(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 111801703 A
(43)申请公布日 2020.10.20

(21)申请号 201980016252.4
(22)申请日 2019.02.14
(30)优先权数据
62/659,129 2018.04.17 US
(85)PCT国际申请进入国家阶段日
2020.08.31
(86)PCT国际申请的申请数据
PCT/US2019/018049 2019.02.14
(87)PCT国际申请的公布数据
WO2019/203920 EN 2019.10.24
(71)申请人 赫尔实验室有限公司
地址 美国加利福尼亚州
(72)发明人 A·F·加里多
J·科鲁兹-阿尔布雷克特
T·J·德罗西耶 S·劳
(74)专利代理机构 北京三友知识产权代理有限公司 11127
代理人 李艳芳 王小东
(51)Int.Cl.
G06T 1/20(2006.01)
G06T 1/60(2006.01)
G06F 15/76(2006.01)

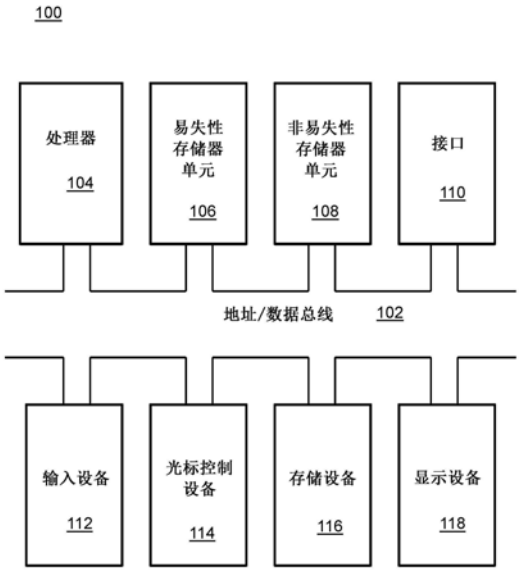
权利要求书2页 说明书13页 附图20页

(54)发明名称

用于图像处理管线的边界框生成的硬件和系统

(57)摘要

描述了一种用于边界框生成的系统。该系统对由均具有一位值的多个像素构成的图像进行操作。边界框是绕图像中的连通分量生成的，连通分量具有像素坐标和像素计数信息。基于像素坐标和像素计数信息，针对各个边界框生成排序分数。基于像素坐标和像素计数信息过滤边界框，以移除超过预定大小和预定像素计数的边界框。还过滤边界框，以移除低于预定排序分数的边界框，从而得到剩余边界框。最后，可以基于剩余边界框控制或以其它方式操作装置。



CN 111801703 A

1.一种用于边界框生成的系统,所述系统包括:

存储器以及一个或更多个处理器,所述存储器具有可执行指令,以使在执行所述指令时,所述一个或更多个处理器执行以下操作:

接收图像,所述图像由每像素具有一位值的像素构成;

围绕所述图像中的连通分量生成边界框,所述连通分量具有像素坐标和像素计数信息;

基于所述像素坐标和所述像素计数信息,针对各个边界框生成排序分数;

基于所述像素坐标和所述像素计数信息,过滤所述边界框,以移除超过预定大小和预定像素计数的边界框;以及

过滤所述边界框,以移除低于预定排序分数的边界框,从而得到剩余边界框;以及

基于所述剩余边界框控制设备。

2.根据权利要求1所述的系统,其中,所述处理器是现场可编程门阵列(FPGA)。

3.根据权利要求1所述的系统,其中,生成所述边界框还包括以下操作:

对所述图像中的连续像素进行分组;以及

将连通像素合并为连通分量,所述边界框由包围所述连通分量的框形成。

4.根据权利要求1所述的系统,其中,控制所述设备包括:使视频平台移动,以将所述剩余边界框中的至少一个剩余边界框保持在所述视频平台的视场内。

5.一种用于边界框生成的计算机程序产品,所述计算机程序产品包括:

非暂时性计算机可读介质,所述非暂时性计算机可读介质上编码有可执行指令,以使在由一个或更多个处理器执行所述指令时,所述一个或更多个处理器执行以下操作:

接收图像,所述图像由每像素具有一位值的像素构成;

围绕所述图像中的连通分量生成边界框,所述连通分量具有像素坐标和像素计数信息;

基于所述像素坐标和所述像素计数信息,针对各个边界框生成排序分数;

基于所述像素坐标和所述像素计数信息,过滤所述边界框,以移除超过预定大小和预定像素计数的边界框;以及

过滤所述边界框,以移除低于预定排序分数的边界框,从而得到剩余边界框;以及

基于所述剩余边界框控制设备。

6.根据权利要求5所述的计算机程序产品,其中,所述处理器是现场可编程门阵列(FPGA)。

7.根据权利要求5所述的计算机程序产品,其中,生成所述边界框还包括以下操作:

对所述图像中的连续像素进行分组;以及

将连通像素合并为连通分量,所述边界框由包围所述连通分量的框形成。

8.根据权利要求5所述的计算机程序产品,其中,控制所述设备包括:使视频平台移动,以将所述剩余边界框中的至少一个剩余边界框保持在所述视频平台的视场内。

9.一种用于边界框生成的计算机实现的方法,所述方法包括以下动作:

使一个或更多个处理器执行编码在非暂时性计算机可读介质上的指令,以使在执行所述指令时,所述一个或更多个处理器执行以下操作:

接收图像,所述图像由每像素具有一位值的像素构成;

围绕所述图像中的连通分量生成边界框,所述连通分量具有像素坐标和像素计数信息;

基于所述像素坐标和所述像素计数信息,针对各个边界框生成排序分数;

基于所述像素坐标和所述像素计数信息,过滤所述边界框,以移除超过预定大小和预定像素计数的边界框;以及

过滤所述边界框,以移除低于预定排序分数的边界框,从而得到剩余边界框;以及

基于所述剩余边界框控制设备。

10.根据权利要求9所述的方法,其中,所述处理器是现场可编程门阵列(FPGA)。

11.根据权利要求9所述的方法,其中,生成所述边界框还包括以下操作:

对所述图像中的连续像素进行分组;以及

将连通像素合并为连通分量,所述边界框由包围所述连通分量的框形成。

12.根据权利要求9所述的方法,其中,控制所述设备包括:使视频平台移动,以将所述剩余边界框中的至少一个剩余边界框保持在所述视频平台的视场内。

用于图像处理管线的边界框生成的硬件和系统

[0001] 政府权益

[0002] 本发明是在编号为HR0011-13-C-0052的题为“Revolutionary Analog Probabilistic Inference Devices for Unconventional Processing of Signals for Data Exploitation(用于数据开发的非常规信号处理的革命性模拟概率推断设备)”(RAPID-UPSIDE)的美国政府合同的政府支持下完成的。政府拥有本发明的特定权利。

[0003] 相关申请的交叉引用

[0004] 本申请是2016年9月21日提交的美国申请No.15/272,247的部分继续申请,该美国申请No.15/272,247是2015年9月21日提交的美国临时申请No.62/221,550的非临时申请,上述申请的全部内容通过引用并入于此。

[0005] 美国申请No.15/272,247是2016年3月24日提交的美国申请No.15/079,899的部分继续申请,该美国申请No.15/079,899是2015年3月24日提交的美国临时申请No.62/137,665的非临时申请,上述申请的全部内容通过引用并入于此。美国申请No.15/079,899也是2015年4月30日提交的美国临时申请No.62/155,355的非临时申请,上述申请的全部内容通过引用并入于此。

[0006] 美国申请No.15/272,247也是2016年2月12日提交的美国申请No.15/043,478的部分继续申请,上述申请的全部内容通过引用并入于此。

[0007] 美国申请No.15/272,247也是2016年7月6日提交的美国申请No.15/203,596的部分继续申请,该美国申请No.15/203,596是2015年9月21日提交的美国临时申请No.62/221,550的非临时申请。

[0008] 本申请还要求2018年4月17日提交的非临时专利申请US.62/659,129的权益,上述申请的全部内容通过引用并入于此。

[0009] 发明背景

(1)技术领域

[0010] 本发明涉及一种图像处理系统,并且更具体地,涉及一种用于在图像中生成边界框以进行图像处理的系统。

[0011] (2)相关技术的描述

[0012] 图像处理用于多种实现(包括跟踪和监测应用)。在跟踪或监测中,边界框用于识别对象,并且在理想情况下,用于跨图像帧和场景跟踪该对象。可以通过为连通分量设置框来形成边界框。例如,Walczyk等人的工作结果描述了执行二进制图像的连通分量标记(参见Robert Walczyk、Alistair Armitage和TD Binnie的“Comparative Study on Connected Components Labeling Algorithms for Embedded Video Processing Systems(用于嵌入式视频处理系统的连通分量标记算法的比较研究)”,2010年图像处理、计算机视觉和模式识别(IPCV)国际大会的会议记录(CSREA,2010),其全部内容通过引用并入于此)。尽管Walczyk等人公开了执行连通分量标记,但是该公开仅针对图像的标记,而未能进一步有效地处理框或图像。

[0013] 因此，仍然需要生成边界框，同时有效地计算边界框坐标和边界框对象像素计数，以便于随后对对象框进行排序和过滤，以进行图像处理。

发明内容

[0014] 本公开提供了一种用于边界框生成的系统。在各个方面中，所述系统包括存储器以及一个或更多个处理器。所述存储器具有可执行指令，以使在执行所述指令时，所述一个或更多个处理器执行多个操作，诸如接收图像，所述图像由每像素具有一位值的多个像素构成；围绕所述图像中的连通分量生成边界框，所述连通分量具有像素坐标和像素计数信息；基于所述像素坐标和所述像素计数信息，针对各个边界框生成排序分数；基于所述像素坐标和所述像素计数信息，过滤所述边界框，以移除超过预定大小和预定像素计数的边界框；以及过滤所述边界框，以移除低于预定排序分数的边界框，从而得到剩余边界框；并且基于所述剩余边界框控制设备。

[0015] 在另一方面中，所述处理器是现场可编程门阵列(FPGA)。

[0016] 在又一方面中，生成所述边界框还包括以下操作：对所述图像中的连续像素进行分组；以及将连通像素合并为连通分量，其中所述边界框由包围所述连通分量的框形成。

[0017] 另外，控制所述设备包括使视频平台移动，以将所述剩余边界框中的至少一个剩余边界框保持在所述视频平台的视场内。

[0018] 最后，本发明还包括计算机程序产品和计算机实现的方法。所述计算机程序产品包括在非暂时性计算机可读介质上存储的计算机可读指令，所述计算机可读指令能够由具有一个或更多个处理器的计算机执行，使得在执行所述指令时，所述一个或更多个处理器执行本文列出的操作。另选地，计算机实现的方法包括使计算机执行这种指令并且执行所得到的操作的动作。

附图说明

[0019] 结合参考以下附图，本发明的目的、特征以及优点将从本发明的各个方面的以下详细描述变得显而易见，其中：

[0020] 图1是示出了根据本发明的各个实施方式的系统的组件的框图；

[0021] 图2是具体实现本发明的一个方面的计算机程序产品的示图；

[0022] 图3是例示了根据本发明的各个实施方式的在准备期间的变量与数组之间的关系的流程图；

[0023] 图4是根据本发明的各个实施方式的用于找到像素标记值的搜索块的示图；

[0024] 图5是例示了根据本发明的各个实施方式的搜索/标记处理的流程图；

[0025] 图6A是示出了根据本发明的各个实施方式的部分图像和对应标记的示图；

[0026] 图6B是示出了根据本发明的各个实施方式的部分图像和对应标记的示图；

[0027] 图6C是根据本发明的各个实施方式的如在图6A和图6B中部分示出的完整图像和对应标记的示图；

[0028] 图7是例示了根据本发明的各个实施方式的合并区域的流程图；

[0029] 图8是例示了根据本发明的各个实施方式的状态转变的流程图；

[0030] 图9A是例示了根据本发明的各个实施方式的状态1的流程图；

[0031] 图9B是根据本发明的各个实施方式的状态2代码的示例；

[0032] 图10是例示了根据本发明的各个实施方式的增量器的流程图；

[0033] 图11是例示了根据本发明的各个实施方式的状态2的流程图；

[0034] 图12是根据本发明的各个实施方式的当前标记模块的示图；

[0035] 图13是例示了根据本发明的各个实施方式的状态3的流程图；

[0036] 图14是例示了根据本发明的各个实施方式的状态4、状态5和状态6的流程图；

[0037] 图15是例示了根据本发明的各个实施方式的状态7和回调操作的流程图；

[0038] 图16是例示了根据本发明的各个实施方式的状态7和排序操作的流程图；

[0039] 图17是例示了根据本发明的各个实施方式的状态7和排序操作以及排序模块的流程图；

[0040] 图18是示出了根据本发明的各个实施方式的示例输入图像的示图，其中各个像素位置具有一位值；

[0041] 图19是示出了具有经过根据本发明的各个实施方式的边界框处理和过滤之后所得到的边界框的图像的示图；以及

[0042] 图20是示出了根据各个实施方式的设备的控制的框图。

具体实施方式

[0043] 本发明涉及一种图像处理系统，并且更具体地，涉及一种用于在图像中生成边界框以进行图像处理的系统。呈现以下描述以使本领域普通技术人员能够作出和使用本发明并且将其结合到特定应用的上下文中。多种修改以及不同应用中的多种用途对于本领域技术人员来说将是显而易见的，并且本文限定的总体原理可以应用于广泛多个方面。因此，本发明不旨在限于所呈现的各个方面，而是涵盖与本文所公开的原理和新颖特征相一致的最广范围。

[0044] 在下面的详细说明中，阐述了许多具体细节，以使得能够更加彻底地理解本发明。然而，本领域技术人员将明白，本发明可以在不限于这些具体细节的情况下实施。在其它情况下，公知结构和设备按框图形式示出而不被详细示出，以免模糊本发明。

[0045] 读者应留意与本说明书同时提交的所有文件和文档，这些文件和文档与本说明书一起公开以供公众查阅，所有这些文件和文档的内容通过引用并入于此。本说明书(包括任何所附权利要求、摘要以及附图)中公开的所有特征可以由用于相同、等同或相似目的的替代特征来代替，除非另有明确说明。因此，除非另有明确说明，否则所公开的各个特征仅是典型系列的等同或相似特征的一个示例。

[0046] 此外，权利要求中的未明确陈述用于执行指定功能的“装置”或用于执行特定功能的“步骤”的任何要素不被解释为在35U.S.C.第112节第6款中指定的“装置”或“步骤”条款。具体地，在本文的权利要求中使用“…的步骤”或“…的动作”不旨在援引35U.S.C.第112节第6款的规定。

[0047] 在详细描述本发明之前，首先提供了对本发明的各个主要方面的说明。随后，介绍部分为读者提供了本发明的总体理解。最后，提供本发明的各个实施方式的具体细节，以给出具体方面的理解。

[0048] (1)主要方面

[0049] 本发明的各个实施方式包括三个“主要”方面。第一方面是一种用于图像处理的系统。该系统通常采用计算机系统操作软件的形式或采用“硬编码”指令集的形式或作为现场可编程门阵列(FPGA)。该系统可以结合到提供不同功能的广泛多种设备中。第二主要方面是使用数据处理系统(计算机)运行的通常采用软件形式的方法。第三主要方面是计算机程序产品。所述计算机程序产品通常表示存储在诸如光学存储设备(例如,光盘(CD)或数字通用盘(DVD))或磁存储设备(诸如,软盘或磁带)的非暂时性计算机可读介质上的计算机可读指令。计算机可读介质的其它非限制性示例包括硬盘、只读存储器(ROM)以及闪存型存储器。这些方面将在下文进行更详细说明。

