[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06T 3/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 200510000355.1

[45] 授权公告日 2008 年 1 月 9 日

[11] 授权公告号 CN 100361156C

[22] 申请日 2000.9.8

[21] 申请号 200510000355.1

分案原申请号 00814363.3

[30] 优先权

[32] 1999.6.16 [33] AU [31] PQ2890

[73] 专利权人 西尔弗布鲁克研究股份有限公司

地址 澳大利亚新南威尔士州

[72] 发明人 保罗·拉普斯顿

西蒙·R·沃姆斯利

[56] 参考文献

WO 99/04368 A1 1999.1.28

WO 99/04551 A1 1999.1.28

US 4639769 A 1987.1.27

EP 0732859 A2 1996.9.18

US 5146328 A 1992.9.8

EP 0709825 A2 1996.5.1

CN 1142723 A 1997.2.12

A GENERIC 2D SHARPNESS ENHANCEMENTALGORITHM FOR LUMINANCE SIGNALS. Egbert G. T. Jaspers et al. IEE IPA97, No. No. 443. 1997

Image Processing Considerations for Digital Photography. Ping Wah Wong et al. PROCEEDINGS OF IEEE COMPCON 97. 1997

审查员 梁 燕

[74] 专利代理机构 北京集佳知识产权代理有限公司

代理人 王学强

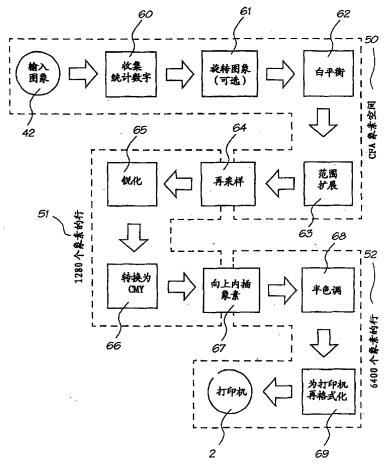
权利要求书 2 页 说明书 100 页 附图 43 页

[54] 发明名称

用于打印图象的方法和装置

[57] 摘要

一种接收表示图象的图象数据，并向打印机提供相应于所述图象数据的打印数据的方法，所述打印数据为预定的打印格式且具有预定的打印分辨率，该方法包括步骤：接收图象数据；将图象数据转换为具有第一预定分辨率的第一格式数据；将第一格式数据转换为打印数据；以及使打印数据可用于打印机打印。 一种接收表示图象的图象数据，并向打印机提供相应于所述图象数据的打印数据的装置，所述打印数据为预定的打印格式且具有预定的打印分辨率，该装置包括：用于接收图象数据的输入装置；用于将图象数据转换为具有第一预定分辨率的第一格式数据的采样装置；用于将第一格式数据转换为打印数据的处理装置；以及使打印数据可用于打印机打印的输出装置。 一种包括这里所述装置的光电照相机。



1、一种接收表示图象的图象数据并向打印机提供相应于所述图象数据的打印数据的方法，所述打印数据为预定的打印格式且具有预定的打印分辨率，该方法包括步骤:

接收图象数据;

将图象数据转换为中分辨率数据;

将中分辨率数据转换为打印数据；以及

使打印数据可用于打印机打印。

2、如权利要求 1 的方法，其中中分辨率不同于打印分辨率。

3、如权利要求 1 或 2 的方法，其中图象数据为拜尔格式且具有预定的图象分辨率。

4、如权利要求 1 或 2 的方法，其中中分辨率数据具有预定的连续色调分辨率。

5、如权利要求 1 或 2 的方法，其中打印数据具有预定的二值点分辨率。

6、如权利要求 1 或 2 的方法，其中图象数据为红、绿和蓝（RGB）格式并且打印机对青、品红和黄（CMY）格式响应，该方法还包括将打印数据从红、绿和蓝（RGB）格式转换为青、品红和黄（CMY）格式的附加步骤。

7、如权利要求 1 或 2 的方法，该方法还包括处理图象数据、中分辨率数据和打印数据中至少一个的步骤。

8、如权利要求 7 的方法，其中图象数据被选择性地调整，用于以预定的旋转方向提供图象。

9、一种接收表示图象的图象数据并向打印机提供相应于所述图象数据的打印数据的装置，所述打印数据为预定的打印格式且具有预定的打印分辨率，该装置包括:

用于接收图象数据的输入装置;

用于将图象数据转换为中分辨率数据的采样装置；

用于将中分辨率数据转换为打印数据的处理装置；以及

使打印数据可用于打印机打印的输出装置。

10、如权利要求9的装置，其中中分辨率不同于打印分辨率。

11、如权利要求9的装置，其中图象数据为拜尔格式且具有预定的图象分辨率。

12、如权利要求9的装置，其中中分辨率数据具有预定的连续色调分辨率。

13、如权利要求9的装置，其中打印数据具有预定的二值点分辨率。

14、如权利要求9的装置，其中图象数据为红、绿和蓝（RGB）格式并且打印机对青、品红和黄（CMY）格式响应，该装置还包括将打印数据从红、绿和蓝（RGB）格式转换为青、品红和黄（CMY）格式的附加装置。

15、如权利要求9的装置，该装置还包括用于处理图象数据、中分辨率数据和打印数据中至少一个的装置。

16、如权利要求15的装置，包括用于选择性地调整图象数据以便以预定的旋转方向提供图象的装置。

17、一种包括权利要求9至16的任何一项的装置的光电照相机。

用于打印图象的方法和装置

本申请是申请日为 2000 年 9 月 8 日，申请号为 00814363.3（PCT/AU00/01075），名称为“根据拜尔图象生成打印图象的方法和装置”的分案申请。

技术领域

本发明涉及根据拜尔图象生成打印的方法和装置.

本发明主要用于数字照相机，包括用于在相纸上打印照相机拍摄的图象的整体打印机，并且以下将参照该应用描述本发明．然而，应该理解，本发明并不限于特定应用领域.

背景技术

交叉引用的申请

参考文献为澳大利亚临时专利申请号 PQ2890（申请日期为 1999 年 9 月 16 日）的共同未决申请优先权要求．共同未决申请描述用于实现小型打印系统的有关模块和方法．与本发明同时申请的共同未决申请为:

国际专利申请号	参考号	发明的题目
PCT/AU00/01074	PCP02	对图象进行锐化处理的方法和装置
PCT/AU00/01073	PCP03	用于向上内插拜尔图象的方法和装置
PCT/AU00/01076	PCP04	用于旋转拜尔图象的方法和装置

发明内容

本发明提供了一种接收表示图象的图象数据并向打印机提供相应于所述图象数据的打印数据的方法，所述打印数据为预定的打印格式且具有预定的打印分辨率，该方法包括步骤:

接收图象数据;

将图象数据转换为中分辨率数据;

将中分辨率数据转换为打印数据；以及

使打印数据可用于打印机打印。

优选地，中分辨率不同于打印分辨率。

优选地，图象数据为拜尔格式且具有预定的图象分辨率。

优选地，中分辨率数据具有预定的连续色调分辨率。

优选地，打印数据具有预定的二值点分辨率。

优选地，图象数据为红、绿和蓝（RGB）格式并且打印机对青、品红和黄（CMY）格式响应，该方法还包括将打印数据从红、绿和蓝（RGB）格式转换为青、品红和黄（CMY）格式的附加步骤。优选地，该方法还包括处理图象数据、中分辨率数据和打印数据中至少一个的步骤。

优选地，图象数据被选择性地调整，用于以预定的旋转方向提供图象。

本发明还提供一种接收表示图象的图象数据并向打印机提供相应于所述图象数据的打印数据的装置，所述打印数据为预定的打印格式且具有预定的打印分辨率，该装置包括:

用于接收图象数据的输入装置;

用于将图象数据转换为中分辨率数据的采样装置;

用于将中分辨率数据转换为打印数据的处理装置；以及

使打印数据可用于打印机打印的输出装置。

优选地，中分辨率不同于打印分辨率。

优选地，图象数据为拜尔格式且具有预定的图象分辨率。

优选地，中分辨率数据具有预定的连续色调分辨率。

优选地，打印数据具有预定的二值点分辨率。

优选地，图象数据为红、绿和蓝（RGB）格式并且打印机对青、品红和黄（CMY）格式响应，该装置还包括将打印数据从红、绿和蓝（RGB）格式转换为青、品红和黄（CMY）格式的附加装置。

优选地，该装置还包括用于处理图象数据、中分辨率数据和打印数据中至少一个的装置。

优选地，该装置包括用于选择性地调整图象数据以便以预定的旋转方向提供图象的装置。

本发明还提供一种包括任一上述装置的光电照相机。

根据本发明的第一方面，提供一种用于提供图象的方法，以便以预定的二值点分辨率进行打印，二值点分辨率相当于预定的连续色调分辨率，该方法包括以下步骤:

接收表示图象的第一数据集，即预定的连续色调分辨率的数据集;

将第一数据集转换为预定的连续色调分辨率的第二数据集;

将第二数据集转换为预定的二值点分辨率的第三数据集；以及

使打印机以预定的二值点分辨率获得第三数据集。

第一分辨率最好与预定的二值点分辨率匹配。然而，在其他实施方式中，第一分辨率大于预定的二值点分辨率。在更进一步的实施方式中，第一分辨率小于预定的二值点分辨率。

同时，第一数据集最好为红、绿和蓝（RGB）格式，而打印机与青、品红和黄（CMY）格式相对应，并且该方法还包括附加步骤，即将第三数据集从RGB格式转换为CMY格式。

在一种最佳形式中，该方法包括以下步骤，即对第二数据集进行锐化处理。作为选择，该方法最好包括对第一数据集进行锐化处理的步骤。

最好从曝光装置中获得第一数据集，并且该方法包括以下步骤，即补偿第一数据集，以抵消曝光装置中的非线性。补偿步骤最好包括将第一数据集从众多 x 比特采样转换为众多 y 比特采样，其中 x>y。并且最好 x=10，y=8。

同时，该方法还包括以下步骤，即将第一数据集平面化为一个红色平面、一个绿色平面和一个蓝色平面。

在最佳形式中，该方法还包括以下步骤:

为第一数据集定义 m%的最暗象素和 n%的最亮象素;

调整第一数据集以使 m%的最暗象素相等；以及

调整第一数据集以使 n%的最亮象素相等。

该方法最好包括附加步骤，即调整第一数据集以便提供预定的白平衡。该方法最好包括以下附加步骤，即调整第一数据集以便提供预定的范围扩展。最好增加第一数据集的彩色分辨率，同时保持相同的空间分辨率。

在一种最佳形式中，选择调整第一数据集，以便沿预定的旋转方向提供图象。

根据本发明的第二方面，提供用于提供图象的装置，以便以预定的二值点分辨率进行打印，二值点分辨率相当于预定的连续色调分辨率，该装置包括:

输入装置，用于接收表示图象的第一数据集，该数据集的格式为

第一分辨率的拜尔格式;

采样装置，用于将第一数据集转换为预定的连续色调分辨率的第二数据集;

处理装置，用于将第二数据集转换为预定的二值点分辨率的第三数据集；以及

使打印机获得第三数据集，以便以预定的二值点分辨率进行打印。

第一分辨率最好与预定的二值点分辨率匹配。作为选择，第一分辨率大于预定的二值点分辨率。在其他实施方式中，第一分辨率小于预定的二值点分辨率。

同时，第一数据集最好为红、绿和蓝（RGB）格式，而打印机与青、品红和黄（CMY）格式相对应，其中处理装置将第三数据集从RGB格式转换为CMY格式。

同时，该装置最好对第二数据集进行锐化处理。然而，在其他实施方式中，该装置对第一数据集进行锐化处理。

在一种最佳形式中，最好从曝光装置中获得第一数据集，并且输入装置补偿第一数据集，以抵消曝光装置中的非线性。补偿非线性最好包括将第一数据集从众多x比特采样转换为众多y比特采样，其中x>y。并且最好x=10，y=8。

输入装置将第一数据集平面化为一个红色平面、一个绿色平面和一个蓝色平面。

输入装置最好:

为第一数据集定义m%的最暗象素和n%的最亮象素;

调整第一数据集以使m%的最暗象素相等；以及

调整第一数据集以使n%的最亮象素相等。

在最佳形式中，输入装置调整第一数据集以便提供预定的白平衡。输入装置最好调整第一数据集以便提供预定的范围扩展。输入装置最好增加第一数据集的彩色分辨率，同时保持相同的空间分辨率。

输入装置最好选择调整第一数据集，以便沿预定的旋转方向提供图象。

根据本发明的第三方面，提供一种照相机，包括：

一个 CCD 阵列，用于提供拜尔图象；

一台打印机，用于选择提供打印的图象；以及

一种用于接收拜尔图象，然后向打印机提供第三数据集以便生成打印图象的装置。

附图说明

通过参照以下说明书和附图，举例说明本发明的最佳实施方式。

图 1 表示 PCP 的高级图象流程。

图 2 以分离形式表示 PCP 框图。

图 3 表示与 Printcam 硬件相连的 PCP 的框图。

图 4 表示长为 4 英寸的 Memjet 打印头。

图 5 表示 4 英寸打印头中各段的结构。

图 6 表示容器中喷嘴的结构，序号表示定位顺序。

图 7 表示容器中喷嘴的结构，序号表示加载顺序。

图 8 表示色品容器（chrompod）。

图 9 表示容器群（podgroup）。

图 10 表示相群（phasegroup）。

图 11 表示段、喷射群（firegroup）、相群、容器群和色品容器之间的关系。

图 12 表示打印奇数和偶数点时 AEnable 和 BEnable 脉冲的轮廓。

图 13 表示基于 CFA 图象打印格式的方向。

图 14 表示图象获取链的框图。

图 15 表示拜尔 CFA 2G 镶嵌面中各象素的结构。

图 16 表示线性化 RGB 过程。

图 17 表示平面化 RGB 过程。

图 18 表示图象打印链的框图。

图 19 表示单色平面的采样颜色范围。

图 20 表示白平衡和范围扩展涉及的步骤。

图 21 表示能够执行白平衡和范围扩展的装置的框图.

图 22 表示与 CFA 分辨率有关的各种颜色平面象素.

图 23 表示将绿色平面旋转 45 度的结果.

图 24 表示绿色平面的旋转象素之间的距离.

图 25 表示将不旋转 CFA 空间中的运动映射到旋转 CFA 空间的过程.

图 26 表示锐化过程的框图.

图 27 表示利用 3×3 内核对单亮度象素进行高通滤波的过程.

图 28 表示从 RGB 转换到 CMY 的变换.

图 29 表示利用三线内插将 RGB 转换到 CMY.

图 30 表示将单个象素复制为一个 5×5 块的象素复制.

图 31 表示半色调过程的框图.

图 32 表示为打印机重新格式化所有点的过程.

图 33 表示图象获取装置的框图.

图 34 表示图象存取装置的框图.

图 35 表示图象直方图装置的框图.

图 36 表示打印接口的框图.

图 37 表示 Memjet 接口的框图.

图 38 表示 AEnable 和 BEnable 脉冲宽度的生成过程.

图 39 表示点数计数逻辑的框图.

图 40 表示打印生成装置的接口.

图 41 表示打印生成装置的框图.

图 42 表示测试模式存取装置的框图.

图 43 表示缓冲器 5 的框图.

图 44 表示缓冲器 4 的框图.

图 45 表示向上内插、半色调和重新格式化过程的框图.

图 46 表示将标准抖动单元映射为交错抖动单元的方式.

图 47 表示将 RGB 转换为 CMY 过程的框图.

图 48 表示缓冲器 2 的框图.

图 49 表示使用 3x3 内核的基本高通空间滤波器.

图 50 表示锐化装置的框图.

图 51 表示缓冲器 1 的结构.

图 52 表示重新采样和创建亮度信道过程的框图.

图 53 表示卷积装置的框图.

图 54 表示从接收器生成的象素的顺序.

图 55 表示在旋转和不旋转空间中沿 x 和 y 方向的运动.

图 56 表示缓冲器 1 的绿色子缓冲器中的条目的地址.

图 57 表示依赖于旋转的绿色条目之间的关系.

图 58 表示绿色信道的 4x4 采样.

图 59 表示 4x4 绿色采样类型 1.

图 60 表示 4x4 绿色采样类型 2.

图 61 表示用于绿色的两类行寻址.

图 62 表示缓冲器 1 的红色和绿色子缓冲器中的条目的寻址.

