应”应当被更宽泛地解释，以包含第一和第二图像流的帧之间的其它类型的时间关系。

[0023] 在为第二图像流 Fs_2 恢复附加帧中，图像处理器 102 确定用于插入第二图像流中的第 M 现有帧和第 (M+1) 现有帧之间的 R+1 个附加帧。如下面将详细描述的，所确定的用于插入第二图像流中的第 M 现有帧和第 (M+1) 现有帧之间的 R+1 个附加帧是基于第一图像流中的对应的 R+1 个连续现有帧和第二图像流中的第 M 和第 (M+1) 现有帧确定的。例如，恢复附加帧可以涉及：识别第一图像流的一个或多个现有帧的部分和第二图像流的一个或多个现有帧的部分之间的对应性，以及使用来自已识别部分的图像信息形成至少一个所述附加帧。在一些实施例中，这些部分被更特定地称为“簇”。

[0024] 从上述可见，本实施例中的图像处理器 102 被配置来生成修改的第二图像流 Fs_2'，其包括在给定的时间段内的一个或多个附加帧，并且因此其具有比原始输入第二图像流 Fs_2 高的帧速率。

[0025] 应理解，仅通过说明性的例子呈现了图 2 中示出的特定布置，而在其它实施例中，可以存在第一和第二图像流的帧之间的其它类型的对应性。本公开的技术可以以直接了当的方式适用于任何具有帧速率比 $\frac{Fr_1}{Fr_2}>1$ 的多流布置中，其中 Fr_1 表示第一图像流 Fs_1 的帧速率而 Fr_2 表示第二图像流 Fs_2 的帧速率。

[0026] 如图 1 所示的图像处理器 102 包括帧捕获模块 108，其被配置用于从第一图像序列 Fs_1 和第二图像序列 Fs_2 捕获帧以用于进一步处理。应理解，在本上下文中“帧捕获”通常指获得第一和第二图像序列的特定帧以供图像处理器进一步处理的一种方式。这区别于由相关联的图像传感器对图像的原始捕获。

[0027] 处理捕获的帧以恢复第二图像序列的附加帧，从而为第二图像序列提供较高的帧速率。如上所指出的，较高速率的第二图像序列在此处也称为修改的第二图像序列 Fs_2'，因为其包括恢复的附加帧，所述恢复的附加帧不是原始输入第二图像序列的一部分。

[0028] 使用图像处理器 102 的仿射变换模块 110、聚簇模块 112、簇对应模块 114、深度填充模块 116 和后处理模块 118 来实现为第二图像序列恢复附加帧。将结合图 3 至图 7 的流程图更加详细地描述这些模块的操作。尽管在图 1 实施例中分开地示出了这些示例性的模块，但是在其它实施例中，这些模块中的两个或更多个可以组合到更少数目的模块中。

[0029] 由图像处理器 102 产生的修改的第二图像流 Fs_2' 可以经网络 104 提供到一个或多个处理设备 106。处理设备 106 可以包括例如计算机、移动电话、服务器或存储设备的任何组合。一个或多个该设备还可包括例如显示屏或其它用于呈现由图像处理器 102 产生的图像的用户界面。因此，处理设备 106 可以包括多种不同的目标设备，其经网络 104 从图像处理器 102 接收处理过的图像流，包括例如从图像处理器 102 接收一个或多个处理过的图像流的至少一个服务器或存储设备。

[0030] 尽管在本实施例中，图像处理器 102 被示出为与处理设备 106 分离，但是图像处理器 102 也可以至少部分地与一个或多个处理设备结合。因此，例如，可以至少部分地使用处理设备 106 中给定的一个来实现图像处理器 102。例如，计算机或移动电话可以被配置为包括图像处理器 102 及可能的一个或多个图像源 107。因此，图像源 107 可以包括与计算机、移动电话或其它处理设备相关联的摄像机或其它成像器。因此，显然，图像处理器 102 可以至少部分地与一个或多个图像源或图像目标结合在普通处理设备上。

[0031] 构思使用至少一个处理设备实现本实施例中的图像处理器 102，并且图像处理器 102 包括耦接到存储器 122 的处理器 120。处理器 120 执行存储在存储器 122 中的软件代码以控制图像处理操作的执行。图像处理器 102 还包括支持经网络 104 的通信的网络接口 124。

[0032] 处理器 120 可以包括例如下列的任意组合：微处理器、专用集成电路 (ASIC)、现场可编程门阵列 (FPGA)、中央处理器 (CPU)、算术逻辑单元 (ALU)、数字信号处理器 (DSP) 或其它类似的处理设备组件，以及其它类型和布置的图像处理电路。

[0033] 存储器 122 存储由处理器 120 在实现图像处理器 102 的部分功能 (例如，模块 110、112、114、116 和 118) 中执行的软件代码。存储由对应的处理器执行的软件代码的给定的该存储器是此处更一般地称作其中嵌入有计算机程序代码的计算机可读介质或其它类型的计算机程序产品的一个例子，并且可以包括，例如下列的任何组合：电子存储器，诸如，随机存取存储器 (RAM) 或只读存储器 (ROM)；磁存储器；光学存储器；或，其它类型的存储设备。如上所述的，处理器可以包括微处理器、ASIC、FPGA、CPU、ALU、DSP 或其它图像处理电路的组合或部分。

[0034] 还应理解，可以以集成电路的形式实现本发明的实施例。在给定的该集成电路实现方式中，通常在半导体晶片表面上以重复的图案形成相同的管芯 (die)。每一个管芯包括此处所描述的图像处理电路并可以包括其它结构或电路。从晶片切片或切割单独的管芯，并随后将其封装为集成电路。本领域的技术人员知晓如何对晶片切片和封装管芯以生产集成电路。如此制作的集成电路被认为是本发明的实施例。

[0035] 如图 1 中示出的图像处理系统 100 的特定配置仅仅是示例性的，并且在其它实施例中系统 100 可以包括另外的或替代那些具体示出的元件的其它元件，包括在这类系统的常规实现方式中常见的形式的一个或多个元件。

[0036] 例如，在一些实施例中，图像处理系统 100 被实现为视频游戏系统或处理图像流以识别用户姿势的其它类型的基于姿势的系统。所公开的技术可以类似地适于在需要基于姿势的人机界面的许多其它系统中使用，并且还可以适于除了姿势识别以外的其它应用，例如在机器人和其它工业应用中的机器视觉系统。

[0037] 现参照图 3，示出了用于提高第二图像流 Fs_2 的帧速率的示例处理 300。设想该处

理由图像处理器 102 至少使用其模块 108、110、112、114、116 和 118 实现。该实施例中的处理包括步骤 302 至 312，对于多次迭代的每一次重复这些步骤。在每一次迭代中，该处理获得 R+3 个帧，包括来自第一图像流 Fs_1 的 R+1 个帧和来自第二图像流 Fs_2 的两个帧，并恢复 R-1 个帧以用于插入第二图像流 Fs_2 的两个帧之间以提高该图像流的帧速率。该处理过程恢复的附加帧此处还被称为第二图像流的“连接”帧。另外，该实施例中的帧被称为相应的图像。此处使用的术语“帧”应当被宽泛地理解，以包括图像或它类型和布置的图像信息。

[0038] 在步骤 302 中，图像处理器 102 在图 3 处理的给定的迭代中，从 Fs_1 和 Fs_2 图像流“获取”或者获得特定的指定图像。这些图像包括来自第一图像流 Fs_1 的被表示为 $Im_1(N)$ 到 $Im_1(N+R)$ 的 R+1 个图像，以及来自第二图像流 Fs_2 的被表示为 $Im_2(M)$ 和 $Im_2(M+1)$ 的两个图像。为了清楚简洁的描述，假定所有图像 $Im_i(j)$ 具有相同的尺寸或像素分辨率，尽管在其它实施例中可以使用不同尺寸或分辨率的图像。

[0039] 在步骤 302 的上下文中所引述的 R+3 个图像此处更一般地是指上述来自第一图像流 Fs_1 的 R+1 帧和来自第二图像流 Fs_2 的两帧。可以使用帧捕获模块 108 从相应的图像流中获得所述图像。

[0040] 在步骤 304 中，至少对在步骤 302 中获得的 R+3 个图像的子集应用仿射变换。例如，仅对来自图像流 Fs_1 的 R+1 个图像，或仅对来自第二图像流 Fs_2 的两个图像，应用仿射变换。仿射变换是用于基本上使图像的视点均等以利于后续的聚簇操作的校准变换的一种类型的例子。仿射变换可以基于在传感器制造或设置时执行的图像传感器校准的结果，并使用图像处理器 102 的仿射变换模块 110 来应用仿射变换。在其中来自第一和第二图像流的图像的视点已基本上均等的实施例中（例如，由于用于第一和第二图像流的图像传感器的布置或设置），可以省略仿射变换操作。