[0050] 图1提供了示出本发明的系统(即,计算机系统100)的示例的框图。计算机系统100被配置成执行与程序或算法相关联的计算、处理、操作和/或功能。在一个方面中,本文讨论的某些处理和步骤被实现为存在于计算机可读存储器单元内并由计算机系统100的一个或更多个处理器执行的一系列指令(例如,软件程序)。在执行时,这些指令使计算机系统100执行特定动作并呈现特定行为,诸如本文所描述的。

[0051] 计算机系统100可以包括被配置成传送信息的地址/数据总线102。另外,一个或更多个数据处理单元(诸如处理器104(或多个处理器))与地址/数据总线102联接。处理器104被配置成处理信息和指令。在一个方面中,处理器104是微处理器。另选地,处理器104可以是不同类型的处理器,诸如并行处理器、专用集成电路(ASIC)、可编程逻辑阵列(PLA)、复杂可编程逻辑器件(CPLD)或现场可编程门阵列(FPGA),其被配置成执行本文描述的操作。

[0052] 计算机系统100被配置成利用一个或更多个数据存储单元。计算机系统100可以包括与地址/数据总线102联接的易失性存储器单元106(例如,随机存取存储器(“RAM”)、静态RAM、动态RAM等),其中,易失性存储器单元106被配置成存储用于处理器104的信息和指令。计算机系统100还可以包括与地址/数据总线102联接的非易失性存储器单元108(例如,只读存储器(“ROM”)、可编程ROM(“PROM”)、可擦除可编程ROM(“EPROM”)、电可擦除可编程ROM(“EEPROM”)、闪存存储器等),其中,非易失性存储器单元108被配置成存储用于处理器104的静态信息和指令。另选地,计算机系统100可以执行诸如在“云”计算中从在线数据存储单元取回的指令。在一个方面中,计算机系统100还可以包括与地址/数据总线102联接的一个或更多个接口(诸如,接口110)。所述一个或更多个接口被配置成使得计算机系统100能够与其它电子设备和计算机系统对接。由所述一个或更多个接口实现的通信接口可以包括有线通信技术(例如,串行电缆、调制解调器、网络适配器等)和/或无线通信技术(例如,无线调制解调器、无线网络适配器等)。

[0053] 在一个方面中,计算机系统100可以包括与地址/数据总线102联接的输入设备112,其中,输入设备112被配置成将信息和命令选择传送至处理器100。根据一个方面,输入设备112是可以包括字母数字键和/或功能键的字母数字输入设备(诸如键盘)。另选地,输入设备112可以是除字母数字输入设备之外的输入设备。在一个方面中,计算机系统100可以包括与地址/数据总线102联接的光标控制设备114,其中,光标控制设备114被配置成将用户输入信息和/或命令选择传送至处理器100。在一个方面中,光标控制设备114使用诸如鼠标、轨迹球、触控板、光学跟踪设备或触摸屏的设备来实现。尽管如此,但在一个方面中,诸如响应于使用与输入设备112相关联的特殊键和键序列命令,光标控制设备114经由来自输入设备112的输入被引导和/或激活。在另选方面中,光标控制设备114被配置成由语音命

令来引导或指导。

[0054] 在一个方面中，计算机系统100还可以包括与地址/数据总线102联接的一个或更多个可选计算机可用数据存储设备(诸如存储设备116)。存储设备116被配置成存储信息和/或计算机可执行指令。在一个方面中，存储设备116是诸如磁或光盘驱动器(例如，硬盘驱动器(“HDD”)、软盘、光盘只读存储器(“CD-ROM”)、数字通用盘(“DVD”))的存储设备。依据一个方面，显示设备118与地址/数据总线102联接，其中，显示设备118被配置成显示视频和/或图形。在一个方面中，显示设备118可以包括阴极射线管(“CRT”)、液晶显示器(“LCD”)、场发射显示器(“FED”)、等离子体显示器或适于显示视频和/或图形图像以及用户可识别的字母数字字符的任何其它显示设备。

[0055] 本文所提出的计算机系统100是根据一个方面的示例计算环境。然而，计算机系统100的非限制性示例并不严格限于是计算机系统。例如，一个方面规定了计算机系统100表示可以根据本文所述各个方面使用的一种数据处理分析。此外，还可以实现其它计算系统。实际上，本技术的精神和范围不限于任何单个数据处理环境。因此，在一个方面中，使用通过计算机执行的计算机可执行指令(诸如，程序模块)来控制或实现本技术的各个方面的一个或更多个操作。在一个实现中，这样的程序模块包括被配置成执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件和/或数据结构。另外，一个方面规定了通过利用一个或更多个分布式计算环境来实现本技术的一个或更多个方面，诸如，在分布式计算环境中，由通过通信网络链接的远程处理设备执行任务，或者诸如，在分布式计算环境中，各种程序模块位于包括存储器-存储设备的本地和远程计算机存储介质中。

[0056] 图2示出了具体实现本发明的计算机程序产品(即，存储设备)的示图。计算机程序产品被示出为软盘200或诸如CD或DVD的光盘202。然而，如先前提到的，计算机程序产品通常表示存储在任何兼容的非暂时性计算机可读介质上的计算机可读指令。关于本发明所使用的术语“指令”通常指示要在计算机上执行的一组操作，并且可以表示整个程序的片段或者单个可分离的软件模块。“指令”的非限制性示例包括计算机程序代码(源代码或目标代码)和“硬编码”电子器件(即，编码到计算机芯片中的计算机操作)。“指令”被存储在任何非暂时性计算机可读介质上，诸如存储在计算机的存储器中或软盘、CD-ROM以及闪存驱动器上。在任一种情况下，这些指令被编码在非暂时性计算机可读介质上。

[0057] (2)介绍

[0058] 本公开提供了用于图像处理管线的边界框生成的系统和对应硬件实现。在各个方面中，该系统在现场可编程门阵列(FPGA)上实现，该FPGA接收二进制图像，从该二进制图像检测对象像素并且围绕连通分量生成边界框。该系统实现了一种连通分量标记方法，该连通分量标记方法用于对贯穿图像找到的连续像素进行分组，然后合并连通像素以创建唯一加框位置(boxed location)。这些唯一框被保存为单独单元，该单独单元包含边界框坐标和所包含的对象像素的计数。该系统还基于边界框的高度和宽度以及所包含的对象像素的数量计算排序分数，该排序分数用于基于大小和纵横比进行对象的后续过滤。该处理旨在提供该边界框信息，同时最小化FPGA资源并实现足够的吞吐量，以跟上所需的输入图像帧速率(例如，每秒30帧)。

[0059] 在执行二进制图像的连通分量标记时，系统还同时记录边界框坐标和针对各个边界框检测到的对象像素的数量。该附加信息用于基于大小和纵横比对对象进行后续排序和

过滤，并且无需大量额外计算时间和硬件资源即可收集该附加信息。另外，本发明的设计被优化，以同时最小化FPGA利用和计算时间。所述优点允许本发明将被用作以高图像帧速率(例如，>每秒30帧)运行的小尺寸、轻重量和低功率(SWAP)图像处理管线(诸如在美国申请No.15/272,247中所描述的)的一部分。通过减少整个图像上的对检测到的特定对象的必要计算，本公开的使用处理将更好地简化图像处理管线。

[0060] 可以将本文所述的系统和处理实现为低SWAP图像处理管线的关键组成部分。此外，所述系统和处理可以应用于多种实现(包括具有严重受限的SWAP的无人驾驶自主交通工具和平台)。通过对传感器附近的硬件上的任务相关目标和障碍物进行快速检测，本发明能够改善任务响应性并减少必须在受约束的通信带宽上传输的原始传感器数据量。此外，所述系统和处理可以用于主动安全性和自主驾驶应用两者。通过在相机附近的低功耗、低成本硬件中执行目标检测，汽车可以更快速且更鲁棒地检测道路上的障碍物，从而向驾驶员提供更及时的警告或在自主交通工具中提供对障碍物的更迅速的自动响应。下面提供了另外的详细信息。

[0061] (3)各个实施方式的具体细节

[0062] (3.1)边界框介绍

[0063] 边界框是如下方法：系统通过该方法接受单个位数据的矩阵作为输入图像，系统将该矩阵用作创建框的数组的基础。各个“框”将包含两个x位置、两个y位置的坐标以及有效像素计数。作为一个示例，来自边界框数组的一组框数据将包含x位置(Xmin＝100，Xmax＝150)和y位置(Ymin＝80，Ymax＝90)，其中像素计数＝70。这将是包含70个有效像素的10个像素高且50个像素宽的“框”。将其它像素集放入框中，并指派其各自的x位置、y位置和有效像素计数。将各个框分开的是有效像素的邻近和分离。下面分别关于软件实现和硬件实现进一步描述该区别。

[0064] (3.2)Matlab中的软件边界框设计

[0065] 可以使用任何合适的软件产品来实现边界框处理。作为一个示例，边界框是在Matlab中实现的。基于软件的边界框设计可以被限定为3个不同部分：准备、搜索/标记和合并区域。各个部分稍后将被转换成在硬件设计上实现。

[0066] (3.2.1)准备

[0067] 准备处理是变量和数组到其默认值的简单实例化和初始化。初始化的变量将是区域计数、先前y位置和当前y位置。初始化的数组将是“Image(图像)”、“Labeled Image(标记图像)”、“Merge to Region(合并到区域)”和“Bounding Box Data(边界框数据)”。区域计数将用作找到的标记像素的“票据”值。先前/当前Y位置用作减小标记图像矩阵大小的参考。减小的标记图像总大小允许数字电路在这样的电路中实现边界框方法时使用较少的硬件资源。由于标记算法搜索模式使用先前位置，所以必须通过在图像周围设置额外空白像素来容纳较大的图像大小，从而将宽度和高度的大小增加2。标记图像的高度为2，宽度设置为图像宽度。合并到区域是为最大区域计数的长度的单维数组集合，该单维数组集合是限于最大期望分布多少标记的集合。该限制是为了反映在数字实现期间使用的有限资源。边界框数据(Bounding Box Data)是二维的，其宽度为5份：两个x位置、两个y位置以及有效像素计数。边界框数据高度被设置为最大区域计数。

[0068] 例如，图3示出了在各个矩阵和变量的准备处理的实例化期间在各个矩阵和变量

之间的关系。既然已经创建了变量，则必须将变量初始化为正确的起始值。例如，系统启动多维度图像(即，大小)加上像素边界(即，填充多个维度的空白像素)的数组316，以形成图像318。合并到区域300和标记图像302两者的所有值均已被初始化至最大区域计数304。标记图像302继续改变312值，以使$Y_{先前}$(Y Previous)被设置为0并且$Y_{当前}$(Y Current)被设置为1。边界框数据306的最小x值和最小y值314分别被发送至图像宽度和图像高度的最大大小308。$Y_{先前}$被设置为1，并且$Y_{当前}$被设置为2。因此，上述矩阵中的值被初始化为非0值。初始化的最后一件事是被设置为0的区域计数(Region Count)310。这将跟踪生成了多少标记，将防止处理超过最大数组大小。

[0069] (3.2.2)搜索/标记

[0070] 在准备处理之后，系统继续进行搜索/标记处理，如图5所示。如标题所示，系统将搜索图像并且标记找到的像素(从图像的顶部到底部，或任何其它预定顺序)。像素是0或1范围内的二进制数，从而“找到的”像素是具有一值的像素。图4示出了如何进行搜索的示例。将图像中的当前像素位置用作(X,Y)，系统首先进行检查以查看(X,Y)是否具有有效像素。如果是，则系统继续检查(X-1,Y-1)、(X,Y-1)、(X+1,Y-1)和/或(X-1,Y)是否已被标记。如图5所示，该处理继续，直到通过500搜索完图像(例如，搜索到其底部或顶部等)为止，此时完成502搜索/标记。另选地，假定这是被找到的第一像素；因此，当前没有位置具有标记(例如，图像尚未被读取到其边缘504，并且在(x,y)处存在有效像素506，并且没有邻近像素已被标记508)。如果是这种情况，则该像素应被标记508有为1的区域计数，并且区域计数510增加。使用区域计数，系统将存储像素计数的边界框数据数组编索引至为1的值，并且使用当前x位置、当前y位置存储其最小值/最大值。由于找到的框的大小仅为一个像素，因此x位置和y位置的最大值和最小值将相等。现在，假定下一个像素也是有效的。这便创建了(X-1,Y-1)、(X,Y-1)、(X+1,Y-1)和/或(X-1,Y)已被标记的条件。然后，系统比较各个邻居的标记，以找到最低标记位置并将其称为“当前标记(Current Label)”512。为此，该处理使用先前Y和当前Y来对标签图像编索引；然而，使用(X-1)、(X)、(X+1)沿另一维度(即，沿x维度而不是y维度)移动。使得想到，在该示例中，通过使$(X-1,Y_{当前})$最低为值“1”，标记图像数组被加载有针对该区域的最大值。然后，将值“1”指派给该像素的标记。在完成对一个像素位置的评估之后，系统实现框526，增加“X”索引，以移动更接近图像的边缘。此外，在如框504所示到达图像的边缘之后，系统开始实现框524，框524将增加“Y”维度并且交换$Y_{当前}$值和$Y_{先前}$值。请记住，$Y_{当前}$和$Y_{先前}$用于保持小标记图像数组，因此，$Y_{当前}$和$Y_{先前}$交换将保留下一次评估所需的数据，同时打开将被重写的新集合。

[0071] 然后，继续进行该处理，以通过将新数据“(X,Y)”与所存储的最大/最小X和Y值进行比较，确定是否已更新边界框数据。如果所存储的最大/最小X和Y值小于/大于新数据，则系统将使用新的X位置和Y位置更新边界框数据。使用以上示例，系统必须通过增加Xmax来更新514边界框数据中的一些值，因为连通像素沿x方向将框大小增加1，并且像素计数将增加1。

[0072] 在更新边界框数据之后，注意稍后合并不可见像素，因为处理可能没有机会在当前扫描中重新标记这些像素。如下所述，在更复杂的示例的情况下，该概念将变得更有意义。为了记录合并，首先考虑识别516哪些邻居具有有效像素。具有有效像素的任何邻居都将与当前标记相比较来检查其合并到区域位置518，以找到最小标记。然后，将两者中的最

小者存储回其同一合并到区域位置520。在该示例中，通过用先前像素位置的标记对合并到区域编索引并且然后存储当前标记的值来结束搜索。最后，如果当前像素是无效522的，则标记图像数组通过$y_{当前}$被编索引，并且x被设置为针对区域计数+1的最大值。

[0073] (3.2.3)合并区域

[0074] 为了理解合并区域处理，查看和理解更复杂的“搜索/标记”示例很有帮助。例如，假定系统正在处理如图6C所示的图像。假设系统正在扫描图6C所示的像素图像，系统将从图6A至图6B至图6C逐渐“看到”图像。注意，在图6A和图6B中，图像在扫描期间包含分离的分量，使得人们将不知道两个分离的分量是连通的。在这种情况下，作为一个示例，假定左侧具有标记1，而右侧具有标记2。只有当像素在两侧之间桥接时，才知道这两侧是同一图像的一部分，并且应该被相应地标记；然而，系统/处理无法改变在标记2中找到的信息，因为此时不知道分量是否连通，也不知道多少个分量被连通。为了解决该问题，合并到区域数组利用为1的值设置位置2。稍后，该改变将被用于将为2的所有标记位置转换成为1的标记位置。