图 63 表示为计算第一象素而读取的前 16 个采样.

图 64 表示从蓝色和红色缓冲器中读取的 4x4 重叠块的最坏情况.

图 65 表示旋转、白平衡和范围扩展装置的框图.

图 66 表示所生成的坐标空间内的有效图象区域.

具体实施方式

1 PCP 概述

1.1 高级功能概述

Printcam 中央处理器(PCP)拥有用于 Printcam 的所有处理能力，并且是专为 Printcam 数码照相机系统而设计的. PCP 3 连接图象传感器 1(用于图象获取)，和 Memjet 打印机 2(用于图象输出). 如图 1 所示，在图象处理方面，可以将 PCP 看作获取图象到输出图象的变换器:

- 图象传感器 1 是一个 CMOS 图象传感器，用于获取 1500x1000 的 RGB 图象. 图象传感器是图象输入设备.

- 打印头 2 是一个 4 英寸长的 1600dpi 的 Memjet 打印头，能够打印三种颜色：青、品红和黄。打印头为图象*输出*设备。
- PCP 3 记录图象传感器 1 的图象，处理该图象，然后将该图象的最终形式发送到打印头 2 进行打印。由于图象传感器 1 获取的图象为 RGB 格式，而打印头 2 按照 CMY 格式打印，所以 PCP 3 必须从 RGB 颜色空间转换到 CMY 颜色空间。PCP 3 包含中间图象处理需要的所有装置，其中中间图象处理包括白平衡、颜色校正、色彩转换、图象锐化和半色调处理。另外，PCP 3 控制用户界面和整个打印过程，以支持各种图象格式。PCP 3 还包含导入、导出相片的接口，其中相片符合 DPOF（数码打印指令格式）标准。

1.2 高级内部概述

PCP 3 是为使用 0.25 微米 CMOS 工艺进行生产而设计的，大约有 10,000,000 个晶体管，其中大部分为快闪存储器或静态 RAM。其估计面积为 16mm^2。其估计制造成本为 4$（2001 年）。PCP 3 是一个比较直接的设计，通过使用数据通路编译技术、宏单元和 IP core，可以简化其设计。PCP 3 包括:

- 一个低速 CPU/微型控制器核心 10
- 1.5MB 的多级快闪存储器（每单元 2 比特）11
- 图象获取单元 12 内的 CMOS 图象传感器接口 98
- 16KB 快闪存储器 13，用于程序存储
- 4KB RAM 14，用于程序变量存储

PCP 3 用于以约 100MHz 的时钟速度运行，其外部电压为 3V，内部电压为 1.5V，以将功耗降到最底程度。实际操作频率为打印头操作频率的整数倍。CPU 10 是一个微型控制器类型的 CPU，运行频率约为 1MHz。CPU 10 和 CMOS 传感器接口 12 是可以为厂商提供的核心。

图 2 以分离形式表示 PCP 3 的框图。

PCP 3 是为 Printcam 系统设计的。图 3 表示与 Printcam 硬件的

其他部分相连的 PCP 3 的框图。

2 打印头背景

PCP 3 是专为连接 4 英寸（10cm）的 Memjet 打印头 2 设计的。打印头 2 作为一个页宽打印机，用于在不移动打印头 2 的情况下生成 4 英寸宽的打印图象。如图 4 所示，当纸张 20 移动通过打印头 2 时，在纸张 20 上进行打印。

2.1 4 英寸打印头的组成

4 英寸长的打印头 2 包括 8 段，每段的长度为 1/2 英寸。段 21 的每一段在纸张的不同部分上打印二值青、品红和黄点，以生成最终图象。各段的位置如图 5 所示。

由于打印头 2 以 1600dpi 的精度打印圆点，所以每个圆点的直径为 22.5μm，间隔为 15，875μm。因此，每段（长为 1/2 英寸）打印 800 点，其中 8 段的对应位置为：

表 1. 每段访问的最终图象点

段	第一点	最后一点
0	0	799
1	800	1,599
2	1,600	2,399
3	2,400	3,199
4	3,200	3,999
5	4,000	4,799
6	4,800	5,599
7	5,600	6,399

尽管每段生成最终图象的 800 点，但是利用二值青、品红和黄墨的组合表示每一点。由于打印是二值的，所以应对输入图象进行抖动或误差扩散以获得最佳结果。

每段 21 包括 2400 个喷嘴：青、品红和黄各 800 个。一个 4 英寸的打印头 2 包含 8 段 21，总计 19,200 个喷嘴。

2.1.1 对各段内的喷嘴进行分组

对各段 21 内的喷嘴 22 进行分组，目的是为了物理稳定性，并且将打印期间的功耗降到最底程度。在物理稳定性方面，10 个喷嘴共享同一墨池。在功耗方面，分组的目的是启用低速和高速打印模式。

打印头 2 支持两种打印速度，以便能在不同产品配置中提供不同的速度/功率的权衡。

在低速打印模式中，同时加热 4 英寸打印头 2 的 96 个喷嘴 22。应使被加热喷嘴 22 的间距最大，因此对每段的 12 个喷嘴 22 进行加热。要加热所有的 19,200 个喷嘴，必须加热 200 组不同喷嘴，每组包括 96 个喷嘴。

在高速打印模式中，同时加热 4 英寸打印头 2 的 192 个喷嘴 22。应使被加热喷嘴 22 的间距最大，因此对每段的 24 个喷嘴进行加热。要加热所有的 19,200 个喷嘴，必须加热 100 组不同喷嘴，每组包括 192 个喷嘴。

低速模式的功耗为高速模式的一半。然而，请注意，在两种情况中，打印每一行、每一页所消耗的功率是相同的。

在诸如电池供电的 Printcam 情况中，功耗需求规定使用低速打印。

2.1.1.1 10 个喷嘴构成一个容器

一个容器 23 包括 10 个喷嘴 22，10 个喷嘴 22 共享同一墨池。5 个喷嘴 22 在一行上，5 个在另一行上。每个喷嘴 22 生成一个直径为 22.5μm、间隔 15.875μm 的圆点。图 6 表示单个容器的结构，喷嘴 22 的序号表示加热顺序。

尽管按照以上顺序对喷嘴 22 进行加热，但是喷嘴 22 的关系以及打印页上的点的物理位置是不同的。其中一行上的喷嘴 22 代表该页上某一行中的偶数点，而另一行上的喷嘴代表该页上相邻行中的奇数点。

图 7 表示具有喷嘴 22 的同一容器 23，序号表示其加载顺序。

因此，容器 23 内的喷嘴 22 的逻辑间隔为 1 个点的宽度。喷嘴 22 之间的精确距离依赖于 Memjet 加热机制的性质。打印头 2 设计有交错喷嘴，交错喷嘴的目的是与纸张 20 的移动匹配。

2.1.1.2 3 个容器构成一个色品容器

将每种颜色（青、品红和黄）的一个容器 23 分组为一个色品容器 24。一个色品容器 24 代表由 10 个点组成的同一水平集合在不同行上的不同颜色分量。不同颜色容器 23 之间的精确距离取决于 Memjet 操作参数，并且随 Memjet 设计的不同而不同。该距离被认为是固定数目的点宽度，因此，在打印时必须考虑该距离：青色喷嘴打印的圖点与品红或黄色喷嘴打印的圖点在不同行上。打印算法必须考虑颜色之间多达 8 个点宽度的可变距离（有关细节见表 3）。图 8 表示单个色品容器 24。

2.1.1.3 5 个色品容器构成一个色品群

将 5 个色品容器 24 组织为一个色品群 25。由于每个色品容器包含 30 个喷嘴 22，所以每个色品群包含 150 个喷嘴 22：青、品红和黄色喷嘴各 50 个。图 9 表示其结构，其中色品容器序号为 0-4。请注意，为了清晰而夸大了相邻色品容器之间的距离。

2.1.1.4 2 个容器群构成一个相群

将 2 个容器群 25 组织为一个相群 26。由于在指定的加热相位（阶段）中同时加热某个相群内的喷嘴组 23（以下详细说明），所以称其为相群 26。利用 2 个容器群组成一个相群的目的在于，通过 2 条 PodgroupEnable（启用容器群）线进行低速和高速打印。

在低速打印中，在指定的加热脉冲中，只设置两条 PodgroupEnable 线中的一条，因此两个容器群中只有一个容器群加热喷嘴。在高速打印中，同时设置两条 PodgroupEnable 线，因此两个

容器群加热喷嘴。所以，低速打印需要的时间是高速打印所需时间的两倍，原因在于高速打印同时加热的喷嘴数是低速打印的两倍。

图10表示相群的组成。为了清晰而夸大了相邻容器群之间的距离。

2.1.1.5 2个相群构成一个加热群

将2个相群（相群A和相群B）组织为一个加热群27，每段4个相群。由于它们同时加热同一喷嘴27，所以将其称为加热群27。两条启用线（AEnable和BEnable）允许在不同加热相位中独立加热相群A的喷嘴和相群B的喷嘴。图11表示其结构。为了清晰而夸大了相邻分组之间的距离。

2.1.1.6 喷嘴分组概要

表2为打印头中喷嘴分组的概要。

表2.4英寸打印头的喷嘴分组

组名	构成	重复比例	喷嘴数
喷嘴22	基本元件	1:1	1
容器23	喷嘴/容器	10:1	10
色品容器24	容器/CMY色品容器	3:1	30
容器群25	色品容器/容器群	5:1	150
相群26	容器群/相群	2:1	300
加热群27	相群/加热群	2:1	600
段21	加热群/段	4:1	2,400
4英寸打印头2	段/4英寸打印头	8:1	19,200

2.2 加载和打印周期

一个4英寸的打印头2总共包含19,200个喷嘴22。一个打印周期包括根据打印信息加热所有喷嘴。一个加载周期包括向打印头加载需要在随后的打印周期中打印的信息。

各喷嘴 22 具有一个相关的 *NozzleEnable* 位，后者确定该喷嘴是否在打印周期中加热。通过一组移位寄存器加载 NozzleEnable 位（每喷嘴一个）。

逻辑上，每段有 3 个移位寄存器（每种颜色一个），每个寄存器的长度为 800（比特）。当将 NozzleEnable 位移动到指定颜色的移位寄存器时，将其引向交错脉冲上的上下喷嘴。在内部，每个 800 深度的移位寄存器由两个 400 深度的移位寄存器组成：一个用于上喷嘴，一个用于下喷嘴。将交错位移动到内部交错寄存器中。然而，至于有关的外部接口，只有一个 800 深度的移位寄存器。

在全部加载所有移位寄存器后（800 个加载脉冲），以并行方式将所有位传送到适当的 NozzleEnable 位中。这等同于并行传送 19,200 比特。当传送发生时，打印周期开始。只要在打印周期结束时并行加载所有的 NozzleEnable 位，打印周期和加载周期就可以同时出现。

2.2.1 加载周期

加载周期牵涉到向打印头的移位寄存器加载下一个打印周期的 NozzleEnable 位。

每段 21 具有 3 个与青、品红和黄色移位寄存器有直接关系的输入。这些输入称为 *CDataIn*、*MDataIn* 和 *YDataIn*。由于有 8 段，所以每个 4 英寸打印头共有 24 条彩色输入线。*SRClock* 线路上的一个脉冲（由 8 段共享）将 24 位传送到适当的移位寄存器中。交错脉冲分别将这些位传送到上下喷嘴中。由于有 19,200 个喷嘴，所以该传送操作共需 800 个脉冲。在传送全部 19,200 位之后，共享 *PTransfer* 线路上的一个脉冲将移位寄存器中的数据并行传送到适当的 NozzleEnable 位。

通过 PTransfer 上的脉冲的并行传送必须在打印周期结束后发生。否则，正在打印的行的 NozzleEnable 位将是错误的。

由于利用单一 SRClock 脉冲加载全部 8 段 21，所以打印过程必须按照正确顺序生成打印头的数据。例如，第一 SRClock 脉冲将传送下一打印周期的点 0、800、1600、2400、3200、4000、4800 和 5600 的

CMY 位。第二 SRClock 脉冲将传送下一打印周期的点 1、801、1601、2401、3201、4001、4801 和 5601 的 CMY 位。在 800 个 SRClock 脉冲后，提供 PTransfer 脉冲。

重要的是，在同一打印周期中打印的奇数和偶数 CMY 输出并不出现在同一物理输入线上。打印头内奇偶喷头之间的物理间隔以及不同颜色的喷嘴之间的间隔，确保它们在该页的不同行上生成圆点。当向打印头加载数据时，必须解决有关差异。各行中的实际差异取决于打印头适用喷墨机制的特性。利用变量 D_1 和 D_2 定义其差异，其中 D_1 是不同颜色的喷嘴之间的距离，D_2 是相同颜色的喷嘴之间的距离。表 3 表示在前 4 个脉冲上向打印头的段 n 传送的所有点。

表 3. 向 4 英寸打印头传送的点的顺序

脉冲	点	黄线	品红线	青线
1	800S[a]	N	N+D_1[b]	N+2D_1
2	800S+1	N+D_2[c]	N+D_1+D_2	N+2D_1+D_2
3	800S+2	N	N+D_1	N+2D_1
4	800S+3	N+D_2	N+D_1+D_2	N+2D_1+D_2

a. S=段号（0-7）

b. D_1=某种颜色的喷嘴和下一种颜色的喷嘴之间的线数（可能=4-8）

c. D_2=同一颜色的两行喷嘴之间的线数（可能=1）

等等，直至全部 800 个脉冲。

可以以 20 MHz 的最高速度，将数据输入到打印头中，20 MHz 的速度将在 40μs 内加载下一行的所有数据。

2.2.2 打印周期

一个 4 英寸打印头 2 包含 19,200 个喷嘴 22。要同时加热所有喷嘴

将消耗大量功率，并且会引起墨回补问题和喷嘴干扰。因此，定义两种加热模式：高速打印模式和低速打印模式：

- 在低速打印模式中，有 200 个相位，每个相位加热 96 个喷嘴。这等同于每段 12 个喷嘴，即每个加热群 3 个喷嘴。
- 在高速打印模式中，有 100 个相位，每个相位加热 192 个喷嘴。这等同于每段 24 个喷嘴，即每个加热群 6 个喷嘴。

根据以下信号确定在指定加热脉冲中加热的喷嘴：

- 3 比特 *ChromapodSelect*（从加热群 27 中选择 5 个色品容器 24 中的一个色品容器）
- 4 比特 *NozzleSelect*（从容器 23 的 10 个喷嘴 22 中选择一个喷嘴）
- 2 比特的 *PodgroupEnable* 线路（选择要加热的 0、1 或 2 容器群 25）

当设置一条 PodgroupEnable 线路时，只有指定的容器群的 4 个喷嘴加热，作为 ChromapodSelect 和 NozzleSelect 确定的喷嘴。当设置两条 PodgroupEnable 线路时，两个容器群同时加热其喷嘴。对于低速模式，需要两个加热脉冲，即 PodgroupEnable = 01 和 10。对于高速模式，只需要一个加热脉冲，即 PodgroupEnable = 11。

由 AEnable 和 BEnable 线路指定加热脉冲的持续时间，AEnable 和 BEnable 分别加热所有加热群中的相群 A 和相群 B 喷嘴。加热脉冲的典型持续时间为 1.3-1.8μs。脉冲的持续时间取决于墨的粘性（取决于温度和墨特性）以及打印头使用的功率。有关为补偿温度变化而利用打印头之反馈的详细信息，请参见第 18 页上的 2.3 一节。

AEnable 和 BEnable 是独立线路，目的是加热脉冲能够重叠。因此，200 个相位的低速打印周期包括 100 个 A 相位和 100 个 B 相位，从而能够有效提供 100 组相位 A 和相位 B。同样，100 个相位的高速打印周期包括 50 个 A 相位和 50 个 B 相位，从而能够有效提供 50 组相位 A 和相位 B。

图 12 表示典型打印周期中的 AEnable 和 BEnable 线路。在高速打印中有 50 个 2μs 的周期，而在低速打印中有 100 个 2μs 的周期。

对于高速打印模式，加热顺序为:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 2, NozzleSelect 0, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 3, NozzleSelect 0, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 4, NozzleSelect 0, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 0, NozzleSelect 1, PodgroupEnable 11（相位 A 和 B）
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 11（相位 A 和 B）
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 11（相位 A 和 B）

