[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06K 9/00 (2006.01)
H04N 7/14 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200580041468.4

[43] 公开日 2007 年 11 月 7 日

[11] 公开号 CN 101069191A

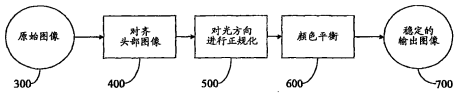
[22] 申请日 2005.11.4
[21] 申请号 200580041468.4
[30] 优先权
[32] 2004.12.2 [33] GB [31] 0426523.7
[86] 国际申请 PCT/GB2005/004261 2005.11.4
[87] 国际公布 WO2006/059060 英 2006.6.8
[85] 进入国家阶段日期 2007.6.1
[71] 申请人 英国电讯有限公司
地址 英国伦敦
[72] 发明人 杰里米·迈克尔·索恩
[74] 专利代理机构 北京三友知识产权代理有限公司
代理人 孙海龙
权利要求书 3 页 说明书 29 页 附图 5 页

[54] 发明名称

视频处理

[57] 摘要

本发明涉及视频处理。公开了一种处理视频数据(300)的方法。视频数据(300)包括多个帧数据组并且是由移动视频数据拍摄装置拍摄的。所述方法包括：(a)在视频的各帧中寻找面部(400)；以及(b)处理对应的帧数据组以：(i)将图像中被面部占据的区域保持为基本恒定(400)；并(ii)将入射到面部上的光的视方向保持为基本恒定(500)；并且/或者(iii)将面部的表观颜色保持为基本恒定。



1、一种处理由移动视频数据拍摄装置拍摄的视频数据的方法，所述视频数据包括多个帧数据组，所述方法包括以下步骤：

(a) 在所述视频的各帧中寻找面部；以及

(b) 处理对应的帧数据组以：

（i）将图像中被所述面部占据的区域保持为基本恒定；并

（ii）将入射到所述面部上的光的视方向保持为基本恒定；并且/或者

（iii）将所述面部的表观颜色保持为基本恒定。

2、根据权利要求 1 所述的方法，其中，所述寻找面部的步骤包括识别所述面部上的一个或更多个特征。

3、根据权利要求 2 所述的方法，其中，所述识别一个或更多个特征的步骤包括：

对所述帧数据的区域与存储的特征模板进行比较，各个所述特征模板包括面部特征的图像并且在大小上与所述区域相对应；以及

通过识别所述帧数据的区域与所述模板之一之间的匹配，来识别各个特征。

4、根据权利要求 3 所述的方法，其中，所述对区域进行比较的步骤包括对区域中的各像素与其在模板中的对应像素进行比较，所述方法还包括：产生对区域中的像素与其在存储的眼睛模板中的对应像素进行的各比较的得分，并且通过选择具有最高得分的像素来识别特征。

5、根据权利要求 2 到 4 中任一项所述的方法，其中，所述特征包括所述面部上的眼对。

6、根据权利要求 5 所述的方法，该方法还包括：检查所述识别出的眼对中的各只眼睛之间的距离和旋转角度。

7、根据任一前述权利要求所述的方法，其中，所述保持入射到所述面部上的光的视方向的步骤包括：

对所述帧数据进行低通滤波；

从所述帧数据中减去所述低通滤波的版本的所述帧数据；以及

将低通滤波版本的先前存储的参考帧数据与所述低通滤波版本的所述帧数据相加，其中所述先前存储的参考帧数据包括在中性光照下的所述面部的图像。

8、根据权利要求 7 所述的方法，其中，低通滤波是通过将所述帧数据与预定义的滤波核进行卷积而实现的。

9、根据任一前述权利要求所述的方法，其中，所述保持所述面部的表观颜色的步骤包括：单独调整所述帧数据的各个颜色通道以将各个颜色通道的相对偏移和绝对值保持为基本恒定。

10、根据权利要求 9 所述的方法，该方法还包括：将所述帧数据的各个颜色通道中的平均像素值移位预定的移位量。

11、根据权利要求 10 所述的方法，其中，对于选择的颜色通道，所述移位量对应于所述帧数据中的所述选择的通道的平均像素值与包括中性光照下的所述面部的图像的先前存储的参考帧数据中的所述选择的通道的平均像素值之间的差。

12、一种视频处理装置，该视频处理装置包括：

其中记录有处理器可读代码的存储介质，所述代码可用于处理由移动视频数据拍摄装置拍摄的视频数据，所述视频数据包括多个帧数据组，所述代码包括：

识别代码，可用于在所述视频的各帧中识别面部；和

帧数据处理代码，可用于处理对应的帧数据组以：

（i）将图像中被所述面部占据的区域保持为基本恒定；并

（ii）将入射到所述面部上的光的视方向保持为基本恒定；并且/或者

（iii）将所述面部的表观颜色保持为基本恒定。

13、一种便携式装置，该便携式装置包括：

视频数据拍摄装置，所述视频数据包括多个帧数据组；

存储介质，其中记录有可用于处理所述拍摄的视频数据的处理器可读代码，所述代码包括：

识别代码，可用于在所述视频的各帧中识别面部；和

帧数据处理代码，可用于处理对应的帧数据组以：

（i）将图像中被所述面部占据的区域保持为基本恒定；并

（ii）将入射到所述面部上的光的视方向保持为基本恒定；并且/或者

（iii）将所述面部的表观颜色保持为基本恒定。

14、根据权利要求 13 所述的便携式装置，所述便携式装置包括手持式装置。

15、一种载有指令程序的数字数据载体，所述指令程序可被处理装置运行以执行根据权利要求 1 到 11 中任一项所述的方法步骤。

视频处理

技术领域

本发明涉及一种用于处理视频数据的方法和设备，更具体地讲，涉及对在移动视频数据拍摄装置中拍摄的视频数据进行处理。

背景技术

视频会议是位于不同地点但是能够利用电子通信彼此看见和听见的两人或更多人之间的讨论。提供视频会议的一个目的是通过传送会话的更多的非语言方面来增强会议效果。假定对这些方面主要起作用的因素是会议参与者的面部表情的样子，则期望尽可能清楚且一致地呈现这些表情。

在移动视频会议中，视频会议参与者中的至少一个使用装备有移动通信装置的摄像机，从而能够在保持在该摄像机的视野内的同时移动。

由装备有移动通信装置的摄像机拍摄的视频帧中的视频会议参与者的外貌在位置、大小、朝向、光照方向和颜色方面易变化。

由于参与者和摄像机之间的相对全局移动，导致位置、大小和朝向变化。由于诸如背景减除（background subtraction）和帧差（frame differencing）的用于检测移动物体的传统技术假定背景是静止的，而如果用户自己在移动的话背景就不是静止的，所以这些传统技术不起作用。

当会议参与者移动通过具有许多光源的环境或者使移动通信装置旋转时，由于落在面部上的入射光的变化或者从彩色物体反射的光的结果，光落在参与者的面部上的方向会发生实质变化并且面部的颜色也会发生实质变化。这导致了组成面部图像的像素的强度和颜色的逐渐但实质性的变化。如果随后对原始视频数据进行编码以满足低“最大比特率”需求，则在对这些变化进行编码与面部表情的清晰度之间需要进行权衡，结果，这些变化会减小面部表情的清晰度。

另外，在当前的装备有移动通信装置的摄像机中，有时候对图像应用白平衡补偿。白平衡补偿寻求将图像的红色、绿色和蓝色(RGB)通道的增益调整为使这些通道的总亮度相等。这会导致面部（其仅形成图像的一部分）显得绿或蓝。

发明内容

根据本发明的第一方面，提供了一种处理由移动视频数据拍摄装置拍摄的视频数据的方法，所述视频数据包括多个帧数据组，所述方法包括以下步骤：

(a) 在所述视频的各帧中寻找面部；以及

(b) 处理对应的帧数据组以：

（i）将图像中被所述面部占据的区域保持为基本恒定；并

（ii）将入射到所述面部上的光的视方向保持为基本恒定；并且/或者

（iii）将所述面部的表观颜色保持为基本恒定。

通过在由移动视频数据拍摄装置拍摄的视频数据的帧中识别面部并且将所述帧数据进行变换以补偿面部的移动、面部大小的变化、入射到面部上的光的方向的变化以及面部颜色的变化，可以减少面部的外貌在帧之间的变化并且使面部表情的清晰度最大化。此外，如果不调整光方向和颜色，则在发送所识别的面部区域的变化时就会浪费掉许多位，这些位对摆出的面部表情没有贡献，因此消耗了更多的处理能力和/或导致无法最佳地利用带宽。

术语“颜色”旨在包括彩色颜色（具有色调（hue）的颜色，色调是颜色的一种性质，通过该性质能够在按光的主波长确定的从红到紫的范围内感知到这些颜色）和非彩色颜色（没有色调的颜色，即白色、灰色和黑色）。

优选的是，寻找面部的步骤包括识别所述面部上的一个或更多个特征，这包括：对所述帧数据的区域与存储的特征模板进行比较，各个所述特征模板包括面部特征的图像并且在大小上与所述区域相对应；以及

通过识别所述帧数据的区域与所述模板之一之间的匹配来识别各个特征。优选的是，所述特征包括所述面部上的眼对。眼睛能够良好地用于模板匹配，这是因为当从不同角度观看时眼睛的形状保持为相对静止，还因为在所述模板内有大量的对比。

优选的是，所述方法还包括：检查所述识别的眼对中的各只眼睛之间的距离和旋转角度。这样，能够拒绝无效的眼对（因为它们离得太远或者它们之间旋转角度太大）。

优选的是，保持入射到所述面部上的光的视方向的步骤包括：对所述帧数据进行低通滤波；从所述帧数据中减去所述低通滤波的版本的所述帧数据；以及将低通滤波版本的先前存储的参考帧数据与所述低通滤波版本的所述帧数据相加，其中所述先前存储的参考帧数据包括在中性光照下的所述面部的图像。通过加入低通滤波版本的先前存储的参考帧数据，得到的帧看起来更自然。