[0075] 在完整图像中并且如图6C所示，线600可以用于切割或以其它方式分割图像，从而示出了在该示例中，A和C将包含被标记为1的所有那些分量。B将包含被标记为2并且在该示例中仍被标记为2的所有那些分量。D将包含本该被标记为2但是现在反而被标记为1的所有那些分量。

[0076] 最后一步是合并区域处理，如图7所示。在该步骤中，将在合并到区域数组中找到的更新提醒设置在适当的边界框中。在该部分开始之前，创建700新数组，该新数组将跟踪有效边界框数据。有效数据数组将以为“真”的项达到区域计数并且为“假”的上述任何项达到最大区域计数(具有与边界框数据的长度相同的长度)开始。所存储的边界框数据值可以合并到中央位置，从而使一组或更多组数据无效。由于已知标记是从小数到大数分配的，因此系统通过从当前区域计数(先前用于计数分配了多少标记)中追溯边界框数据，在for循环中向下计数至1来开始。for循环是循环重复直到完成的处理。存在不同类型的这些循环，但通常for循环执行一些动作，直到达到结束条件为止。通常在循环中，要么向下计数，要么向上计数，以达到结束处理并退出循环的条件。

[0077] 通过查看for循环的当前索引并将其与由同一值索引的合并到区域702进行比较，来确定更新的必要性。如果在搜索/标记阶段期间的某个时间找到至不同的较小标记的连通，则合并到区域702数组应该被更新704为比索引更低的值。如果这种情况从未发生过，那么合并到区域自然将具有来自原始准备阶段的更大数量。因此，如果for循环的索引大于存储在具有该同一索引的合并到区域中的信息，则系统需要更新706边界框数据以包括最新找到的信息。边界框数据将包含Xmin、Ymin、Xmax、Ymax和像素计数信息；因此，为了更新706，该处理继续查看两个索引位置中的边界框数据(即，(1)当前索引的边界框数据和(2)存储在具有该同一索引的合并到区域中的值的边界框数据)。使用这两个位置；比较最小值以查看哪一者具有较小值，比较最大值以查看哪一者具有较大值，并组合像素计数值。然后将信息存储回由合并到区域值索引的边界框数据中，该合并到区域值由将包含较低标记的for循环编索引。一旦数组到达该较低索引708，边界框数据便在该位置利用与最小计数值、最大计数值和像素计数值有关的大多数当前信息被完全更新。重复该处理，直到区域计数为一710，这是最低标记，并且不可能有另一位置合并到其中。此时，合并区域处理终止712。现在，输出将是与验证数组一起存储为边界框数据的值，该验证数组识别边界框数据的哪

些部分包含具有未被合并或具有最新合并信息的区域的框。

[0078] (3.3)硬件边界框实现

[0079] 如上所述,本公开还提供了一种用于生成边界框的数字硬件实现。创建数字硬件实现需要将边界框减小到某个已知值范围。出于例示的目的,关于宽为512个像素、长为256个像素的图像描述实现,其中各个像素都包含一位值。该设计将从外部模块进行控制,使得边界框模块将接收启动信号,并且需要允许根据请求对必要图像位置编索引。为了满足图像处理设计的其它规范,还实现了附加过滤,并将所提供的结果减少到排序前15位的框。找到所有边界框并将其存储在存储器中;然而,该模块将具体提供15个边界框,所述15个边界框按照下文进一步详细讨论的方式进行排序。

[0080] 就像在软件设计中一样,硬件可以概括为三个阶段,即,准备、搜索/标记和合并区域。然而,在该硬件实现中,还有一个称为回调和排序的附加阶段。硬件中的转换还要求功能按特定时钟周期完成。在这种情况下,算法已分解为不同状态。另外,为了减少使用许多触发器的硬件负担,将大边界框数组存储在块随机存取存储器(BRAM)中。触发器是存储位的一种类型的寄存器。该触发器可以在FPGA的构造中找到,并且为了减少存储创建数据所需的量,可以将该触发器放入BRAM(这是FPGA中的另一组件)。这进一步要求将算法分解为多个状态,其中一些状态用于隐藏索引并且从BRAM接收信息。

[0081] (3.3.1)准备

[0082] 硬件准备部分翻译为算法中所需的变量的实例化和一些初始化。如之前所讨论的并且如图8所示,硬件的限制要求标记变量的大小。该处理在图8中例示,其中,一些框表示数组800,而其余框均表示值。最大限制是可以提供的标记数量。在一个示例中,标记数量减少到256,这反过来将设置许多数组的范围。就像前文所描述的,针对输入图像801包括标记图像数组802,在该示例中,输入图像801的高度将是255个维度并且宽度是511(图像的宽度)(维度)。包含的数据将具有最大为255的最大区域计数804,这意味着8位的大小可以表示这些范围。合并到区域806的宽度将为256,其中值为255那样大(这意味着8位的大小)。根据经验,关于宽度的任何项将为511那样大(需要9位),而高度将为255那样大(需要8位)。被呈现以供使用的像素的宽度为1位。由于BRAM 808的大小与最大标记(其为256)一样长,因此写入和读取地址的宽度将为8位。BRAM将包含来自边界框BRAM数组808的所有信息810。如在上述软件情况中一样,所包含的信息810将是Xmax、Ymax、Xmin、Ymin和像素计数的信息。然而,除此之外,硬件实现还包括Xsize和Ysize,其是简单的(Xmax-Xmin或Ymax-Ymin)计算。

[0083] 就像在软件中一样,所有变量都必须被初始化。可以在重置和状态0中初始化变量。在重置中,标记图像和合并到区域806中的所有值均被设置为255,这将是最大值标记,将所有其它值设置为0。由于不知道BRAM中包含的内容,因此边界框BRAM808将不被设置为任何值。相反,期望跟踪写指针的位置,以获知BRAM的哪个部分有效。在状态0 812中,当给出启动命令时,标记图像和合并到区域值被设置为255,状态被设置为1,$Y_{先前}$被设置为0,$Y_{当前}$被设置为1,所有其它值被设置为0。作为提醒,数字实现从0开始而不是从1开始对第一行编索引,这先前在软件实现中完成。

[0084] (3.3.2)搜索/标记

[0085] 与软件一起,后续是搜索/标记部分。由于边界框位于BRAM中,因此搜索/标记功能

被拆分，以允许读取BRAM。考虑到该约束，可以将搜索/标记软件设计进一步分成不同状态部分，以利用时钟延迟。因此，搜索/标记通过特殊增量器阶段被分为三个状态。

[0086] 如图9A和图10所示，状态1将包含用于在当前搜索位置找到新像素或未找到像素的条件。如果不存在硬性停止条件900，并且如果找到902当前有效像素，则期望确定像素的邻居(如图4所示)中的任一个是否具有有效像素904。如果没有一个邻居具有有效像素，则标记图像数和边界框BRAM被更新906，并且增量器908被激活918。对边界框BRAM的写命令将花费一个时钟周期，但是由于该处理(经由增量器908)使写入地址递增，以决不重写BRAM中的区域，并且状态1不读取BRAM，所以该处理可以返回到状态1 910而没有任何问题，从而不需要不必要的等待状态。为了进一步理解，FPGA/硬件由每个时钟周期并行运行的处理组成，并且将在每个时钟周期结束时确认动作。还应该注意，BRAM在某些时钟周期延迟下运行。BRAM是系统将要写入和读取的位置。考虑到FPGA在时钟周期结束时确认动作并且BRAM中存在延迟，因此期望在写入操作完成之前不会不正确地访问BRAM。换句话说，系统不尝试读取当前正在写入的位置，而是仅应在写入延迟结束后才读取该位置。还应注意，状态1不重写位置或不需要读取，它仅在处理返回到状态1 910的情况下才进行写入。这将隐藏写入时钟周期，使得BRAM准备就绪并且无需将等待时钟添加到该状态。

[0087] 如果相邻像素904中存在有效像素，则移动至状态2 912。数字实现的独特之处是硬性停止900和增量器908的功能。增量器908将充当for循环，从而移动当前像素并请求下一组像素值。一旦读取了整个图像，增量器908就将状态机移动至状态4 914，以开始合并区域部分。然而，由于给出过多标记而存在溢出的机会，因此状态1实现了硬性停止900，查看该处理何时比最大标记范围小。如果找到了，则不需要继续搜索图像，而是处理前进至状态4 916以开始合并区域部分。如果系统没有成功实现硬性停止并且没有找到有效像素，则920标记图像必须将存储在该像素位置的数据重置为最大区域计数。这将确保在系统比较标记图像位置(参见图12)时，最低位置是最新的，并且仅包含与该图像的该部分有关的有效数据。

[0088] 此外，图10例示了增量器908处理，示出了进行至状态1 910或状态4 914之间的决策。在激活增量器908之后，如果该处理尚未读取到图像的边缘1000，则系统使“X”索引递增，以移动更接近图像的边缘1002并进行至状态1 910。另选地，如果该处理已读取到图像的边缘1000，则将“X”索引重置为1 1004，并且确定处理是否在图像的末尾处1006。如果不是，则使“Y”索引递增，以移动更接近图像的末尾，并且交换“$Y_{当前}$”和“$Y_{先前}$”的值1008，并且进行至状态1 910。另选地，如果是，则将“$Y_{当前}$”和“$Y_{先前}$”重置为其初始值，并且将有效区域计数锁定为区域计数的当前值1010，并且进行至状态4 914。

[0089] 如图11所示，状态2使用被称为当前标记模块1100的另一模块，该当前标记模块1100在图12中进一步详细示出。这里，时钟周期延迟被用于执行当前标记指派1112，以准备用于该阶段(注意上面关于FPGA时钟周期和BRAM读取/写入延迟的讨论)。再次参考图11，由于边界框BRAM 1102将花费一个时钟周期来读取，因此必须与当前标记1104一起发送读取信号以读取包含将被组合的数据的地址。BRAM需要信号以及地址来指示该处理将进行读取。“当前标记”1102将是读取地址。请注意上面关于“搜索/标记”部分的功能的注释，其中该系统在此处组合标记以供之后合并。

[0090] 状态2将仅包含设置合并到区域的部分。存在状态2以利用数据设置“合并到区域”

1108数组，该数据稍后将在状态3a 1110和“合并区域”阶段期间使用。

[0091] 就像在软件方面一样，将当前标记1104与存储在合并到区域中的内容进行比较1106，以基于有效邻近像素1114进行更新，针对有效邻近像素1114，系统将“合并到区域”数组中找到的值与最低标记值进行比较。较低“赋值”标记将是存储在“合并到区域”数组中的值。因为标记是按连续顺序提供的，所以较低标记应该是合并区域部分中随后使用的参考。参见例如例示了状态2代码的图9B和例示了当前标记模块1102的图12。

[0092] 图13所示的状态3涵盖了搜索/标记阶段的最后部分。在状态3中，边界框BRAM 1102被更新。然而，由于在状态2中发送了读取，因此该数据仅在一个时钟周期后才有效。因此，将状态3参考为两个部分。状态3A 1300将等待一个时钟周期以获得来自BRAM 1102的有效数据，然后状态3B 1302将更新边界框BRAM 1102。假定在第二周期内接收到数据，则可以执行对边界框BRAM 1102的更新。更新包括将从计算出的当前标记位置读出的数据与当前信息进行组合。就像在软件中一样，期望比较x位置和y位置的最大值和最小值，然后将较大和较小的那个值设置回到边界框BRAM 1102中。针对找到的新像素，还期望将像素计数增加到更大的值。数字实现还增加了另一部分来计算X和Y方向的大小。这稍后将被用作针对不需要的边界框的过滤器。另一增加是过滤出不满足像素大小的特定范围1308的不需要框。下边界和上边界像素计数被用于验证边界框BRAM 1102存储(其中像素计数位于被指定为有效的有效像素范围1310内，而有效像素范围之外的像素计数被指定为无效1312)。因此，有效区域数组被用于通过将值“1”指派给容纳有效数据集的地址来确定应在稍后阶段测试存储在BRAM中的哪些信息。稍后将循环通过有效区域数组，以查看系统在回调和排序阶段期间是否应从该地址位置读取存储在BRAM中的数据。最终，系统激活增量器908并将处理1304发送至状态1。然而，如果增量器908检测到该处理在最后像素位置，则它反而将处理1306发送回至状态4，如图10所示。

[0093] 如图14所示，状态4、状态5和状态6涵盖了合并区域阶段。在状态4中，系统通过开始于区域计数1400然后向下计数以确定是否需要合并数据，来在合并到区域1402中进行搜索。如果是，则需要来自边界框BRAM 1102的两个地址(区域计数1400和合并到区域[区域计数]1402)，并且系统写回到边界框BRAM 1002位置。如果不是，则系统基于在状态6 1426 1430期间创建并最终通过1418实现的数据，通过减小区域计数1422和减小有效区域计数1420来继续在合并到区域中进行搜索。

[0094] 由于BRAM 1102读取，所以该处理必须返回至状态4，以涵盖读取的一个时钟周期延迟并请求不同地址。为了清楚起见，状态4被分为两部分，用于确定是否需要合并的状态4a 1404以及包括时钟周期等待和读取下一地址的状态4b 1406。

[0095] 状态5 1408非常简单，因为已知BRAM 1102具有来自状态4a 1404的有效数据。因此，必须保存1410已到达的数据，使得稍后可以使用该数据与在状态4b 1406中读取的地址进行比较。

[0096] 状态6 1412然后将比较1414从两个边界框BRAM 1102读取接收的两组数据，然后将合并信息相应地存储回具有较低地址的边界框BRAM 1102中。由于写入将发生在状态4a 1404期间，因此在激活下一地址的读取之前，系统将已正确写入。到达状态6指示当前区域计数将被合并，因此系统将使该区域对于有效区域数组无效，并且减小有效区域计数1424。如上所述，可以添加过滤器以过滤掉不满足像素计数的特定范围1426的区域(不需要的

框)，其中像素计数位于被指定为有效的有效像素范围1428内，而将有效像素范围之外的像素计数指定为无效1430。

[0097] 该实现的独特之处是状态7 1416承载的附加回调和排序阶段。具体地并且如图15和图16所示，状态7分别集中于回调操作和排序操作。如图15的回调操作所示，该阶段将从BRAM 1102回调所有有效信息，直到该处理读出了已经存储的所有有效区域为止。根据先前状态，该处理一直对边界框BRAM 1102中有多少个位置在过滤后仍是有效的进行计数，并利用有效数组跟踪地址。因此，为了回调所有有效位置，有效数组用于检查BRAM 1102中的区域是否具有有效数据，然后使有效数据计数递减。在读取时，BRAM具有延迟的两个时钟周期。通过在地址零1512处的有效区域1510开始从BRAM读取1508，并且通过在返回状态7 1516之前迫使一个时钟周期等待1514，来处理从状态4a移动至状态7。该思想是不断从边界框BRAM 1102读取地址，使得该处理仅比每次读取晚一个时钟周期1500。状态7 1416将基于从边界框BRAM 1102的读取是否有效1502并且满足附加过滤来进行过滤。在返回至状态0 1506之前，在1504中包括延迟时钟周期，以解决附加过滤延迟。框1518示出了对有效区域的搜索的结束，因此必须强制系统停止对1520中看到的地址进行持续读取。