对于低速打印模式，加热顺序类似。对于 PodgroupEnable 为 11 的高速模式的每个相位，替换为以下两个相位 PodgroupEnable = 01 和 10:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 01（相位 A 和 B）
- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 10（相位 A 和 B）
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 01（相位 A

和 B）

- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 10（相位 A 和 B）
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 01（相位 A 和 B）
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 10（相位 A 和 B）
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 01（相位 A 和 B）
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 10（相位 A 和 B）

当喷嘴 22 加热时，大约需要 100μs 进行回补。不能在回补时间过去前加热喷嘴 22。从而每行的最高打印速度为 100μs。在高速打印模式中，打印一行的时间是 100μs，因此从加热某一行中的喷嘴到加热下一行中的喷嘴的时间与回补时间匹配，所以高速打印模式是可接受的。低速打印模式比高速打印模式慢，因此也是可接受的。

加热喷嘴 22 会在有限时间内在该喷嘴之容器 23 的公用墨池内造成声学扰动。该扰动干扰同一容器 23 内的其他喷嘴的加热。所以，容器内的喷嘴的加热应尽量偏移。因此，我们加热色品容器 24 内的 3 个喷嘴（每种颜色一个喷嘴 22），然后移动到容器群 25 内的下一个色品容器 24。

- 在低速打印模式中，独立加热容器群 25。因此，在第一色品容器再次加热前，两个容器群中的 5 个色品容器 24 必须全部加热，共计 10×2μs 周期。所以，每 20μs 加热容器 23 一次。
- 在高速打印模式中，同时加热容器群 25。因此，在第一色品容器再次加热前，一个容器群中的 5 个色品容器 24 必须全部加热，

共计 5×2μs 周期。所以，每 10μs 加热容器 23 一次。

由于墨通道的长度为 300μm，并且声音在墨中的速度约为 1500m/s，所以墨通道的谐振频率为 2.5MHz，从而低速模式允许 50 个谐振周期供声学脉冲衰减，而高速模式允许 25 个谐振周期。因此在两种情况中都能将声学干扰降到最底程度。

2.2.3 采样定时

例如，正如 Printcam 要求的那样，考虑在 2 秒内打印一张 4″×6″相片的定时。为了在 2 秒内打印一张相片，4 英寸打印头必须打印 9600 行（6 × 1600）。在 2 秒内完成 10,000 行，相当于每行 200μs。必须在该时间内完成一个打印周期和一个加载周期。另外，打印头外部的物理处理必须适当地移动相纸。

从打印观点看，低速打印模式允许 4 英寸打印头在 200μs 内打印一整行。在低速打印模式中，每加热脉冲中有 96 个喷嘴 22 加热，从而能够在特定时间内打印一整行。

打印头 2 的 800 个 SRClock 脉冲（每个时钟脉冲传送 24 位）必须在 200μs 的行时间内发生。SRClock 脉冲的长度不能超过 200μs/800 = 250ns，这表示打印头的时钟周期为 4MHz。另外，计算各比特值（19,200 个喷嘴的每个喷嘴）的平均时间不能超过 200μs/19,200 = 10ns。从而需要按如下速度运行的点生成器：

- 100 MHz，生成 1 位（点）/周期
- 50 MHz，生成 2 位（点）/周期
- 25 MHz，生成 4 位（点）/周期

2.3 打印头的反馈

打印头 2 生成几行反馈（累计 8 段的反馈）。反馈线用于调整加热脉冲的定时。尽管各段生成相同反馈，但是所有段的反馈共享相同

的三态总线。所以，每次只有一段 21 能够提供反馈。

SenseSegSelect 线路上的脉冲与有关青色数据的逻辑与启用该段的传感线路。反馈传感线路将来自所选段，直至下一个 SenseSegSelect 脉冲。反馈传感线路如下：

- *Tsense* 通知控制器打印头有多热。这允许控制器调整加热脉冲的定时，因为温度影响墨的粘性。
- *Vsense* 通知控制器致动器能使用的电压。从而控制器能够通过调整脉冲宽度来补偿其电压下降的电池或高压电源。
- *Rsense* 通知控制器致动加热器的电阻率（每方欧姆）。从而控制器能够调整脉冲宽度以便维持在恒定功率，而不考虑加热器的电阻率。
- *Wsense* 通知控制器加热器的关键部分的宽度，由于平板印刷和蚀刻，其宽度会有±5%的变化。从而控制器能够适当地调整脉冲宽度。

2.4 特殊周期

2.4.1 预热周期

打印过程的一个显著趋势是保持在平衡温度。为了确保第一次选择的打印相片具有一致的圆点尺寸，必须在打印任何圆点前满足平衡温度。这是通过预热周期实现的。

预热周期包含应用于所有喷嘴的一个加载周期（即，设置所有要加热的喷嘴），该周期为 1 秒，以及应用于各喷嘴的许多短暂加热脉冲。脉冲的持续时间必须不足以喷射墨滴，但足以加热墨。每个喷嘴总共需要 200 个脉冲，按照标准打印周期的相同次序发出脉冲。

在预热模式期间，由 Tsense 提供反馈，直至达到平衡温度（约高于环境 30°C）。预热模式的持续时间约为 50 毫秒，并且取决于墨的成分。

在每个打印作业前执行预热。因为在向打印机传送页面数据时执行预热，所以这并不影响打印机的性能。

2.4.2 清洁周期

为了降低喷嘴堵塞的可能性，可以在每个打印作业前执行清洁周期。每个喷嘴向吸收海绵喷射几次。

清洁周期包含应用于所有喷嘴的一个加载周期（即，设置所有要加热的喷嘴），该周期为 1 秒，以及应用于各喷嘴的许多加热脉冲。通过与标准打印周期相同的喷嘴加热次序，清洁各喷嘴。各喷嘴 22 的加热次数取决于墨的成分，以及打印机的闲置时间，如同预热一样，清洁周期并不影响打印机的性能。

2.5 打印头接口概要

一个 4 英寸的打印头 2 具有以下连接:

表 4.4 英寸打印头连接

名称	引线数	描述
ChromapodSelect	3	选择要加热的色品容器（0-4）
NozzleSelect	4	从容器中选择要加热的喷嘴（0-9）
PodgroupEnable	2	启用要加热的容器群（选择: 01、10、11）
AEnable	1	相群 A 的加热脉冲
BEnable	1	相群 B 的加热脉冲
CDataIn[0-7]	8	段 0-7 的青色移位寄存器的青色输入
MDataIn[0-7]	8	段 0-7 的品红色移位寄存器的品红色输入
YDataIn[0-7]	8	段 0-7 的黄色移位寄存器的黄色输入
SRClock	1	SRClock 上的一个脉冲（ShiftRegisterClock）将 CDataIn[0-7]、MDataIn[0-7]和 YDataIn[0-7]的当前值加载到 24 个移位寄存器中。
PTransfer	1	将移位寄存器中的数据并行传送到内部 NozzleEnable 位（每个喷嘴一位）。
SenseSegSelect	1	SenseSegSelect 上的一个脉冲与 CDataIn[n]上的

		数据的逻辑与选择段 n 的传感线路。
Tsense	1	温度检测
Vsense	1	电压检测
Rsense	1	电阻率检测
Wsense	1	宽度检测
逻辑 GND	1	逻辑接地
逻辑 PWR	1	逻辑功率
V-	母线	致动器接地
V+		致动器功率
总计	44	

在 4 英寸打印头的内部，每段到连接容器的电路为：

表 5.4 英寸打印头内部段的连接

名称	引线数	描述
ChromapodSelect	3	选择要加热的色品容器（0-4）
NozzleSelect	4	从容器中选择要加热的喷嘴（0-9）
PodgroupEnable	2	启用要加热的容器群（选择：01、10、11）
AEnable	1	相群 A 的加热脉冲
BEnable	1	相群 B 的加热脉冲
CDataIn	1	青色移位寄存器的青色输入
MdataIn	1	品红色移位寄存器的品红色输入
YDataIn	1	黄色移位寄存器的黄色输入
SRClock	1	SRClock 上的一个脉冲（ShiftRegisterClock）将 CDataIn、MDataIn 和 YDataIn 的当前值加载到 3 个移位寄存器中。
PTransfer	1	将移位寄存器中的数据并行传送到内部 NozzleEnable 位（每个喷嘴一位）。
SenseSegSelect	1	SenseSegSelect 上的一个脉冲与 CDataIn 上的数

		据的逻辑与选择该段的传感线路。
Tsense	1	温度检测
Vsense	1	电压检测
Rsense	1	电阻率检测
Wsense	1	宽度检测
逻辑 GND	1	逻辑接地
逻辑 PWR	1	逻辑功率
V-	21	致动器接地
V+	21	致动器功率
总计	65	（65×8 段 =520 段，对所有段）

3 图象处理链

以上各节只设计 PCP 功能性的最高级概要，将 CFA 图象映射为各种输出打印格式。实际上，从图象传感器中取得图象，然后生成高质量的输出打印涉及许多步骤。可以将高级处理划分为两个图象处理链，每个处理链具有许多步骤:

- 图象获取链
- 打印链

图象获取链涉及从图象传感器中获取图象，并且在 Printcam 内本地存储。打印链涉及取得存储的图象，然后打印。以上两个处理链按以下方式映射到基础 Printcam 功能性:

- 取象并打印 = 图象获取链，然后是打印链
- 重新打印 = 打印链

例如，用户可以打印一张小图（取象并打印），并且如果对结果满意的话，打印几份标准副本（重新打印）。

本章描述满足 Printcam 质量要求的独立图象处理链的实现。在此阶段中，我们并不精确考虑如何以硬件方式实现该处理，而是考虑必

须进行哪些处理。必须将这些功能映射到 PCP 内的各种装置。

不管 PCP 的实现，有许多约束:

- 输入图象是基于连续色调 RGB 图象的 CFA。
- 输出图象用于 CMY 颜色空间中的 Memjet 打印头（二值点，分辨率为 1600dpi），并且输出宽度相同（4 英寸宽）。

3.0.1 支持的打印格式

如表 6 所示，PCP 3 支持各种输出打印格式。在所有情况中，图象的宽度总是 4 英寸（与打印头的宽度匹配）。只有打印输出的长度改变。

表 6. 支持的图象格式

格式名	长宽比	输出尺寸（英寸）	输出分辨率（1600dpi）	旋转
标准 30	2:3	4″×6″	6400×9600	90
护照 31	2:3	4″×6″	6400×9600	90
全景 33	4:6	4″×12″	6400×19200	90
小图 32	2:3	4″×2.67″	6400×4267	0

图象传感器并不提供方向信息。所有图象都是以相同分辨率（1500 × 1000）获取的，并且可能需要在打印输出前旋转 90 度。图 13 表示获取的 CFA 图象和支持的各种打印格式之间的映射。请注意，尽管将该图表示为逆时针旋转 90 度，也可以顺时针或逆时针旋转该图象。

3.1 图象获取链

图象获取链负责从图象传感器中取得图象，并在 Printcam 内本地存储。图象获取链涉及许多只在图象获取期间执行的处理。图 14 表示图象获取链，以下几部分详细说明其子部件。

3.1.1 图象传感器 1

输入图象来自图象传感器 1。尽管可以使用各种图象传感器，但是我们只考虑拜尔彩色滤镜阵列（CFA）。拜尔 CFA 具有许多本文定义的属性。

假设已经充分过滤了 CMOS 传感器 1（通过取象透镜）获取的图象，以消除锯齿现象。传感器本身的长宽比为 3:2，具有 1500×1000 分辨率的采样。最可能的象素结构为拜尔彩色滤镜阵列（CFA），其中图 15 表示在 2G 镶嵌面中排列的各 2×2 象素块。

R、G、B（分别对应于红、绿、蓝）的各连续色调采样是 10 比特。请注意，镶嵌面的每个象素仅包含有关 R、G、B 之一的信息。必须在打印输出前估计丢失的彩色信息。

认为 CFA 执行适当的固定模式噪声（PFN）抑制。

3.1.2 线性化 RGB 40

- 图象传感器 40 未必有完整的线性响应。因此，必须认为来自 CFA 的 10 比特 RGB 采样是非线性的。通过查找表（每种颜色一个）将这些非线性采样转换为 8 比特的线性采样。

来自 CFA 0 行、2 行、4 行等的象素编入 R 和 G 表，而来自 CFA 1 行、3 行、5 行等的象素编入 G 和 B 表。该处理完全独立于照相机的方向。图 16 表示该处理。每个查找表需要的总存储量为 $2^{10}\times 8$ 比特。因此，3 个查找表 45 共需 3KB（3×2^{10} 字节）。

3.1.3 平面化 RGB41

由于象素的拜尔镶嵌面的性质，所以从 CFA 获得的象素使其颜色平面交错。这意味着在偶数水平线上，一个红色象素后跟一个绿色象素，然后是另一个红色象素，不同颜色平面互相交错。在某些图象处理系统中，交错格式非常有用。然而，在 Printcam 处理系统中，该算

法在平面 RGB 上更有效。

平面化的图象是已经分成其组成色的图象。在 CFA RGB 的情况中，有 3 个独立图象：一个仅包含红色象素的图象，一个仅包含蓝色象素的图象，以及一个仅包含绿色象素的图象。请注意，每个平面仅代表实际采样颜色的象素。在平面化处理中不进行重新采样。因此，不互相登记 R、G、B 平面，并且 G 平面为 R 平面或 B 平面的两倍。图 17 表示该处理。

实际处理非常简单 — 取决于读取的象素的颜色，将输出象素发送到适当颜色平面的图象中的下一位置（因此，与 CFA 中的方向相同）。

红色 45 和蓝色 47 平面图象为原始 CFA 图象的四分之一。它们是每个维数的分辨率的一半。因此，红色和蓝色图象是 750×500 象素，在 CFA 空间（1500×1000）中，红色图象在 x 和 y 维数上偏离蓝色图象一个象素。

尽管绿色平面图象 46 的大小是原始 CFA 图象的一半，但是其布局设计并不象红色和蓝色那样简单。原因在于绿色平面的交错排列布局。在绿色平面的一行上是*奇数*象素，在绿色平面的下一行上是*偶数*象素。因此，绿色平面的交错行代表 CFA 图象内的奇数象素和偶数象素。所以绿色平面图象是 750×1000 象素。从而具有重新采样处理分支（参见 28 页上的 “重新采样 64”）。

3.1.4 存储的图象 42

将线性化的 RGB 图象的各颜色平面写入存储器中，以便临时存储。存储器应为快闪存储器 11，以便在关闭电源后继续保存图象。

平面化的线性 RGB 图象需要的总存储量是 1,500,000 字节（约为 1.5MB），其排列为：

- R: 750×500=375,000 字节
- G: 750×500=375,000 字节
- B: 750×1000=750,000 字节

3.2 打印链

打印链包括从存储器中取得现有图象 42，然后打印到 Memjet 打印机 2。通常获取图象后就打印该图象，尽管也可以重新打印（即，无需重新获取）。

为了从获取的 CFA 图象中生成高质量的打印，图象处理链包括许多步骤。图 18 表示打印链。将打印链划分为三个工作分辨率。第一分辨率是原始图象获取空间 50（与 CFA 的图象空间相同），第二分辨率是中分辨率 51（每行有 1280 个连续色调象素），第三分辨率是打印机分辨率 52，每行有 6400 个二值点。

3.2.1 输入图象

正如 3.1.4 部分中说明的图象获取链的存储方式一样，输入图象是以平面形式存储的线性化的 RGB 图象 42。

3.2.2 收集统计数字

在执行诸如白平衡和范围扩展之类的处理前，需要收集整个图象的许多统计数字。对于特定的获取图象 42 的所有打印，只需收集一次统计数字，并且可以分别收集红、绿、蓝色平面图象的统计数字。

3.2.2.1 构建直方图

第一步是构建各颜色平面的 8 比特值的直方图。每个 1500×1000 CFA 图象总共包含:

- 375,000 个红色象素（至少需要一个 19 比特的计数器）
- 375,000 个蓝色象素（至少需要一个 19 比特的计数器）
- 750,000 个绿色象素（至少需要一个 20 比特的计数器）

因此，存储直方图需要一个 256×20 比特的表。

正如以下伪码所示，构建直方图的过程非常简单:

```
For I = 0 to 255
  Entry[I] = 0
EndFor
For Pixel = ImageStart to ImageEnd
  p = Image[Pixel]
  Entry[p] = Entry[p] + 1
EndFor
```