在权利要求中限定了本发明的其他方面。

附图说明

现在将参照附图仅作为示例来描述本发明的实施例，其中，相似的附图标记表示相似的部分，并且在附图中：

图 1 是通用计算机系统的系统组件的系统框图的示图；

图 2 是指示用户头部的移动方向的示图；

图 3 是示出了图像处理方法的操作的流程图；

图 4 是示出了图像处理方法的对齐头部图像阶段的操作的流程图；

图 5 是示出了图像处理方法的对齐头部图像阶段的跟踪头部阶段的操作的流程图；

图 6 是示出了图像处理方法的对齐头部图像阶段的跟踪头部阶段的全局搜索阶段的操作的流程图；

图 7 是示出了图像处理方法的对光方向进行正规化阶段的操作的流程图；以及

图 8 是示出了图像处理方法的颜色平衡阶段的操作的流程图。

具体实施方式

现在将参照图 1 来描述在本发明的一些实施例中使用的通用计算机系统。其他的实施例可能使用手持式装置、笔记本电脑、大型计算机、微型计算机、多处理器系统、分布式系统等。考虑到可以进行编程以实现本发明的硬件的范围广，下面一般将本发明的操作描述为由该计算机系统的计算机执行的计算机可执行指令，诸如程序模块。这些程序模块可以包括执行任务或实现特定抽象数据类型的进程、程序、对象、组件、数据结构、数据变量等。在分布式计算环境中，可将多个计算机系统连接到通信网络，并且可将本发明的个体程序模块分布在这些计算机系统之中。

本领域公知的通用计算机系统（图 1）包括：台式主机（tower base）单元 100，其中包含有计算机处理单元、主板、一个或多个硬盘驱动器、系统存储器；和光盘驱动器 110，其能够从诸如 CD、CD-R、CD-RW、DVD 等的可移动光盘读取和/或对其写入。此外，主机单元 100 还容纳有能够从磁性软盘接收并读取和/或对其写入的磁性软盘驱动器 112。

应该明白，图 1 仅示出了示例性计算机系统，并且可以有能够与本发明一起使用的其他结构的计算机系统。具体地讲，主机单元 100 可以是台式结构，或者另选地将该计算机系统实现在膝上型、笔记本型或手持型结构中，该计算机系统就可以是便携式的。

该计算机系统的内部组件包括主板，在主板之上安装有系统存储器 118，该系统存储器 118 自身包括随机存取存储器 120 和只读存储器 130。此外，设置有系统总线 140，该系统总线 140 将包括系统存储器 118 在内的各种系统组件与处理单元 152 相连接。系统总线 140 还连接有：软盘驱动器接口 156，其控制软盘驱动器 112 以从插入其中的任一软盘读取数据或将数据写入该软盘；光盘驱动器接口 160，其控制光盘驱动器 110 以从插入其中的可移动光盘读取数据以及将数据写入该可移动光盘；红外(IR)端口接口 153，其控制 IR 端口 116；以及蓝牙（Bluetooth™）卡接口 155，其控制蓝牙 PC 卡 118。IR 端口 116 和蓝牙 PC 卡 118 使该计算机系

统可以与其他相似实现的装置进行通信。还将网卡等形式的网络接口 162 连接到系统总线 140，网络接口 162 被布置为使该计算机系统可以通过网络 190 与其他计算机系统进行通信。网络 190 可以是局域网、广域网、局域无线网等。具体地讲，可以具体使用 IEEE 802.11 无线 LAN 网络以使计算机系统可以移动。网络接口 162 使该计算机系统可以通过网络 190 与诸如服务器、路由器或对等级别计算机的其他计算机系统形成逻辑连接，以进行程序或数据的交换。

关于网络接口 162，虽然前面已经描述了为何其优选地为无线 LAN 网卡，但是同样也应该理解，该计算机系统可以设置有调制解调器，该调制解调器安装到串行或并行端口接口(未示出)，并且被布置为通过公共交换电话网(PSTN)形成从该计算机系统到其他计算机的逻辑连接。

在网络环境中使用该计算机系统的情况下，还应该明白，可以本地存储在该计算机系统中的应用程序、其他程序和其他数据还可以另选或另外地存储在远程计算机上，并且由该计算机系统通过在网络 190 上形成的逻辑连接对其进行访问。

此外，还设置有连接到系统总线 140 的硬盘驱动器接口 166，该硬盘驱动器接口 166 控制从硬盘驱动器 168 读取数据或程序或者将数据或程序写入硬盘驱动器 168。硬盘驱动器 168、光盘驱动器 110 使用的光盘、或者软盘驱动器 112 使用的软盘全都提供对该计算机系统的计算机可读指令、数据文件、程序模块以及其他数据的非易失性存储。尽管这里描述了这三种特定类型的计算机可读存储介质，但是预期的读者应该明白，还可以使用能够存储数据的其他类型的计算机可读介质，具体地讲为盒式磁带、闪存卡、磁带存储驱动器、数字多功能盘等。

诸如硬盘驱动器 168、或任何软盘或光盘的计算机可读存储介质中的每一个都可以存储多种程序、程序模块或数据。具体地讲，本实施例中的硬盘驱动器 168 具体地存储有应用程序 175、应用程序数据 174、计算机系统 1 或用户所需的其他程序 173、诸如 Microsoft®Windows®、Linux™、Unix™等的计算机系统操作系统 172 以及其他用户数据 171。该硬盘驱动器 168 提供对上述程序和数据的非易失性存储，从而在没有

电的情况下也能永久地存储这些程序和数据。

为了使该计算机系统运行这些应用程序或者利用存储在硬盘驱动器168或其他计算机可读存储介质上的数据进行工作，系统存储器118设置有随机存取存储器120，当该计算机系统需要时，该随机存取存储器120为应用程序、程序数据、其他程序、操作系统和用户数据提供存储器存储。当这些程序和数据被加载到随机存取存储器120中时，存储器的特定部分125将保持应用程序，另一部分124可以保持程序数据，第三部分123可以保持其他程序，第四部分122可以保持操作系统，第五部分121可以保持用户数据。应该明白，该计算机系统可以按需要将各种程序和数据移入和移出随机存取存储器120。更具体地讲，在一程序或数据不是正被该计算机系统使用的情况下，可能不将该程序或数据存储在随机存取存储器120中，而是将其返回到硬盘168上的非易失性存储器。

该系统存储器118还设置有只读存储器130，该只读存储器130为基本输入输出系统(BIOS)提供存储器存储，该BIOS包括用于在该计算机系统内的系统组件之间传送信息的基本信息和命令。BIOS在系统启动时是必不可少的，用以提供关于各种系统组件如何彼此通信的基本信息并且使系统可以引导启动（boot-up）。

在本发明的第一实施例中，提供了一种视频通信系统。该通信系统包括已经描述过的计算机系统和诸如Nokia 3650移动电话的便携手持式装置199。

当接通时，该计算机系统运行存储在硬盘168上的Microsoft® Windows®2000操作系统程序。本领域技术人员可以理解，该操作系统程序支持包括Microsoft® DirectShow应用编程接口(API)的DirectX 8.1。

装置199包括能够拍摄视频并且根据国际电信联盟(ITU)的H.263标准对该视频进行编码的摄像机。拍摄的视频作为3GPP多媒体文件被存储在移动装置199上，该3GPP多媒体文件是由第三代合作伙伴项目定义的文件标准。装置199可以使用已经预装在该计算机系统1上并存储在硬盘驱动器168上的软件通过红外线或蓝牙与该计算机系统交换信息。存储在该计算机系统上的还有称作mov2avi的免费软件工具， mov2avi利

用苹果计算机公司的 QuickTime 软件将（从移动装置 199 接收的）3GPP 多媒体文件转换成能够使用在 DirectShow API 中提供的功能进行读取的 AVI 文件。

安装在该计算机系统上的还有 OpenCV 软件库和图像处理库(IPL)，它们均来自 Intel®公司，2200 Mission College Blvd., Santa Clara, CA 95052, USA。

在本实施例中，将图像处理程序提供在 CD ROM 上，并使用光盘驱动器 110 将其安装在该计算机系统上并存储在硬盘驱动器 168 上。程序代码是由 C++编程语言写成的，并且当该程序代码被编译和运行时，对表示由装置 199 的摄像机拍摄的视频数据的 AVI 文件进行处理。

现在将结合图 3 到图 8 来描述该图像处理程序。在下面的描述中，作出如下假设：（1）用户正在观看摄像机；（2）用户的两只眼睛都是可见的；（3）用户的头的倾斜（pitch）、转动（roll）和摇摆（yaw）较小（即，小于 10°）；以及（4）摄像机保持在正常的观看距离（即，15 至 50cm）。参照图 2 来定义用户的头的倾斜、转动和摇摆。倾斜与点头有关，转动与使耳朵向肩膀移动有关，而摇摆与摇头有关。

下面给出了该图像处理程序（图 3）的主循环的伪代码：

```
mainLoop()
{
setup()

state:trackState
state.eyesInitialised=false

while(Iquit)
    {
    captureImage(rawImage)
    mobileFacePreprocessor(rawImage,state,stabilisedCImage,stabilisedECImage)
    doSomething(stabilisedCImage,stabilisedECImage)
```

```
    }
}
```