[0041] 在步骤 306 中，通过对这些图像应用聚簇操作，将该 R+3 个图像划分到簇中。使用图像处理器 102 的聚簇模块 112 实现该操作。更具体地，聚簇操作涉及为 R+3 个图像中的每一个生成单独的簇映射。更特定地，为输入图像集合 $\{Im_1(N), Im_1(N+1), \cdots, Im_1(N+R); Im_2(M), Im_2(M+1)\}$ 中的每一个图像生成单独的簇映射。不同图像的簇映射可以具有不同的特征，例如，不同数目的簇和不同的簇编号顺序。

[0042] 例如，可以以如下的方式定义图像 $Im_i(j)$ 的给定的簇映射 $Cm_i(j)$。假定来自图像 $Im_i(j)$ 的所有像素的集合被划分到像素的不交叉的子集中，每一个这样的子集代表一个簇。在这种情况下的簇映射可以是具有与图像 $Im_i(j)$ 相同大小的矩阵 $Cm_i(j)$ 的形式。来自 $Cm_i(j)$ 的元素 (m,n) 对应于具有坐标 (m,n) 的图像像素所属的 $Im_i(j)$ 的特定簇的索引。在其它实施例中可以使用其它类型的簇映射。因此，此处使用的术语“簇映射”应被宽泛地解释。例如，由簇对应模块 114 产生的簇对应信息的至少一部分也可以以一个或多个簇映射的形式产生。

[0043] 在实现步骤 306 中可以使用许多不同的聚簇技术。在此将在其它地方描述基于统计区域合并 (SRM) 的聚簇技术的详细例子。在 R.Nock 和 Nielsen 所著的“Statistical region merging”(IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.26, No.11, 2004 年 11 月）中公开了 SRM 的常规方面，通过引用将其合并在此。该实施例中的聚簇技术通常试图确保所识别的簇的边界包括成像场景中对应的物体的显著边界，即使这些物体可能位于离相应的第一和第二图像流的成像传感器不同的距

离，或者，可能呈现不同的颜色或在两个图像流中具有其它不同的特征。

[0044] 在步骤308中，基于在步骤306中为相应的图像生成的簇映射确定簇对应信息。由图像处理器102的簇对应模块114执行该确定。簇对应信息指示步骤306中生成的簇映射的簇之间的对应性，并且如上所述的，表达簇对应信息本身可以至少部分地使用其它类型的簇映射（例如，本文其它地方表示为 Cg_1 和 Cg_2 的簇映射）来表示。

[0045] 作为一个例示，步骤308中的簇对应操作可以接受来自于步骤306的簇映射 $Cm_1(N),\cdots,Cm_1(N+R)$；$Cm_2(M),Cm_2(M+1)$，和输入图像的集合 $\{Im_1(N),Im_1(N+1),\cdots,Im_1(N+R)$；$Im_2(M),Im_2(M+1)\}$ 作为其输入。在这种情况下，确定簇对应涉及寻找不同图像中的簇的集合之间的关系，使得例如，来自第一图像流的一个输入图像 $Im_1(j)$ 的簇的非交叉集合对应于来自第二图像流的另一输入图像 $Im_2(k)$ 的簇的非交叉集合。

[0046] 因此，簇对应可以将 $Im_1(j)$ 中的多个簇的集合识别为对应于 $Im_2(k)$ 中的多个簇的不同的集合，两者簇都与相同的成像物（一个或多个）相关联。假定 $Im_1(j)$ 和 $Im_2(k)$ 的簇的集合基本上覆盖这些对应的图像的图像像素的整个集合。还假定由各自的图像传感器在基本上相同的时间捕获图像 $Im_1(j)$ 和 $Im_2(k)$，例如图2的例子中的帧N和M或帧N+R和M+1。将在下面结合图5至图7描述示例性的簇对应确定的附加细节。

[0047] 在步骤310中，使用在步骤308中获得的簇对应信息应用深度填充操作。这一般涉及将来自第一和第二图像流的图像中的对应簇的深度数据合并到一个或多个附加帧中。更具体地，如图中所指示的，在恢复帧N+i中，用来自第二图像流 Fs_2 的帧M和M+1的对应簇的深度数据填充来自第一图像流 Fs_1 的帧N+i的至少一个簇。在深度填充操作中填充到恢复的帧中的深度数据可以以在本文中其它地方描述的方式预处理。在其它实施例中，在基于簇对应信息恢复附加帧中可以使用不同于深度数据的图像信息，或者除了深度数据以外还使用其它图像信息。

[0048] 如步骤312所示的，对于 $i=1$ 到R-1，重复步骤310的深度填充操作，以恢复用于提高第二图像流 Fs_2 的帧速率的R-1个附加帧的完整集合。由图像处理器102的深度填充模块116执行这些深度填充操作。

[0049] 尽管在图3的实施例中没有明确示出，但是在步骤310和312的深度填充操作之后，可以应用后处理操作以增强恢复的附加帧的质量。使用图像处理器102的后处理模块118执行该后处理操作。在其它实施例中，可以省略后处理或将后处理至少部分地与深度填充操作结合。另外，在实现处理300中，可以使用其它的附加或替换操作的集合，并因此，图中所示出的特定操作应被认为仅仅是示例性的。

[0050] 在图4中示出了用于提高低帧速率图像流 Fs_2 的帧速率的另一示例性处理过程400。如所示出的，该处理取来自第一图像流 Fs_1 的R+1个帧402和来自第二图像流 Fs_2 的两个帧404作为其输入，并产生第二图像流 Fs_2 的R-1个恢复的附加帧406作为其输出。在该图中，来自第一图像流的R+1个帧402被表示为帧N到N+R，来自第二图像流的两个帧404被表示为帧M到M+1，并且R-1个输出帧406被表示为恢复的帧1到R-1。再次地，在其它实施例中可以使用其它数量和类型的输入帧和输出帧。

[0051] 处理过程400包括用于仿射变换410、聚簇412、簇对应414、深度填充416和后处理418的处理操作，在附图中示出为相应的块，设想这些操作由图像处理器102的相应的模块110、112、114、116和118实现。如前所述的，这些处理操作仅仅是示例性的，并且其它实

施例可以使用附加或替换的操作集合来提高图像流的帧速率。

[0052] 在本实施例中，仅对来自第一图像流 Fs_1 的 R+1 个帧 402 应用仿射变换。这在图中通过仿射变换块 410-1、410-2、...、410-(R-1) 示出，其处理来自输入帧 402 的集合的相应的输入帧 N、N+1、...、N+R。直接将包含第二图像流 Fs_2 的两个帧 M 和 M+1 的输入帧 404 施加到聚簇块 412 和深度填充块 416。如附图中所出的，还将在仿射变换块 410-1、410-2、...、410-(R-1) 的输出处的仿射变换过的帧施加到聚簇块 412 和深度填充块 416。聚簇块 412 以先前描述的方式产生簇映射，并且簇对应块 414 确定来自第一和第二输入流的帧中的簇之间的对应性。所得到的簇对应信息（如上所述的，其可以包括附加的簇映射）被提供到深度填充块 416，深度填充块 416 使用该信息恢复第二图像流的 R-1 个附加帧。如后处理块 418-1、418-2、...、418-(R-1) 所示的，分别对这些附加帧中的每一个应用后处理。这些后处理块 418 在其相应的输出处提供 R-1 个输出帧 406。

[0053] 如图 3 的实施例中那样，对于多次迭代中的每一次迭代重复图 4 的实施例的处理操作，每一次这样的迭代确定用于插入在第二图像流 Fs_2 的给定对的连续现有帧之间以提高该图像流的帧速率的附加帧的集合。

[0054] 现将更加详细地描述由图像处理器 102 的仿射变换模块 110 执行的示例性仿射变换操作。

[0055] 设 $Im_i(N)$ 为在图像流 Fs_i 的第 N 帧中捕获的图像，其中 $i = 1,2$。可以确定仿射变换 T 以使得将 $T(Im_1(N))$ 置入 $Im_2(N)$ 的坐标系中。

[0056] 在其中图像传感器相对于彼此固定并且预先已知其并置 (collocation) 的实施例中，可以在传感器制造或设置时定义仿射变换 T 为 $T_1 = T(T_2)$，其中 T_i 是 3D 空间中第 i 图像传感器的向量基 (vector basis)，并且仿射变换 T 是放大矩阵，其中放大矩阵通常提供将一组向量映射到另一组向量的线性变换。因此，例如，可以使用给定的放大矩阵来将第一图像传感器的向量基变换到第二图像传感器的向量基，其中在单一 3D 空间中考虑所述向量基。

[0057] 根据使用的图像传感器的类型，期望与仿射变换 T 相关联的关键点在 2D 或 3D 空间中，并且可以手动地或通过自动技术（例如，使用边缘分析）选择关键点。对于许多实际应用，关键点的合适的数目将在大约 20 的量级上，然而在其它实施例中可以使用其它数目的关键点。