3.2.2.2 确定高低阈值

在构造颜色平面的直方图后，可以使用直方图来确定高低阈值。在打印期间，白平衡和范围扩展的自动化处理需要以上阈值。

根据直方图中象素数的阈值，我们认为可以“牺牲”*n%的最暗象素*，从而使这些象素相等。同样，我们认为可以“牺牲”*n%的最亮象素*，从而使这些象素相等。期望 n 的准确值为 5%，这取决于 CFA 的响应特性。

确定 n%的最暗值很简单。包括从 0 计数开始向上（即，0，1，2，3 等）遍历该颜色平面的直方图，直至到达总数的 n%，即从零开始遍历到某个总数。我们认为这些值中的最大值为该颜色平面的低阈值。尽管这些最暗值之间有差异，但是为了范围扩展和颜色平衡可以牺牲这些差异。

确定 n%的最亮值的过程类似。包括从 255 计数开始向下（即，255，254，253 等）遍历该颜色平面的直方图，直至到达总数的 n%，即从 255 开始遍历到某个总数。我们认为这些值中的最小值为该颜色平面的高阈值。尽管这些最亮值之间有差异，但是为了范围扩展和颜色平衡可以牺牲这些差异。

在离开 0 或 255 一定距离后停止的原因在于补偿以下两类图象：

- 原始动态范围低的图象，或
- 图象中没有白色或黑色的图象

在以上两种情况中，我们不希望牺牲n%的高低值，因为开始范围较小。我们可以将高73和低72阈值设置为实际采样的象素值的范围外。精确距离取决于CFA，但是将是两个常数。

图19表示颜色平面的采样颜色范围。请注意，尽管图象颜色平面的象素范围有可能是0-255，但是该特定图象的范围较小。同时，n%的直方图范围70、71表示的低端范围70大于高端范围71。原因在于与低端相比，该直方图包含的高值象素更接近。

必须分别确定各颜色平面的高73和低72阈值。该信息用于计算白平衡和范围扩展中使用的范围比例和偏移因子。

以下伪码表示确定两个阈值的过程（为了查找低阈值，StartPosition=255，Delta=1。为了查找高阈值，StartPosition=0，Delta=-1）。该伪码假设Threshold是一个8比特值，其中在进行加法时回绕。

```
Threshold = StartPosition
Total = 0
TotalDelta = 0
While ((TotalDelta < MaxDelta) AND (Total < MaxPixels))
 Threshold = Threshold + Delta
 Total = Total + Entry[Threshold]
 TotalDelta = TotalDelta + 1
EndWhile
Return Threshold
```

3.2.3 旋转图象61

在获取、打印和重新打印过程中，旋转图象61是一个可选步骤。

正如图13所示，不同打印格式需要相对与CFA方向将该图象旋转0度或90度。旋转量取决于当前选择的打印格式。尽管旋转方向并

不重要（由于新方向仅仅是为了方便打印头的宽度，所以可以顺时针或逆时针旋转），旋转方向影响 3 个颜色平面的相关读数。表 7 概要说明各种打印格式需要从原始 CFA 方向旋转的角度。

表 7. 各种打印格式需要从 CFA 方向旋转的角度

打印格式	旋转
标准 30	90
护照 31	90
全景 32	90
小图 33	0

由于只需旋转 0 度或 90 度，所以在旋转处理中不丢失信息。如果旋转 0 度，则可以逐行读取图象，如果旋转 90 度，可以逐列读取图象。读取 3 个颜色平面必须考虑旋转方向。

3.2.4 白平衡 62 和范围扩展 63

相片很少是在理想照明条件下拍摄的。即使非常“完美的照明条件”也是充满主观性的，包括照相师方面的主观性和主体方面的主观性。然而，在所有情况中，利用来自光源（如日光或室内照明）或相片主体特有发光体（如霓虹灯）的光线，照射相片主体。

在大部分照明条件中，照相师看作“白”光的光源通常并不是白色的。例如，室内光通常具有黄色光线，该黄色光线将出现在未修正的相片上。对大部分人而言，最终未修正相片上出现黄色光线是不正常的。尽管与拍摄相片时的观察条件一致，但它与物体的感觉色彩不一致。因此，在打印输出前对相片进行白平衡是很关键的。

同样，如果扩展颜色的动态范围以便与各颜色平面的满标度一致，则会感觉到图象的质量较高。在以更高的分辨率对图象重新采样前，以上处理非常有用。如果动态范围较高，则在内插的象素位置中可使用中间值，从而避免出现分级或块状图象。范围扩展的目的是向实际

采样值提供全部 256 个值范围。在最佳情况中，将最小值映射为 0，将最大值映射为 255。将所有中间值按比例映射为 0 到 255 之间的值。

数学上，所执行的操作是将 LowThreshold 72 变化为 0，然后乘以某个比例。公式为：

$$Pixel' = (Pixel - LowThreshold) \times RangeScaleFactor$$

$$\text{其中 } RangeScaleFactor = \frac{256}{(HighThreshold - LowThreshold)}$$

将 RangeScaleFactor 限制在最大值，以免将范围扩得太大。；有关计算 LowThreshold 72 的细节，请参见 3.2.2 一节中的 “收集统计数字”。每种颜色的 LowThreshold 和 RangeScaleFactor 各不相同，并且每个图象只需计算一次。

如图 20 所示，可以同时进行以上两项工作。

由于此步骤涉及缩放处理，因此在映射值中可以保留分数部分，如，值 12 映射到 5.25。将一个 10 比特的结果（8 比特整数，2 比特分数）传送到图象处理链的下一阶段，而不是丢弃分数部分。虽然提供的存储器不能以高于 8 比特的分辨率存储整幅图象，但是可以在重新采样阶段中使用更高分辨率。因此，输入图象是 8 比特，输出图象的每种颜色分量是 10 比特。图 21 表示其逻辑处理。

在进行减法运算时，重要的是具有最低限度 0，从而将低于 LowThreshold 72 的所有值映射为 0。同样，乘法运算结果的整数部分必须具有最高限度 255，从而将高于 HighThreshold 73 的输入值映射为 255。

3.2.5 重新采样 64

CFA 图象只提供每个象素（x,y）坐标单一颜色分量。为了生成最终打印图象，需要获得各象素的其他颜色分量。最后，我们需要各象素的青、品红和黄色分量，但是，要获得青、品红和黄色，我们需要

红、绿、蓝。如果有某个特定位置的红色分量，则需要估计蓝和绿。或则如果有绿色分量，则需要估计红和蓝。

即使拥有各 CFA 分辨率象素的红、绿、蓝分量，该 CFA 分辨率图象也不是最终输出分辨率。另外，尽管输出格式不同，但是打印图象的物理宽度不变（4 英寸，分辨率为 1600dpi）。因此，打印头的固定宽度为 6400 点。

需要考虑两种极端情况:

- *内插到 CFA 分辨率（最小内插），然后执行锐化、颜色转换。最后按比例提高到打印分辨率。*其优点在于，锐化内核固定，并且以低分辨率进行颜色转换。然而，其缺点是，为内插图象存储的各颜色分量高于 8 比特，否则在最终按比例提高到打印分辨率时，将不能正确内插中间值。另一个缺点是，需要一个按比例提高部件，该部件能够在每个周期内生成 1 个打印分辨率内插值。
- *内插到打印分辨率，然后执行锐化和颜色转换。*其优点在于，只有一个重新采样处理，从而提供最高准确度。然而，其缺点是，需要一个按比例提高部件，该部件能够在每个周期内生成 1 个双三次内插值，同时平均在每个周期内执行锐化和颜色转换。锐化内核必须足够大，以便将 CFA 分辨率内核应用于高分辨率图象。更糟的是，对锐化处理，至少必须在输出图象上保持 3 个窗口（每个窗口包含 6400 行），这是因为在一个打印周期中，青、品红和黄点表示互不相同的 6 行中的点。

以上两种情况都没有考虑以下事实，即最终打印输出是二值图象而不是连续色调图象。因此，可以对重新采样进行折衷，并且获得最佳结果。

解决方案是内插到*中分辨率*。以中分辨率进行锐化和颜色变换，然后按比例提高到打印分辨率。中分辨率必须足够低，以采用较小的锐化内核和颜色变换定时。但是，中分辨率必须足够高，从而在按比例提高到打印分辨率的二值图象时不损失质量。其效果必须与一次（而

不是二次）内插到打印分辨率的效果相同。

由于打印图象是以 1600dpi 抖动二值点的方式打印的，所以能够利用 320dpi 的连续色调图象无损表示。因此，1280 连续色调象素的中分辨率提供 6400 二值点上的无损质量。从 1280 提高到 6400 的缩放比率为 1:5。

为了确定重新采样的最佳方法，最好考虑与 CFA 分辨率有关的各颜色平面。图 22 表示旋转 0 度的颜色平面。

3.2.5.1 红色 45 和蓝色 47

考虑红色 45 和蓝色 47 平面，通过将每个维数的采样象素数放大 2 倍，可以创建该颜色平面的全 CFA 分辨率图象。借助重建过滤器（如 Lanczos 或指数过滤器），可以生成中间象素。因为内核是对称的，所以只需一维内核。由于红色和蓝色在 CFA 采样空间内的初始表示的偏移不同，所以内核中的初始位置不同。

将输出坐标（在 1280 空间中）映射到输入坐标依靠图象的当前旋转，原因在于登记旋转象素变化的缘故（0 度或 90 度，取决于打印格式）。因此，对于红色和蓝色，以下关系成立:

$$\left.\begin{aligned} x' &= \left(\frac{x}{mps}\right) + k_1 \\ y' &= \left(\frac{y}{mps}\right) + k_2 \end{aligned}\right\}$$

其中

x，y=中分辨率空间中的坐标

x′，y′=输入空间中的坐标

mps=每个输入空间采样的中分辨率象素

$k_{1,2}$={0，-0.5}取决于旋转

这意味着给定输入空间中的起始位置，通过将 1/mps 的Δx 和Δy 以及 0 分别增加 1279 次，可以生成一行新的中分辨率象素。可以直接

使用输入空间中 x 和 y 的分数部分查找用于图象重建和重新采样的内核系数。

请注意，k_1 和 k_2 为 0 和-0.5，取决于将该图象旋转 0 度还是旋转 90 度。表 8 表示红色和蓝色平面中 k_1 和 k_2 的值，其中旋转 90 度是逆时针旋转。

表 8. k1 和 k2 的旋转结果（旋转是逆时针旋转）

格式	从原始 CFA 图象旋转	红色		蓝色	
		k_1	k_2	k_1	k_2
标准 30	90	0	-0.5	-0.5	0
护照 31	90	0	-0.5	-0.5	0
全景 33	90	0	-0.5	-0.5	0
小图 32	0	0	0	-0.5	-0.5

每个采样的中分辨率象素数（mps）取决于打印格式。假设不旋转时，平面化的 RGB 图象的红色和蓝色平面分辨率为：R: 750×500, B: 750×500，则不同输出格式（见第 17 页上的图 13）的比例系数如表 9 所示。请注意，对于护照图象格式，将整个图象重新采样为输出空间的 1/4。

表 9. 图象格式的红色和蓝色比例系数

格式	映射	mps	1/mps
标准 30	500 ⇒ 1280	2.56	0.390625
护照 31	500 ⇒ 640	1.28	0.78125
全景 33	250 ⇒ 1280	5.12	0.1953125
小图 32	750 ⇒ 1280	1.71	0.5848

正如从表 9 中看到的那样，所有图象都需要按比例放大红色和蓝

色图象。因此，重新采样处理不会引入锯齿现象。

3.2.5.2 绿色 46

不能按照红色和蓝色平面的方法，简单地按比例放大绿色平面 46，原因在于绿色平面的每一行表示不同象素，即交错行上的奇数和偶数象素。尽管就象素数而言，表示绿色图象为 750×1000，但是可以将该图象说成是 1500×500。上述混淆是由绿色象素的交错排列性质造成的，其中沿 x 和 y 维数方向的象素之间的距离不相等，并且不能准确映射到图象重建和重新采样。其他系统为绿色平面重建所使用的内插方法的数目就是一个证明 — 从最近邻域复制到线性内插到双线性内插和启发式重建。

对绿色平面而言，将输出坐标（在 1280 空间中）映射到输入坐标与用于红色和绿色映射的概念相同。该映射依靠图象的当前旋转，原因在于登记旋转象素变化的缘故（0 度或 90 度，取决于打印格式）。因此，对于绿色平面以下关系成立:

$$\left.\begin{aligned} x' &= \left(\frac{x}{mps}\right) + k_1 \\ y' &= \left(\frac{y}{mps}\right) + k_2 \end{aligned}\right\}$$

其中

x，y=中分辨率空间中的坐标

x′，y′=输入空间中的坐标

mps=每个输入空间采样的中分辨率象素

$k_{1,2}$={0，-0.5}取决于旋转

对于红色 45 和蓝色 47 平面，每个采样的中分辨率象素数（mps）取决于打印格式。假设不旋转时，平面化的 RGB 图象的平面分辨率为: R: 750×500，B: 750×500，G: 750×1000，则不同输出格式（见图 13）的比例系数如表 10 所示。请注意，对于护照图象格式，将整个图象重

新采样为输出空间的 1/4。

表 10. 图象格式的绿色平面比例系数

格式	映射	mps	1/mps
标准 30	1000 ⇒ 1280	1.28	0.78125
护照 31	1000 ⇒ 640	0.64	1.5625
全景 33	500 ⇒ 1280	2.56	0.390625
小图 32	1500 ⇒ 1280	0.85	1.17648

以上比例系数允许CFA分辨率输入空间和中分辨率空间之间的坐标映射。然而，如果拥有 CFA 分辨率输入空间中的坐标，则不能象红色和蓝色那样，在采样上执行图象重建和重新采样，原因在于绿色平面的交错排列性质。

为了高质量的图象重建和重新采样，可以认为绿色信道是一个旋转 45 度的图象。如图 23 所示，当按此种方式考虑象素时，高质量图象重建和重新采样方法是清楚的。

考虑图 23，现在沿 x 和 y 方向的采样象素之间的距离相等。正如图 24 所示，采样象素之间的距离为√2。

因此，绿色信道的解决方案是在*旋转空间*中执行图象重建和重新采样。尽管使用用于红色和蓝色重新采样的重建过滤器，但其内核不同。因为绿色采样率和该信号中最高频率之间的关系，与红色和蓝色平面的关系不同。另外，就内核坐标而论，应正规化内核，以便采样之间的√2 距离成为 1（然而，仍然使用重新采样坐标之间的非正规化距离来确定是否出现锯齿）。因此，需要两种变换:

- 第一种变换是将不旋转的 CFA 空间映射到旋转的 CFA 空间。通过将各纵坐标乘以 1/√2 实现该变换，因为旋转 45 度（cos45 = sin45 = 1/√2）。
- 第二种变化是缩放坐标以匹配正规化内核，通过将各纵坐标乘以 1/√2 实现。

以上两种变换联合得到放大系数 1/2。因此，在不旋转的 CFA 空间中，当将 x 增加 k 时，将内核 x 增加 k/2，将内核 y 减少 k/2。同样，当将 y 增加 k 时，将内核 x 增加 k/2，将内核 y 增加 k/2。

通过考虑根据CFA空间输入图象生成一行中分辨率象素时发生的事情，可以说明不同坐标系之间的关系。给定 CFA 输入空间中的 y 纵坐标，我们从 x=0 开始，每次增加 1/mps，共计 1280 次，从而在每个新位置生成一个新象素。可以将在不旋转的 CFA 空间中移动 1/mps，分解为在旋转 CFA 空间中沿 x 和 y 方向的移动。图 25 表示该过程。

由于 cos45=sin45=1/√2，所以在不旋转的 CFA 空间中移动 1/mps 等价于沿 x 和 y 移动 1/(mps/√2)。必须对该数值进行缩放，以匹配正规化内核。缩放等价于再乘以 1/√2。因此，在不旋转的 CFA 空间中移动 1/mps 等价于在内核 x 和内核 y 中移动 1/2mps。表 11 列出不同格式的三种坐标系之间的关系：

表 11. 不同图象格式的绿色平面内核Δ值

格式	比例系数（mps）	不旋转的 CFA 空间Δ 1/mps	旋转的 CFA 空间Δ 1/mps√2	内核Δ 1/2mps
标准	1.28	0.78125	0.552	0.391
护照	0.64	1.5625	1.105	0.781
全景	2.56	0.391	0.276	0.195
小图	0.85	1.17648	0.832	0.601