[0098] 在状态3和状态6期间，像素计数有效范围被用作验证边界框BRAM 1102数据的初始条件。现在，期望通过比较Xsize(Xmax-Xmin)与Ysize(Ymax-Ymin)来过滤出形状奇怪的“框”。如果Xsize/Ysize或Ysize/Xsize<＝30％(或任何其它预定值)，则这些框无效。除此之外，期望找到填充有像素斑点的框，从而覆盖找到的框的良好区域。因此，系统还通过检查以查看(像素计数)/Xsize×Ysize<＝30％(或任何其它预定值)并将这些位置设置为无效来进行过滤。如果数据已通过所有过滤器，则将其标记为有效排序并传递到“排序”部分。只有那些有效的框才被排序，并且在本地保存以用于其它模块。由于排序中的延迟，状态7可能在最后一次从BRAM 1102读取有效数据之后完成8个时钟周期。因此，例如，该处理在确定边界框已完成并且当前被保存之前等待8个时钟周期。由于排序部分地在另一模块中完成，因此将重置值，重置是从状态0到状态1的转变。

[0099] 如图16所示，在状态7和另外的排序模块1600中完成排序。有效bram读取1508和边界框数据1102被过滤1604以识别有效排序。另外，读取的结果和有效排序具有添加的一个时钟延迟1606、1608。如果发现数据是有效排序1602，则由排序模块1600进一步对其进行排序。

[0100] 为了进一步理解，图17示出了状态7的流程图，其集中于各个单独排序模块中的排序操作。各个排序模块首先通过确定1700与边界框或区域相关联的排序是否有效或是否已经发布重置排序来开始。如果由有效排序值0确定排序是无效排序，则模块将发出传递无效排序命令和相关数据的“0”值。如果排序是重置排序1704，则系统清空存储在排序中的数据1706。存储在排序中的清空数据1706是指本地存储的排序数据。该排序数据稍后将保存从回调BRAM读取中找到的值，其最终本地存储像素计数、xmax、xmin、ymax、ymin和(Xsize＝xmax-xmin与Ysize＝ymax-ymin)之间的最大者。排序模块均通过将像素计数除以Xsize与Ysize之间的最大者来创建排序编号。该排序编号和有效排序在排序模块之间传递1708，使得较高排序编号保持在顶部，而较低排序编号下降以开放排序或完全离开保存区域。通过首先确定1710传入排序是否大于当前存储数据的较高排序来进行该处理。如果是，则将传入排序设置为1712当前存储数据的较高排序，其中先前存储的较高排序数据然后被设置

1714为所存储的较低数据，并且从排序模块中移除1716。如果传入排序小于当前存储数据的较高排序，则确定1718传入排序是否大于先前存储数据的较低排序。如果否，则从排序模块中移除1722该传入排序数据。另选地，如果传入排序大于先前存储数据的较低排序，则将传入排序数据设置1720为较低排序存储数据，并且将其从排序模块传递出去1716。

[0101] (3.3.3)硬件实现结果

[0102] 进行模拟，其中使已知图像经历上述边界框实现。遵守算法需求并且如图18所示，已知图像1800的大小为512×256个像素，具有单个位像素(即，一位值用于各个像素位置)。通过使图像1800通过过滤器，系统识别出220个标记位置，这些标记位置随后合并为182个唯一边界框。通过排序，系统将边界框过滤成只剩前15位“排序”位置(如图19所示)。因此，这表明了本公开的边界框处理在图像中的被识别对象中并且围绕这样的对象生成边界框是有效的。基于此，可以对视频图像中的连续帧实现本文所述的边界框处理，以按照任何期望设置用作高效且有效的移动跟踪器。

[0103] (3.4)设备的控制。

[0104] 如图20所示，处理器2000可以用于基于边界框生成来控制设备2002(例如，移动设备显示器、虚拟现实显示器、增强现实显示器、计算机监测器、马达、机器、无人机、相机等)。设备2002的控制可以用于将对象的定位转换成表示对象的静止图像或视频。在其它实施方式中，可以基于鉴别和定位来控制设备2002，以使设备移动或以其它方式发起物理动作。

[0105] 在一些实施方式中，可以控制无人机或其它自主交通工具移动到基于图像确定对象的定位的区域。在又一些其它实施方式中，可以通过将移动边界框保持在视场内来控制相机，以跟踪所识别的对象。换句话说，致动器或马达被激活以使相机(或传感器)移动，以将边界框保持在视场内，使得操作者或其它系统可以识别并跟踪对象。作为又一示例，该设备可以是自主交通工具(诸如，无人飞行器(UAV))，该自主交通工具包括相机和本文所述的边界框设计。在操作中并且当由在UAV中实现的系统生成边界框时，可以使UAV操纵跟随对象，使得边界框保持在UAV的视场内。例如，UAV的转子和其它组件被致动以使UAV追踪并跟随对象。

[0106] 最后，虽然已经根据多个实施方式对本发明进行了说明，但本领域普通技术人员应当容易地认识到，本发明可以在其它环境中具有其它应用。应注意，可以有许多实施方式和实现。另外，“用于…的装置”的任何用语旨在引发要素和权利要求的装置加功能的解读，而未特别使用“用于…的装置”用语的任何要素不应被解读为装置加功能要素，即使权利要求以其它方式包括了“装置”一词。此外，虽然已经按特定顺序陈述了特定方法步骤，但这些方法步骤可以按任何期望的顺序进行，并且落入本发明的范围内。

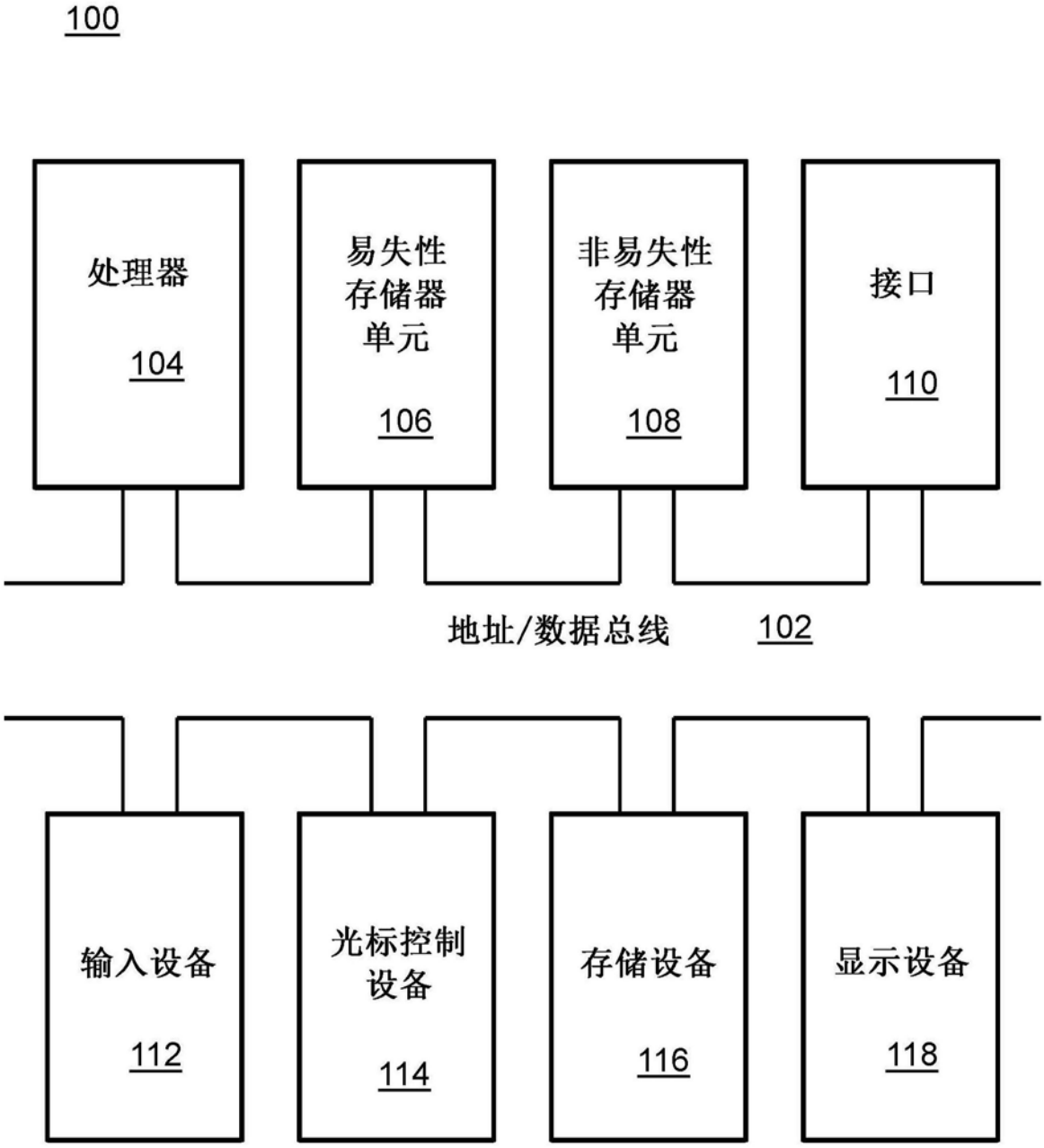

100
处理器
104
易失性
存储器
单元
106
非易失性
存储器
单元
108
接口
110
地址/数据总线
102
输入设备
112
光标控制
设备
114
存储设备
116
显示设备
118