换言之，参照图 3，该图像处理程序 3 涉及将原始图像数据 300 处理成稳定的输出图像数据 700。原始图像数据 300 还没有被图像处理程序 3 处理，其在这种意义上是原始的，并且原始图像数据 300 包括一组视频帧数据，该组视频帧数据被装置 199 拍摄、以 3GPP 文件的形式进行存储、传递到计算机系统 1 并且转换成 AVI 文件。然后使用 DirectShow API 从该 AVI 文件读取帧，这些帧形成原始图像数据 300。

在主循环中调用的 setup 函数用于加载和初始化在主函数（mobileFacePreprocessor（））中使用的一些变量。下面将更加详细地描述在 setup 函数内调用的各种函数。

然后在三个阶段 400、500、600 中对原始图像数据 300 进行处理。第一阶段在原始图像数据的每帧中寻找并对齐头部的图像（400），第二阶段将落在该头部的面部上的光的方向正规化（500），第三阶段校正图像的彩色平衡（600）。下面给出了该图像处理程序（图 3）中的主函数（mobileFacePreprocessor（））的伪代码示例：

```
mobileFacePreprocessor(in rawImage:image, inout t:trackstate,
out stabilisedECImage:image, out stabilisedCImage:image)
{
alignHeadImage(rawImage,t,extremeCloseupImage,closeupImage)
normaliseLightDirection(extremeCloseupImage,extremeCloseupBlurNeutral,balancedECImage)
normaliseLightDirection(closeupImage,closeupBlurNeutral,balancedCImage)
colourBalance(balancedECImage,stabilisedECImage)
colourBalance(balancedCImage,stabilisedCImage)
}
```

下面将更加详细地描述这些阶段中的每个阶段。

对齐头部图像（400）

现在参照图 4 来描述对齐原始图像数据的每帧中的头部图像的阶段（400）。对齐头部图像包括两个阶段：（i）跟踪头部（401）；以及（ii）

对原始图像数据 300 进行仿射卷绕（affine warp）(线性变换和旋转的组合)和修剪（crop），以使头部在所有帧上保持相同的位置、朝向和标度（450）。下面给出了对齐头部图像 400 阶段的伪代码示例：

```
alignHeadImge(in rawImage:image, inout t:trackState,
out extremeCloseupImage:image, out closeupImage:image)
{
trackHead(rawImage,t)

//取得极限特写（closeup）
desiredEyeWidth:int=0.6*extremeCloseupImage.width
desiredRightEyeLocation:point=(0.2*extremeCloseupImage.width,0.3*extremeCloseupImage.wi
dth)
affineWarpAndCrop(rawImage,t.eyeLocations,desiredEyeWidth,desiredRightEyeLocation,extre
meCloseupImage)

//取得特写
desiredEyeWidth=0.4*closeupImage.width
desiredRightEyeLocation=(0.3*closeupImage.width,0.41*closeupImage.height)
affineWarpAndCrop(rawImage,t.eyeLocations,desiredEyeWidth,desiredRightEyeLocation,close
upImage)
}
```

现在将更加详细地描述各个阶段。

跟踪头部（401）

跟踪是跟随用户的头部的某些特征的跨帧的运动的重复处理。作为另一选择，可以执行在原始图像中的所有像素中寻找期望的特征的新搜索。然而，跟踪较好，这是因为在给定了初始估计的情况下，可以仅围绕该估计进行搜索，因此使用较少的计算处理资源。在本实施例中，通过跟踪用户的眼睛的运动来实现对头部的跟踪。如果在被检查的小区域内无法找到特征，则跟踪会失败。例如，如果特征被遮挡（被一些其他

物体（例如用户的手）遮掩）或者如果特征移到正在检查的区域之外（例如，摄像机的突然移动使特征在帧内剧烈移动），则这会发生。

在本实施例中，通过下面的伪代码来描述跟踪头部（401）的方法：

```
trackHead(in rawImage:image,inout t:trackState)
{
lostTrack:boolean=false
if(t.eyesInitialised)
    {
    updateKalmanByTime(t)
    localSearch(rawImage,t.predictedEyeLocation,eyeTemplates,newEyeLocations)
    if(validPair(newEyeLocation,rawImage.width))
        {
        t.measuredEyeLocations=newEyeLocations
        updateKalmanByMeasurement(t)
        }
    else
        {
        lostTrack=true
        }
    }
else
    {
     lostTrack=true
    }
if(lostTrack)
    {
    globalSearch(rawImage,eyeTemplates,newEyelLocations)
    if(validPair(newEyeLocations,rawImage.width))
        {
```

```
            t.measuredEyeLocations=newEyeLocations
            t.eyesInitialised=true
            initialiseKalman(t)
            }
        else
            {
            t.eyesInitialised=false
            //t.eyeLocations 保持在最后的位置
            }
    }
}
```

换言之，参照图 5，该处理通过以下步骤开始：检查是否存在眼睛的初始位置（403），即针对前一帧是否已经找到眼对的坐标。如果该检验的结果是肯定的，即针对前一帧已经找到眼对的坐标，则可以跟踪头部，并且使用先前找到的坐标作为初始估计在当前帧中对用户的眼睛执行局部搜索（405）。另一方面，如果所述结果是否定的，则对头部的跟踪失败，并且代替地在整个原始图像中对用户的眼睛进行全局搜索（407）。下面将更加详细地描述局部搜索(405)和全局搜索(407)。

局部搜索 405

在本实施例中，通过下面的伪代码来描述进行局部搜索（405）的方法：

```
localSearch(in rawImage:Image, in oldEyeLocations:eyeLocations, in eyeTemplates:imagePair,
out newEyeLocations:eyeLocations)
{
localSearch2(rawImage,oldEyeLocations.left,eyeTemplates.left,newEyeLocations.left)
localSearch2(rawImage,oldEyeLocations.right,eyeTemplates.right,newEyeLocations.right)
}
localSearch2(in rawImage:image, in oldEyeLocation:point, in eyeTemplate:image, out
newEyeLocation:point)
```

```
{
searchRange:point=(25+template.width/2,25+template.width/2)
//25 是原始图像宽度的函数。25=320/12.8=rawImage.width/12.8
rawImage.roi=createROI(oldEyeLocation-
searchRange,oldEyeLocation+searchRange)copyImage(rawImage,region)
colorToGrey(region, greyRegion)
matchTemplate(greyRegion,template,match)
locateMaximum(match,newEyeLocation)
}
```

对用户的左眼和右眼都进行局部搜索，并且使用正规化互相关，通过模板匹配来进行该局部搜索。

模板匹配包括对包含关注特征的已知示例的模板与原始图像数据中的区域进行比较以寻找匹配。在本实施例中，预先存储了两个模板，一个模板是先前拍摄的用户的左眼图像（eyeTemplate.left），用于检测左眼在原始图像数据中的位置，而另一个模板是先前拍摄的用户的右眼图像（eyeTemplate.right），用于检测右眼在原始图像数据中的位置。这两个模板根据在装置 199 上拍摄的静止图像创建，因此在外观和大小上与正在检查的帧内所期望的非常相似。正规化互相关是一种用于为原始图像的区域和模板之间的比较产生得分的数学技术。

首先基于存储的模板的大小来建立方形的搜索范围，然后使用 IPL 软件库的 createROI 函数来选择原始图像数据中的关注区域。这个区域从先前找到的眼坐标偏移了该搜索范围。模板匹配是对该图像的灰度版本（单个 2D 整数数组（即单通道），每个整数表示一位置（x,y）处的强度）进行的，因此随后使用 IPL 软件库的 colorToGrey 函数将该区域转换成灰度图像。（该模板可以存储为灰度图像或者可以当需要时将其转换成灰度图像）。然后使用 OpenCV 软件库的 cvMatchTemplate 函数在整个灰度区域上执行灰度模板的模板匹配和正规化互相关。将新的眼位置（用户的左眼和右眼的坐标组）取为该区域中具有最高比较得分的位置，并且这是使用 OpenCV 软件库的 cvMinMaxLoc 函数计算出的。

然后对从局部搜索获得的眼对坐标进行测试以检验它们是否有效（409）。即，检查左眼和右眼的相对位置以确保它们不是分离得太远或靠得太近并且它们之间的旋转角小。在本实施例中，通过下面的伪 C＋＋代码来描述执行有效眼对测试（409）的方法：

```
validPair(eyeLocations:eyeLocations,imageWidth:int):boolean
{
  edx:double=eyeLocations.left.x-eyeLocations.right.x;
  edy:double=eyeLocations.left.y-eyeLocations.right.y;
  ed:double=sqrt(edx*edx+edy*edy);
  ea:double=tan-1(edy/edx);
  badPair:boolean=(ed<0.1*imageWidth‖ed>0.4*imageWidth‖ea>0.5‖ea<-0.5)
  return(!badPair)
}
```

在本实施例中，如果左眼和右眼之间的距离 ed 小于 0.1*imageWidth 或大于 0.4*imageWidth(其中，ed 和 imageWidth (图像的宽度)是按像素测量的)，或者头部的转动 ea 大于 0.5 或小于-0.5（其中，ea 是按弧度测量的），则认为该眼对无效。然而，这些限制将根据装置 199 中的摄像机的视场而变化。如果发现该眼对无效，那么对头部的跟踪失败，并且代替地执行在整个原始图像中对用户的眼睛的全局搜索（407）。如果发现该眼对有效，则随着找到眼对坐标组 413，头部跟踪阶段 401 完成。

在局部搜索 405 期间，除了眼对的坐标以外，还可以考虑针对前一帧找到的坐标在先前帧的序列中观察到的“速度”。这具有在局部搜索期间改进眼对的估计位置的效果。可以使用 Kalman 滤波器（提供用于估计一过程的状态的有效计算手段的一组数学等式）将该功能增加到局部搜索，并且在本实施例中，通过 OpenCV 软件库的 cvKalmanUpdateByTime 和 cvKalmanUpdateByMeasurement 函数来提供该功能。