表 11 表示内核空间中的移动总是乘以一个小于 1 的数，而在旋转的 CFA 空间中，只有护照图象的Δ值大于 1。因此，护照打印格式出现锯齿，而其他格式不会出现锯齿。然而，假设Δ近似等于 1，并且四种图象的大小仅为 1/4，不会出现明显锯齿，特别是在图象获取期间对绿色采取低通滤波时。

3.2.5.3 红、绿、蓝的重建过滤器

要使用的精确重建过滤器取决于许多问题。通常在构造原始信号中使用的采样数、信号重建所需的时间和重新采样图象的质量之间进行折衷。在该情况中，一个令人满意的折衷是正在重建的维数中的 5 个象素采样，以估计位置 X 为中心，即 X-2，X-1，X，X+1，X+2。由于利用 5 个采样点进行重建，因此，卷积内核中的条目只需要 4 个系数。

对于每种颜色分量，创建一个具有 n 个条目的内核系数查找表。每个条目具有 4 个系数。当我们在输出空间中前进时，将输出空间中的变化映射为输入空间和内核空间中的变化。当前内核空间中的分数部分的最高有效位用于编写内核系数表的索引。如果内核表中有 64 项，则利用分数的前 6 位查找系数。64 项足够 Printcam 中的重新采样。

3.2.6 锐化 65

在打印前，必须对 CFA 获取的图象进行锐化处理。理论上，应该在 CFA 分辨率范围内应用锐化过滤器。然而，在该图象的获取分辨率，我们没有各象素的彩色信息。我们仅有指定象素位置的红、蓝或绿色信息。独立锐化每个颜色平面会引起颜色偏移。应该对图象的亮度信道应用锐化，从而指定象素的色度和饱和度不变。

锐化包括将 RGB 图象变换为颜色空间，在颜色空间中，分离亮度与其他颜色信息（如 HLS 或 Lab）80。然后对亮度信道 81 进行锐化 82（通过按比例添加亮度的高通过滤模型）。最后，将整幅图象转换为 RGB 83（或转换为 CMY，由于我们要以 CMY 格式进行打印）。图 26 表示该过程。

然而，如果考虑向该图象添加高通过滤的 L 的结果，即改变图象的亮度，则可以避免许多颜色转换步骤。利用线性 R、G、B 中的相同改变，可以很好地逼近给定象素的亮度改变。因此，只需生成 L（高

通过滤器 L），然后将部分结果平等地应用于 R、G、B。

3.2.6.1 将 RGB 转换为 L 80

考虑 CIE 1976 $L^*a^*b^*$ 颜色平面，其中感觉上 L 是一致的。为了从 RGB 转换到 L（亮度信道），按下式计算 R、G、B 的最大值和最小值的平均值:

$$L = \frac{MIN(R,G,B) + MAX(R,G,B)}{2}$$

3.2.6.2 高通过滤器 L 84

然后，将高通过滤器 84 应用于亮度信息。由于在中分辨率空间而不是在 CFA 分辨率空间中进行过滤，所以可以按比例放大锐化内核的大小，或适当缩放高通结果。锐化的准确数量取决于 CFA，但是一个 3×3 卷积内核 85 足以生成一个好结果。

如果要增加内核的大小，则表 12 表示应用于 1280 分辨率空间时 CFA 空间中的 3×3 卷积需要的有效缩放比例 86，其中使用绿色信道作为内核缩放的基础。可以从该表中看到，应用于中分辨率空间的 7×7 大小的内核适合所有锐化。

表 12. 卷积过滤器的比例系数

格式	比例	3×3	中分辨率（1280）空间中的内核
标准 30	1.28	3.84	3×3 或 5×5
护照 31	0.64	1.92	无，或 3×3
全景 33	2.56	7.68	7×7
小图 32	0.85	2.55	无，或 3×3

如果在中分辨率图象上应用 3×3 过滤器 85，则根据普通图象缩放操作中使用的比例系数，对结果进行缩放 86。给定表 2 中的数量（特

别是标准打印格式），可以使用3×3过滤器85，然后对结果进行缩放。图27表示生成单个过滤L象素的过程。

使用的实际内核可以为一组标准高通过滤器内核中的任意内核。图50表示实现PCP时使用的令人满意的基本高通过滤器。

3.2.6.3 向RGB添加过滤器L

下一项工作是，向亮度信道添加部分作为结果的高通过滤亮度值。然后将图象转换为RGB（或者，转换为CMY）。然而，可以利用RGB中的相同改变合理逼近亮度改变（只要颜色空间是线性的即可）。因此，通过向R、G、B添加相同比例的高通过滤亮度值，就可以避免颜色转换。可以利用比例系数定义高通过滤图象的准确比例。

如果L是高通过滤亮度象素，k是固定比例系数，则可以按下式

$$\left.\begin{aligned} R' &= R + kL \\ G' &= G + kL \\ B' &= B + kL \end{aligned}\right\} \text{（最大值255）}$$

定义锐化R、G、B的变换：

当然，可以把应用于L的比例系数和高通过滤处理（见3.2.3.6一节）中的比例系数结合起来，作为一个比例系数。

在向RGB象素应用锐化处理后，可以将图象转换为CMY 83，以便打印输出。

3.2.7 转换为CMY 83

理论上，从RGB到CMY的转换很简单：

C = 1 – R

M = 1 – G

Y = 1 – B

上述转换假设CMY空间具有线性响应，然而，对于有颜色的墨，

该假设并不成立，并且对于基于着色的墨，也只有部分成立。特定设备（输入和输出）的不同颜色轮廓大不相同。因此，考虑到准确转换以及未来的传感器、墨和打印机， Printcam 需要更精确的模型。

图28表示需要的变换。选择Lab的原因在于感觉上它是一致的(不同于XYZ)。关于从图象传感器色彩转换为打印机色彩，打印机色彩通常完全包含在传感器色彩内。

通过基于3组3D查找表的三线性转换，可以得到非常好的结果，而不用穷尽上述变换。查找表包含利用RGB作为索引的特定条目的合成变换。需要三个表：一个将RGB映射为C的表90，一个将RGB映射为M的表91，以及一个将RGB映射为Y的表92。对于表中未包含的条目，通过使用三线性内插得到最终结果。图29表示该过程。

三线性内插需要从查找表中读取8个值，然后执行7次线性内插（第一维数需要4次，第二维数需要2次，第三维数需要1次）。可以对中间值使用高精度，尽管输出值只有8位。

需要的查找表的大小取决于变换的线性性。在本申请中每个表的推荐大小是17×17×17 [注1]，每条目8位。一个17×17×17的表为4913字节（小于5KB）。

为了编入每维17个数值的查找表，将8位输入颜色分量看作定点数（4:4）。4比特整数作为索引，4比特分数用于内插。

注1：尽管一个17×17×17的查找表能够提供非常好的结果，但是也可以仅仅使用一个9×9×9的转换表(729字节)。可以通过模拟确定表的大小。本文选择保守但结果明确的5K方法。

3.2.8 向上内插

现在，必须对中分辨率（1280宽度）CMY图象进行向上内插，以得到最终打印分辨率（6400宽度）。两个维数的精确比例都是1:5。

尽管可以对25个值（X和Y维数均为1:5）进行双线性内插，但是不能以连续色调打印合成值。需要对结果进行抖动，并以二值方式打印。假设将连续色调1600dpi结果变为抖动二值点，则从320dpi到1600dpi的双线性内插的准确性并不明显（这正是选择中分辨率的原因）。因此，象素复制生成良好结果。

象素复制仅仅包括取得一个象素，并使用该象素作为更大区域的

值。此时，将一个象素复制为25个象素（一个5×5块）。如果每个象素是连续色调，则结果将是块状的，但是由于已经对象素进行了抖动，所以25个合成二值点不具有连续色调值。图30表示该过程。

3.2.9 半色调68

打印头2只能以二值形式打印圆点。因此，必须从连续色调CMY转换为抖动CMY图象。更确切地说，通过使用一个随机抖动单元，生成一个分散的圆点级的抖动，其中随机抖动单元将连续色调CMY图象转换为抖动二值CMY图象。

比较8比特1600dpi连续色调值与抖动单元93中的当前位置的值。如果80比特连续色调值大于抖动单元值，则生成一个输出比特1。否则，生成一个输出比特0。最终将输出比特发送到打印头，并控制单个喷嘴生成单个C、M、Y圆点。输出比特表示是否加热给定颜色和位置的特定喷嘴。

抖动单元93中的相同位置可用于C、M、Y。因为实际打印头2在同一打印周期中生成不同行的C、M、Y圆点。不同颜色点的交错排列生成抖动单元中的交错排列。

图31表示半色调处理。

抖动单元93的大小取决于输出圆点的分辨率。由于我们生成1600dpi的圆点，所以单元的大小应大于32×32。另外，为了使处理顺序与打印头区段匹配，抖动单元的大小最好均匀分成800（由于打印头的每段有800点）。

50×50的抖动单元足以生成高质量结果，并均匀分成800（16倍）。抖动单元的每个条目为8比特，总共2500字节（约1.5KB）。

3.2.10 为打印机重新格式化69

在发送到打印机之前，需要将所有圆点格式化为发送到打印头的正确顺序。必须以正确顺序，如2.2.1中定义的每次24点的顺序，将这些圆点发送到打印头。

如果能够以正确的打印顺序生成这些圆点（即，向上内插和抖动功能生成正确顺序的数据），则只需收集这些点值（每个值为 1），然后以 24 个值为一组发送。图 32 表示该过程。

通过 Memjet 接口 15，将 24 比特组发送到打印头 2。

4 CPU 内核与存储器

4.1 CPU 内核 10

PCP 3 安装了一个微型控制器 CPU 内核 10，用于同步图象获取和打印图象处理链，以及执行 Printcam 的通用操作系统任务，包括用户接口。可以使用各种 CPU 内核：只要具有足以执行所需计算的处理能力，并且具有能够快速满足消费者期望的控制功能的处理器即可。

由于利用专用硬件进行图象处理，所以 CPU 不需要处理象素。因此，CPU 可以非常简单。然而，CPU 的速度必须能够在打印期间驱动步进电机（步进电机需要 5KHz 处理）。例如，可以使用 1MHz 的飞利浦 8051 微型控制器。

无需保持不同 Printcam 型号之间的指令集的连续性。由不同厂商制造不同的 PCP 芯片设计，而不需要许可或移植 CPU 核心。设备独立性避免芯片厂商封锁，如 Intel 对 PC 市场的封锁。

与 CPU 内核关联的是程序 ROM 13 和小型程序擦除存储器 RAM 14。

CPU 10 通过存储器映象 I/O，与 PCP 3 内的其他部件通信。特定地址范围映射到特定装置，并且在每个范围内，映射到特定装置内的特定寄存器。包括并行和串行接口。

4.2 程序 ROM 13

将小型程序快闪 ROM 13 安装到 PCP 3 内。ROM 的大小取决于选择的 CPU，但是应不大于 16-32KB。

4.3 程序 RAM 14

同样，将一个小型擦除 RAM 区域 14 安装到 PCP 3 内。由于程序代码不需要处理图象，所以不需要一个很大的擦除区域。RAM 的大小取决于选择的 CPU（如，堆栈机制，子程序调用约定，寄存器大小等），但是应不大于 4KB。

4.4 CPU 存储器译码器 16

CPU 存储器译码器 16 是一个满足 CPU 数据存取的简单译码器。该译码器将数据地址翻译成内部低速总线上的内部 PCP 寄存器存取，因此，支持 PCP 寄存器的存储器映象 I/O。

5 通信接口

5.1 USB 串行端口接口 17

该接口是一个标准 USB 串行端口，连接到内部芯片低速总线 18。由 CPU 10 控制 USB 串行端口。串行端口向/从 Printcam 传送图象，并且在外部控制下，以 DPOF（数码打印指令格式）打印传送的相片。

5.2 QA 芯片串行接口 19

该接口是两个标准低速串行端口，连接到内部芯片低速总线 18。使用两个端口之间的 CPU 仲裁协议认证打印滚筒[1，2]，并执行以下功能:

- 获取墨特性
- 获得推荐的墨滴量
- 记录打印的相纸数量，并且当相纸不足以打印请求的打印格式时，请求新的打印滚筒。

具有两个端口的原因在于，连接到使用独立线路的照相机上的 QA 芯片 4 和打印滚筒的 QA 芯片 5 以认证芯片[2]的方式实现两个 QA 芯片。如果只使用一条线路，则克隆打印滚筒厂商能够盗用认证机制[1]。

5.2.1 打印滚筒的 QA 芯片 5

每个打印滚筒消耗品包括其特有的 QA 芯片 5。QA 芯片包含维持最佳打印质量所需要的信息，并且使用认证机制[2]实现该芯片。按以下方式分配 256 位数据：

表 13. 打印滚筒的 256 位（16

M[n]	访问	描述
0	RO[a]	基本头，标记等（16 位）
1	RO	序号（16 位）
2	RO	批号（16 位）
3	DO[b]	相纸余量，单位 mm（16 位）
4	RO	青色墨属性（32 位）
5	RO	
6	RO	品红色墨属性（32 位）
7	RO	
8	RO	品红色墨属性（32 位）
9	RO	
10-12	RO	供未来扩展=0（48 位）
13-15	RO	随机位，各芯片不同（48 位）

a.只读

b.只能减少

在每次打印前，由 CPU 检测相纸余量，以确保足够当前指定的打印格式使用。在打印开始后，由 CPU 减少打印滚筒的 QA 芯片中的相纸余量。

5.3 并行接口 6

并行接口 6 将 PCP 3 连接到各静电信号。通过低速总线，CPU 能够控制作为存储器映象 I/O 的各连接（有关存储器映象 I/O 的细节，

请参见 4.4 一节）。

表 14.并行接口的连接

连接	方向	引线
相纸传送步进电机	输出	4
切纸电机	输出	1
聚焦电机	输出	1
封盖线圈	输出	1
闪光触发器	输出	1
状态 LCD 区段驱动	输出	7
状态 LCD 公共驱动	输出	4
相纸牵引传感器	输入	1
按钮	输入	4
	总计	24

5.4 JTAG 接口 7

为了进行测试，PCP 3 包括一个标准 JTAG（联合测试行动组）接口 7。由于芯片的复杂性，需要各种测试技术，包括 BIST（内置的自检测）和功能块隔离。芯片区域中 10%的总开销用于全部芯片测试电路。

6 图象 RAM 11

图象 RAM 11 用于存储获取的图象 42。图象 RAM 是多级快闪存储器（2 为/单元），以便在关闭电源后继续保存图象。

平面化的线性 RGB 图象需要的总存储量是 1,500,000 字节（约为 1.5MB），其排列为：

- R: 750×500=375,000 字节
- G: 750×500=375,000 字节
- B: 750×1000=750,000 字节

由图象获取装置写入图象，由图象直方图单元 8 和打印生成单元 99 读取该图象。CPU 10 不对图象存储器进行直接随机存取。它通过图象访问装置访问图象象素。

7 图象获取单元 12

图象获取装置包含 3.1 所述的图象获取链需要的所有功能性。图象获取装置通过图象传感器接口 98 接受象素数据，通过查找表 96 线性化 RGB 数据，最后，以平面格式将线性化的 RGB 图象写到 RAM 中。图 33 表示该过程。

7.1 图象传感器接口 98

图象传感器接口（ISI）98 是一个状态机，为了读取图象，状态机向 CMOS 图象传感器发送控制信息，包括帧同步脉冲和象素时钟脉冲。大部分 ISI 可能是来自图象传感器厂商的源单元。由图象获取装置的状态机本身控制 ISI。

7.1.1 图象传感器格式

尽管可以使用各种图象传感器格式，但是我们只考虑拜尔彩色滤镜阵列（CFA）。拜尔 CFA 具有许多本文定义的属性。

假设已经充分过滤了 CMOS 传感器（通过取象透镜）获取的图象，以消除锯齿现象。传感器本身的长宽比为 3:2，具有 1500×1000 分辨率的采样。最可能的象素结构为拜尔彩色滤镜阵列（CFA），其中图 15 表示在 2G 镶嵌面中排列的各 2×2 象素块。

R、G、B（分别对应于红、绿、蓝）的各连续色调采样是 10 比特。请注意，镶嵌面的每个象素仅包含有关 R、G、B 之一的信息。必须在打印输出前估计丢失的彩色信息。