图1

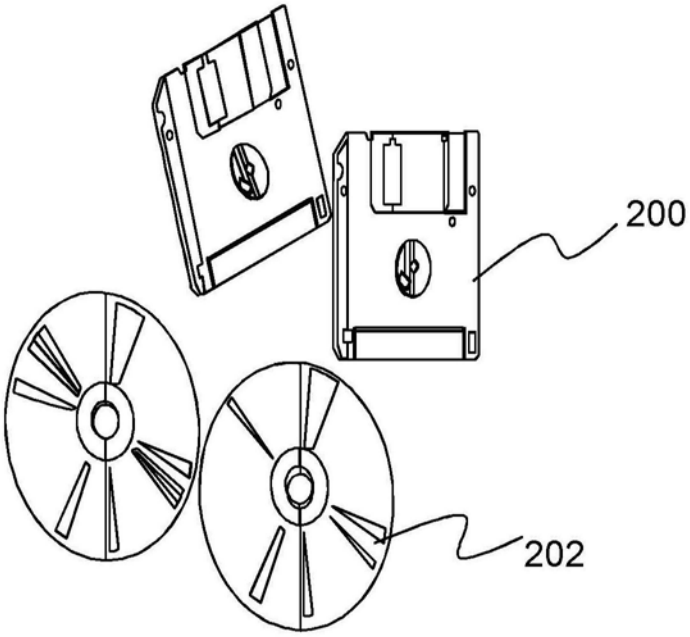


图2

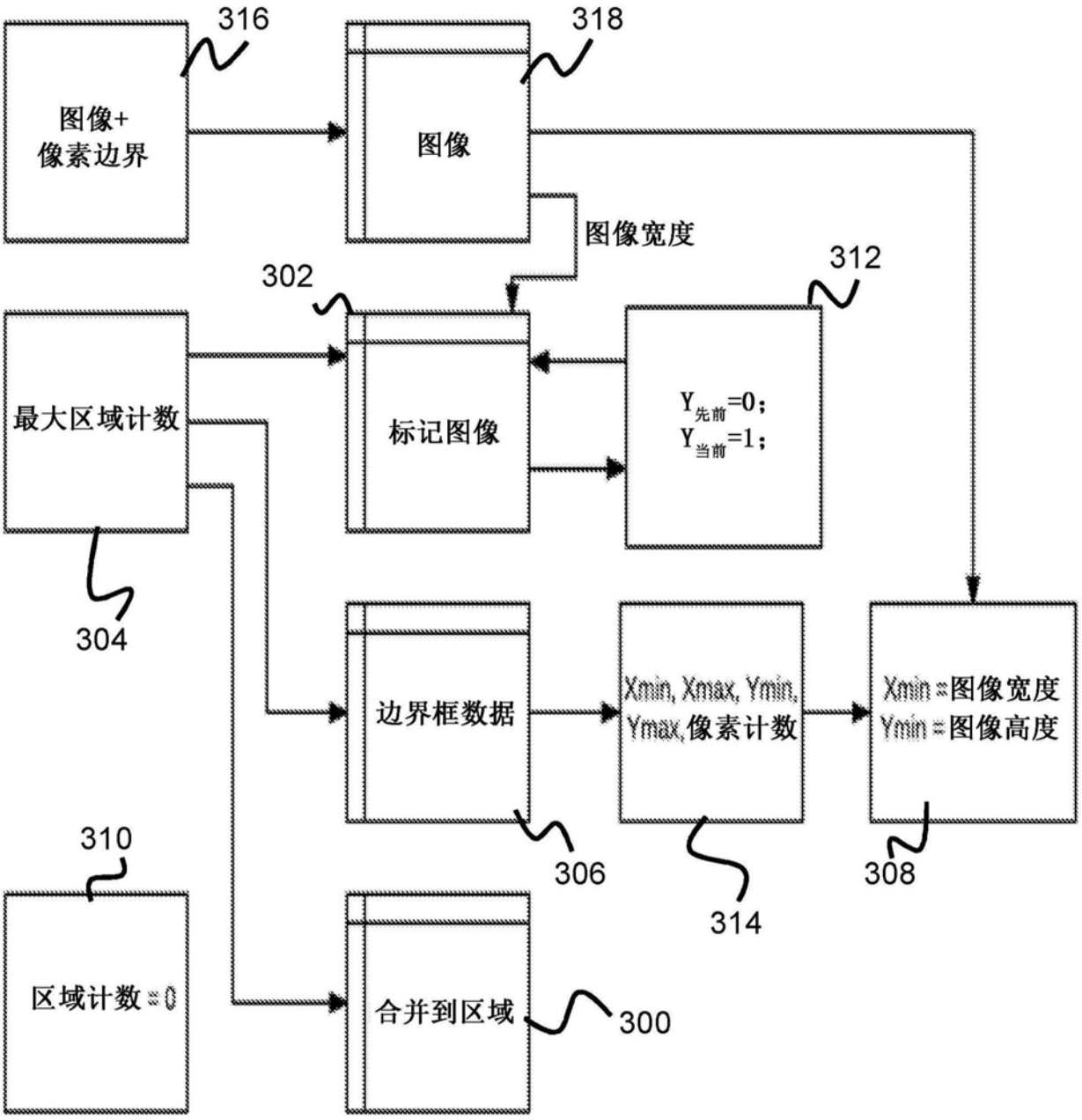


图3

X-1,Y-1	X,Y-1	X+1,Y-1
X-1,Y	X,Y	

图4

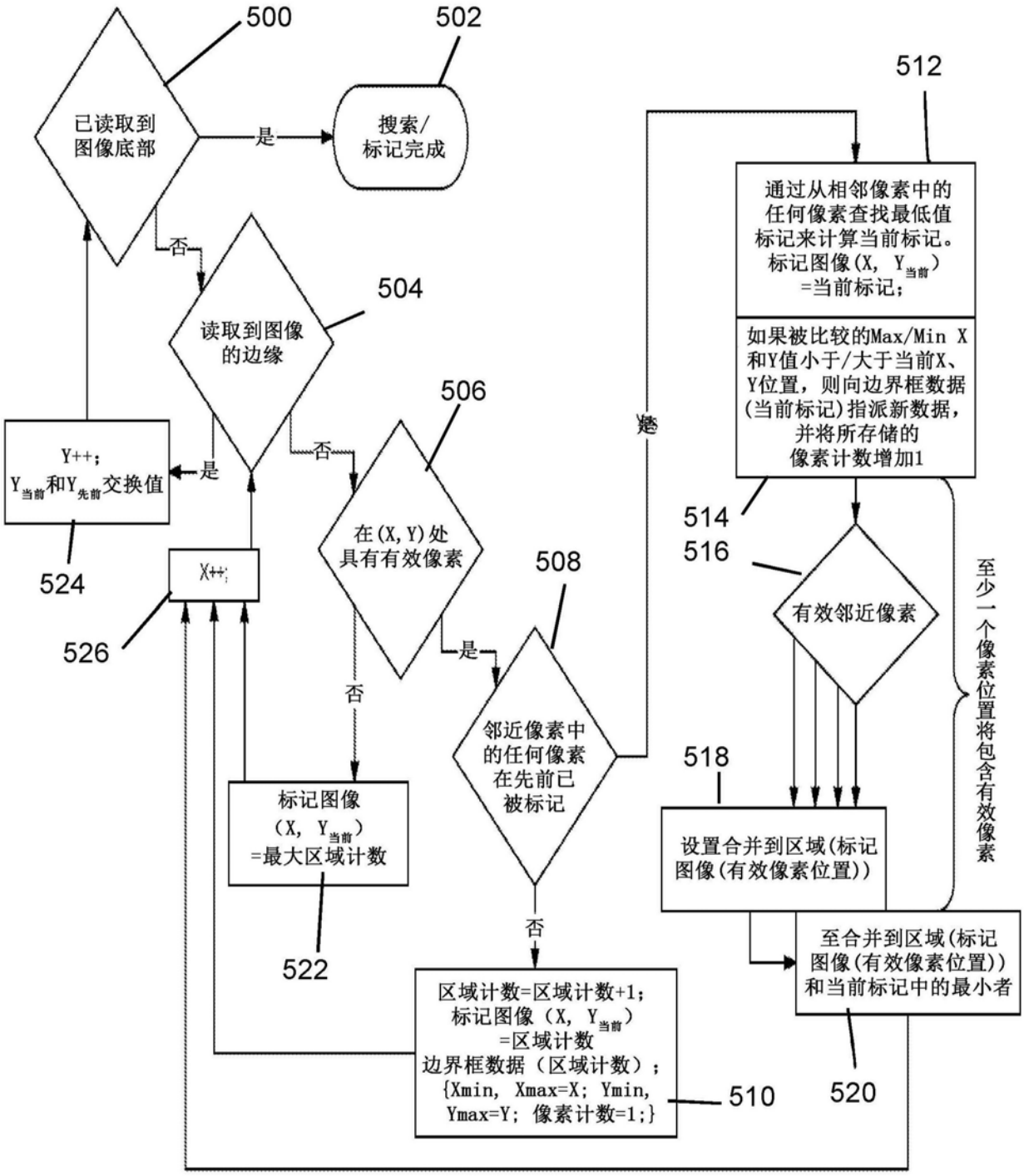

500
已读取到图像底部
是
502
搜索/标记完成
否
504
读取到图像的边缘
是
Y++;
$Y_{当前}$和$Y_{先前}$交换值
524
否
506
在(X,Y)处具有有效像素
是
508
邻近像素中的任何像素在先前已被标记
是
512
通过从相邻像素中的任何像素查找最低值标记来计算当前标记。标记图像(X, $Y_{当前}$)=当前标记;
如果被比较的Max/Min X和Y值小于/大于当前X、Y位置，则向边界框数据(当前标记)指派新数据，并将所存储的像素计数增加1
514
516
有效邻近像素
至少一个像素位置将包含有效像素
518
设置合并到区域(标记图像(有效像素位置))
至合并到区域(标记图像(有效像素位置))和当前标记中的最小者
520
否
区域计数=区域计数+1;
标记图像（X, $Y_{当前}$）=区域计数
边界框数据（区域计数）;
{Xmin, Xmax=X; Ymin, Ymax=Y; 像素计数=1;}
510
X++;
526
标记图像(X, $Y_{当前}$)=最大区域计数
522
否

图5

图6A

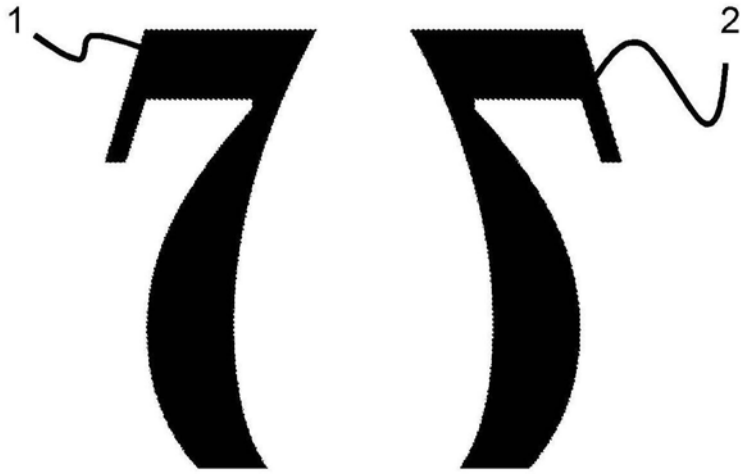


图6B

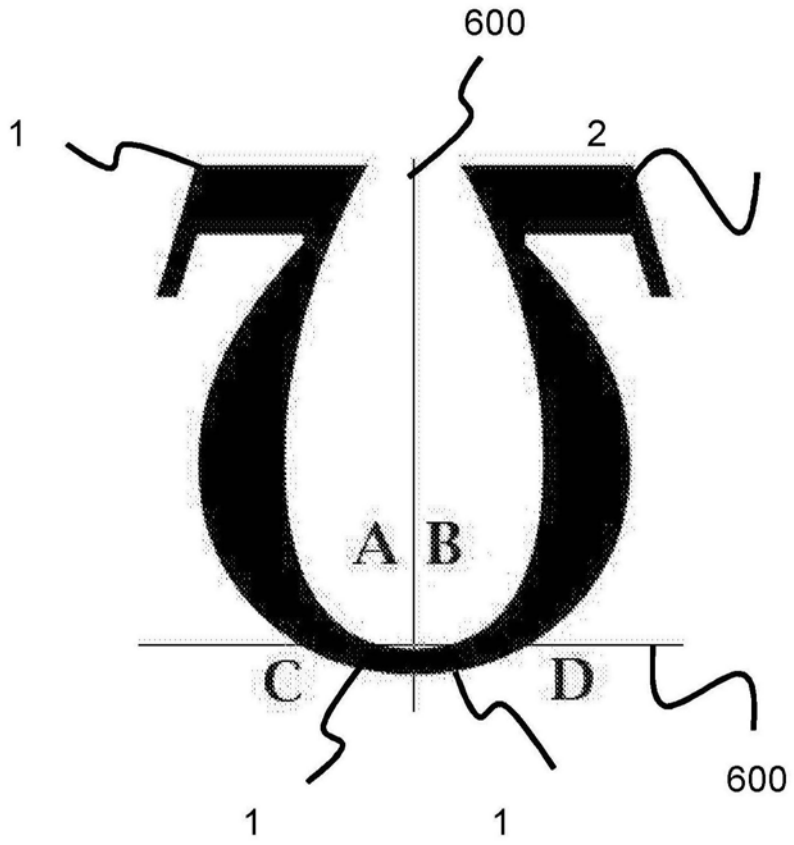


图6C

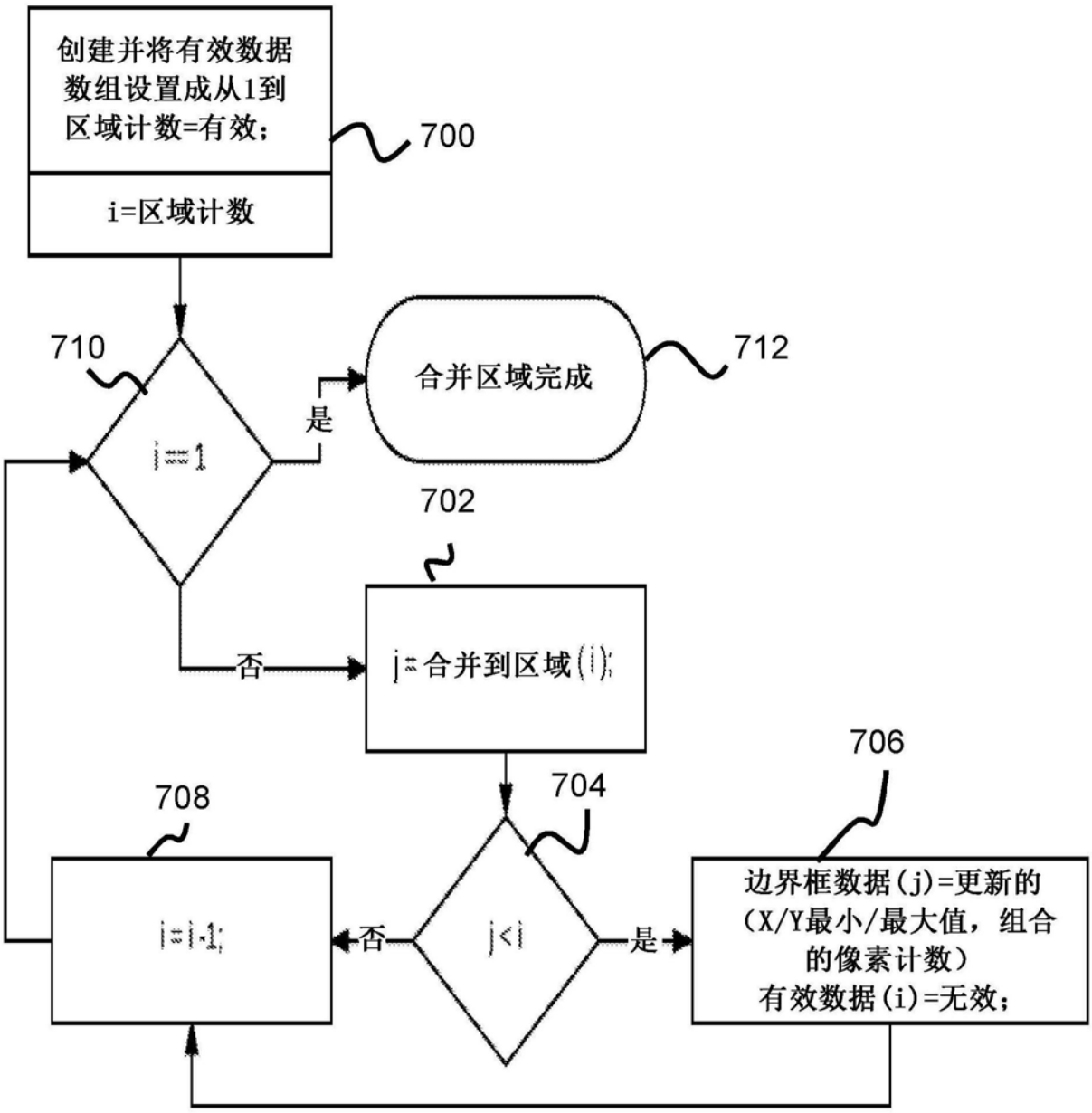

创建并将有效数据数组设置成从1到区域计数=有效；
700
i=区域计数
710
i==1
是
合并区域完成
712
否
702
j=合并到区域(i);
706
704
708
i=i-1;
否
j<i
是
边界框数据(j)=更新的(X/Y最小/最大值，组合的像素计数)
有效数据(i)=无效；

图7

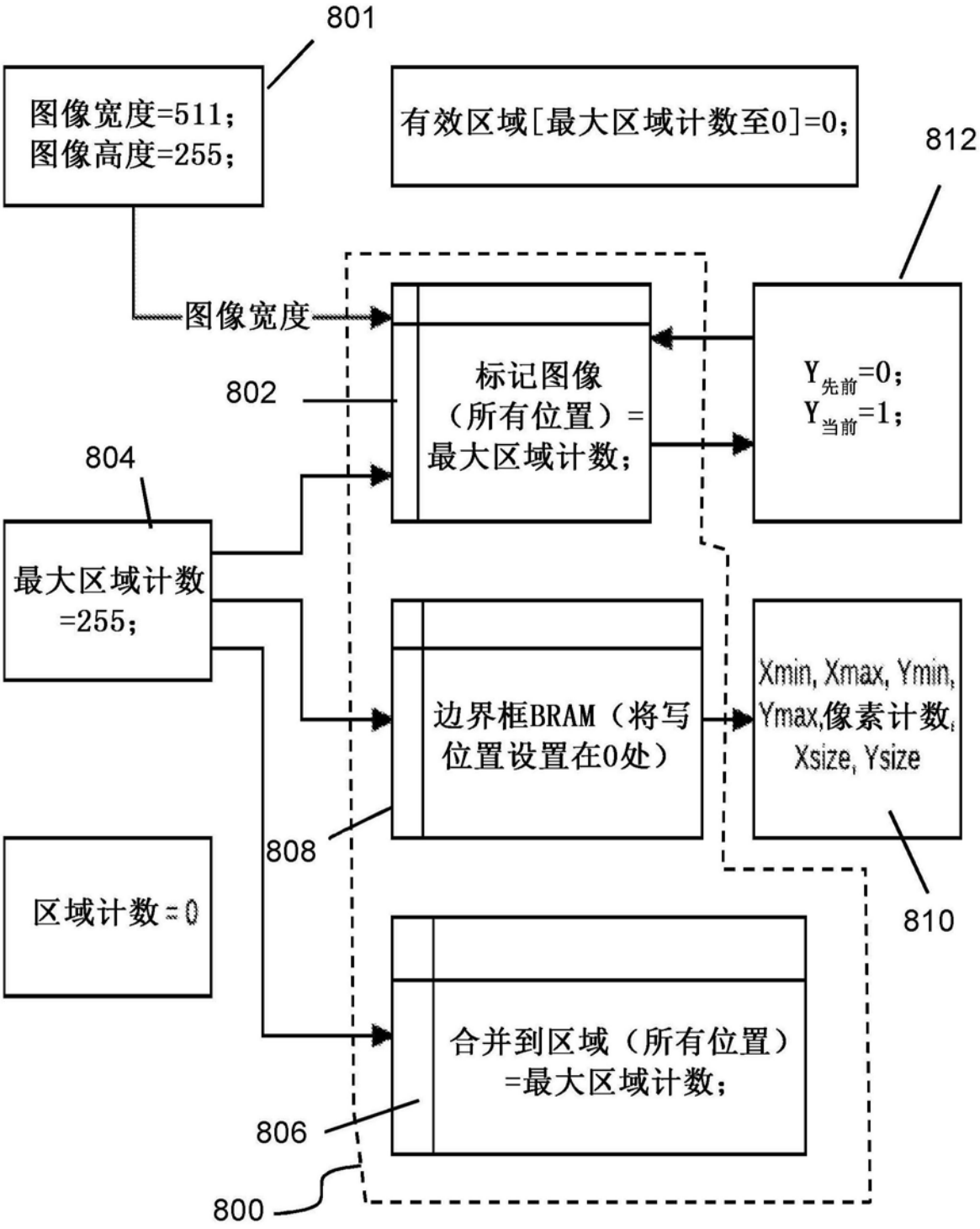

801
图像宽度=511；
图像高度=255；
有效区域[最大区域计数至0]=0；
812
图像宽度
802
标记图像
（所有位置）=
最大区域计数；
$Y_{先前}$=0；
$Y_{当前}$=1；
804
最大区域计数
=255；
边界框BRAM（将写
位置设置在0处）
Xmin, Xmax, Ymin,
Ymax,像素计数,
Xsize, Ysize
808
810
区域计数=0
合并到区域（所有位置）
=最大区域计数；
806
800