全局搜索 407

在本实施例中，通过下面的伪代码来描述进行全局搜索（407）的方法：

```
globalSearch(in rawImage:image, in eyeTemplates:imagePair,
out newEyeLocations:eyeLocations)
{
  colorToGrey(rawImage,grey)

  //templateMatch(rawImage,template,match)
  matchTemplate(grey,eyeTemplate.left,match.left)
  matchTemplate(grey,eyeTemplate.right,match.right)

  //blur(match,blur)
  blur(match.left,blur.left)
  blur(match.right,blur.right)

  //shift(blur,rawImage.size,shift)
  shift(blur.left,shift.left,-0.17*imageSize.width,0.004*imageSize.height)
  shift(blur.right,shift.right,0.17*imageSize.width,-0.004*imageSize.height)

  //multiply(match,shift,bestGuess)
  multiply(match.left,shift.right,bestGuess.left)
  multiply(match.right,shift.left,bestGuess.right)

  //locateMaxima(bestGuess,newEyeLocation)
  locateMaximum(bestGuess.left,newEyeLocation.left)
  locateMaximum(bestGuess.right,newEyeLocation.right)
}
```

换言之，参照图 6，该处理从如下步骤开始：对右眼和左眼执行的全局模板匹配（419、429）—在整个原始图像数据的灰度版本上与灰度模板进行正规化互相关。与前面相同，使用 OpenCV 软件库的 cvMatchTemplate 函数来执行模板匹配和正规化互相关。可将根据各位置

的互相关而得到的得分看作在该具体位置处存在眼睛的估计概率的映射。

然后可以基于左眼相对于右眼的相对位置的知识来估计左眼的位置，并且还可以基于右眼相对于左眼的相对位置的知识来估计右眼的位置。根据在装置 199 上拍摄的简单视频训练序列的统计（当头部移动足够慢从而不需要进行全局搜索头部跟踪就能成功时）已经示出：用户的左眼位置形成一分布，其从用户的右眼位置偏移了平均眼间隔。因此，可以通过用将（从右眼全局模板匹配 419 获得的）右眼估计与高斯（Gaussian）（用于使图像“模糊”并且去除细节和噪声的算子）进行卷积并且进行偏移，来形成左眼的位置的概率映射。还可以通过将（从左眼全局模板匹配 429 获得的）左眼估计与高斯进行卷积并且进行偏移，来形成右眼的位置的概率映射。这在上述伪代码中由“blur”和“shift”函数表示，而在图 6 中由步骤 421/423 和 431/433 表示。在本实施例中，对于左眼，（使用训练序列计算出的）x 和 y 移位分别是-0.17*ImageWidth 和 0.004*ImageHeight；对于右眼，x 和 y 移位分别是 0.17*ImageWidth 和 -0.004*ImageHeight。

在本实施例中，通过下面的伪代码来描述进行“模糊”的方法：

```
blur(in match:image,out blur:image)
{
convolveSep2D(match,blur,gaussianKernelx,gaussianKernely)
}
```

其中，convolveSep2D 函数是由 Intel IPD 软件库提供的。下面给出了能够用来建立高斯核（Gaussian kernel）的函数的伪代码示例：

```
createGaussianKernels(in filterLength:int,out kernelx:convKernel,out kernely:convKernel)
{
//简单高斯
filterCoeffs:int[filterLength]
m:double=filterLength/2
s:double=filterLength/8
```

```
g:double=255/(s*sqrt(2*pi))
f:double=2*s*s
for(int i=0;i<filterLength;i++ )
        {
        x:double=i-m
        filterCoeffs[i]=(int)(g*exp(-x*x/f))
        }
 kernelx=filterCoeffs
 kernely=filterCoeffs
}
```

为了建立高斯核 gaussianKernelx 和 gaussianKernely，可以作为 setup 函数的一部分如下调用此函数。

```
CreateGaussianKernels(21,gaussianKernelx,gaussianKernely)
//滤波器长度为 21
```

在本实施例中，通过下面的伪代码来描述执行“移位”（即，上述的偏移）的方法：

```
shift(in in:image,in x:int,in y:int,out shift:image)
{
fillImage(shift,0)
if(x>=0&&y>=0)
   {
   in.roi=createROI((0,0),(in.width-x,in.height-y))
   shift.roi=createROI((x,y),(in.width-x,in.height-y))
   }
   else if(x<0&&y>=0)
   {
   in.roi=createROI((-x,0),(in.width+x,in.height-y))
   shift.roi=createROI((0,y),(in.width+x,in.height-y))
   }
```

```
else if(x<0&&y<0)
{
in.roi=createROI((-x,-y),(in.width+x,in.height+y))
shift.roi=createROI((0,0),(in.width+x,in.height+y))
}
else if(x>=0&&y<0)
{
in.roi=createROI((0,-y),(in.width-x,in.height+y))
shift.roi=createROI((x,0),(in.width-x,in.height+y))
}
//从一个关注区域 (roi)复制到另一关注区域
copyImage(in,shift)
}
```

createROI 函数是由 Intel IPD 软件库提供的，fillImage 和 copyImage 函数是由 Intel OpenCV 软件库的 cvFillImage 和 cvCopyImage 函数提供的。

因此，通过模板匹配（419/429）、模糊（421/431）和偏移（423/433），存在两个对右眼位置的估计，一个估计是通过右眼全局模板匹配而获得的（在上述伪代码中称作“match.right”），一个估计是通过对左眼的全局模板匹配进行模糊和偏置而获得的（在上述伪代码中称作“shift.left”）。

每只眼睛的“最佳”位置是模板匹配估计和偏移并模糊的模板匹配估计都良好的位置。可以通过将这两个估计相乘而将其组合，并且这在上述伪代码中由“multiply”函数表示并且在图 6 中由步骤 425/435 表示。在本实施例中，multiply 函数是由 Intel IPL 软件库的 Multiply 函数提供的，该函数执行两个图像 a 和 b 的像素级（pixelwise）乘法以形成新的图像 c，从而 c(x,y)=[a(x,y)*b(x,y)/255]。在上述伪代码中，输入图像称作“match”和“shift”，输出图像称作“BestGuess”。然后使用 Intel OpenCV 软件库的 cvMinMaxLoc 函数来执行对全局最大值的搜索（上述伪代码中的“locateMaximun”函数和图 6 中的阶段 427/437），这会产生最佳的眼对坐标 439。

再次参照图 5，然后对通过全局搜索获得的最佳的眼对坐标进行测试以检查其是否有效(411)。该检查在本质上与如上面关于步骤 409 所描述的对通过局部搜索获得的眼对坐标执行的检查相似。如果发现该眼对有效，则当找到一组眼对坐标 413 时，头部跟踪阶段 401 完成。然而，如果发现该眼对无效，则对头部的跟踪失败 415。在这种情况下，一种选择是不提供这帧数据的输出并且在没有眼睛的初始位置的情况下对下一帧数据重新开始该处理。这导致了在步骤 403 中执行的测试失败，确保执行另一全局搜索 407。另外一种选择是：（例如，通过使用 Kalman 滤波器）相对于最后的已知眼睛位置，估计眼对坐标（417）。

仿射卷绕和修剪（450）

获得眼对坐标 413 之后，可以对原始图像 300 执行仿射卷绕和修剪（450），以使头部在所有数据帧上保持在相同的位置、朝向和标度。在本实施例中，仿射卷绕和修剪仅仅解决头部的 x、y 位置和标度以及转动的变化。它不解决头部的倾斜和摇摆的变化或由于不同的透视失真（即，当面部非常靠近摄像机时，面部看起来透视缩短（foreshortened）了）导致的变化。然而，如果用户使眼睛与摄像机保持接触，则这些变化可能较小。

可以调整进行修剪时的特写程度。如果进一步的机器处理是目标，则只有面部（前额到下巴）的特征的极限特写提供最稳定的图像。然而，这不会产生在审美上令人愉悦的图像。示出整个面部的更加自然的头部照片也是可以的。由变量 desiredRightEyeLocation 和 desiredEyeWidth 来控制特写的程度，这些变量分别表示输出图像中的右眼的期望坐标和输出图像中的眼睛之间的期望距离。参照上述用于寻找头部图像的伪代码，在本实施例中，对于极限特写，将这些变量分别设置为以像素为单位的极限特写图像的宽度的 0.6、0.2 和 0.3 倍。对于特写，将这些变量设置为以像素为单位的特写图像的宽度的 0.4、0.3 和 0.41 倍。

在本实施例中，通过下面的伪代码来描述进行仿射卷绕和修剪（450）的方法：

```
affineWarpAndCrop(in rawImage:image,in eyeLocations:eyeLocations,
```

```
in desiredEyeWidth:double,in desiredRightEyeLocation:point, out alignedImage:image)
{
coeffs:double[2][3]

outex:int=desiredRightEyeLocation.x
outey:int=desiredRightEyeLocation.y
outew=desiredEyeWidth
lex:int=eyeLocations.left.x
rex:int=eyeLocations.right.x
ley:int=eyeLocations.left.y
rey:int=eyeLocations.right.y

//计算输入中的眼睛之间的距离
edx:int=lex-rex
edy:int=ley-rey
ed:double=sqrt(edx*edx+edy*edy)

//计算输入和输出之间的标度因子
double s:double=outew/ed

//设置一些单位矢量
ev1x:double=edx*s/ed
ev1y:double=edy*s/ed
ev2x:double=-edy*s/ed
ev2y:double=edx*s/ed

//设置系数
coeffs[0][0]=ev1x
coeffs[0][1]=ev1y
```

```
coeffs[0][2]=-rex*ev1x-rey*ev1y+outrex
coeffs[1][0]=ev2x
coeffs[1][1]=ev2y
coeffs[1][2]=-rex*ev2x-rey*ev2y+outrey

//进行卷绕
warpAffine(rawImage,alignedImage,coeffs)
}
```