认为 CFA 执行部分固定模式噪声（PFN）抑制。需要附加的 FPN 抑制。

7.2 查找表 96

查找表 96 为 ROM，后者将传感器的 RGB 映射为线性 RGB。它匹配 3.1.2 中描述的线性化 RGB 过程 40。同样，ROM 为 3KB（3×1024×8 — 比特）。10 位地址来自 ISI，而 2 位 TableSelect 是由图象获取装置的状态机 97 生成的。

7.3 状态机 97

图象获取装置的状态机 97 生成图象传感器接口 1 的控制信号，并且生成用于线性化 RGB 40 和用于平面化图象数据 41 的地址。

发送到 ISI 98 的控制信号通知 ISI 开始获取象素，停止获取象素等。

发送到查找表 96 的 2 位地址匹配正从 ISI 读取的当前行。对于偶数行（0，2，4 等），2 位地址是红、绿、红、绿等。对于奇数行（1，3，5 等），2 位地址是绿、蓝、绿、蓝。不管照相机的方向如何，以上总是正确的。

发送到图象 RAM 的 21 位地址是该图象的写地址。三个寄存器保存红、绿、蓝平面的当前地址。将地址增量作为象素写入各平面。

7.3.1 寄存器

图象获取装置包含许多寄存器：

表 15. 图象获取装置中的寄存器

名称	位	描述
MaxPixels	12	每行的象素数
MaxRows	12	图象中象素的行数
CurrentPixel	12	当前提取的象素
CurrentRow	12	当前处理的行
NextR	21	存储下一个红色象素的图象 RAM 中的地址。在图

		象获取前，设置为红色平面的开始地址。在图象获取后，该寄存器将指向红色平面后的字节。
NextG	21	存储下一个绿色象素的图象 RAM 中的地址。在图象获取前，设置为绿色平面的开始地址。在图象获取后，该寄存器将指向绿色平面后的字节。
NextB	21	存储下一个蓝色象素的图象 RAM 中的地址。在图象获取前，设置为蓝色平面的开始地址。在图象获取后，该寄存器将指向蓝色平面后的字节。
EvenEven	2	偶数行/偶数象素使用的地址
EvenOdd	2	偶数行/奇数象素使用的地址
OddEven	2	奇数行/偶数象素使用的地址
OddOdd	2	奇数行/奇数象素使用的地址
Go	1	写入 1 开始获取。写入 0 停止图象获取。在获取到 MaxPixels 的 MaxRows 后，由状态机自动写入 0。

另外，图象传感器接口 98 包含许多寄存器。确切寄存器取决于选择的图象传感器 1。

8 图象访问单元 9

图象访问单元 9 生成供 CPU 10 访问 ImageRAM 11 中的图象的方法。CPU 10 能够读取 ImageRAM 11 中的图象的象素，并将象素写入 ImageRAM 11。

为了图象存储（如，通过 USB）17 或简单图象处理而读取象素。在图象处理后，将象素写入 ImageRAM 11，作为先前保存的图象（通过 USB 加载），或者作为测试图形的图象。测试图形可以是合成图象，特定测试图象（通过 USB 加载），或者是通过打印生成单元 99 直接加载到打印头的 24 比特喷嘴加热值。

图象访问单元 9 是 ImageRAM 11 的直接访问机构，并且根据图 16 所示的 3 个寄存器，其操作相当简单。

表 16. IAU 寄存器

名称	位	描述
ImageAddress	21	在 ImageRAM 中进行读写的地址
Mode	3	0=从 ImageAddress 读入到 Value 中。 1=将 Value 写入 ImageAddress。
Value	8	在 ImageAddress 存储的值（如果 Mode=Read） 要在 ImageAddress 存储的值（如果 Mode=Write）

正如图 35 所示，图象访问装置的结构非常简单。

每当 CPU 10 写入 Mode 寄存器时，状态机 101 就读取/写入 ImageRAM 11。

9 图象直方图单元 8

图象直方图单元（IHU）8 的目的是生成 3.2.2 描述的打印图象处理链需要的图象的直方图。IHU 只生成 8 比特采样的平面格式图象的直方图。

每个打印通常使用图象直方图单元 8 三次。生成三个不同的直方图，每个颜色平面一个。每次收集直方图时，都要分析结果，以便确定高低阈值，比例系数等，供其他打印处理使用。有关使用直方图的详细信息，请参见 3.2.2.2 和 3.2.4。

9.1 直方图 RAM 102

在一个有 256 项的 RAM 102 中存储直方图本身，每一项为 20 位。只从 IHU 内访问直方图 RAM。作为 20 位数值读写每一项。

9.2 状态机和寄存器 103

状态机 103 遵循 3.2.2.1 中描述的伪码。由表 17 所示的寄存器控制状态机。

表 17. 图象直方图单元中的寄存器

名称	位	描述
TotalPixels	20	要计数的象素数（递减到 0）
StartAddress	21	开始计数的地址
PixelRemaining	20	尚须计数的象素数
PixelValue	8	写入此寄存器将向 PixelCount 加载直方图的 PixelValue 项。
PixelCount	20	在当前直方图中计数的 PixelValue 象素数。在写入 PixelValue 后才有效。
ClearCount	1	确定是否在开始直方图处理时清除直方图计数。1 表示清除计数，0 表示保留计数不变（即，下一个直方图添加到现有计数）。
Go	1	写入 1 开始直方图处理。写入 0 停止直方图处理。在 TotalPixels 递减到 0 之后，由状态机自动写入 0。

寄存器的典型用途是，向 TotalPixels 提供计数中包括的总象素数（如，红色 375,000），向 StartAddress 提供红色平面的地址，向 ClearCount 提供 1，并将 1 写入 Go 寄存器。在完成计数后，通过将 0-255 写入 PixelValue 然后读取相应的 PixelCount，确定直方图中的每个值。

10 打印头接口 105

PCP 3 利用打印头接口（PHI）105 向 Memjet 打印头 2 加载需要打印的点，并控制实际的点打印过程。PHI 是许多装置的逻辑封装:

- Memjet 接口（MJI）15，该装置向 Memjet 打印头传送数据，并且在打印期间控制喷嘴加热顺序。
- 打印生成单元（PGU）99，该单元实现第 24 页上 3.2 中描述的大部分打印链，并提供生成测试图形的装置。PGU 从

ImageRAM 11 取出平面化的线性 RGB，并且实时生成 Memjet 接口 15 需要的 1600 dpi 的抖动 CMY 图象，其中平面化的线性 RGB 是从 CFA 格式获取的图象得到的。另外，PGU 具有测试图形模式，该模式允许 CPU 10 精确确定需要在打印期间加热的喷嘴。

由 CPU 编程的许多寄存器控制 PHI 内的单元。

图 37 表示打印头接口的内部结构。

10.1 Memjet 接口 15

Memjet 接口（MJI）15 将 PCP 连接到外部 Memjet 打印头，用于提供数据和适当信号，以便控制打印期间喷嘴的加载和加热顺序。

Memjet 接口 15 只是一个状态机 106（见图 38），该状态机遵循 2.2 中描述的打印头加载和加热次序，并且包括 2.4.1 和 2.4.2 中描述的预热周期和清洁周期的功能性。

MJI 15 向从以下两个数据源中选择的打印头加载数据：

- 全 1。意味着在随后的打印周期中所有喷嘴将加热，这是为预热或清洁周期加载打印头的标准机制。
- PGU 99 的 Transfer（传送）寄存器中保存的 24 位输入。这是打印图象的标准方法，无论是拍摄的相片还是测试图形。将 PGU 中的 24 位值直接发送到打印头，然后向 PGU 发送 1 位“Advance（前进）”控制脉冲。在每行的结尾，向 PGU 发送 1 位“AdvanceLine（前进一行）”脉冲。

必须在 PGU 99 准备好第一个 24 位传送值后，启动 MJI 15。只有这样，第一次传送到打印头的 24 位数据输入才有效。

因此，直接将 MJI 15 连接到打印生成单元 99 和外部打印头 2。图 38 表示其基本结构。

101.1 打印头的连接

MJI 15到打印头2的连接如下表所示，连接用于读取有关MJI 15的输入和输出。名称匹配打印头上的引线连接（见第2节）。

图18. 打印头连接

名称	引线数	I/O	描述
ChromapodSelect	4	O	选择要加热的色品容器（0-9）
NozzleSelect	4	O	从容器中选择要加热的喷嘴（0-9）
AEnable	1	O	相群A的加热脉冲
BEnable	1	O	相群B的加热脉冲
CDataIn[0-7]	8	O	段0-7的青色移位寄存器的青色输入
MDataIn[0-7]	8	O	段0-7的品红色移位寄存器的品红色输入
YDataIn[0-7]	8	O	段0-7的黄色移位寄存器的黄色输入
SRClock	1	O	SRClock上的一个脉冲（ShiftRegisterClock）将CDataIn[0-7]、MDataIn[0-7]和YDataIn[0-7]的当前值加载到24个移位寄存器中。
PTransfer	1	O	将移位寄存器中的数据并行传送到打印头的内部NozzleEnable位（每个喷嘴一位）。
SenseSegEnable	1	O	SenseSegEnablet上的一个脉冲与CDataIn[n]上的数据的逻辑与选择段n的传感线路。
Tsense	1	I	温度检测
Vsense	1	I	电压检测
Rsense	1	I	电阻率检测
Wsense	1	I	宽度检测
总计	41		

10.1.2 加热脉冲持续时间

AEnable和BEnable上的加热脉冲的持续时间取决于墨的粘性(取决于温度和墨特性)以及打印头使用的功率。典型脉冲持续时间为1.3到1.8μs。因此，MJI包含一个可编程脉冲持续时间表，利用打印头的反馈作为索引。脉冲持续时间表允许使用低成本电源，并且有助于保持更精确的墨滴喷射。

脉冲持续时间表有256项，利用当前的Vsense和Tsense设置作为索引。地址的前4位来自Vsense，地址的低4位来自Tsense。每一项为8位，并且表示0-4μs范围内的定点值。图39表示生成AEnable和BEnable线路的过程。

在打印相片前，由CPU 10写入256字节的脉冲持续时间表。表中的每个8位脉冲持续时间项组合:

- 亮度设置
- 墨的粘性曲线(来自QA芯片)5
- Rsense
- Wsense
- Tsense
- Vsense

10.1.3 点数

MJI 15保持从打印头2喷射的每种颜色的点数。每种颜色的点数是一个32位值，在处理器的控制下单独清除。每个点数可以保持69个6英寸打印的最大平均点数，尽管在典型使用中，在每次打印后读取并清除点数。

在最初的Printcam产品中，消耗品包括相纸和墨，可以想象，不同Printcam型号只有可更换墨的消耗品。最初的Printcam产品递减以毫米为单位的相纸的余量(在QA芯片5中存储，见5.2)，以确定相纸是否足够打印所需格式。墨足以覆盖提供的所有相纸。在其他

Printcam 产品中，CPU 10 使用点数更新 QA 芯片 5，以便预测墨盒何时用完墨。处理器从 QA 芯片 5 中了解墨盒中 C、M、Y 的墨量。通过计算墨滴数，无需使用墨传感器，并且防止墨通道变干。在每次打印后，将更新的墨滴数写入 QA 芯片 5。如果墨不足，则不打印新相片，并允许用户更换墨，从而用户不会得到一张需要重新打印的无用相片。

图 40 表示青色点数计数器的布局。其他两个点数计数器（用于品红的 MdotCount 和用于黄色的 YDotCount）具有相同结构。

10.1.4 寄存器

CPU 10 通过寄存器组与 MJI 15 通信。寄存器允许 CPU 用参数表示打印机，并接收有关打印进展的反馈。

MJI 包含以下寄存器:

表 19. Memjet 接口寄存器

寄存器名称	描述
打印参数	
NumTransfers	加载打印头需要的传送数（通常为 800）。该数字为 SRClock 上的脉冲数以及为给定行传送的 24 为数据值的数目。
PulseDuration	用于确定 ColorEnable 线路上单一脉冲的持续时间的定点数。持续时间范围=0-6μs。
NumLines	执行的加载/打印周期数。
监视打印机	
Status	Memjet 接口的状态寄存器。
LinesRemaining	未打印的行数。只有 Go=1 时才有效。初值为 NumLines。
TransferRemaining	在考虑为当前行加载打印头前，剩余的传送数。只有 Go=1 时才有效。
SenseSegment	在随后的反馈 SenseSegSelect 脉冲期间，在青色数据线

	路上放置的 8 位值。仅仅设置 8 位中的 1 位，该位与 8 段中某一段对应。
SetAllNozzles	如果非零，则在 LoadDots 处理期间写入打印头的 24 位值全为 1，从而在随后的 PrintDots 处理期间加热所有喷嘴。在预热和清洁周期中使用。如果为零，则写入打印头的 24 位值来自打印生成装置。这正是相片和测试图象的实际打印情况。
行动	
Reset	此寄存器的写入操作将复位 MJI，停止所有加载和打印处理，并向所有寄存器加载 0。
SenseSegSelect	向此寄存器写入任何值将清除 Status 寄存器的反馈位，并且在 SenseSegSelect 线路上发送一个脉冲，如果 LoadingDots 和 PrintingDots 状态位全为 0 的话。如果设置了任意状态位，则清除反馈位，并且不再进行任何处理。 在测试各种传感线路后，将这些值置于 Tsense、Vsense、Rsense 和 Wsense 寄存器中，然后设置 Status 寄存器的反馈位。在随后的打印操作期间，反馈继续。
Go	在此位写入 1 启动 LoadDots/PrintDots 循环。总共打印 NumLines 行，每行包含 NumTransfers 的 24 位传送。在打印各行时，LinesRemaining 减 1，并且再次向 TransferRemaining 加载 NumTransfers。Status 寄存器包含打印状态信息。当完成 NumLines 时，加载/打印处理停止，并清除 Go 位。在最后的打印周期中，不向打印头加载数据。 在此位写入 0 停止打印处理，但并不清除其他寄存器。
ClearCounts	此寄存器的写入操作清除 CdotCount、MdotCount、YdotCount 寄存器，如果分别设置了位 0、1、2 的话。写入 0 无效。

反馈	
Tsense	发送到段 SenseSegment 的上一个 SenseSegSelect 脉冲的 Tsense 的只读反馈。只有设置了 Status 寄存器的 FeedbackValid 位才有效。
Vsense	发送到段 SenseSegment 的上一个 SenseSegSelect 脉冲的 Vsense 的只读反馈。只有设置了 Status 寄存器的 FeedbackValid 位才有效。
Rsense	发送到段 SenseSegment 的上一个 SenseSegSelect 脉冲的 Rsense 的只读反馈。只有设置了 Status 寄存器的 FeedbackValid 位才有效。
Wsense	发送到段 SenseSegment 的上一个 SenseSegSelect 脉冲的 Wsense 的只读反馈。只有设置了 Status 寄存器的 FeedbackValid 位才有效。
CDotCount	发送到打印头的青色点的 32 位计数，只读。
MdotCount	发送到打印头的品红色点的 32 位计数，只读。
YDotCount	发送到打印头的黄色点的 32 位计数，只读。

MJI 的 Status 寄存器是一个带位翻译的 16 位寄存器。

表 20. MJI 状态寄存器

名称	位	描述
LoadingDots	1	如果设置，则 MJI 正在加载点，数目为 TransferRemaining 中需要传送的剩余点数。 如果清除，则 MJI 当前未加载点。
PrintingDots	1	如果设置，则 MJI 正在打印点。 如果清楚，则 MJI 当前未打印点。
PrintingA	1	如果 AEnable 线路上有脉冲，则设置此位。
PrintingB	1	如果 BEnable 线路上有脉冲，则设置此位。

FeedbackValid	1	如果反馈值 Tsense、Vsense、Rsense 和 Wsense 有效，则设置此位。
Reserved	3	-
PrintingChromaphod	4	当设置 PrintingDots 状态位时，保持正在加热的色品容器。
PrintingNozzles	4	当设置 PrintingDots 状态位时，保持正在加热的喷嘴。

10.1.5 预热和清洁周期

通过设置适当的寄存器，完成清洁和预热周期。

- SetAllNozzles=1
- 将 PulseDuration 寄存器设置为低持续时间（在预热模式中），或设置为合适的墨滴喷射持续时间（对于清洁模式）。
- 将 NumberLines 设置为喷嘴的加热次数。
- 设置 Go 位，然后等待打印周期结束时清除 Go 位。