[0058] 如下面将描述的，可以通过通过使用最小二乘法求解超定线性方程组来确定仿射变换。设 m 表示关键点的数目，定义 D_{xyz} 为 2×m 或 3×m 矩阵，其包括逐列书写的来自 $Im_1(N)$ 的每 m 个点的坐标，并定义 T_{xyz} 为为 2×m 或 3×m 矩阵，其包括逐列书写的来自 $Im_2(N)$ 的对应的 m 个点。令 A 和 TR 分别表示在最小均方意义上优化的仿射变换矩阵和相关联的变换向量。对于 3D 情况：

[0059] $$\left\{ A \cdot \begin{pmatrix} x_1^1 & y_1^1 & z_1^1 \\ \cdots & & \\ x_m^1 & y_m^1 & z_m^1 \end{pmatrix}^T + \overline{TR} = \begin{pmatrix} x_1^2 & y_1^2 & z_1^2 \\ \cdots & & \\ x_m^2 & y_m^2 & z_m^2 \end{pmatrix}^T \right.,$$

$$D_{xyz}=\begin{pmatrix} x_1^1 & y_1^1 & z_1^1 \\ \cdots & & \\ x_m^1 & y_m^1 & z_m^1 \end{pmatrix}^T, T_{xyz}=\begin{pmatrix} x_1^2 & y_1^2 & z_1^2 \\ \cdots & & \\ x_m^2 & y_m^2 & z_m^2 \end{pmatrix}^T$$

[0060] 其中 (x_i^j, y_i^j, z_i^j) 是 $Im_j(N)$ 第 i 个关键点的坐标，并且矩阵 A 和向量 TR 共同定义仿射变换：

[0061] $T_{xyz} = A \cdot D_{xyz} + TR,$

[0062] 可以作为如下的优化问题的解而获得矩阵 A 和向量 TR：

[0063] $||A \cdot D_{xyz} + TR - T_{xyz}||^2 \rightarrow$ 最小值。

[0064] 使用元素方式的标记 $A = (a_{ij})$，其中 $(i, j) = (1,1) \cdots (3,3)$，并且 $TR = (tr_k)$ 其中 $k = 1..3$，在最小均方意义下的优化问题的解基于如下线性方程组，其包括总共 12 个变量和 12m 个方程式：

[0065] $dR/da_{ij} = 0, i = 1,2,3, j = 1,2,3,$

[0066] $dR/dtr_k = 0, k = 1,2,3。$

[0067] 以如上所述的方式确定了仿射变换参数 A 和 TR 之后，如下，将图像 $Im_1(N)$ 变换到 $Im_2(N)$ 的坐标系中：

[0068] $D_{1xyz} = A \cdot D_{xyz} + TR。$

[0069] 作为应用仿射变换的结果，在所得到的变换的图像 D_1 中的像素的坐标 (x,y) 不总是整数，而是，更一般地为有理数。可以使用例如最近毗邻或插值的技术，来将这些有理数坐标映射到 $Im_2(N)$ 的规则等距正交整数格子。该映射提供具有与 $Im_2(N)$ 相同分辨率的图像 $Im_1(N)$，尽管由于映射可能在标准格子中留下未填充的一些点，使得可能存在空像素位置（即，未定义的数据）。

[0070] 前述仅是可以通过图像处理器 102 实现的映射变换的一个例子，其它实施例可以使用其它类型的变换或技术来将用于产生第一图像流和第二图像流的图像传感器的坐标系对准。

[0071] 现在将更加详细地描述由图像处理器 102 的聚簇模块 112 执行的示例性聚簇操作。首先应注意，聚簇模块可以实现需要不同水平的计算资源的若干不同的聚簇技术，并且可以基于图像处理器 102 的当前计算负载在这些技术之间切换。

[0072] 如前所述的，适于在聚簇模块 112 中使用的聚簇技术可以基于统计区域合并或 SRM。该技术通常抗随机噪声，并具有适中的计算复杂度和良好的定量误差界 (quantitative error bounds)。另外，可以以允许动态控制计算需求的方式调节划分程度。

[0073] 在基于 SRM 的聚簇技术的更特定的例子中，通过与优化图像 $Id_i(j)$ 相关的独立分布随机变量表示实际图像 $Im_i(j)$ 的每一个像素，实际图像 $Im_i(j)$ 被认为是优化图像 $Id_i(j)$ 的特定观察。使用指定每一个统计区域内像素具有相同的期望并且相邻区域的期望不同的均质性规则 (homogeneity rules)，将实际图像 $Im_i(j)$ 和优化图像 $Id_i(j)$ 每一个分到优化统计区域中。

[0074] 该示例性的基于 SRM 的技术使用规定定的合并谓词 (predicate)P 实现递归合并。考虑任意图像 $Im_i(j)$。设由 Q 随机变量表示 $Im_i(j)$ 的每一个像素。那么，对于 $Im_i(j)$ 的任

意两个区域 R_1, R_2，合并谓词 P 可以如下表示：

[0075] $$P(R_1,R_2)=\begin{cases}\text{真}, & \text{如果}|R_1-R_2|\le\sqrt{b^2(R_1)+b^2(R_2)}\\ \text{假}, & \text{其它情况}\end{cases}$$，其中

$$b(R)=G\sqrt{\frac{1}{2Q|R|}\cdot\ln(|R|^{|R|}/\delta)}$$

[0076] 其中 |R| 表示区域 R 中像素的数目，G 表示当前图像的给定像素的最大可能值（例如，对于来自 Kinect 图像传感器的图像 $G=2^{12}$），而 δ 是小于 1 的正值。因此，$|R_1-R_2|$ 表示区域 R_1 中的像素数和区域 R_2 中的像素数之差的大小。如果 $P(R_1,R_2)$ =真，则该技术将区域 R_1 和 R_2 合并到单一的簇。

[0077] 该技术以像素级别开始，每一个像素初始地被认为是单个的区域。针对谓词 P 测试区域合并的顺序遵照不变式 A，其指示两个不同区域的两个部分之间的任何测试何时发生，这意味着在这两个区域内的所有测试先前已经发生。可以使用函数 $f(pix_1,pix_2)=|pix_1-pix_2|$ 获得不变式 A，其中 pix_i 是图像像素值。

[0078] 然后，以如下的方式进行基于 SRM 的技术。首先，以函数 $f(pix_1,pix_2)=|pix_1-pix_2|$ 的升序排序所有可能的像素对 (pix_1,pix_2)，并所得到的顺序仅遍历一次。对于任何 $R(pix_1)\neq R(pix_2)$ 的当前像素对 (pix_1,pix_2)（其中 R(pix) 表示 pix 所属的当前区域），执行测试 $P(R(pix_1),R(pix_2))$，并且，当且仅当测试返回真 (true) 时，合并 $R(pix_1)$ 和 $R(pix_2)$。在对当前图像的合并处理完成时，图像像素已经分到多个簇中，所述簇通过先前描述的类型的簇映射来表征。

[0079] 在该实施例中使用函数 $f(pix_1,pix_2)=|pix_1-pix_2|$ 作为不变式 A 的近似，然而也可以使用其它函数。另外，在上述基于 SRM 的技术中，可以改变合并谓词和其它参数。另外，可以使用不是基于 SRM 的多种聚簇技术。

[0080] 现在将参照图 5 至图 7 更详细地描述由图像处理器 102 的簇对应模块 114 执行的示例性的簇对应操作。图 5 示出了总体的簇对应处理过程 500，其包括分别在图 6 和图 7 的处理过程 600 和处理过程 700 中示出的被标示为 (1) 和 (2) 的部分。在步骤 502 中，有部分 (1) 的单一的实例，并且在相应的步骤 504、506、508 和 510 中有部分 (2) 的多个单独的实例。部分 (1) 通常涉及确定两个簇映射之间的对应性，并且部分 (2) 的给定的实例通常涉及将簇的一个集合映射到簇的另一个集合。然而应理解，在其它实施例中可以使用附加的或可替换的簇对应操作。

[0081] 首先参照图 5，簇对应处理 500 包括与部分 (1) 相关联的步骤 502 和每一个与部分 (2) 的例子相关联的步骤 504、506、508 和 510。

[0082] 在步骤 502 中，确定簇映射 $Cm_2(M)$ 和 $Cm_2(M+1)$ 之间的对应，并形成簇映射 $Cg_2(M)$ 和 $Cg_2(M+1)$。