其中，warpAffine 函数是从 Intel IPL 软件库提供的。

仿射卷绕和修剪（450）的结果（也是对齐头部图像（400）阶段的结果）是对齐的面部图像（490），该对齐的面部图像（490）根据修剪的程度而存储为 closeupImage 或 extremeCloseupImage。

这完成了对齐头部图像（400）的阶段。

正规化光方向（500）

参照图 7，现在将描述对落在在阶段 400 中找到的头部上的光方向进行正规化（500）的处理。

再次参照图像处理程序的伪代码示例（图 3），将会看到，取决于在对齐头部图像阶段 400 中的修剪的程度，对 extremeCloseupImage 数据或者 closeupImage 数据执行光方向的正规化。

在本实施例中，通过下面的伪代码来描述对光方向进行正规化（500）的方法：

```
normaliseLightDirection(in alignedImage:image, in lowpassneutral:image,
out alignedbalancedImage:image)
{
RGB2YUV(alignedImage,yuv)
normaliseLightDirection2(lowpassneutral,yuv.y)
YUV2RGB(yuv,alignedbalancedImage)
}
normaliseLightDirection2(in lowpassneutral:image,inout grey:image)
```

```
{
//模糊
convolveSep2D(grey,blur,lightDirGaussianKernelx,lightDirGaussianKernely)
//从 grey 中减去模糊的
subtract(grey,blur,b)
//添加模糊的中性图像
add(b,lowpassneutral,grey)
}
```

参照图 7，仅对包含在 YUV 颜色空间中的 Y（亮度）通道中的强度信息执行光方向的正规化。将输入的图像数据（在上述伪代码中称作 alignedImage 并且包括 extremeCloseupImage 数据或 closeupImage 数据）从 RGB 颜色空间转换到 YUV 颜色空间，并且提取出 Y 通道（由上述代码中的 yuv.y 变量表示）（501）。在本实施例中，这是使用 Intel IPL 软件库的 RGB2YUV 函数执行的。

假定落在用户的面部上的光的方向导致面部上从亮到暗的缓慢变化。这样，在图像的低频率中拍摄到光强度的大部分变化。高斯模糊（即，与高斯进行卷积）有效地对图像进行了低通滤波（步骤 503）。在本实施例中，这是使用 Intel IPL 软件库的 convolveSep2D 函数执行的。可以使用上述的 createGaussianKernels 函数来建立高斯核 lightDirGaussianKernelx 和 lightDirGaussianKernely，并且在前面提及的 setup 函数中如下调用：

```
CreateGaussianKernels(41,lightDirGaussianKernelx,lightDirGaussianKernely)
//滤波长度为 41≈alignedImage.width/2
```

从原始版本中减去该经模糊的版本得到仅包含高空间频率的图像（步骤 505）。在本实施例中，这是使用 Intel IPL 软件库的 subtract 函数执行的，该 subtract 函数执行两个图像 a 和 b 的像素级减法以形成新的图像 c，从而 c(x,y)=a(x,y)-b(x,y)。

在该阶段，光照的效果已被基本去除，但是得到的图像看起来不太自然，因此通过在中性（neutral）光照（给予面部均匀外观而不会在面部

上投射强阴影的散射光照）下加入头部图像的高斯模糊（即低通滤波）版本，来恢复一些低频信息。

在本实施例中，取决于修剪的程度，根据中性光照下的头部的对齐的特写或极限特写图像来创建该高斯模糊图像。这些对齐的中性图像 517 不是单个的图像而是通过对许多对齐的图像进行求和而创建的平均图像（mean image），这具有使光方向的变化平坦（even out）的效果。

与对齐的图像数据 490 相同，首先将对齐的中性图像数据 517 从 RGB 颜色空间转换到 YUV 颜色空间，并且提取 Y 通道（519）。在本实施例中，这是使用 Intel IPL 软件库的 RGB2YUV 函数执行的。然后使用 Intel IPL 软件库的 convolveSep2D 函数对所提取的数据执行高斯模糊（521）。下面给出了用于对齐的中性图像的处理的示例伪代码：

```
setupLightDirectionBalance(in neutralImage:Image,out lowPassNeutralImage:image)
{
RGB2YUV(neutralImage,yuv)
convolveSep2D(yuv.y,lowPassNeutralImage,lightDirGaussianKernelx,lightDirGaussianKernely)
}
```

对于各修剪程度仅需将该模糊的对齐的中性图像创建一次，因此在前述的 setup 函数中如下调用这个函数：

```
setupLightDirectionBalance(extremeCloseupNeutralImage,extremeCloseupBlurNeutral)
setupLightDirectionBalance(closeupNeutralImage,extremeCloseupBlurNeutral)
```

然后将该模糊的中性对齐图像加到从减法步骤（505）输出的图像数据（507）。在本实施例中，这是使用 Intel IPL 软件库的 add 函数执行的，该 add 函数执行两个图像 a 和 b 的像素级相加以形成新的图像 c，从而 c(x,y)=a(x,y)+b(x,y)。

然后将得到的图像数据转换回 RGB 颜色空间（515）。在本实施例中，这是使用 Intel IPL 软件库的 RGB2YUV 函数执行的。

对光方向进行正规化的阶段（500）的结果是面部的具有平衡的光方向的对齐图像（550），根据修剪的程度，可将该图像存储为 balancedECImage 或 balancedCImage。

这完成了对光方向进行正规化的阶段（500）。

校正颜色平衡（600）

参照图 8，现在将描述对从阶段 500 输出的具有平衡的光方向的对齐图像（550）的颜色平衡进行校正（600）的处理。

应该回想起，出于本发明的目的，术语“颜色”旨在包括彩色颜色（具有色调的颜色，色调是颜色的一种性质，通过该性质能够在按光的主波长确定的从红到紫的范围内感知到这些颜色）和非彩色颜色（没有色调的颜色，即白色、灰色和黑色）。

再次参照图像处理程序（图 3）的伪代码示例，可以看出，根据修剪的程度，对 balancedECImage 数据或者 balancedCImage 数据执行颜色平衡的校正。

在本实施例中，通过下面的伪代码来描述校正颜色平衡（600）的方法：

```
colourBalance(in alignedbalancedImage:image,out stabilisedimage)
{
//蓝色通道
mean:int=findMean(alignedbalancedImage.b)
shift:int=neutralMean.b-mean
shiftPixels(alignedbalancedImage,shift,stabilisedimage)

//绿色通道
mean:int=findMean(alignedbalancedImage.g)
shift:int=neutralMean.g-mean
shiftPixels(alignedbalancedImage,shift,stabilisedimage)

//红色通道
mean:int=findMean(alignedbalancedImage.r)
shift:int=neutralMean.r-mean
shiftPixels(alignedbalancedImage,shift,stabilisedimage)
```

```
}
```

参照图 8，分别对 RGB 图像的每个通道执行颜色平衡的校正，并且在步骤 601 中，计算 RGB 图像的每个通道（红色、绿色和蓝色）中的平均像素值。在上述伪代码中，这由与 Intel OpenCV 软件库的 cvMean 函数相对应的 findMean 函数表示。还计算了中性光照下头部的对齐的 RGB 图像（517）的每个通道中的平均像素值（605）。下面给出了用于处理对齐的中性图像的示例伪代码：

```
setupColourBalance(neutralImage:image)
{
neutralMean.b=findMean(neutralImage.b)
neutralMean.g=findMean(neutralImage.g)
neutralMean.r=findMean(neutralImage.r)
}
```

中性光照下头部的对齐的 RGB 图像（517）的每个通道中的平均像素值仅需计算一次。此外，仅根据面部区域进行计算，因此使用在中性光照下的头部的对齐的极限特写图像。因此，在前述的 setup 函数内如下调用该函数：

```
setupColourBalance(extremeCloseupNeutralImage)
```

然后计算像素移位（603），像素移位会调整平均 R、G、B 值以使其匹配该中性图像，然后将该移位加到各像素（607）。

校正颜色平衡的阶段（600）的结果是面部的具有平衡的光方向和经校正的颜色平衡的对齐图像，换言之，即稳定的输出图像（700），根据修剪的程度，该稳定的输出图像（700）由面部的特写图像（stabilisedCImage）或极限特写图像（stabilisedECImage）构成。

这完成了对光方向进行正规化的阶段（500）。

图像处理程序（图 3）的 mobileFacePreprocessor()函数对视频数据的的效果是要减少帧之间的像素值的变化，这些变化是由对象穿过场景的全局移动、摄像机和对象之间的相对移动、或诸如在处理之后的自动白平衡和曝光补偿的对特定摄像机参数的电子调整所导致的。对象的外貌

的其余变化基本上是由于对象的非刚性变形（例如，在面部的情况下，即表情的变化）和对象的表面性质的变化（例如，在面部的情况下，即起皱纹和脸红）。

再次参照图像处理程序的主循环的上述示例伪代码，一旦获得了稳定的输出图像 700，就可将其用在某些其他函数中，并且这由函数 doSomething 来表示。

例如，该图像处理程序（图 3）可以是视频会议系统的一部分，在此情况下，可以使用传统的视频编解码器（例如，H.264）和网络协议（例如，RTP、TCP）来对该稳定的输出图像进行编码和发送。（还存在这样的可能：在这种视频会议系统中，并不总是需要这种类型的图像处理。例如，会有人的背景和环境比其表情更重要的时候。在这种视频会议系统中，因而可以在对原始数据进行编码/发送以及对稳定的输出数据进行编码/发送之间进行切换。）