图8

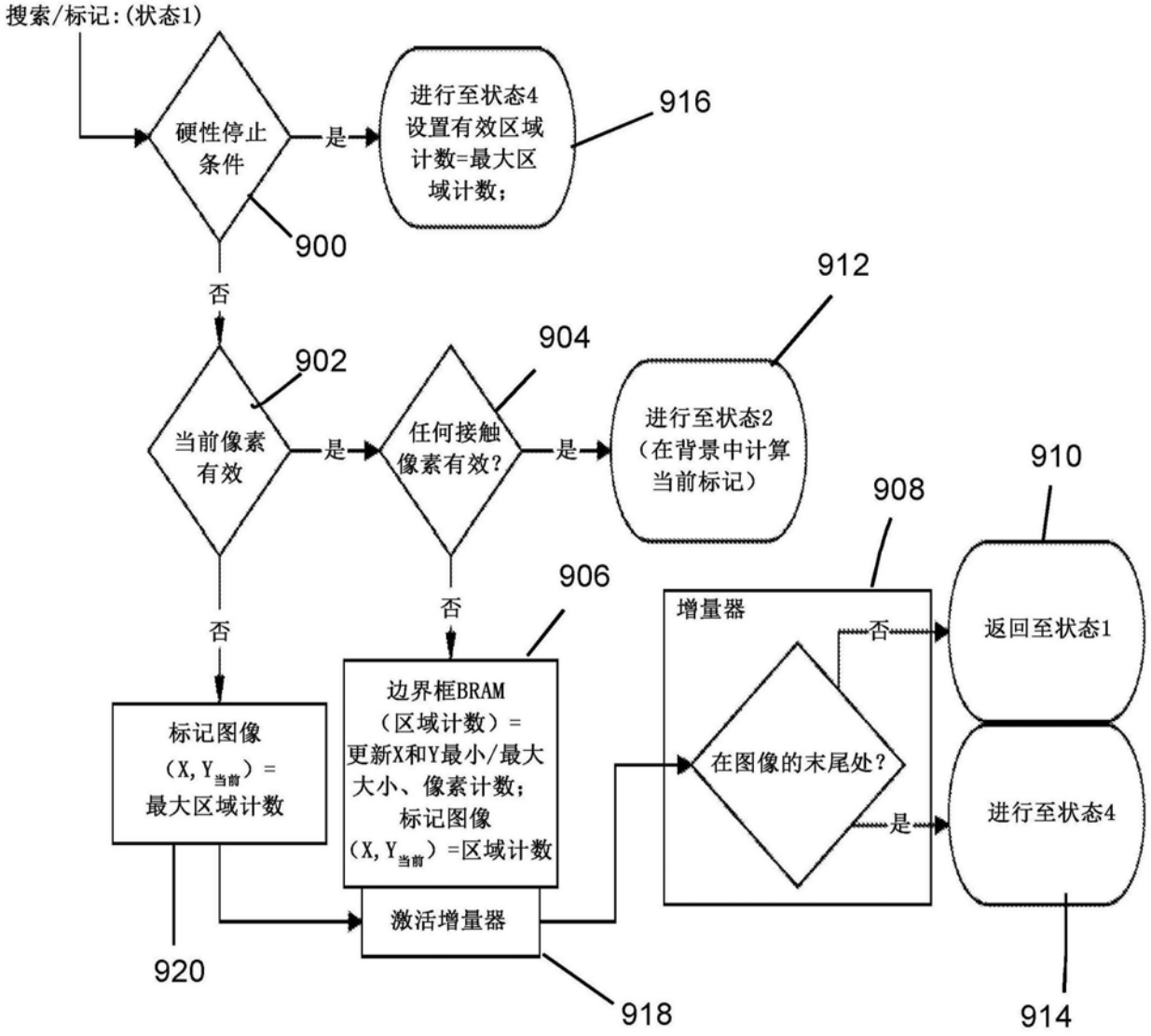

搜索/标记:(状态1)
硬性停止条件
是
进行至状态4
设置有效区域计数=最大区域计数;
916
900
否
912
902
904
当前像素有效
是
任何接触像素有效?
是
进行至状态2
(在背景中计算当前标记)
910
908
906
否
否
增量器
返回至状态1
标记图像
(X,Y当前)=
最大区域计数
边界框BRAM
(区域计数)=
更新X和Y最小/最大大小、像素计数;
标记图像
(X,Y当前)=区域计数
在图像的末尾处?
否
是
进行至状态4
激活增量器
920
918
914

图9A

```
state <= 3; //Move to state 3
//We will be writing BRAM LATER but first
write_addr<= currentlabel; //Just to grab valid label
//We will be reading BRAM
read_enable<=1;
read_addr<= currentlabel;
//Note the location written
written[currentlabel]<=1'b1;

//Bounding Box Algorithm variables
labelImage[LabelyCurr][xcurr] <= currentlabel;

if(xprev_yprev_pixel == 1)begin
    if(mergeToRegion[labelImage[LabelyPrev][xprev]] < currentlabel)begin
        mergeToRegion[labelImage[LabelyPrev][xprev]] <= mergeToRegion[labelImage[LabelyPrev][xprev]];
    end
    else begin
        mergeToRegion[labelImage[LabelyPrev][xprev]] <= currentlabel;
    end
end

if(xcurr_yprev_pixel == 1)begin
    if(mergeToRegion[labelImage[LabelyPrev][xcurr]] < currentlabel)begin
        mergeToRegion[labelImage[LabelyPrev][xcurr]] <= mergeToRegion[labelImage[LabelyPrev][xcurr]];
    end
    else begin
        mergeToRegion[labelImage[LabelyPrev][xcurr]] <= currentlabel;
    end
end

if(xnext_yprev_pixel == 1)begin
    if(mergeToRegion[labelImage[LabelyPrev][xnext]] < currentlabel)begin
        mergeToRegion[labelImage[LabelyPrev][xnext]] <= mergeToRegion[labelImage[LabelyPrev][xnext]];
    end
    else begin
        mergeToRegion[labelImage[LabelyPrev][xnext]] <= currentlabel;
    end
end
if(xprev_ycurr_pixel == 1)begin
    if(mergeToRegion[labelImage[LabelyCurr][xprev]] < currentlabel)begin
        mergeToRegion[labelImage[LabelyCurr][xprev]] <= mergeToRegion[labelImage[LabelyCurr][xprev]];
    end
    else begin
        mergeToRegion[labelImage[LabelyCurr][xprev]] <= currentlabel;
    end
end
```

图9B

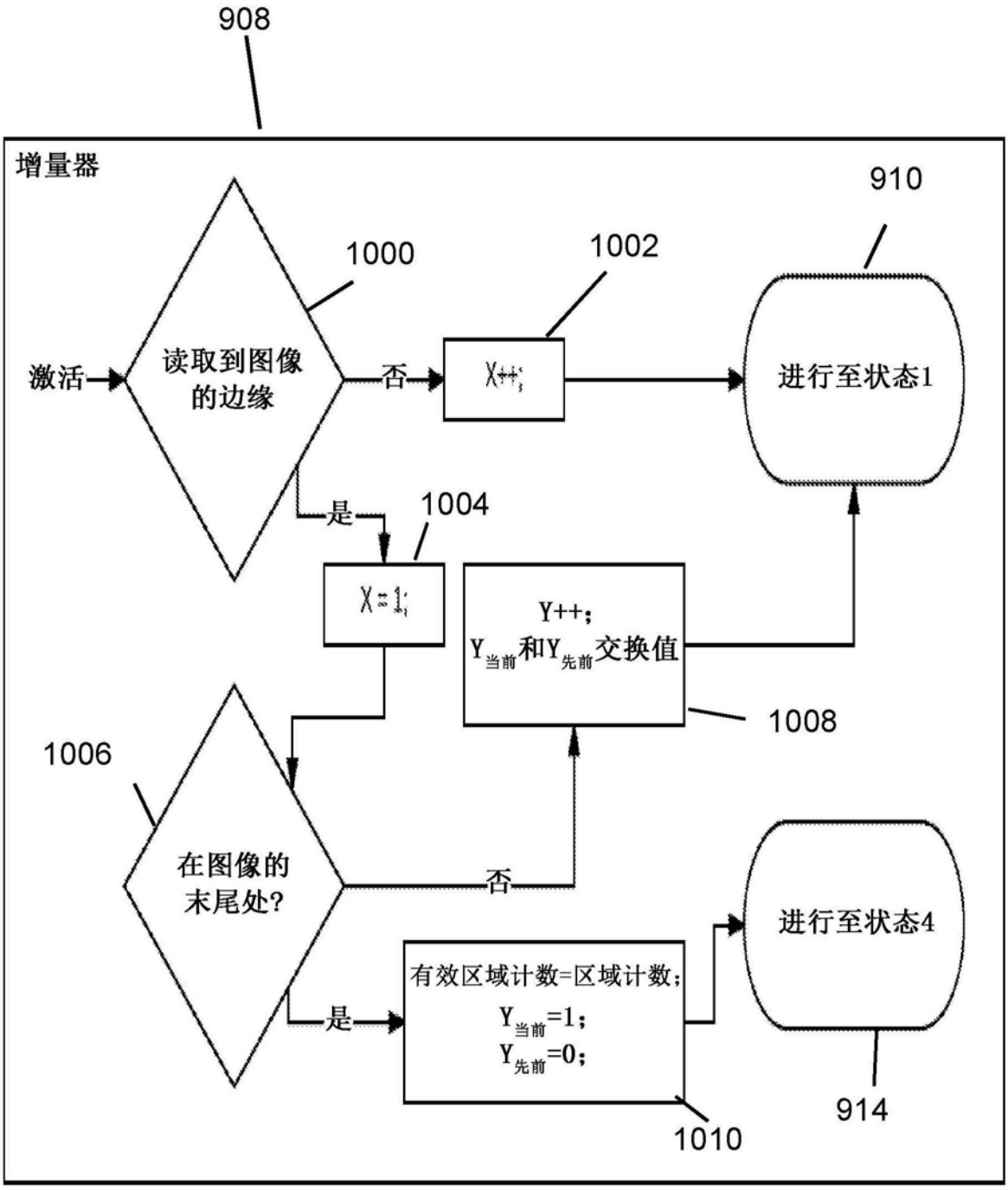

908
增量器
910
1000
1002
激活
读取到图像的边缘
否
X++;
进行至状态1
是
1004
X=1;
Y++;
Y当前和Y先前交换值
1008
1006
在图像的末尾处?
否
进行至状态4
有效区域计数=区域计数;
Y当前=1;
Y先前=0;
914
1010

图10

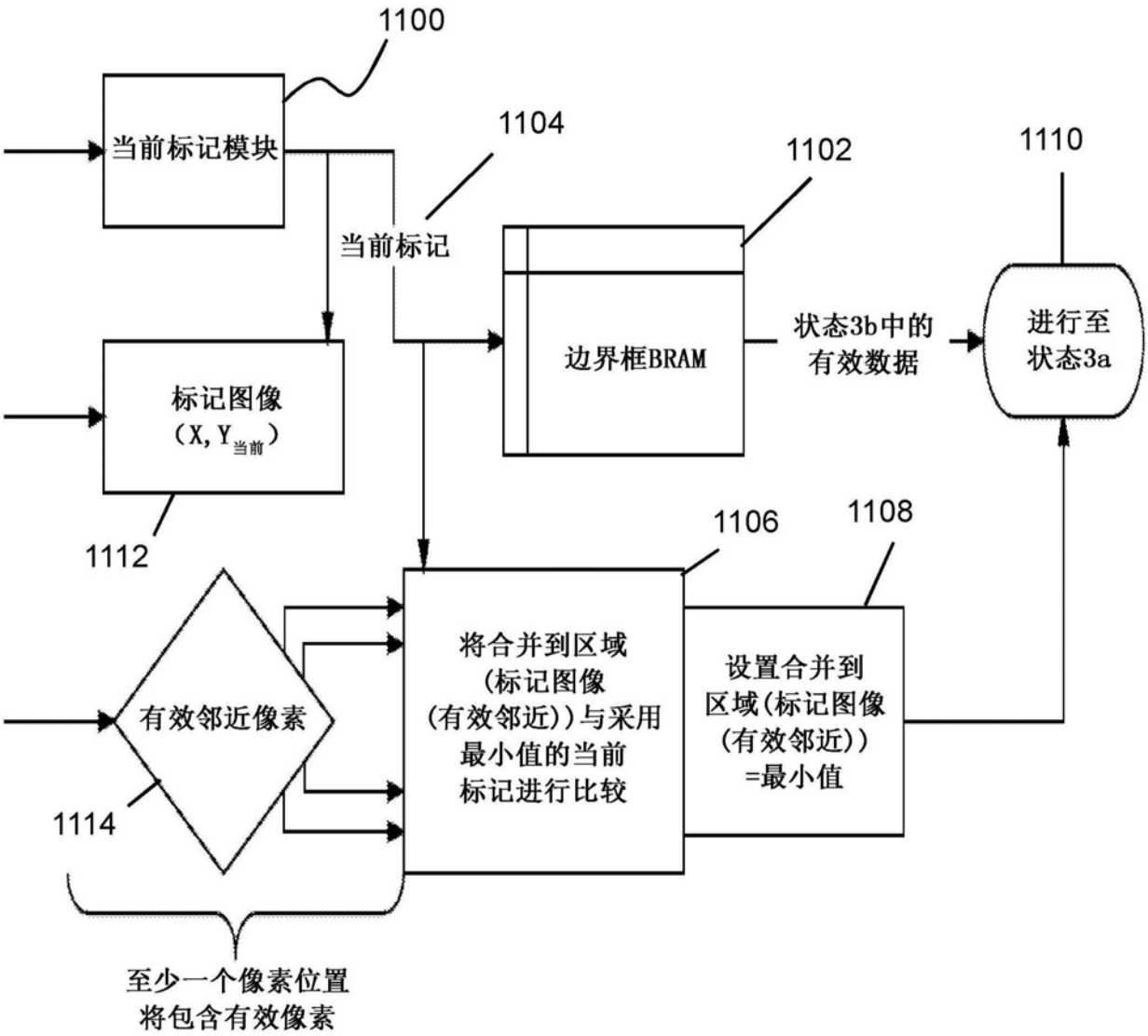

1100
当前标记模块
1104
当前标记
1102
1110
边界框BRAM
状态3b中的
有效数据
进行至
状态3a
标记图像
(X,Y当前)
1112
1106
1108
有效邻近像素
1114
将合并到区域
(标记图像
(有效邻近))与采用
最小值的当前
标记进行比较
设置合并到
区域(标记图像
(有效邻近))
=最小值
至少一个像素位置
将包含有效像素