[0083] 在步骤 504 中，将簇映射 $Cg_2(M)$ 映射到簇映射 $Cm_1(N)$，以获得簇映射 $Cg_1(N)$。

[0084] 在步骤 506 中，将簇映射 $Cg_2(M+1)$ 映射到簇映射 $Cm_1(N+R)$，以获得簇映射 $Cg_1(N+R)$。

[0085] 在步骤 508 中，将簇映射 $Cg_1(N)$ 映射到簇映射 $Cm_1(N+1)$，以获得簇映射 $Cg_1(N+1)$。

[0086] 在步骤 510 中，将簇映射 $Cg_1(N+R)$ 映射到簇映射 $Cm_1(N+R-1)$，以获得簇映射

$Cg_1(N+R-1)$。

[0087] 对一个或多个剩余帧继续进行由步骤 504 和 508 示出的映射操作序列。

[0088] 类似地，对一个或多个剩余帧继续进行由步骤 506 和 510 示出的映射操作序列。

[0089] 如所示出的，处理 500 在完成以上提到的映射操作序列时产生簇映射 $Cg_1(N)$, $Cg_1(N+1)$, …, $Cg_1(N+R)$, $Cg_2(M)$ 和 $Cg_2(M+1)$，以及这些簇映射中的每一个中的簇的数目 k。

[0090] 簇映射 Cg_1 和 Cg_2 也称为对准的簇映射，并且可以被认为是此处更一般地称作“簇对应信息”的概念的例子。如同在此公开的其它簇映射类似，Cg_1 和 Cg_2 簇映射可以被表示为相应的矩阵。在本发明的其它实施例中可以使用许多其它类型的簇对应信息。

[0091] 现在将参考图 6 描述用于执行图 5 的步骤 502 中的部分 (1) 的示例性处理 600。如上所述的，部分 (1) 通常涉及确定两个簇映射之间的对应性。用于实现该功能的处理 600 包括步骤 602 至 618。

[0092] 设 C_1 和 C_2 表示要使用将要描述的处理 600 相互映射的两个簇映射。在图 5 的步骤 502 的上下文中，$C_1 = Cm_2(M)$ 并且 $C_2 = Cm_2(M+1)$。设 N_1 表示 C_1 中簇的数目，并设 N_2 表示 C_2 中簇的数目。

[0093] 考虑来自 C_1 的任意簇 CL_1 和来自 C_2 的任意簇 CL_2。如果满足如下条件，则簇 CL_2 被称作与簇 CL_1 交叉：

[0094] $\rho(CL_1, CL_2) > threshold_1$。

[0095] 这里，$0 \leqslant \rho(CL_1, CL_2) \leqslant 1$ 表示两个像素集合的相对交叉量度。例如，可以使用如下对称和非对称交叉量度中给定的一个：

[0096] $$\rho_1(CL_1, CL_2) = \frac{|CL_1 \cap CL_2|}{|CL_1 \cup CL_2|} \text{ 以及 } \rho_2(CL_1, CL_2) = \frac{|CL_1 \cap CL_2|}{|CL_2|}$$

[0097] 阈值 $threshold_1$ 对应于预定的阈值（例如，0.1），其可以被控制作为处理过程的参数。

[0098] 处理 600 中的簇映射从空的簇映射 Cg_1 和 Cg_2 开始，其可以以零矩阵的形式表示。初始化三个附加变量，包括簇的数目 k 和分别表示来自 C_1 和 C_2 已经使用的簇的集合的变量 $Used_1$ 和 $Used_2$。

[0099] 在步骤 602 中，处理过程的全局初始化设置 $Used_1 = \{\}$, $Used_2 = \{\}$, $Cg_1 = 0$, $Cg_2 = 0$, 以及 $k = 0$。

[0100] 在步骤 604 中，基于每一个簇中像素的数目以大小减少的顺序排序来自 C_1 的簇。

[0101] 在步骤 606 中，确定是否来自 C_1 的所有簇已经用于分组到集合中。如果来自 C_1 的所有簇已经被使用，那么处理过程 600 对 Cg_1 和 Cg_2 中的簇进行后处理，如步骤 607 中所示，并然后如所示的，退出该处理过程。如果不是所有的来自 C_1 的簇已被使用，则处理过程 600 行进到步骤 608。

[0102] 在步骤 608 中，查找来自 C_1 的未使用的簇 CL_1。

[0103] 在步骤 610 中，通过将集合 g_1 初始化为 $\{CL_1\}$ 以及将对应的集合 g_2 初始化为空集合，执行集合搜索初始化。然后处理过程通过步骤 612、614、616 和 617 循环。

[0104] 在步骤 612 中，将 g_2 定义为与来自 g_1 的簇相交叉的来自 CL_2 的簇的集合，如下：

[0105] $\mathbf{g_2}= \{CL_2 \in C_2 \setminus Used_2 \mid \exists cl \in g_1 : \rho(cl, CL_2) > threshold_1\}$。

[0106] 在步骤614中，定义$\hat{g}_1$为与来自步骤612中定义的新的g_2的簇相交叉的来自CL_1的簇的集合，如下：

[0107] $\hat{g}_1 = \{CL_1 \in C_1 \setminus Used_1 \mid \exists cl \in g_2 : \rho(CL_1, cl) > threshold_1\}$。

[0108] 在步骤616中，确定g_1是否等于$\hat{g}_1$。如果g_1不等于$\hat{g}_1$，那么处理过程进行到步骤617，否则进行到步骤618。

[0109] 在步骤617中，设置g_1等于$\hat{g}_1$，并重复步骤612、614和616直到满足条件$\mathbf{g_1} = \hat{g}_1$，此时处理过程进行到步骤618。由于在C_1和C_2中簇的数目有限，因此在有限次的包括步骤612、614、616和617的迭代循环后，该条件将满足。

[0110] 在步骤618中，k增加1，并更新使用的簇的集合，使得$k = k+1$，$Used_1 = Used_1 \cup g_1$以及$Used_2 = Used_2 \cup g_2$。此外，通过将$Cg_1(g_1)$和$Cg_2(g_2)$都设置为等于k，将g_1和g_2添加到所得到的簇映射Cg_1和Cg_2，其中$Cg_1(g_1) = k$意指将用于与来自簇g_1的像素对应的簇映射Cg_1的矩阵的所有元素设置为k，并且类似地，$Cg_2(g_2) = k$意指将用于与来自簇g_2的像素对应的簇映射Cg_2的矩阵的所有元素设置为k。

[0111] 随后，如所示的，处理返回步骤606。如上所述的，如果在步骤606中确定来自C_1的所有簇已被使用，那么如步骤607所示，处理过程600对Cg_1和Cg_2中的簇进行后处理。例如，该后处理（其不同于图4中的后处理块418所执行的后处理）可以涉及识别Cg_1中的任何在Cg_2中具有对应的空簇的簇，以及识别Cg_2中的任何没有分配给Cg_1中任何簇的簇。在这两种情况中，将每一个识别的簇组合到与其共享最长边界的相邻的簇中，其中，两个簇之间的边界长度被定义为这些簇的边界像素的数目，并且其中给定的边界像素是这样的像素，其具有一个或多个来自两个簇的相邻像素。

[0112] 步骤607中的后处理可以涉及使用簇之间的深度距离量度的附加操作。例如，该深度距离量度可以被定义为两个簇的平均深度之间的绝对差值，或者两个簇的边界像素的平均深度之间的绝对距离。所述附加操作可以包括：如果Cg_1中相邻的簇之间的对应的深度距离量度小于预定的阈值threshold2，则合并Cg_1中所述相邻的簇。可以应用图5处理过程的部分(2)来确定Cg_2的对应的合并过程。

[0113] 可以使用替代的技术来替代处理过程600。例如，可以应用第二图像流的图像M和M+1两者中的所有簇集合的穷尽搜索。

[0114] 现在将参考图7描述用于执行图5的步骤504、506、508和510中的特定的一个中的部分(2)的给定的实例的示例性处理过程700。如上所述的，部分(2)的给定的实例通常涉及将簇的一个集合映射到簇的另一个集合，并且可以被认为是处理过程600的简化版本。例如，在该简化版本中，来自Cg的簇被保持不变，而仅将来自Cm的簇分组到集合中。这样的设置在Cm中的簇的数目大于Cg中的簇的数目时尤其有用。用于实现该功能性的处理700包括步骤702至710。

[0115] 在步骤702中，处理过程的全局初始化设置$Used_1 = \{\}$，$Used_2 = \{\}$，$Cg_1 = Cm_1$，$Cg_2 = 0$，以及$k = 0$。

[0116] 在步骤704中，确定是否来自C_1的所有簇已经被用于分组到集合中。如果已经使

用来自 C_1 的所有簇，则处理过程 700 对 Cg_2 中的簇进行后处理，如在步骤 705 中所示，并如所示的随后退出处理过程。如果不是来自 C_1 的所有簇都已被使用，则处理过程 700 进行到步骤 706。

[0117] 在步骤 706 中，查找来自 C_1 的未使用的簇 g_1。

[0118] 在步骤 708 中，如下，定义 g_2 为与来自 g_1 的簇相交叉的来自 CL_2 的簇的集合：

[0119] $$g_2 = \{CL_2 \in C_2 \setminus Used_2 \mid \exists cl \in g_1 : \rho(cl, CL_2) > threshold_1\}。$$