10.2 打印生成单元 99

打印生成单元（PGU）99 实现 3.2 中描述的打印链的大部分功能，并提供生成测试图形的装置。

从最简单的观点看，PGU 提供图象 RAM 11 和 Memjet 接口 15 之间的接口，如图 41 所示。PGU 从 ImageRAM 中取出平面化的线性 RGB，并且实时生成 Memjet 接口需要的 1600dpi 的抖动 CMY 图象，其中平面化的线性 RGB 是从 CFA 格式获取的图象得到的。另外，PGU 99 具有测试图形模式，该模式允许 CPU 10 精确确定需要在打印期间加热的喷嘴。在使用 24 位值后，MJI 15 向 PGU 99 提供一个 Advance 脉冲，并且在行结束时提供一个 AdvanceLine 脉冲。

PGU 99 具有两个图象处理链。第一个处理链是测试图形模式，直接读取图象 RAM 11 中的数据，然后在缓冲器中格式化该数据以输出到 MJI。第二个处理链包括大部分打印链功能（见 3.2）。图 18 所示

的打印链包括以下功能:

- 收集统计数字 60
- 旋转图象 61
- 白平衡 62
- 范围扩展 63
- 重新采样 64
- 锐化 65
- 转换为 CMY 66
- 向上内插 67
- 半色调 68
- 为打印机重新格式化 69

PGU 99 包含除收集统计数字 60 之外的所有功能。为了执行收集统计数字步骤，CPU 10 调用图象直方图单元 8 三次（每个颜色信道一次），并应用某些简单算法。其余功能属于 PGU 99，原因在于精度和速度：关于精度，以高精度保存整幅图象需要太多存储器，关于速度，一个简单 CPU 不能满足 Memjet 打印头 2 的实时高速要求。

PGU 99 将许多参数作为输入，参数包括 RGB 到 CMY 的转换表，执行白平衡和范围扩展的常数，重新采样的比例系数，以及用于旋转的图象访问参数。

图 20 表示两个处理链。最直接的处理链通过测试图形访问处理 110 从图象 RAM 11 到缓冲器 5。另一个处理链包括 5 个并行运行处理。第一处理 111 执行图象旋转、白平衡和范围扩展。第二处理 112 执行重新采样。第三处理 65 执行锐化，第四处理 66 执行颜色转换。最后一个处理 113 执行向上内插、半色调处理，并且为打印机重新格式化。通过缓冲器连接上述处理，某些处理之间只需几个字节，而其他处理需要几 K 字节。

由于打印头的定时驱动所有处理，所以主要以逆序考虑上述处理。特定处理的定时和缓冲器大小需求比较明显。总之，表 21 表示缓冲器

大小。

表 21. 打印生成单元的缓冲器大小

缓冲器	大小（字节）	缓冲器的组成
缓冲器 1	188	红色缓冲器=6 行×6 项/行×10 比特/项=45 字节 蓝色缓冲器=6 行×6 项/行×10 比特/项=45 字节 绿色缓冲器=13 行×6 项/行×10 比特/项=97.5 字节
缓冲器 2	24	6×4 RAM 3 行×4 项/行×8 比特/项=12 字节 3 色×4 项/色×8 比特/项=12 字节
缓冲器 3	3	3 色（RGB）×8 比特/色
缓冲器 4	23,040	3 色（CMY）×6 行/色×1280 连续色调象素/行×8 比特/象素
缓冲器 5	9	3×24 比特
总计	23,264	

除许多寄存器之外，某些处理还需要大量查找表或存储元件。表 22 概括说明这些需求。

表 22. PGU 处理内的存储器需求

装置	大小(字节)	需求的组成
旋转/白平衡/范围扩展	0	
重新采样/转换为 L	1,152	3 个内核，每个内核 64×4×12 比特
锐化	0	
转换为 CMY	14,739	3 个转换表，每个 17×17×17×8 比特
向上内插/半色调/重新格式化	2,500	抖动单元，50×50×8 比特

测试图形访问	0	
总计	18,391	

10.2.1 测试图形访问

测试图形访问处理 110 是生成测试图形的装置。在正常用户环境中，不使用该处理。该处理主要用于诊断。

测试图形访问 110 读取图象 RAM 11，然后将 8 位值直接传送到缓冲器 5 118，以输出到 Memjet 接口。它不对 8 位值做任何修改。CPU 10 通过使用图象访问单元 9 生成图象 RAM 11 中的数据。

以环绕方式读取图象 RAM 11 中的数据。使用两个寄存器来描述测试数据：第一字节的起始地址和字节数。当达到数据结束时，再次从头开始读取数据。

图 43 表示测试图形访问装置 110 的结构。

正如从图 43 中看到的那样，测试图形访问装置 110 与地址生成器 119 没有区别。启动时，对于每个 AdvanceLine 信号，生成器读取 3 个字节，生成一个 TransferWriteEnable 脉冲，读取下 3 个字节，然后等待 Advance 脉冲。在 Advance 脉冲中，提供 TransferWriteEnable 脉冲，读取下 3 个字节，然后再次等待。继续上述处理直至 AdvanceLine 脉冲，其中在该脉冲上，处理再次从当前地址开始。

关于读取 3 个字节，地址生成器 119 只需从图象 RAM 11 中读取三个 8 位值，然后将其写入缓冲器 5 118 中。将第一个 8 位值写入缓冲器 5 的 8 位地址 0，将第二个写入缓冲器 5 的 8 位地址 1，将第三个写入缓冲器 5 的 8 位地址 2。然后，地址生成器 119 等待 Advance 脉冲，以便再次进行相同处理。

为图象 RAM 11 生成的地址基于表 23 所示的起始地址和字节数。

表 23. 测试图形访问寄存器

寄存器名称	描述

TestModeEnabled	如果为 1，则启用 TestMode。 如果为 0，则禁用 TestMode。
DataStart	图象 RAM 中测试数据的起始地址
DataLength	测试数据中的 3 字节数

以下伪码表示地址生成。未示出 AdvanceLine 和 Advance 脉冲。

```
Do Forever
    Adr = DataStart
    Remaining = DataLength
    Read Adr into Buffer 5 (0), Adr = Adr + 1
    Read Adr into Buffer 5 (1), Adr = Adr + 1
    Read Adr into Buffer 5 (2), Adr = Adr + 1
    Remaining = Remaining - 1
    if (Remaining = 0)
        Remaining = DataLenght
EndDo
```

由 CPU 10 确保该数据对打印头 2 有意义。字节 0 是 8 个青色段的喷嘴加热数据（位 0=段 0 等），字节 1 用于品红，字节 2 用于黄色。另一组 24 位数据用于用水平线分割的奇数/偶数象素。

10.2.2 缓冲器 5 118

缓冲器 5 118 保持整个打印生成处理生成的圆点。缓冲器 5 包括：一个 24 位移位寄存器，用于保持每次从 UHRU 113（向上内插、半色调和重新格式化装置）生成的圆点；3 个 8 位寄存器，用于保持从 TPAU（测试图形访问装置）生成的数据；和一个 24 位寄存器，作为传送到 MJI（Memjet 接口）的数据的缓冲器。来自 MJI 的 Advance 脉冲向 24 位 Transfer（传送）寄存器加载全部 24 位数据，或者从 3 个 8 位

寄存器或从一个 24 位移位寄存器。

因此，缓冲器 5 作为所生成的圖点的双重缓冲机制，并且具有图 44 所示的结构。

10.2.3 缓冲器 4 117

缓冲器 4 117 保持计算的 CMY 中分辨率（1280 分辨率）连续色调图象。缓冲器 4 是由颜色转换处理 66 生成的，由向上内插、半色调和重新格式化处理 113 访问，以生成打印机的输出点。

连续色调缓冲器的大小取决于打印头上的喷嘴之间的物理距离。在生成某一物理行的一种颜色的圖点时，生成不同行上不同颜色的圖点。实际结果是打印机一次打印 6 行不同的物理行 — 不同输出行的奇偶点，以及每种颜色的不同行。2.1.1 解释此概念并定义以上距离。

实际结果是，在高分辨率圖点中，从偶数青色圖点通过品红色圖点到黄色圖点是已知距离。为了最小化 RGB 的生成，从而最小化 CMY 的生成，在缓冲器 4 中缓冲生成高分辨率圖点的中分辨率连续色调象素。

由于中分辨率行与高分辨率行的比例是 1:5，所以在每个维数上对各中分辨率采样 5 次。对缓冲器行而言，我们仅关心 1 个维数，因此，仅考虑来自一个象素行的 5 行圖点。不同颜色的喷嘴之间的距离是 4-8 点（取决于 Memjet 参数）。因此，假设为 8 点，则内含距离为 16 点或 17 点间距。最坏情况是，17 行圖点包括给定象素行的上一行圖点。这意味着 5 行象素，生成的圖点行为 1，5，5，5，1，并且允许将喷嘴间隔增加到 10。

为了保证缓冲器的连续色调生成处理写入过程不干扰缓冲器的点生成处理读取过程，额外添加每种颜色的中分辨率行，每种颜色共计 6 行。

因此，连续色调缓冲器为 6 行 3 色，每行包含 1280 个 8 比特连续色调值。需要的总存储量为 3×6×1280=23040 字节（22.5KB）。存储器只需要每周期读出一个 8 比特值，每 25 个周期写入一个 8 比特值（每

个连续色调象素读25次）。图45表示缓冲器4的结构。

可以将缓冲器4实现为以打印头点生成处理的额定速度运行的单周期双存取（读和写）RAM，或者实现为比每周期一次读/写快4%的RAM。

在开始打印处理前，将缓冲器4设置为空白（全0）。

10.2.4 向上内插、半色调和为打印机重新格式化

尽管3.2.8、3.2.9和3.2.10分别将向上内插、半色调和为打印任务重新格式化任务113定义为独立任务，但是在PCP 3的硬件实现中，作为单一处理实现。

向上内插、半色调和重新格式化装置（UHRU）113的输入为连续色调缓冲器（缓冲器4）117，包括预先计算的CMY 1280分辨率（中分辨率）图象。输出是一组需要发送到Memjet接口15的具有正确顺序的24位值，以便通过缓冲器5 118输出到打印头。以每次一位的方式生成24位输出，并发送到缓冲器5 118中的24位移位寄存器。

利用来自MJI 15的Advance和AdvanceLine信号控制以上处理。当UHRU 113启动时，并且在各AdvanceLine脉冲后，生成24位值，然后利用ShiftWriteEnable信号按照时钟信号写入缓冲器5的24位移位寄存器。在写入第24位后，提供TransferWriteEnable脉冲，然后生成下24位。此后，UHRU 113等待来自MJI的Advance脉冲。当Advance脉冲到达时，向缓冲器5 118提供TransferWriteEnable脉冲，并且在再次等待前计算下24位。实际上，在提供第一个Advance脉冲后，开始同步，此后每24个周期出现一个Advance脉冲。

图46表示向上内插、半色调和重新格式化处理。

利用简单的8位无符号比较器120执行半色调任务。比较器的两个输入来自交错抖动单元121和缓冲器4 117。由地址生成状态机122确定向无符号比较器120提供上述值的顺序，状态机122确保进入1280分辨率图象的地址与打印头需要的面向段的顺序匹配。因此，地址生成状态机122执行向上内插和为打印机重新格式化任务。通过正确寻

址连续色调缓冲器（缓冲器4）117，并确保比较器120使用来自抖动单元121的正确查找表匹配交错地址，实现重新格式化，而不仅仅是每次以高分辨率访问一整行，然后根据打印机查找需求重新格式化该行（见3.2.10中的说明）。

半色调任务与3.2.9说明的处理相同。然而，由于点输出是按照打印头的正确顺序生成的，所以选择抖动单元121的大小，以便平均分成800。因此，某一段的抖动单元中的给定位置与剩余7段（的抖动单元中的给定位置）相同。一个50×50的抖动单元能够提供令人满意的结果。正如3.2.9中所述，由于同时生成各种颜色的不同行，所以不同颜色可以使用抖动单元中的相同位置。因此，抖动单元的寻址很简单。我们从交错排列的抖动单元中的特定行（如，0行）开始。使用的第一个抖动单元条目是条目0。使用该条目24次（24个周期）以生成全部8段的3种颜色，然后前进到0行的条目1。在条目49后，返回到条目0。对全部19,200个周期继续上述处理，以生成19,200个点。然后，半色调装置停止并等待AdvanceLine脉冲，该脉冲使地址生成器前进到抖动单元中的下一行。

称为交错抖动单元121的原因在于，它与正常抖动单元的区别在于交错奇数行和偶数行。这是因为我们生成不同行上的奇数和偶数象素（从象素0开始），并且使地址生成器122免于在24个象素的交替集合上前进到下一行，然后再返回。图25表示一个简单抖动单元93，以及将其映射到相同大小的交错抖动单元121的方法。请注意，为了确定给定位置的“奇数性”，我们给指定行0、1、2等中的象素编号。

比较（无符号）来自缓冲器4 117的8位值与来自交错抖动单元121的8位值。如果缓冲器4的象素值大于等于抖动单元值，则将“1”比特输出到缓冲器5 118的移位寄存器。否则，将“0”比特输出到缓冲器5的移位寄存器。

为了对19,200个连续色调象素进行半色调处理，必须读入19,200个连续色调象素。地址生成单元122执行此项任务，生成进入缓冲器

4 117 的地址，有效实现向上内插任务。为读取缓冲器 4 而生成地址的过程比生成抖动单元的地址的过程稍微复杂一些，但并不很复杂。

在写入缓冲器的第一行后，为读取缓冲器 4 的地址生成才开始。缓冲器 4 的剩余行为 0，因此，实际上是白色（没有打印点）。

6 行输出的每一行具有一个存储整数和分数部分的寄存器。使用寄存器的整数部分选择要读取哪个寄存器，以便有效地向上内插特定颜色的奇数和偶数象素的颜色。使用 3 个象素计数器保存段 0 内的当前位置，使用一个临时计数器 P_ADR（象素地址）来偏移剩余 7 段。

总而言之，读取缓冲器 4 的地址生成需要表 24 所示的寄存器。

表 24. 读取缓冲器 4 需要的寄存器

寄存器名称	大小
CyanEven	6 比特（3:3）
CyanOdd	6 比特（3:3）
MagentaEven	6 比特（3:3）
MagentaOdd	6 比特（3:3）
YellowEven	6 比特（3:3）
YellowOdd	6 比特（3:3）
Cyan_P_ADR	14 比特（11:3）
Magenta_P_ADR	14 比特（11:3）
Yellow_P_ADR	14 比特（11:3）
P_ADR	11 比特（只保持 X_P_ADR 的整数部分）

6 个缓冲器行寄存器的初值是喷嘴之间的物理点距离（切记分数部分除以 5 的余数）。例如，如果利用 1 点隔离某种颜色的奇数和偶数输出点，并且利用 8 点隔离一种颜色的喷嘴与下一种颜色的喷嘴，则初值如表 25 中 “第一行”一栏所示。在生成每组 19,200 个圆点后，

这些计数器必须增加 1 分数部分，表示我们沿垂直维数对各象素采样 5 次。表 25 的 “第二行”一栏表示结果值。请注意，5:4+1=0:0，因为只有 6 个缓冲器行数。

表 25. 6 个缓冲器行寄存器的初始设置和第二行的值

		第一行		第二行	
名称	计算	值	缓冲器	值	缓冲器
CyanEven	初始位值	0:0	0	0:1	0
CyanOdd	CyanEven+0:1	0:1	0	0:2	0
MagentaEven	CyanOdd+1:3(8)	1:4	1	2:0	2
MagentaOdd	MagentaEven+0:1	2:0	2	2:1	2
YellowEven	MagentaOdd+1:3(8)	3:3	3	3:4	3
YellowOdd	YellowEven+0:1	3:4	3	4:0	4

接着，6 个缓冲器行寄存器确定读取哪个缓冲器行用于指定颜色的奇数和偶数象素。为了确定从缓冲器 4 的特定行中读取哪些 1280 中分辨率的象素，我们使用 3 个象素地址计数器（每种颜色一个），和一个临时计数器（P_ADR），后者作为各段的索引。每一段与下一段的间隔是 800 点。在中分辨率象素中，该距离为 160。由于 800 能够被 5 整除，所以只需使用 3 个象素地址计数器的整数部分。我们生成偶数青色象素的 8 个地址，然后生成偶数品红色象素的 8 个地址，最后生成黄色偶数象素的 8 个地址。接着，对奇数青、品红和黄色象素进行相同处理。将两组 24 位值（24 个偶数和 24 个奇数）的处理执行 400 次。然后，将象素地址计数器（X_P_ADR）复位到 0，并使 6 个缓冲器行寄存器前进。每前进 5 行，下一个缓冲器行立即变为空闲，并且可以更新（通过转换为 CMY）。表示 26 以简单形式列出上述步骤。