图11

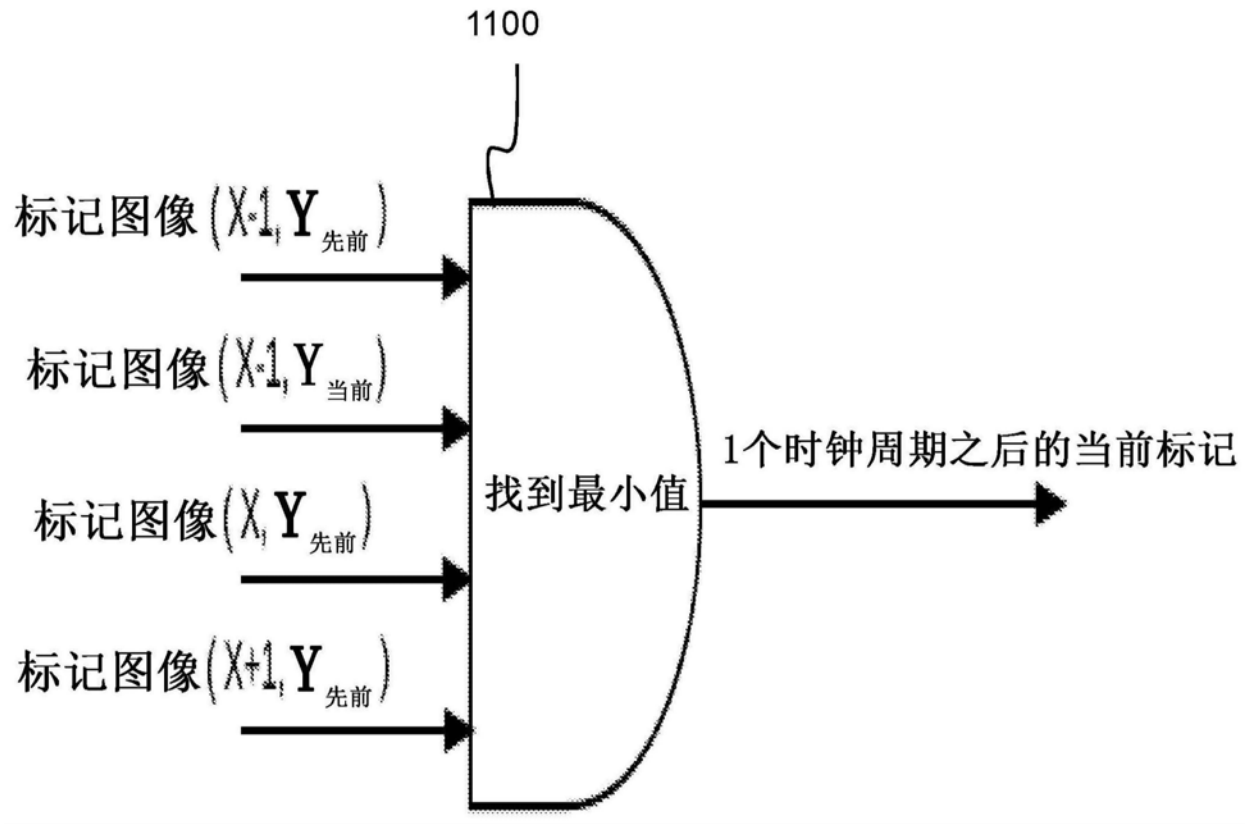

1100
标记图像 $(X-1, Y_{先前})$
标记图像 $(X-1, Y_{当前})$
标记图像 $(X, Y_{先前})$
标记图像 $(X+1, Y_{先前})$
找到最小值
1个时钟周期之后的当前标记

图12

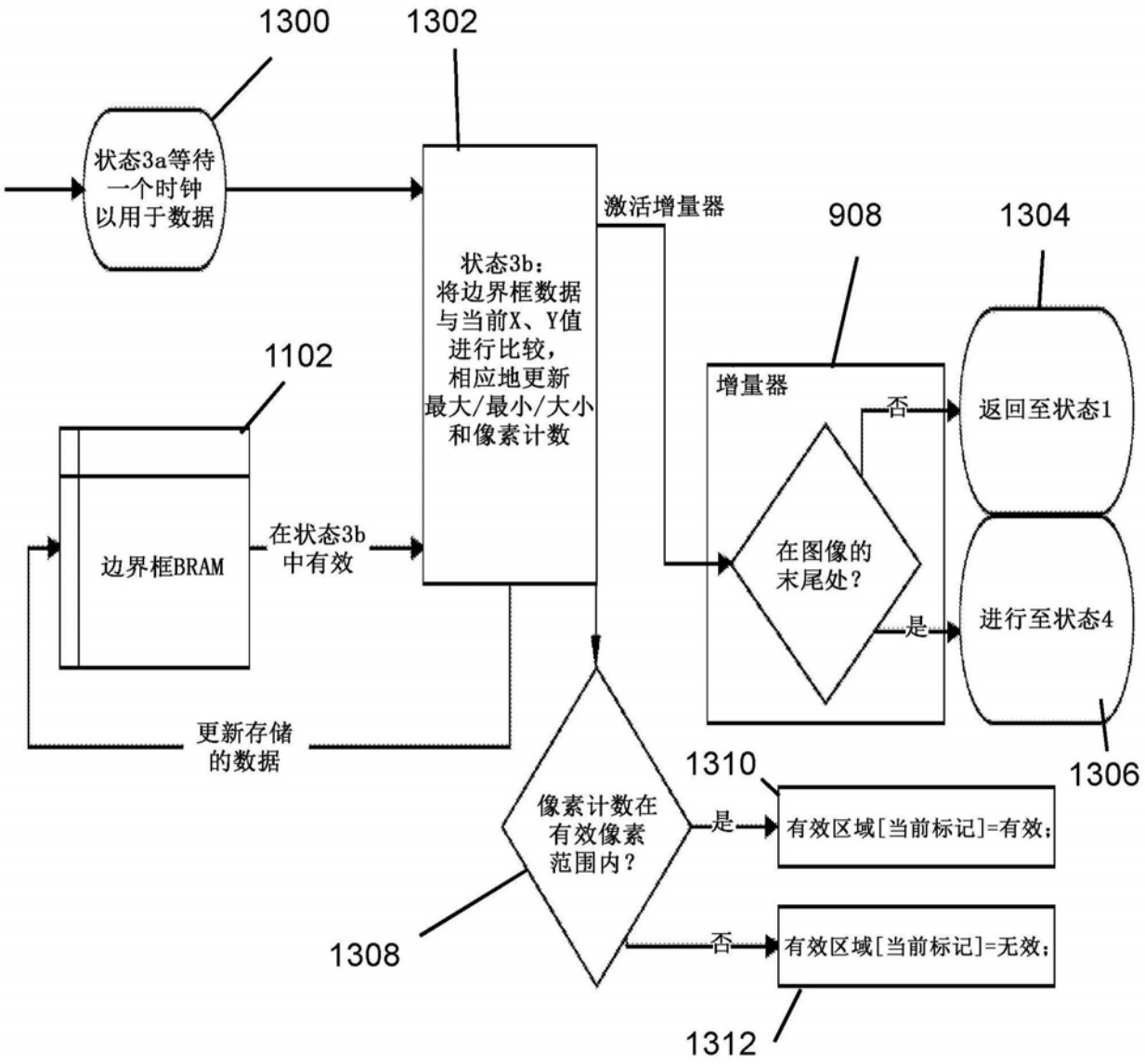

1300
1302
状态3a等待一个时钟以用于数据
激活增量器
908
1304
状态3b：将边界框数据与当前X、Y值进行比较，相应地更新最大/最小/大小和像素计数
增量器
否
返回至状态1
1102
在图像的末尾处？
是
进行至状态4
边界框BRAM
在状态3b中有效
更新存储的数据
1306
1310
像素计数在有效像素范围内？
是
有效区域[当前标记]=有效；
否
有效区域[当前标记]=无效；
1308
1312

图13

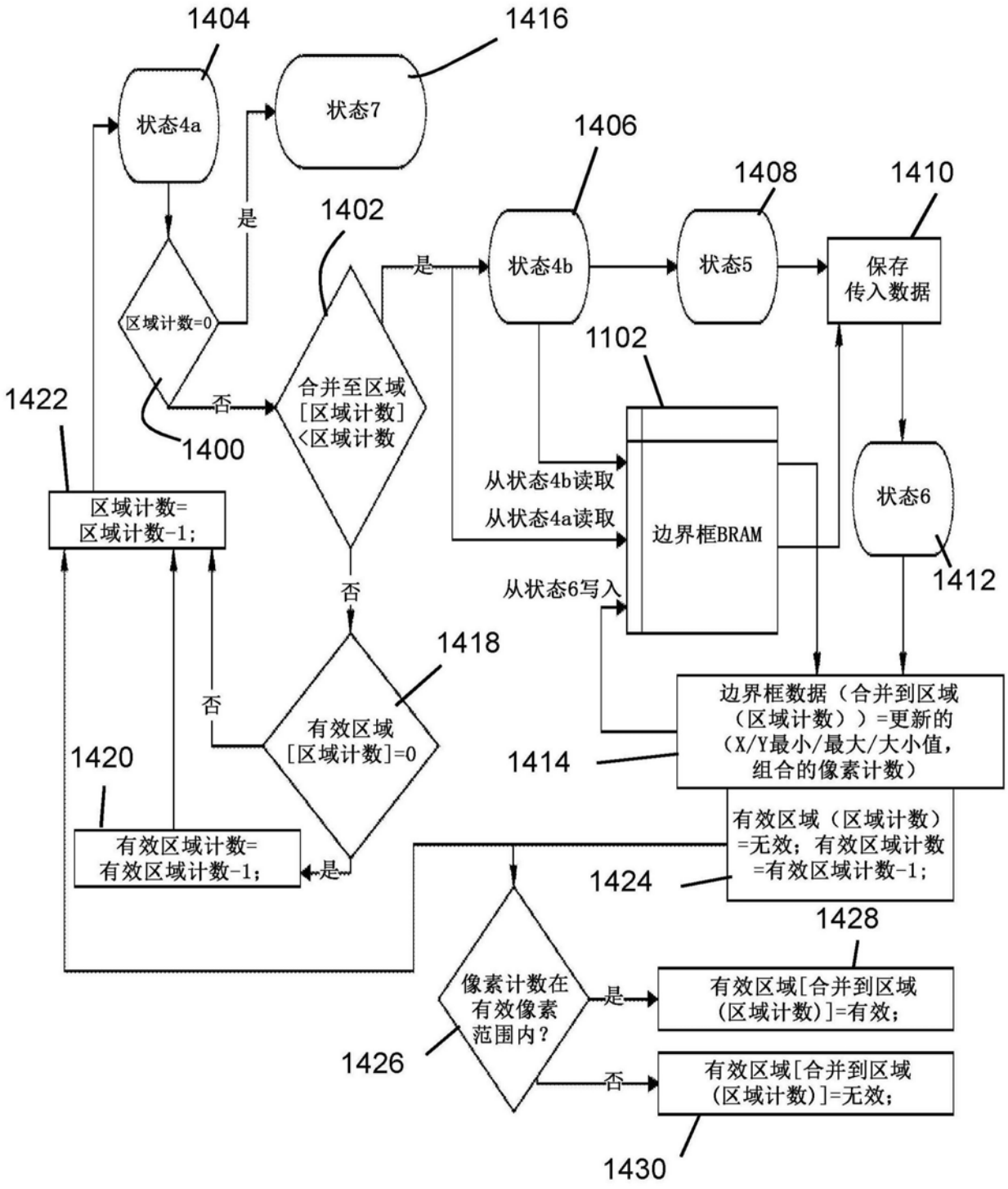

1404
状态4a
1416
状态7
1406
1408
1410
状态4b
状态5
保存
传入数据
1402
是
区域计数=0
是
否
合并至区域
[区域计数]
<区域计数
1400
1102
1422
区域计数=
区域计数-1;
从状态4b读取
从状态4a读取
从状态6写入
边界框BRAM
状态6
1412
否
1418
否
有效区域
[区域计数]=0
1414
边界框数据（合并到区域
（区域计数））=更新的
（X/Y最小/最大/大小值，
组合的像素计数）
1420
有效区域计数=
有效区域计数-1;
是
有效区域（区域计数）
=无效；有效区域计数
=有效区域计数-1;
1424
1428
像素计数在
有效像素
范围内？
是
有效区域[合并到区域
（区域计数）]=有效；
1426
否
有效区域[合并到区域
（区域计数）]=无效；
1430

图14

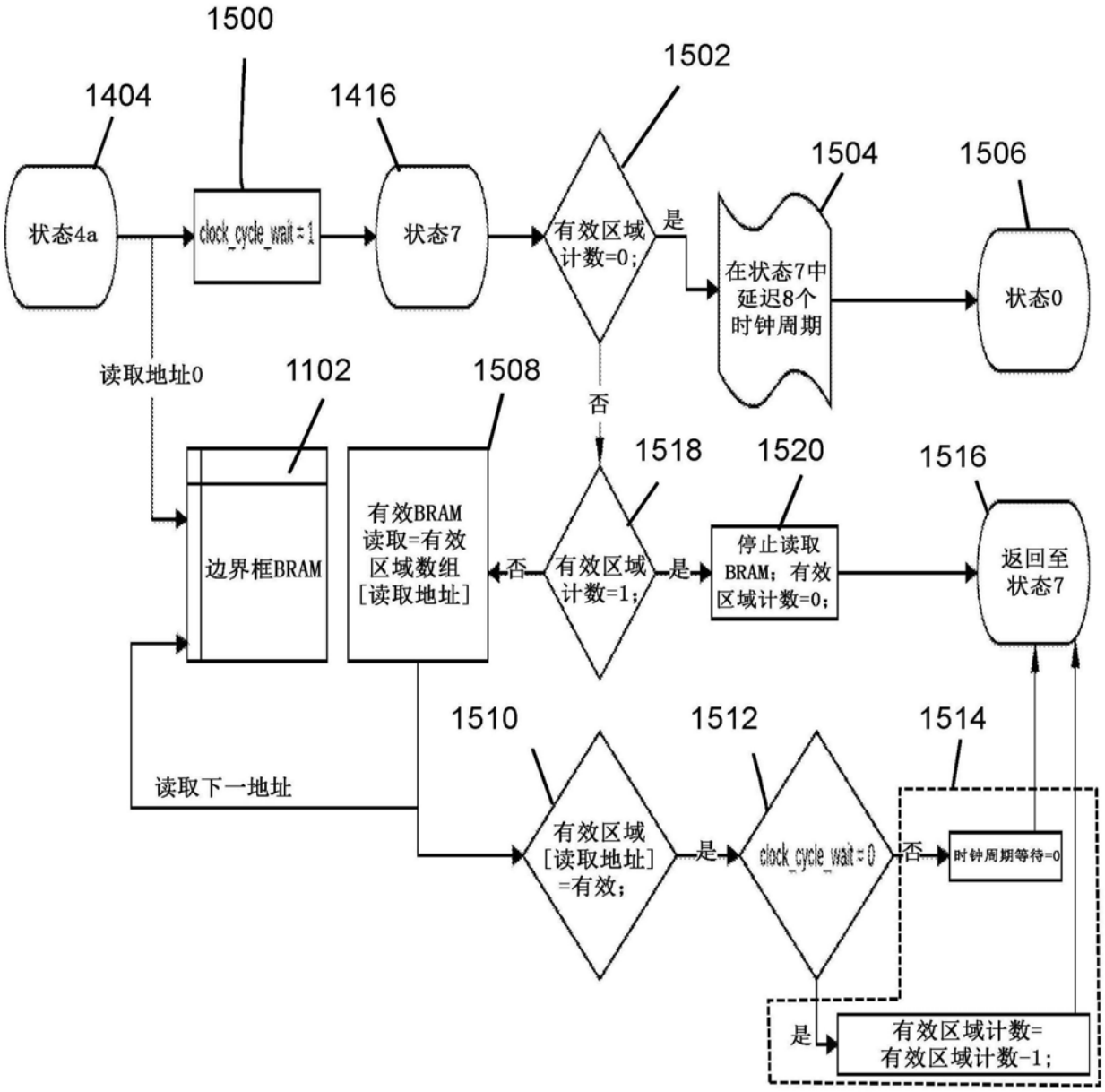

1500
1404
1416
1502
1504
1506
状态4a
clock_cycle_wait=1
状态7
有效区域计数=0;
是
在状态7中延迟8个时钟周期
状态0
读取地址0
1102
1508
否
1518
1520
1516
边界框BRAM
有效BRAM读取=有效区域数组[读取地址]
否
有效区域计数=1;
是
停止读取BRAM；有效区域计数=0;
返回至状态7
1510
1512
1514
读取下一地址
有效区域[读取地址]=有效;
是
clock_cycle_wait=0
否
时钟周期等待=0
是
有效区域计数=有效区域计数-1;