[0120] 在步骤 710 中，k 增加 1，并更新使用的簇的集合，使得 k = k+1，$Used_1 = Used_1 \cup g_1$，并且 $Used_2 = Used_2 \cup g_2$。另外，通过将 $Cg_2(g_2)$ 设置为等于 k，将 g_2 添加到所得到的簇映射 Cg_2，其中 $Cg_2(g_2) = k$ 意指将与来自簇 g_2 的像素对应的簇映射 Cg_2 的矩阵的所有元素设置为 k。

[0121] 随后，处理如所示的返回步骤 704。如上所述的，如果在步骤 704 中确定来自 C_1 所有簇已经被使用，则如步骤 705 中所示，处理过程 700 对 Cg_2 中的簇进行后处理。

[0122] 应理解，在图 3 至 7 的流程图中使用的特定的处理步骤仅是示例性的，其它实施例可以使用不同类型和设置的图像处理操作。例如，在其它实施例中可以改变具体的聚簇操作的类型。另外，在给定的流程图中被示为串行执行的步骤，在其它实施例中可以至少部分地与一个或多个其它步骤并行执行。

[0123] 现在将更详细的描述由图像处理器 102 的深度填充模块 116 执行的示例性深度填充操作。

[0124] 在簇对应模块 114 以如上所述的方式确定簇对应信息之后，该信息被传送到深度填充模块 116，深度填充模块 116 恢复用于插入在第二图像流 Fs_2 的现有帧 M 和 M+1 之间的附加帧。深度填充模块将来自这些现有帧的深度数据添加到附加帧中的特定簇。

[0125] 例如，利用簇对应信息，深度填充模块可以处理 $Cm_1(N+i)$，$I = 1, 2, \cdots, R-1$ 以确定簇的集合 $\{CL_1, \cdots, CL_P\}$ 和 $\{CL_1', \cdots, CL_T'\}$，其中 P 小于来自 $Im_2(M)$ 的 $Cm_2(M)$ 中的不同元素的最大数目，并且 T 小于来自 $Im_2(M+1)$ 的 $Cm_2(M+1)$ 中的不同元素的最大数目。

[0126] 如前所示的，可以在由深度填充模块 116 将深度数据填充到附加帧之前，对深度数据预处理。例如，可以以如下的方式放大深度数据：

[0127]

$$\mathbf{dData(i,j) = max\left(\max_{簇 \in CSET}\left[\max_{pix(i,j) \in 簇}(pix(i,j))\right], \max_{簇 \in CSET'}\left[\max_{pix(i,j) \in 簇}(pix(i,j))\right]\right)}$$

[0128] 其中 dData(i, j) 表示坐标为 (i, j) 的深度数据，并且 pix(i, j) 是坐标为 (i, j) 的图像像素的值。另外或替代地，可以如下对深度数据进行平滑 (smooth)：

[0129] dData = smooth(dData, smooth_str)

[0130] 其中该光滑技术用邻近像素的深度数据的加权和替换给定像素的深度数据 dData(i, j)。通过 smooth_str（例如，smooth_str = 5）给出邻近性模板大小。应理解，这些特定的预处理操作仅是示例性的，并且在其它实施例中可以使用深度填充之前的其它类型的预处理。

[0131] 在为来自 $Im_1(N+i)$，i = 1, 2, .. R-1 的给定的簇构建深度数据 dData 之后，深度填充模块 116 使用该深度数据来填充给定的恢复的图像中的对应的簇。由于在该例子中，给

定的簇是从高帧速率图像流 Fs_1 的图像提取的，因此可以提供比来自低帧速率图像流 Fs_2 的图像准确得多的相关联的成像对象的定位和边缘。

[0132] 恢复的图像可以被如下表示为来自与 $Im_1(N+i)$ 相关的簇映射的所有簇上的指示器 (indicator)Ind 的和：

[0133] $$\mathbf{iData(p,q)} = \sum_{CL \in Cm_1(N+i)} Ind(pix(p,q) \in CL) \cdot dData(p,q),$$

[0134] 其中给定的指示器 Ind 由下式给出：

[0135]

$$Ind(x \in A) = \begin{cases} 1, & \text{如果} x \in A \\ 0, & \text{其它情况} \end{cases}$$

[0136] 并且，其中 p,q 采取由 size($Im_2(M)$) 给出的所有值。在这种情况下，设想第二图像流 Fs_2 的恢复的图像具有与第一图像流 Fs_1 的对应的图像相同的大小。

[0137] 现在将更详细地描述由图像处理器 102 的后处理模块 118 执行的示例性后处理操作。这些后处理操作通常被配置用于增强恢复的图像的质量，并且可以涉及对恢复的图像应用一个或多个过滤器以提高锐度或消除不期望的冗余阴影。

[0138] 作为更特定的例子，可以通过应用如下的后处理操作消除冗余阴影和提高恢复的图像的总体质量。将给定的恢复的图像 $Im_2(N)$ 分到非交叉的簇中，使得每一阴影区域属于特定的簇。假定，如果来自 $Im_2(N)$ 的簇 CL 和其对应的来自 $Im_2(N+1)$ 的簇 CL′ 之间的相关性小于或等于预定阈值乘以 CL 面积 (area)，则来自 $Im_2(N)$ 的簇 CL 未显著改变。因此，假定如果：

[0139] $$\frac{sum(sum(C.*C'))}{sum(sum(C|C'))} > change\_thr$$

[0140] 则簇 CL 未改变，其中 change_thr 表示预定的实值 (例如，0.95)，并且 C 和 C′ 分别是与簇 CL 和 CL′ 相关的矩阵。显然，不随时间显著变化的簇满足该条件。

[0141] 如果来自 $Im_2(N)$ 的簇 CL 包括阴影区域，则保留该簇。然而，如果 $Im_2(N+1)$ 和 $Im_2(N+2)$ 中的对应的簇对于两个结果帧不变，则使用来自 CL 的深度数据减少 $Im_2(N+2)$ 中的阴影区域。更具体地，设 iData 为基于 $Im_2(N+2)$ 簇的恢复的图像，并且设其包含阴影区域，并且还设具有矩阵 C 的 CL 是来自 $Im_2(N)$ 的簇，从而 CL 并不显著地改变，并且对于时间 N+2 相关的簇包含阴影区域。后处理识别所有使得 pix(i,j) = Inf 的坐标 (i,j)，其中 pix(i,j) 是来自 iData 的像素，并且随后，对于 C(i,j)>0，所识别的 (i,j) 坐标的 CL 像素被分配与 Inf 不同的值 m，使得 pix(i,j) = m。这迫使 iData 包含更少的阴影区域。由于阴影区域对于不同的成像器可以有不同的值，因此用于 Inf 的具体值取决于用于产生对应图像的成像器的类型，并且通常表示可以用于指示作为阴影区域的一部分的给定像素的任何值。例如，其可以是大的负数。

[0142] 该示例性的后处理引起 $1/Fr_2$ 秒的附加处理延迟，或一帧的延迟，其并不是可察觉地那么显著，并因此能够适用于实际应用中。再次地，在其它实施例中可以使用各种其它类型的预处理操作。

[0143] 本发明的实施例提供了用于使用至少一个较高速率的图像流提高较低帧速率图

像流的帧速率的有效的技术。例如，所公开的技术允许提高图像流的帧速率，即使所成像的场景包括静态和并不必然跟随线性轨迹的移动物体的组合。然而，使用常规的技术例如移动插值处理这样的流将很困难。

[0144] 应当再次强调，本文所述的本发明的实施例仅仅是示例性的。例如，本发明的其它实施例可以利用与这里所描述的特定实施例中所使用的不同的各种不同类型和布置的图像处理电路、模块和处理操作来实现。此外，这里在描述某些实施例的上下文中所作的特定假设或设想并不是在其它实施例中必须施用的。在所附的权利要求范围内的这些及许多其它替代实施例对于本领域技术人员来说将是显而易见的。