在一另选例中，可以不将稳定的输出图像 700 提供给另一人类用户而是将其提供给一机器，该机器在将稳定的输出图像 700 转发给人之前将执行一些进一步的图像处理。还可以是该机器自身能够自动理解面部表情，这在使用特定面部表情（例如微笑）的动态来许可对例如计算机网络或保密地点的访问的应用中是有用的。

另选的是，可将摄像机安装在车辆的仪表板上并且用于拍摄驾驶员面部的视频。在这种场景下，除了面部/头部的方向会变化以外，落在面部上的光也会显著变化（尤其在夜里）。可以使用稳定的输出图像来监视驾驶员的警惕性，从而监视他的安全控制车辆的能力。

应该明白，通常在发送视频数据之前运行该图像处理程序（图 3），因此将其称作预处理。

从前面的描述中可以显见，在不脱离本发明的情况下，可以对上述实施例作出许多修改或变型。这些修改和变型包括：

尽管在上述实施例中，是在计算机系统 1 上存储并运行图像处理程序，但是还可以在移动装置 199 上存储和运行它。这会实现实时的、人到人的通信。在这种实施例中，还可以在该移动装置和计算机系统之间

分担由该图像处理程序执行的处理。还可以在移动装置与其所连接的基站之间或者在移动装置与连接到网络的另一计算机之间分担某些处理。例如，该移动装置可以执行定位和对齐头部图像所需的处理，而基站/计算机执行对光方向进行正规化和调整颜色平衡所需的处理。另选的是，可以在基站/计算机处执行对眼对的全局搜索，将结果发送回移动装置。

在上述实施例中，在对用户眼睛的局部和全局搜索中使用了模板匹配。在另选例中，可以在对用户面部上的其他特征（例如鼻孔、眼眉）的局部和/或全局搜索中使用模板匹配。

在另选例中，模板匹配可与现在描述的自适应颜色匹配法结合，甚至用其来替换。给定面部的初始位置，可以定义图像的为“面部”和“非面部”的区域。从这两个区域中，可以计算出两个概率分布：在一像素属于“面部”区域的情况下该像素为颜色 c 的概率（P(像素为颜色 c|像素属于面部)）；和在一像素属于“非面部”区域的情况下该像素为颜色 c 的概率（P(像素为颜色 c|像素不是面部的一部分)）。对于新图像，对于每个像素，使用以上两种分布，可以使用贝叶斯（Bayes）理论来确定在一像素为颜色 c 的情况下该像素属于“面部”区域的概率（P(像素属于面部|像素为颜色 c)）。通过检查属于面部的概率高的像素的空间分布，可以确定面部在新图像中的位置。

在本实施例中，最好将颜色表示为 YUV（甚至 HSV）而不是 RGB，并且为了进一步减少计算量，该空间将十分粗糙（例如，4 比特用于 Y，4 比特用于 U，4 比特用于 V）。

然而，由于头部和摄像机移动，导致光照改变，这些静态概率分布不再能良好地表示面部的颜色。因此，可以更新该分布。在新图像中定位了面部之后，又可以确定“面部”和“非面部”区域。可以根据这些区域中的像素来计算新的概率分布，然后可以使用运转平均方法（running average method）来更新全局分布。

例如，如果 pYUV(t)是在一像素属于根据帧 t 中的像素计算出的“面部”区域的情况下该像素具有颜色 Y、U、V 的概率（P(像素为颜色 Y、U、V|像素是面部)），并且 gpYUV(t)是在一像素属于根据帧 t 中的像素计

算出的“面部”区域的情况下该像素具有颜色 Y、U、V 的运转平均概率（P(像素为颜色 Y、U、V|像素是面部)），则：gpYUV(t+1)=gpYUV(t)×(1-a)+ pYUV(t)×a，其中，a 是常量并且较小（例如 0.1）。

另选的是，“面部”和“非面部”区域的概率分布还可用高斯分布来表示，由于高斯可被更简单地刻画为均值和方差，所以高斯分布可能更适合移动装置的实现。

如果面部的大小/位置看起来靠不住或者面部彩色像素的分布太稀疏，则能够检测到面部跟踪的完全丢失。在这种情况下，需要进行重新初始化来定位面部。

如果需要关于面部朝向的附加信息，则可以使用局部特征匹配（例如，如上面关于局部搜索 405 所描述的）来精细调整对面部的位置和朝向的判定。

尽管在上述实施例中在对眼对的全局搜索中使用了高斯分布，但是也可使用其他概率分布。例如，可以使用训练序列来创建眼睛的相对位置的 2D 概率分布，即：在左眼位于(x,y)的情况下右眼位于位置(x+dx,y+dy)的概率（p(右眼位于(x+dx,y+dy)|左眼位于(x,y))）。这种 2D 分布随后可用来直接对模板匹配结果进行卷积，而不进行上述的偏移/与高斯进行卷积的步骤。然而，使用偏移/与高斯进行卷积的步骤的优点在于：由于高斯卷积能够分成两个 1D 卷积并且包括 2n 次的运算量级（与用 2D 分布进行卷积的 n^2 次运算相对），所以该处理更快。

与全局搜索有关的上述处理的另一另选例是：

像前面那样执行左眼和右眼模板匹配⇒用阈值衡量匹配得分⇒定位剩余斑点（blob）的质心以给出左眼的 n 个位置和右眼的 m 个位置⇒对各个可能的左/右眼位置的对进行比较（n*m 个可能的对）⇒基于 p(右眼位于(x+dx,y+dy)|左眼位于(x,y))对这些对打分⇒选择具有最高得分的眼对。这将比前面描述的方法更快（因此更适合实现在移动装置上），但是如果眼睛的校正位置没有通过该用阈值衡量的阶段，则该处理会失败。

上述用于校正颜色平衡的方法假定从该方法输出的像素的颜色是与输入到该方法的像素相同的像素的颜色的某个函数，即，输出颜色＝F(输

入颜色)，其中 F 是一函数，从对面部颜色以及颜色的期望输出分布的测量得出该函数的性质。上述函数的另选函数对本领域技术人员是显而易见的。

尽管在上述对光方向进行正规化的方法中，通过与高斯滤波核进行卷积来实现低通滤波，但是也可使用其他滤波核，例如海宁（Hanning）、恺撒（Kaiser）。然而，由于高斯核不产生太多失真并且能够实现为两个 1D 核（这会导致快速计算），所以高斯核特别有效。通过与滤波核进行卷积在空间域中进行低通滤波的另选方法是：使用 2D 快速傅立叶变换(FFT)将图像转换到频域，与抑制高频的滤波核相乘，并执行逆 FFT 以返回到空间域。

尽管在上述实施例中，校正颜色平衡步骤是在 RGB 颜色空间中执行的，但是也可以在 YUV 颜色空间中执行该步骤（这等同于在 RGB 颜色空间中执行，因为 RGB 和 YUV 之间的转换是线性变换）。在这种实施例中，校正颜色平衡步骤对 Y 通道没有任何影响。此外，应该回想到，仅对 Y 通道执行了对光方向进行正规化的步骤，因此不会影响 U 和 V 通道。因此，执行对光方向进行正规化的步骤和校正颜色平衡的步骤的顺序无关紧要。

就效率而言，在 YUV 颜色空间中执行这两个步骤而不在这两个步骤中间转换回 RGB 颜色空间是有利的。在这种实施例中，将故意忽略对 Y 通道的颜色校正。实际上，应该回想到，头部定位的模板匹配是针对灰度图像执行的（仅 Y 通道）。因此，可以在 YUV 颜色空间中执行整个处理，而输出可以是 YUV 图像（因为许多视频编码器接受 YUV 数据作为输入）。然而，这将涉及计算图像的被修剪掉的区域的 U 和 V，所以在这种实施例中进行该处理的可能顺序是：输入 RGB⇒寻找 Y⇒修剪 RGB 和 Y⇒在修剪的 RGB 上寻找 U、V⇒对 YUV 进行仿射卷绕和定标(scale)⇒针对 Y 对光方向进行正规化⇒针对 U、V 校正颜色平衡⇒输出 YUV。

尽管在上述实施例中，稳定的输出图像 700 包括 RGB 图像，但是期望的输出还可以是头部的灰度图像。在这种情况下，由于原始图像最初就被转换成灰度图像（等同于从 RGB 颜色空间转换到 YUV 颜色空间并

且丢弃 U 和 V 通道），所以图像处理步骤将被简化。于是，在局部或全局搜索阶段不需要将用于模板匹配的关注区域转换成灰度图像，在对光方向进行正规化时，不需要在 RGB 和 YUV 颜色空间之间进行转换，并且在校正颜色平衡时，仅需调整一个通道。

尽管在上述实施例中，来自该图像处理程序的输出是稳定的输出图像，但是由于眼睛保持在头部中的固定位置处并且眼睛的平均位置给出了头部在图像内的良好定位，所以还可以提供关于头部的位置和标度的输出信息。