图15

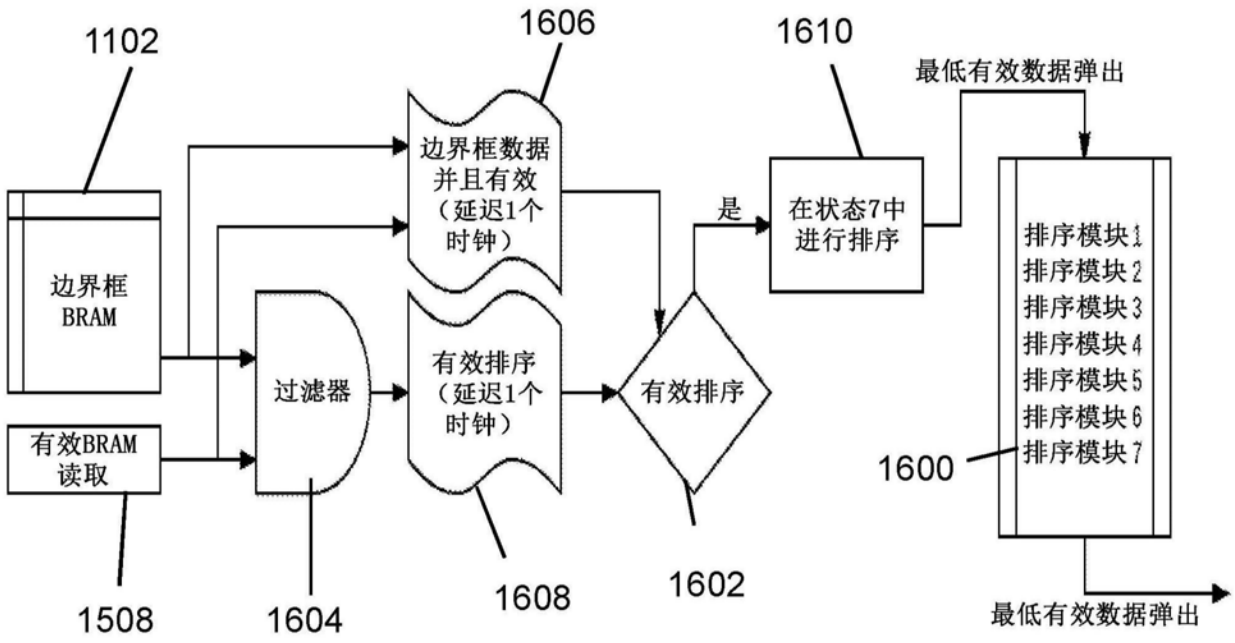

1102
1606
1610
最低有效数据弹出
边界框数据
并且有效
（延迟1个
时钟）
是
在状态7中
进行排序
边界框
BRAM
过滤器
有效排序
（延迟1个
时钟）
有效排序
排序模块1
排序模块2
排序模块3
排序模块4
排序模块5
排序模块6
排序模块7
1600
有效BRAM
读取
1508
1604
1608
1602
最低有效数据弹出

图16

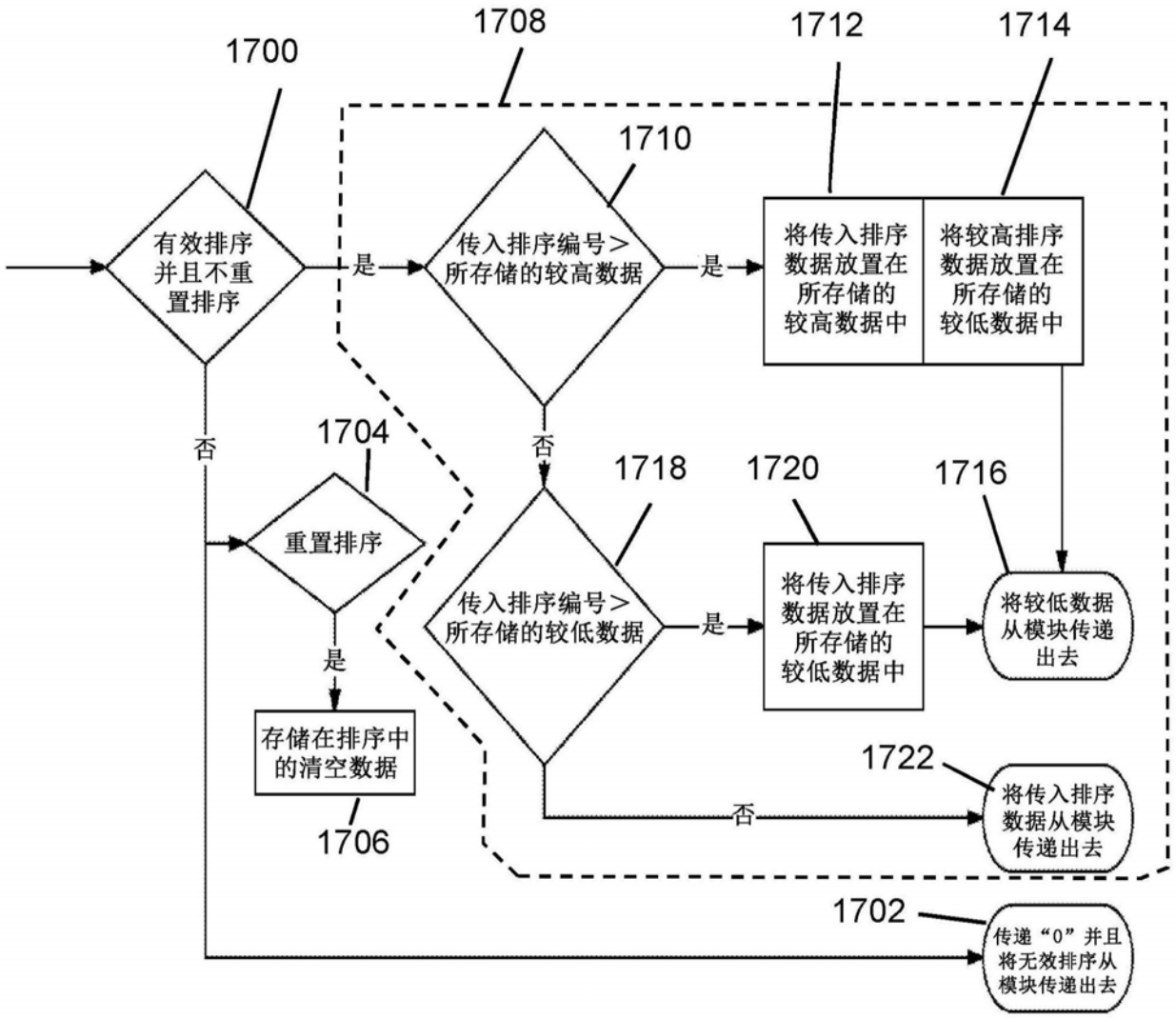

1700
1708
1712
1714
1710
有效排序并且不重置排序
是
传入排序编号>所存储的较高数据
是
将传入排序数据放置在所存储的较高数据中
将较高排序数据放置在所存储的较低数据中
否
否
1704
1718
1720
1716
重置排序
传入排序编号>所存储的较低数据
是
将传入排序数据放置在所存储的较低数据中
将较低数据从模块传递出去
是
存储在排序中的清空数据
1722
否
将传入排序数据从模块传递出去
1706
1702
传递“0”并且将无效排序从模块传递出去

图17

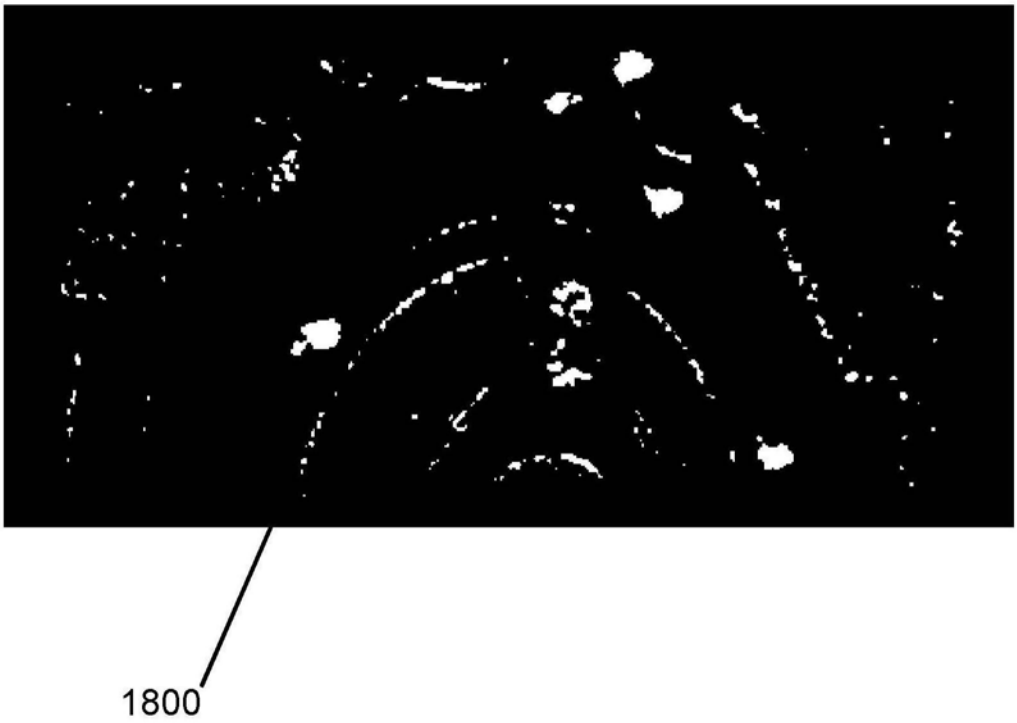


图18

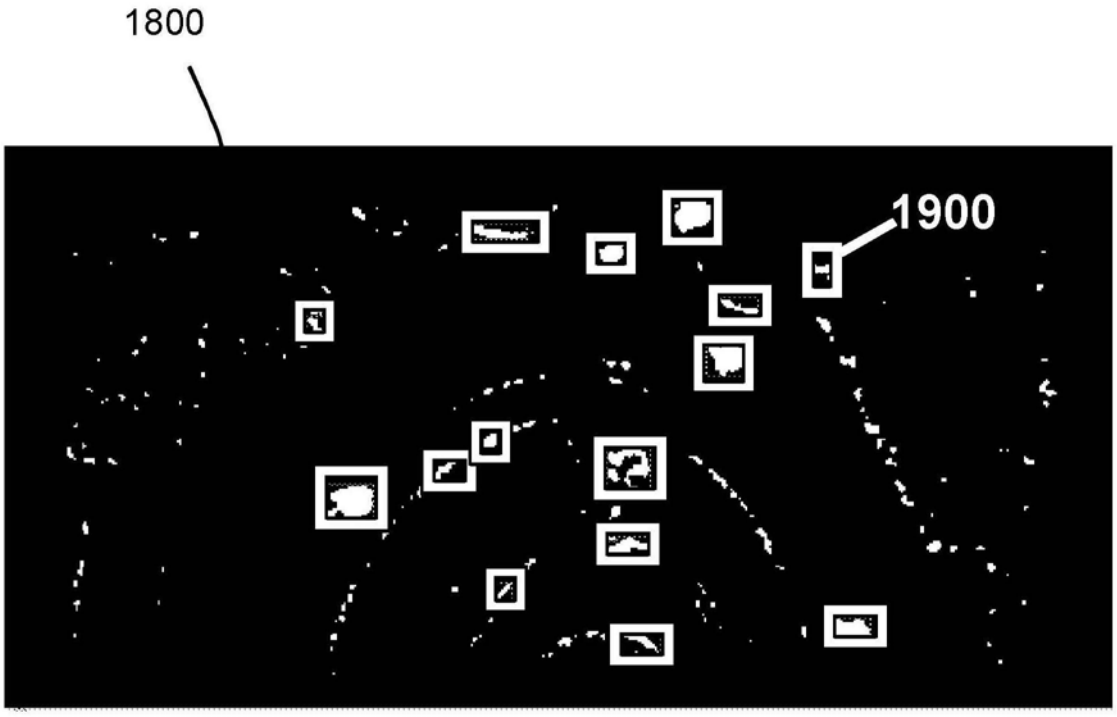


图19

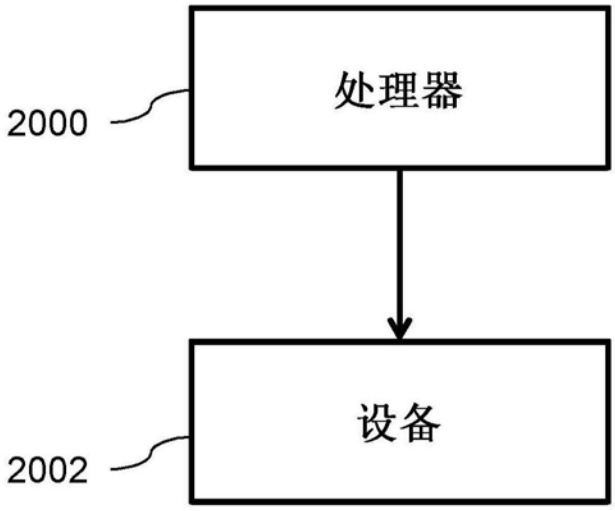


图20