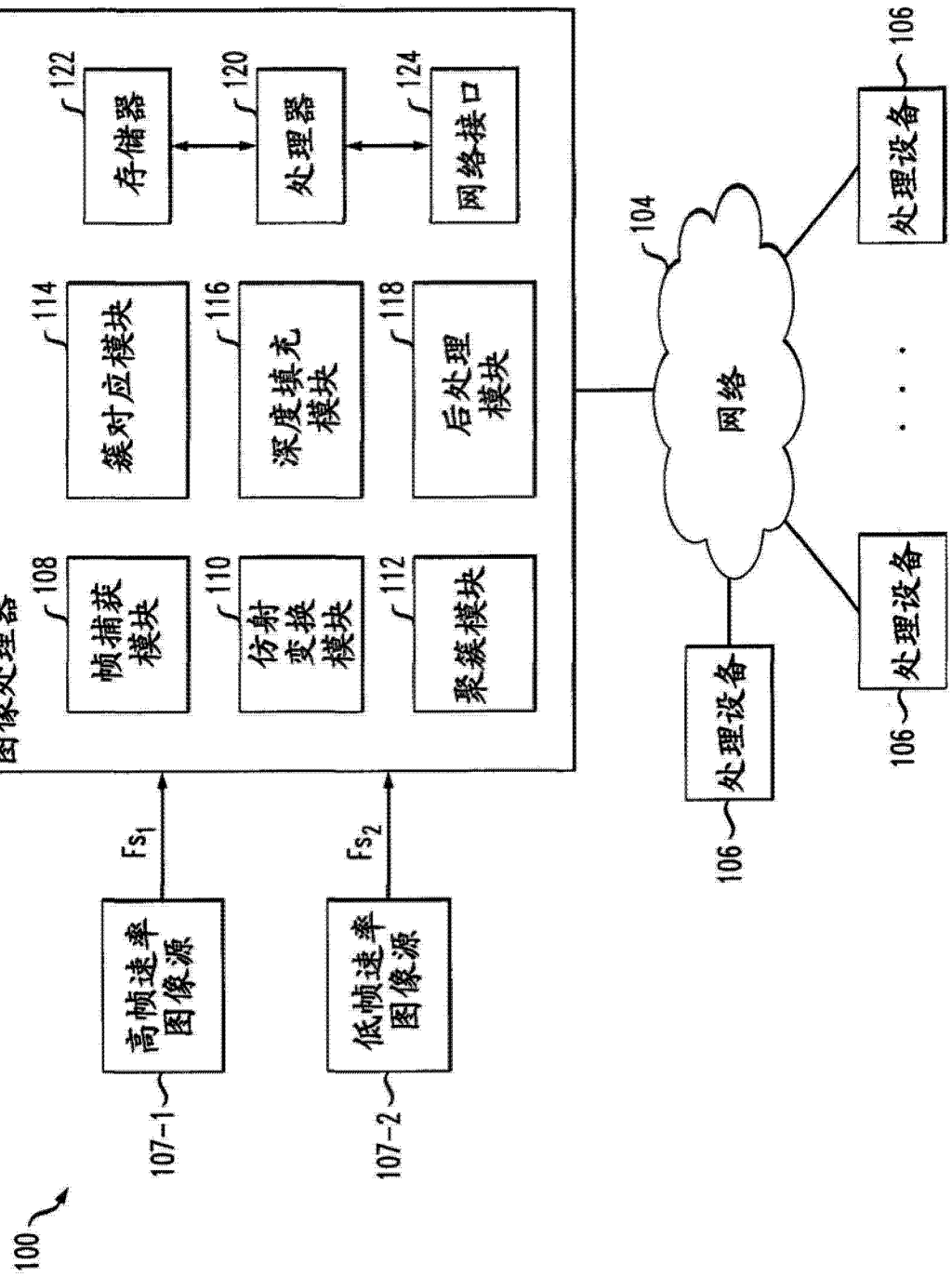

100
102
图像处理器
108
帧捕获模块
110
仿射变换模块
112
聚簇模块
114
簇对应模块
116
深度填充模块
118
后处理模块
120
处理器
122
存储器
124
网络接口
104
网络
106
处理设备
106
处理设备
106
处理设备
107-1
高帧速率图像源
107-2
低帧速率图像源
Fs_1
Fs_2

图 1

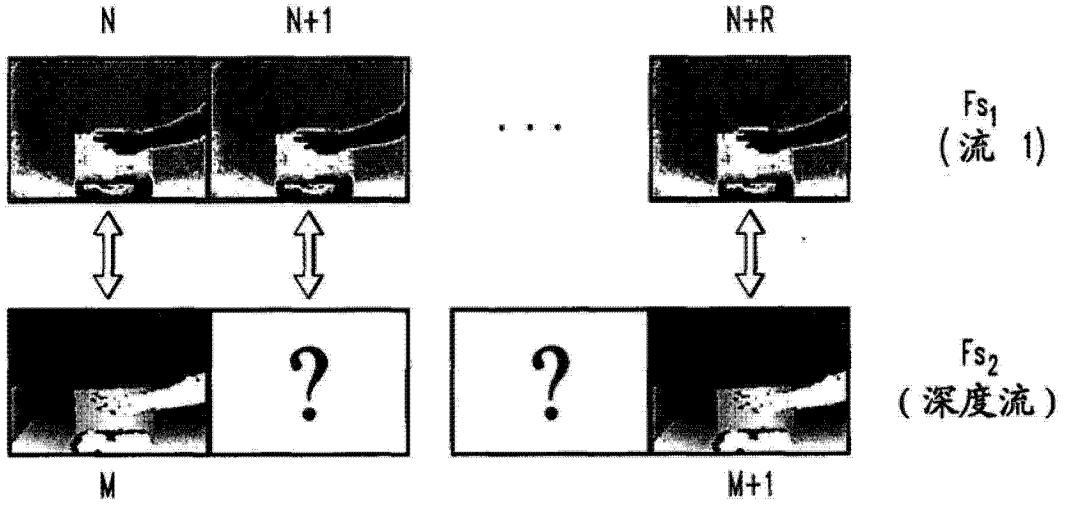

N
N+1
N+R
Fs1
(流 1)
?
?
Fs2
(深度流)
M
M+1

图 2

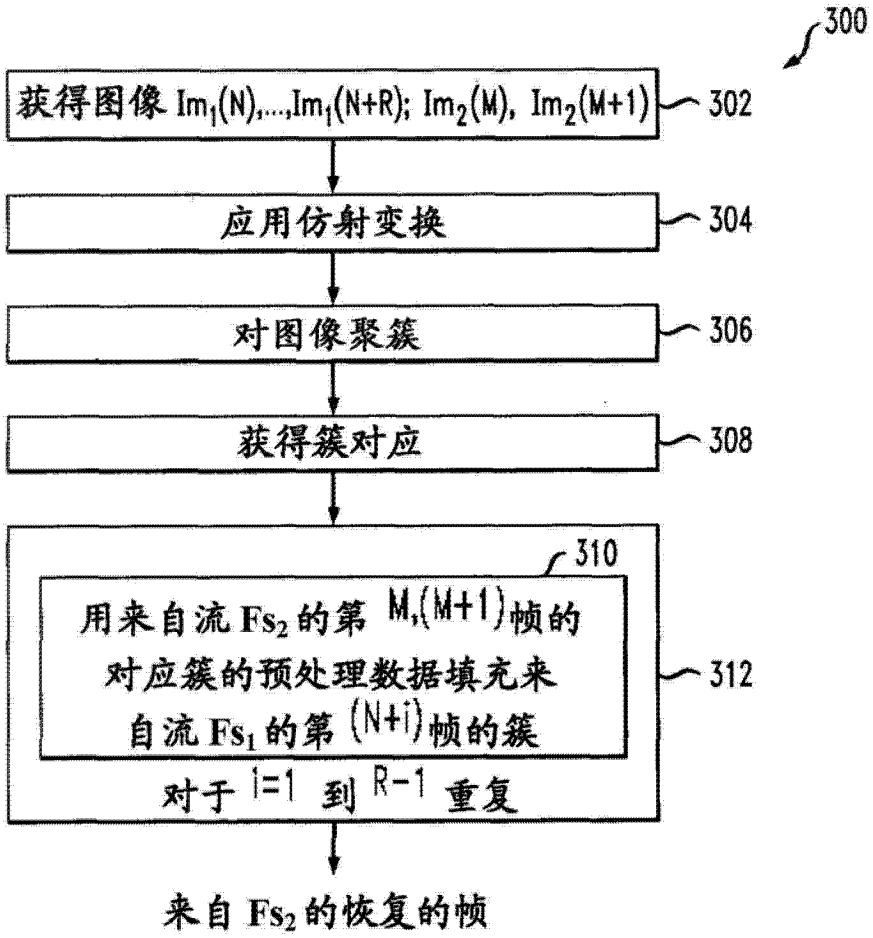

300
获得图像 Im1(N),...,Im1(N+R); Im2(M), Im2(M+1)
302
应用仿射变换
304
对图像聚簇
306
获得簇对应
308
310
用来自流 Fs2 的第 M,(M+1)帧的
对应簇的预处理数据填充来
自流 Fs1 的第 (N+i)帧的簇
312
对于 i=1 到 R-1 重复
来自 Fs2 的恢复的帧