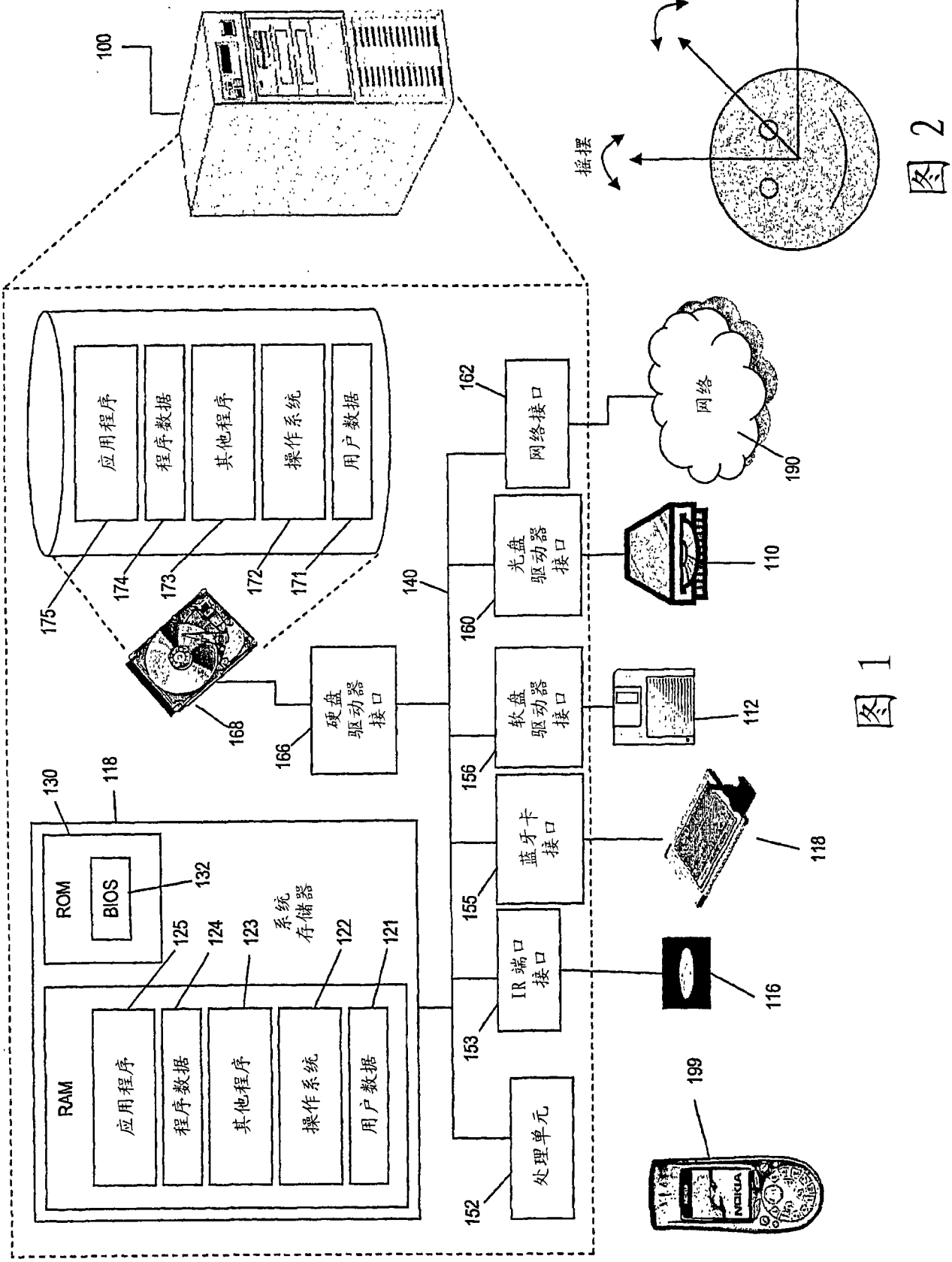

100
应用程序
程序数据
其他程序
操作系统
用户数据
175
174
173
172
171
168
166
硬盘驱动器接口
140
162
网络接口
网络
190
160
光盘驱动器接口
110
156
软盘驱动器接口
112
155
蓝牙卡接口
118
153
IR端口接口
116
152
处理单元
199
130
118
ROM
BIOS
132
系统存储器
125
124
123
122
121
RAM
应用程序
程序数据
其他程序
操作系统
用户数据

图 1

转动
倾斜
摇摆

图 2

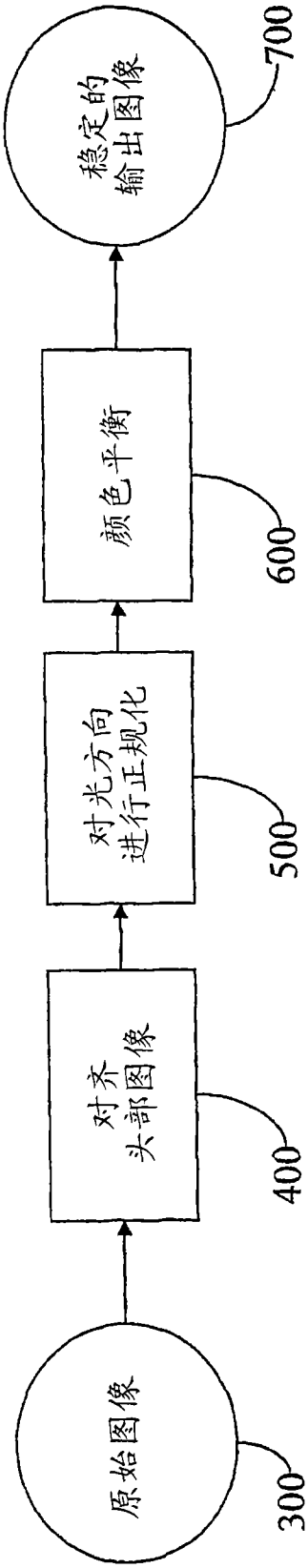

原始图像
300
对齐
头部图像
400
对光方向
进行正规化
500
颜色平衡
600
稳定的
输出图像
700

图 3

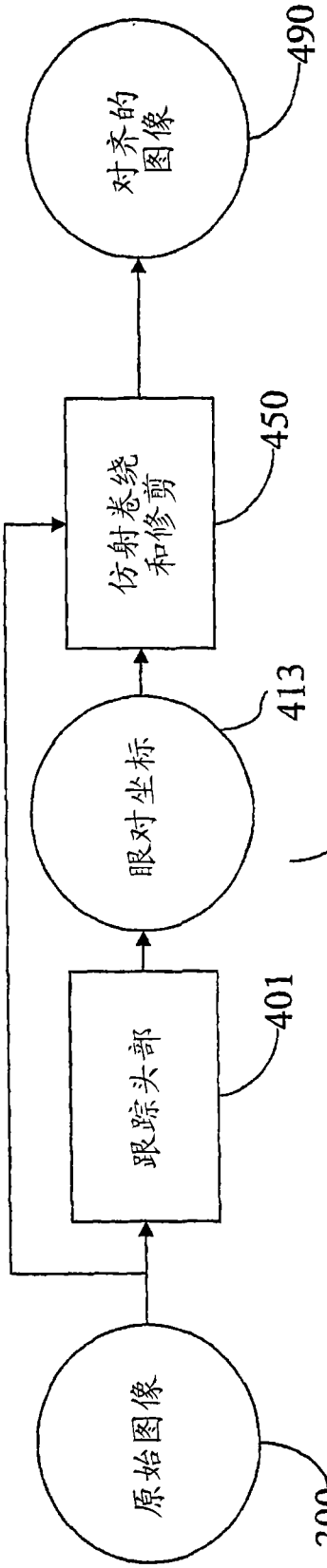

原始图像
300
跟踪头部
401
眼对坐标
413
400
仿射卷绕
和修剪
450
对齐的
图像
490

图 4

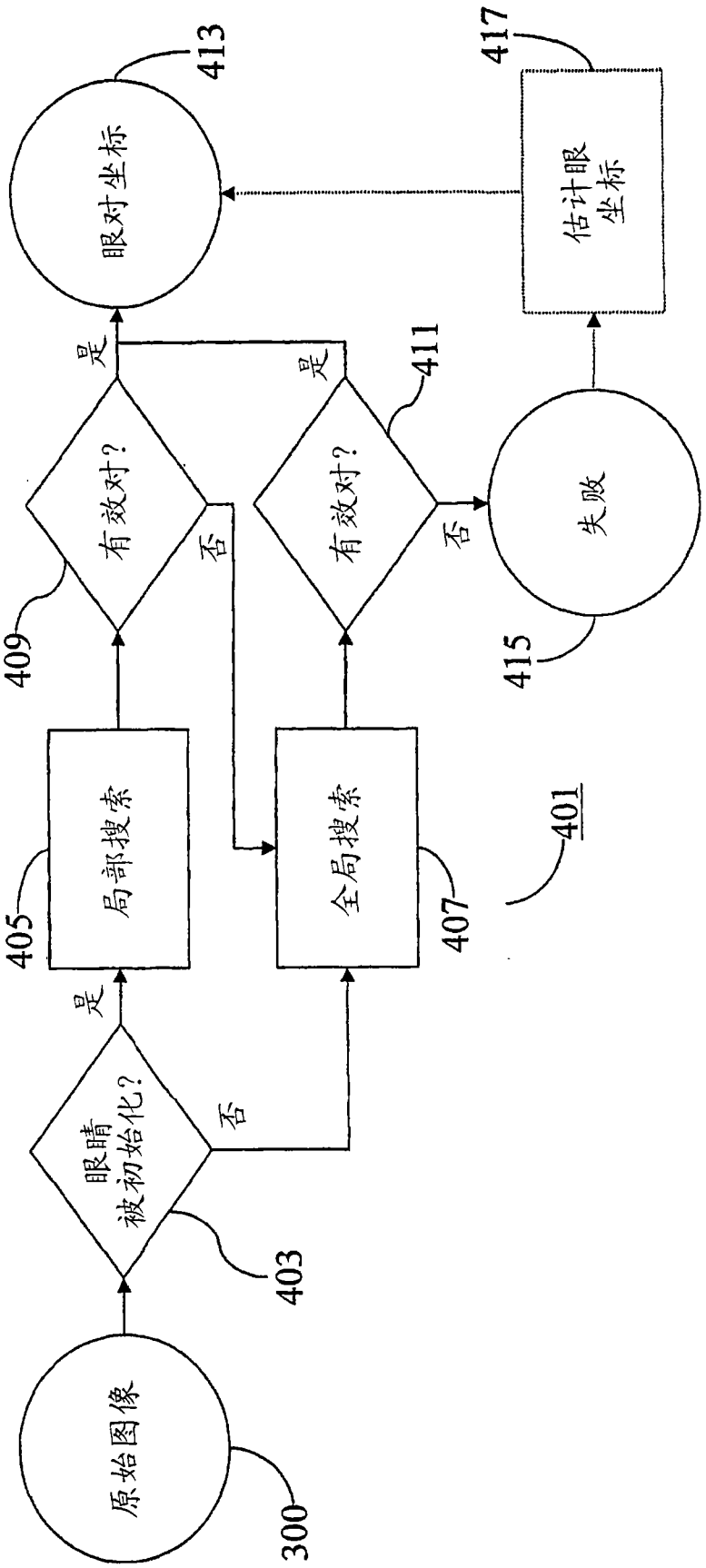

原始图像
300
眼睛被初始化?
403
是
否
局部搜索
405
全局搜索
407
401
有效对?
409
是
否
有效对?
411
是
否
失败
415
估计眼坐标
417
眼对坐标
413