图 3

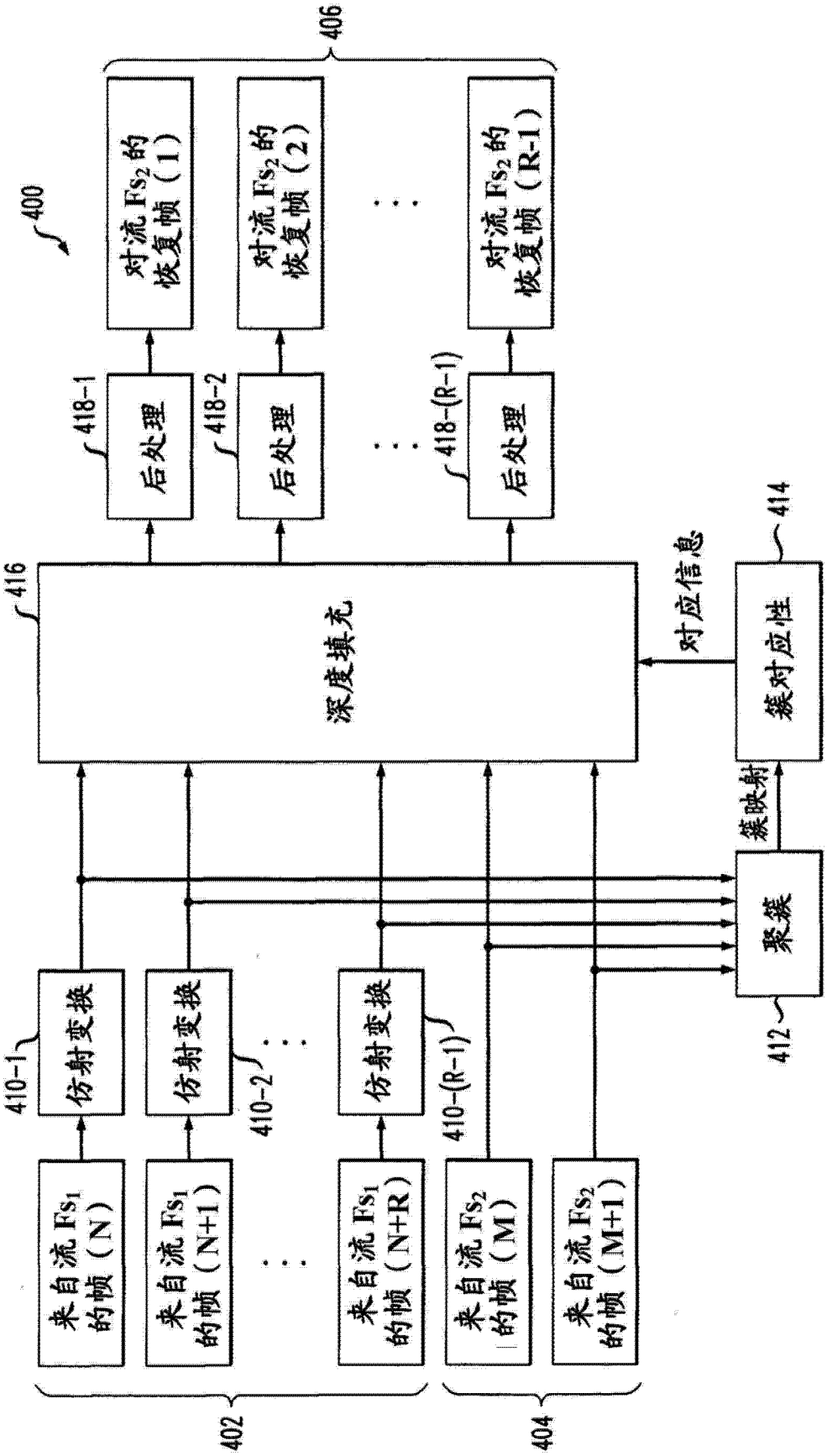

400
406
对流 Fs_2 的恢复帧（1）
对流 Fs_2 的恢复帧（2）
对流 Fs_2 的恢复帧（R-1）
418-1
418-2
418-(R-1)
后处理
后处理
后处理
416
深度填充
对应信息
414
簇对应性
簇映射
聚簇
412
410-1
410-2
410-(R-1)
仿射变换
仿射变换
仿射变换
来自流 Fs_1 的帧（N）
来自流 Fs_1 的帧（N+1）
来自流 Fs_1 的帧（N+R）
来自流 Fs_2 的帧（M）
来自流 Fs_2 的帧（M+1）
402
404

图 4

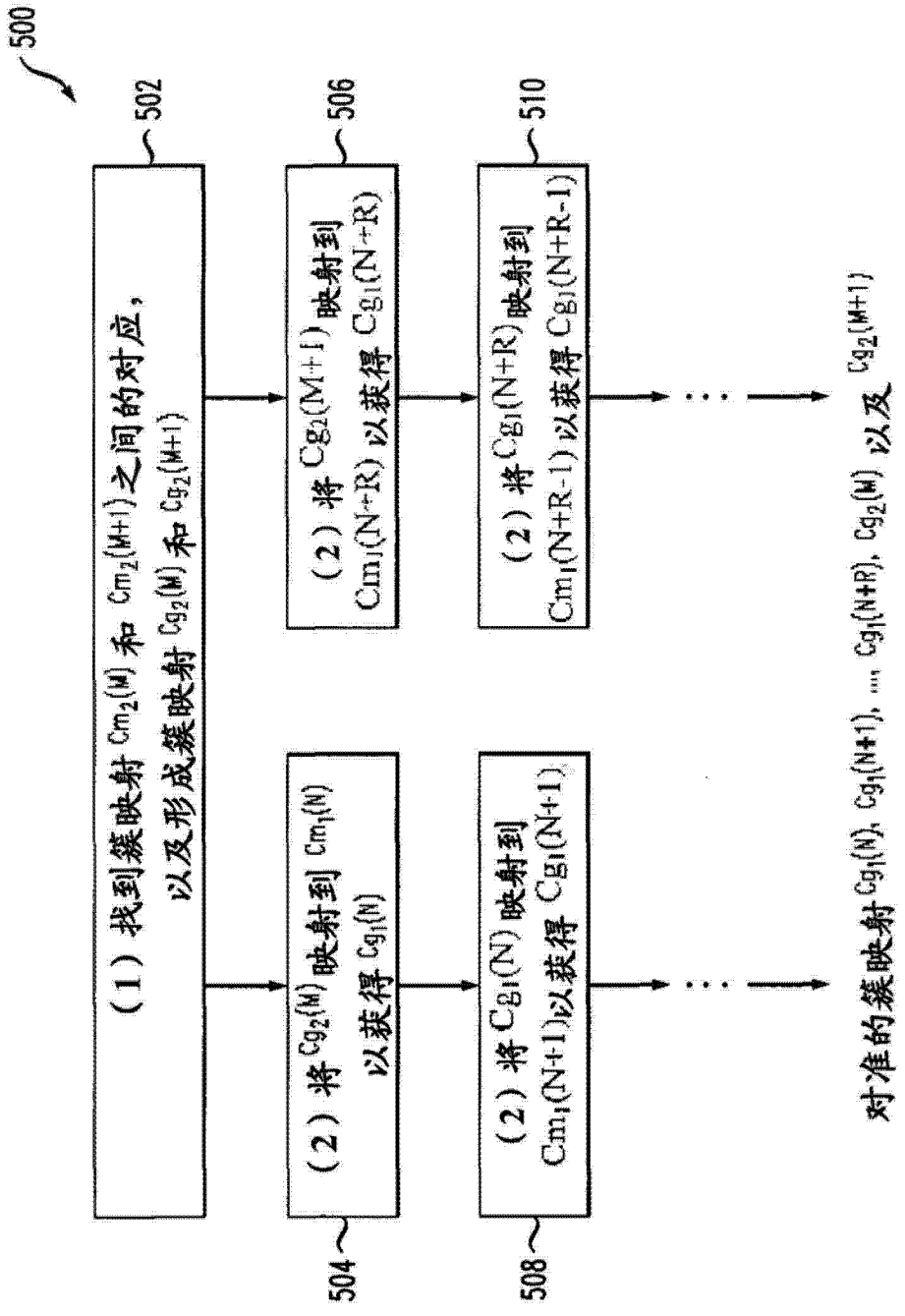

500
502
(1) 找到簇映射 $Cm_2(M)$ 和 $Cm_2(M+1)$ 之间的对应，
以及形成簇映射 $Cg_2(M)$ 和 $Cg_2(M+1)$
504
(2) 将 $Cg_2(M)$ 映射到 $Cm_1(N)$
以获得 $Cg_1(N)$
506
(2) 将 $Cg_2(M+1)$ 映射到
$Cm_1(N+R)$ 以获得 $Cg_1(N+R)$
508
(2) 将 $Cg_1(N)$ 映射到
$Cm_1(N+1)$ 以获得 $Cg_1(N+1)$
510
(2) 将 $Cg_1(N+R)$ 映射到
$Cm_1(N+R-1)$ 以获得 $Cg_1(N+R-1)$
对准的簇映射 $Cg_1(N)$, $Cg_1(N+1)$, ..., $Cg_1(N+R)$, $Cg_2(M)$ 以及 $Cg_2(M+1)$
在每一个映射中簇的数目 k

图 5

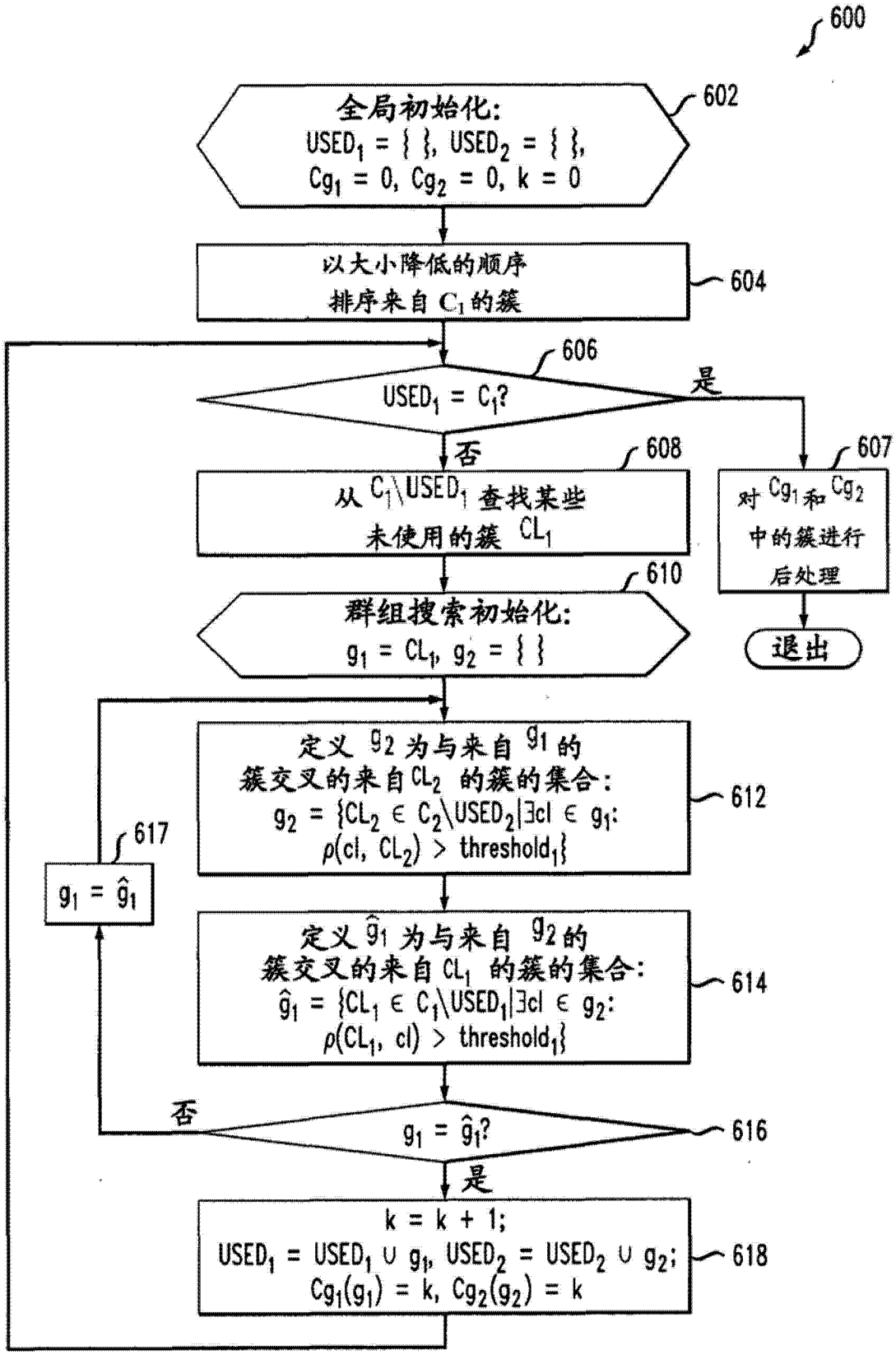

600
全局初始化:
USED1 = { }, USED2 = { },
Cg1 = 0, Cg2 = 0, k = 0
602
以大小降低的顺序
排序来自 C1 的簇
604
606
USED1 = C1?
是
否
607
对 Cg1 和 Cg2
中的簇进行
后处理
退出
608
从 C1\USED1 查找某些
未使用的簇 CL1
610
群组搜索初始化:
g1 = CL1, g2 = { }
定义 g2 为与来自 g1 的
簇交叉的来自 CL2 的簇的集合:
g2 = {CL2 ∈ C2\USED2|∃cl ∈ g1:
ρ(cl, CL2) > threshold1}
612
617
g1 = ĝ1
定义 ĝ1 为与来自 g2 的
簇交叉的来自 CL1 的簇的集合:
ĝ1 = {CL1 ∈ C1\USED1|∃cl ∈ g2:
ρ(CL1, cl) > threshold1}
614
否
g1 = ĝ1?
616
是
k = k + 1;
USED1 = USED1 ∪ g1, USED2 = USED2 ∪ g2;
Cg1(g1) = k, Cg2(g2) = k
618

图 6

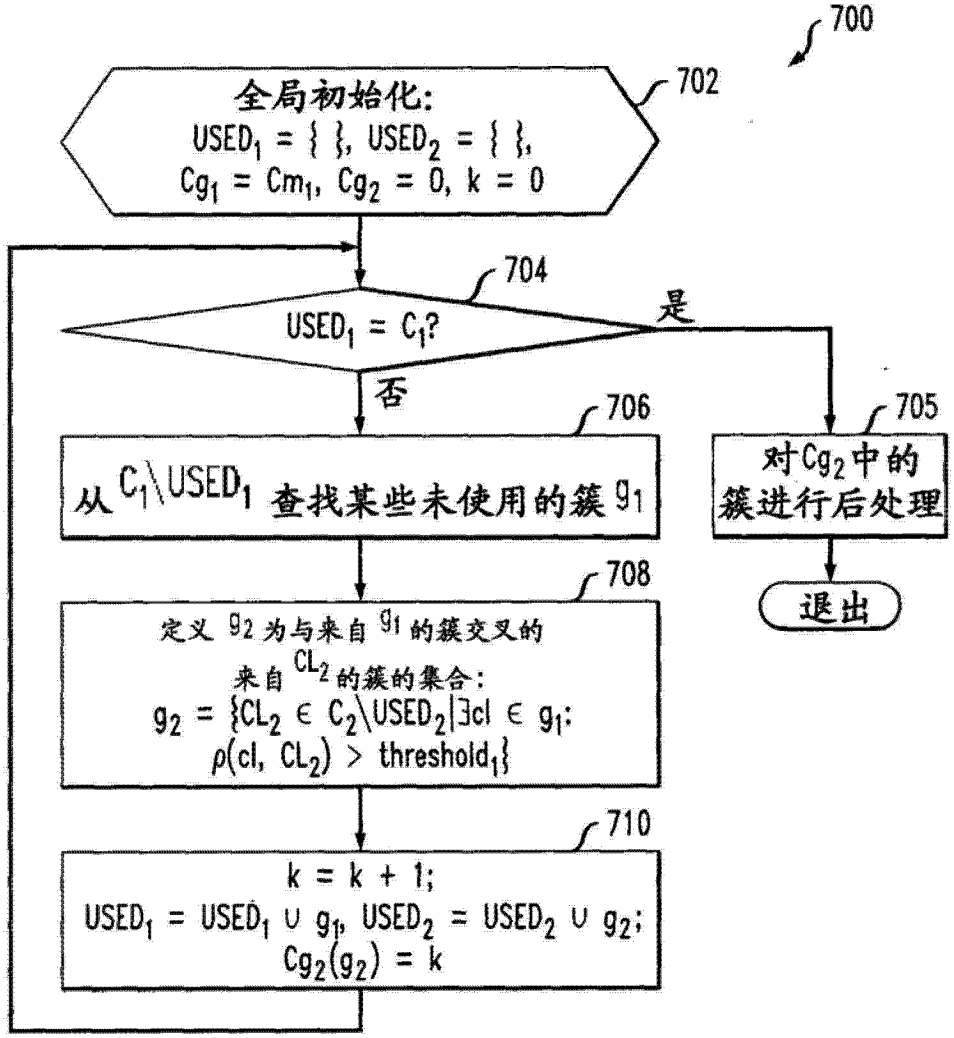

700
全局初始化:
USED1 = { }, USED2 = { },
Cg1 = Cm1, Cg2 = 0, k = 0
702
704
USED1 = C1?
是
否
705
对Cg2中的簇进行后处理
退出
706
从 C1\USED1 查找某些未使用的簇 g1
708
定义 g2 为与来自 g1 的簇交叉的来自 CL2 的簇的集合:
g2 = {CL2 ∈ C2\USED2|∃cl ∈ g1:
ρ(cl, CL2) > threshold1}
710
k = k + 1;
USED1 = USED1 ∪ g1, USED2 = USED2 ∪ g2;
Cg2(g2) = k

图 7