图 5

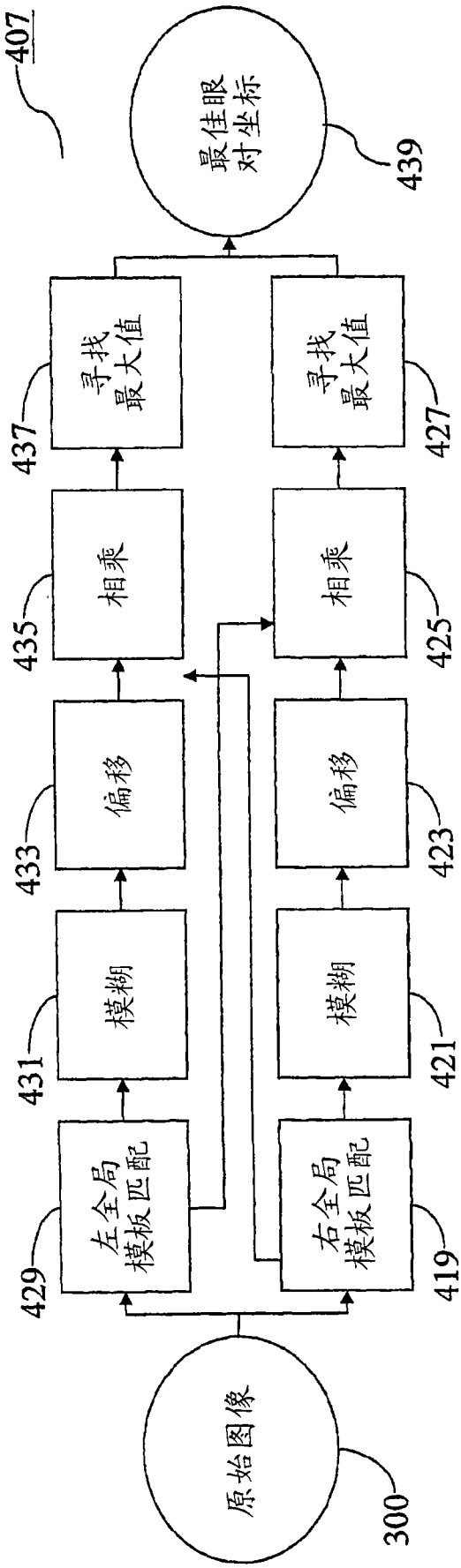

407
最佳眼对坐标
439
寻找最大值
437
寻找最大值
427
相乘
435
相乘
425
偏移
433
偏移
423
模糊
431
模糊
421
左全局模板匹配
429
右全局模板匹配
419
原始图像
300

图 6

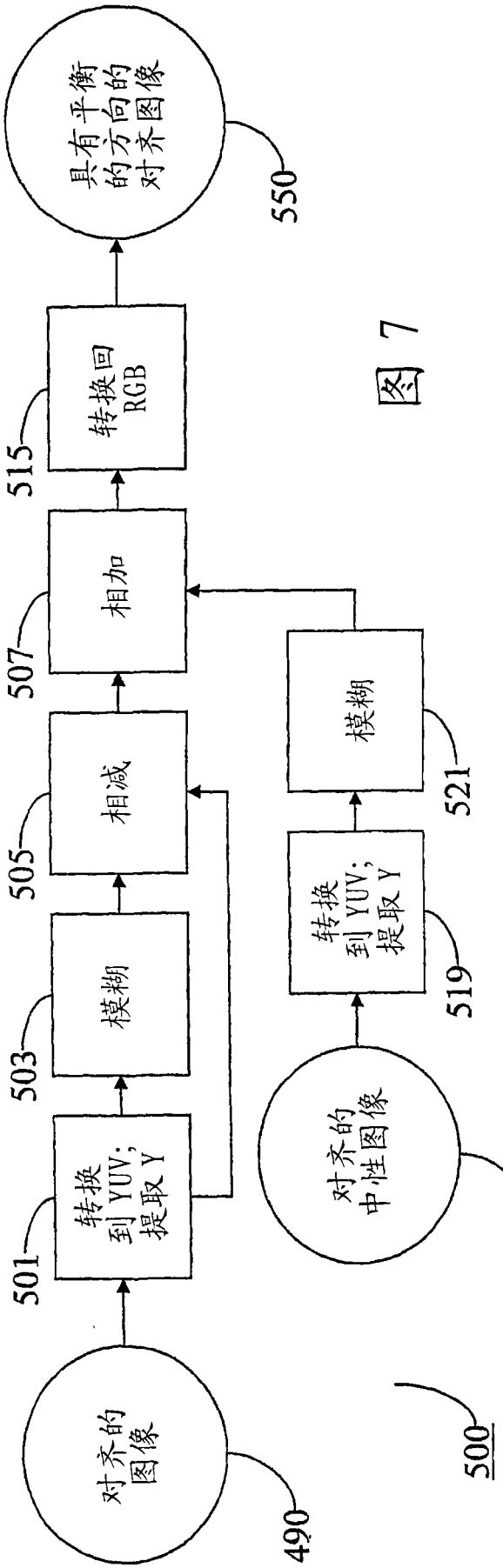

对齐的图像
490
转换到 YUV; 提取 Y
501
模糊
503
相减
505
相加
507
转换回 RGB
515
具有平衡的方向的对齐图像
550
转换到 YUV; 提取 Y
519
模糊
521
对齐的中性图像
517
500

图 7

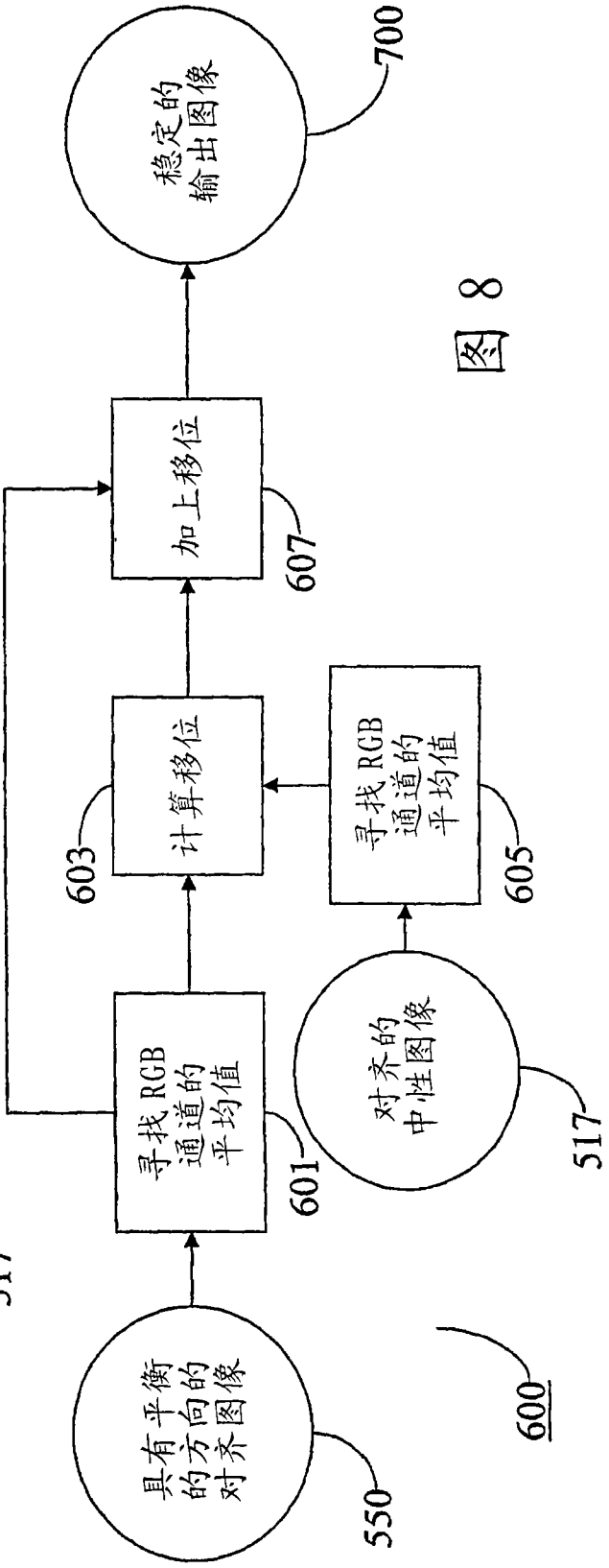

具有平衡的方向的对齐图像
550
寻找 RGB 通道的平均值
601
计算移位
603
加上移位
607
稳定的输出图像
700
对齐的中性图像
517
寻找 RGB 通道的平均值
605
600

图 8