表 26. 读取缓冲器 4 的地址生成处理

#	地址	计算	注释

-		P_ADR=Cyan_P_ADR Cyan_P_ADR+=1(mod5)	生成青色段 0 中偶数象素的地址并前进到青色的下一象素
1	CyanEven:P_ADR	P_ADR+=160	前进到段 1（青色）
2	CyanEven:P_ADR	P_ADR+=160	前进到段 2（青色）
3	CyanEven:P_ADR	P_ADR+=160	前进到段 3（青色）
4	CyanEven:P_ADR	P_ADR+=160	前进到段 4（青色）
5	CyanEven:P_ADR	P_ADR+=160	前进到段 5（青色）
6	CyanEven:P_ADR	P_ADR+=160	前进到段 6（青色）
7	CyanEven:P_ADR	P_ADR+=160	前进到段 7（青色）
8	CyanEven:P_ADR	P_ADR=Magenta_P_ADR Magenta_P_ADR+=1 (mod5)	生成品红色段 0 中偶数象素的地址并前进到品红色的下一象素
9	MagentaEven:P_ADR	P_ADR+=160	前进到段 1（品红色）
10	MagentaEven:P_ADR	P_ADR+=160	前进到段 2（品红色）
11	MagentaEven:P_ADR	P_ADR+=160	前进到段 3（品红色）
12	MagentaEven:P_ADR	P_ADR+=160	前进到段 4（品红色）
13	MagentaEven:P_ADR	P_ADR+=160	前进到段 5（品红色）
14	MagentaEven:P_ADR	P_ADR+=160	前进到段 6（品红色）
15	MagentaEven:P_ADR	P_ADR+=160	前进到段 7（品红色）
16	MagentaEven:P_ADR	P_ADR=Yellow_P_ADR Yellow_P_ADR+=1(mod5)	生成黄色段 0 中偶数象素的地址并前进到黄色的下一象素
17	YellowEven:P_ADR	P_ADR+=160	前进到段 1（黄色）
18	YellowEven:P_ADR	P_ADR+=160	前进到段 2（黄色）
19	YellowEven:P_ADR	P_ADR+=160	前进到段 3（黄色）
20	YellowEven:P_ADR	P_ADR+=160	前进到段 4（黄色）
21	YellowEven:P_ADR	P_ADR+=160	前进到段 5（黄色）
22	YellowEven:P_ADR	P_ADR+=160	前进到段 6（黄色）

23	YellowEven:P_ADR	P_ADR+=160	前进到段 7（黄色）
24	YellowEven:P_ADR	P_ADR=Cyan_P_ADR Cyan_P_ADR+=1(mod5)	生成青色段 0 中偶数象素的地址并前进到青色的下一象素
25	CyanOdd:P_ADR	P_ADR+=160	前进到段 1（青色）
等			

以下列出用于生成缓冲器 4 117 的地址的伪码。请注意，以步骤的顺序集合方式列出伪码。表 26 能够更好地表示地址生成操作的并行性质。

```
%Calculate start positions
CyanEven = 0:0
CyanOdd = CyanEven + 0:1
MagentaEven = CyanOdd + 1:3
MagentaOdd = MagentaEven + 0:1
YellowEven = MagentaOdd + 1:3
YellowOdd = YellowEven + 0:1

Do N times (depend on print size)
 Cyan_P_ADR = 0
 Magenta_P_ADR = 0
 Yellow_P_ADR = 0
 Do 400 times
     %generate the even pixels for the first set of 24 bits
     P_ADR = Integer portion of Cyan_P_ADR
     Cyan_P_ADR += 0:1
     Do 8 times
         ReadBuffer4(line=CyanEven,pixel=P_ADR)
```

```
    P_ADR += 160
EndDo
P_ADR = Integer portion of Magenta_P_ADR
Magenta_P_ADR += 0:1
Do 8 times
    ReadBuffer4(line=MagentaEven,pixel=P_ADR)
    P_ADR += 160
EndDo
P_ADR = Integer portion of Yellow_P_ADR
Yellow_P_ADR += 0:1
Do 8 times
    ReadBuffer4(line=YellowEven,pixel=P_ADR)
    P_ADR += 160
EndDo

%generate the odd pixels for the first set of 24 bits
P_ADR = Integer portion of Cyan_P_ADR
Cyan_P_ADR += 0:1
Do 8 times
    ReadBuffer4(line=CyanOdd,pixel=P_ADR)
    P_ADR += 160
EndDo
P_ADR = Integer portion of Magenta_P_ADR
Magenta_P_ADR += 0:1
Do 8 times
    ReadBuffer4(line=MagentaOdd,pixel=P_ADR)
    P_ADR += 160
EndDo
P_ADR = Integer portion of Yellow_P_ADR
```

```
        Yellow_P_ADR += 0:1
        Do 8 times
            ReadBuffer4(line=YellowOdd,pixel=P_ADR)
            P_ADR += 160
        EndDo

        %Now can advance to next "line"
        CyanEven += 0:1
        CyanOdd += 0:1
        MagentaEven += 0:1
        MagentaOdd += 0:1
        YellowEven += 0:1
        YellowOdd += 0:1
    EndDo
EndDo
```

10.2.5 缓冲器 3 116

缓冲器 3 是 8 比特 R、G、B 值的简单集合。这些 RGB 值为锐化处理 65 生成的锐化后的中分辨率（1280 分辨率）象素，转换为 CMY 处理 66 读取这些值。

不需要使缓冲器 3 116 加倍。原因在于读（转换为 CMY）处理 66 只需要前 39 个周期的 RGB 值，而写（锐化）处理 65 使用实际更新 RGB 值前的 49 个周期。

10.2.6 转换为 CMY 66

正如 3.2.7 所述，在中分辨率空间（1280 分辨率）中执行从 RGB 到 CMY 的转换。

转换处理 66 必须以足以跟上向上内插-半色调-重新格式化处理 113 的速度，生成连续色调缓冲器象素（缓冲器 4）117。由于每个连

续色调值用于 25 个周期（沿 x 和 y 维数各 5 次）所以转换处理占用 25 个周期。三个颜色分量共需 75 个周期。

此处说明的处理每颜色分量只需 14 个周期，并且在 39 个周期后释放输入的 RGB 值。如果实现该处理的逻辑访问输入 RGB 值的周期多余 49 个周期，则必须使缓冲器 3 116 加倍，因为锐化处理 65 需要的该时刻后更新缓冲器。

以三线性内插方式实现转换。转换处理使用三个 17×17×17 查找表：RGB 到青色 90，RGB 到品红 91，以及 RGB 到黄色 92。然而，由于需要 25 个周期执行个三线性内插，因此，不需要快速三线性内插装置。而调用线性内插处理 130 8 次就足够了。

用于查找表索引的地址生成很简单。我们使用 8 位颜色分量的 4 个最高有效位作为地址生成，8 位颜色分量的 4 个最低有效位用于从转换表检索的数值之间的内插。由于查找表的维数是 17 而不是 16，所以查找表的寻址需要一个加法器。幸运的是，将一个 4 比特数 X 乘以 17 得到一个 8 比特数 XX，因此，不需要加法器或乘法器，将一个 4 比特数乘以 17^2（289）稍微复杂一些，需要一次加法。

尽管可以更快地执行内插，但是我们使用一个加法器生成地址，并且有一个单周期内插装置。因此，正如表 27 所示，我们能够计算在 14 个周期内根据 RGB 生成单一颜色分量的内插。为了生成青、品红和黄色分量，需要将该处理重复 3 次。更快的方法也是可行的，但没有必要。

表 27. 用于颜色转换的三线性内插

周期	加载	有效取数	调整 ADR 寄存器	内插
1			ADR=289R	
2			ADR=ADR+17G	
3			ADR=ADR+B	
4	P1	RGB	ADR=ADR+1	
5	P2	RGB+1	ADR=ADR+16	

6	P1	RG+1B	ADR=ADR+1	P3=P1 到 P2，通过 B
7	P2	RG+1B+1	ADR=ADR+271	
8	P1	R+1GB	ADR=ADR+1	P4=P1 到 P2，通过 B
9	P2	R+1GB+1	ADR=ADR+16	P5=P3 到 P4，通过 G
10	P1	R+1G+1B	ADR=ADR+1	P3=P1 到 P2，通过 B
11	P2	R+1G+1B+1		
12				P4=P1 到 P2，通过 B
13				P6=P3 到 P4，通过 G
14				V=P5 到 P6，通过 R

正如表 27 所示，使用一个 ADR 寄存器和加法器生成查找表的地址。可以使用 6 组 8 位寄存器保存中间结果 —2 个寄存器保存从查找表中加载的值，4 个寄存器用于内插装置的输出。请注意，线性内插装置的输入总是一对 8 位寄存器 P1/P2，P3/P4 和 P5/P6。目的是简化寄存器选择逻辑。在周期 14 中，“V”寄存器 131 保存最后计算的 8 位值。可以在下一个周期中，将 8 位结果写入缓冲器 4 117 中的适当位置。

图 48 表示转换为 CMY 处理 66 的框图。

假设第一次运行该处理生成青色，将合成的青色连续色调象素存储到青色 1280 中分辨率连续色调缓冲器中。然后在相同 RGB 输入上再次进行处理以生成品红色象素。将品红色连续色调象素存储到品红色 1280 中分辨率连续色调缓冲器中。最后，根据相同的 RGB 输入生成黄色连续色调象素。将生成的黄色象素存储到黄色 1280 中分辨率连续色调缓冲器中。

用于写入连续色调缓冲器（缓冲器 4）117 的地址生成很简单。使用一个地址（和附随的 ColorSelect 位）写入三个颜色缓冲器的每个缓冲器中。在周期 15 上写入青色缓冲器，在周期 30 上写入品红色缓冲器，在周期 45 上写入黄色缓冲器。每 75 个周期（在写入三种颜色后）将象素地址加 1。每 5 个 AdvanceLine 脉冲，正在写入的行加 1（环绕）。

正在写入的行的顺序为 0-1-2-3-4-5-0-1-2-3 等。因此，写入（25×1280×3）与读取（19200×5）衡消。

10.2.7 缓冲器 2 115

缓冲器 2 接受重新采样-创建亮度（CreateLuminance）处理 112 的输出，其中在处理 112 中生成给定象素坐标的全部 RGB 和 L 象素。缓冲器 2 115 的输出进入锐化处理 65，该处理需要一个 3×3 的亮度值 135，后者以正在锐化的象素为中心。

因此，在锐化处理 65 中，需要访问 3×3 的亮度值，以及中心亮度象素的对应 RGB 值 136。同时，重新采样-创建亮度处理 112 必须计算下三个亮度值以及对应的 RGB 中心值。图 49 表示访问缓冲器 1 115 的逻辑视图。

缓冲器 2 115 的实际实现只是一个 4×6（24 项）的 8 位 RAM，有关读写的寻址提供这些值的有效移位。可以将 2 比特的列计数器加 1（环绕）以提供循环缓冲器，这等价于将整个缓冲器的数据移动 1 列。我们不需要第四列 RGB 数据这一事实没有什么意义，并且只使用 3 个字节，因此无需实现复杂的移位和读/写逻辑。在给定周期中，可以读写 RAM。读写处理具有 75 个周期，在 75 个周期内实现读写以便跟上打印头。

10.2.8 锐化

锐化装置 65 执行 3.2.5 说明的锐化任务。由于在缓冲器 3 116 中存储锐化后的 RGB 象素，所以锐化装置 65 必须跟上转换为 CMY 处理 66，这意味着必须在 75 个周期内锐化全部 RGB 象素。

正如第 35 页上 3.2.6.2 部分表 12 说明的那样，锐化处理包括 L（根据 RGB 数据生成的信道并存储在缓冲器 2 中）的一个高通过滤器，并且将过滤后的 L 添加到 RGB 分量中。如图 50 所示，使用的高通过滤器是一个使用 3×3 卷积内核的基础高通过滤器。

在 10 个周期上计算高通过滤器。第一周期向临时寄存器 140 加载

8×中心象素值（中心象素左移3位）。下8个周期用基数0减去剩余的8个象素值。因此可以用一个加法器完成整个过程。周期10包括将结果乘以常数141。以上常数表示1/9，但是是一个寄存器，可以利用软件按一定比例系数修改以上常数。

然后，将总量加到R、G、B值（上限为255），并且在周期72、73和74内，写入缓冲器3。在75个周期的最后3个周期内计算/写入锐化后的RGB值，从而无需使缓冲器3加倍。

图51表示锐化装置的结构。

与缓冲器2 115相连的加法器142是一个以0为基数的减法器。在周期0（共75个）期间，向TMP 140加载8×第一L值，然后从中减去下8个L值。其结果没有符号，因为减法具有基数0。

在第10个周期（周期9）期间，将TMP 140中的11比特总数乘以一个比例系数（通常为1/9，但是受软件控制，因此可以调整系数），然后写回到TMP 140。只将结果的8整数比特写到TMP中（截去分数），因此乘法器的限制为255。如果使用比例系数1/9，则写入的最大值将是226（255×8/9）。比例系数为8比特分数，其最高位表示1/8。可变比例系数考虑了以下事实，即不同打印格式是利用不同数量缩放CFA图象的结果（因此，3×3卷积将生成适当比例的结果）。

在周期72、73和74期间，计算红、绿、蓝的锐化值，然后写入缓冲器3 116的R、G、B寄存器，每周期写一次。在上述3个周期内执行的计算为，将TMP加到与中心象素对应的缓冲器2中的R、G、B。

地址生成很简单。写入缓冲器3 116分别是周期72、73和74中的R、G、B。缓冲器2 115的读取操作使用缓冲器2的周期性质。地址包括一个2比特的列分量（表示读取4列中的哪一列），和一个3比特值（表示L1、L2、L3、R、G、B）。各行的列号从1开始，并且每75个周期加1（环绕）。表28表示读取缓冲器2的顺序。C寄存器为地址的2比特列分量。C上的所有加法均以4作为模数（在2比特内环绕）。

表 28. 75 个周期内缓冲器 2 的读访问

周期	地址	更新 C
0	C, L2	C=C-1
1	C, L1	
2	C, L2	
3	C, L3	C=C+1
4	C, L1	
5	C, L3	C=C+1
6	C, L1	
7	C, L2	
8	C, L3	C=C-1
9-71	不访问	
72	C, R	
73	C, G	
74	C, B	C=C-1

在周期 74 后，C 寄存器保持下一个计算集合的列数，因此，使得下一个周期 0 中的取数有效。

当在缓冲器 2 中写入足够的 L 和 RGB 象素后(从而高通过滤器有效)，锐化处理才开始。只有缓冲器 2 的写入处理前进 3 列后，锐化处理才开始。

10.2.9 缓冲器 1 114

缓冲器 1 以原始获取空间分辨率保持白平衡和范围扩展的象素。以 10 比特的颜色分辨率保存各象素，而图象 RAM 的图象存储颜色分辨率为每象素 8 比特。

缓冲器 1 的排列为 3 个可独立寻址的缓冲器 — 每个颜色平面一个，红 145，绿 146，蓝 147。图 52 表示缓冲器的简单概述。

在75个周期期间，重新采样处理112从3个缓冲器的各缓冲器中读出16项，共读取3次，并且最多向3个缓冲器写入29个新值（实际数目取决于比例系数，以及重新采样期间当前次象素的位置）。

缓冲器必须足够宽，从而读取和写入操作不会彼此干扰。在读取过程中，从6行中的每一行中读取4个象素。如果比例系数很大（如，按比例放大为全景），则可以多次读取同一象素（使用不同内核位置进行重新采样）。最后，需要下一个象素。如果放大系数不是很大，则在下一个象素生成周期之前需要新象素（即，在75个时钟周期内）。

考虑表9和表11中的比例系数，缩放的最坏情况是护照格式31:

- 绿色平面具有的护照的Δ值为1.5625，表示6个CFA象素位置内可以包含4个位置。然而，每一行绿色采样仅包含各交替象素。这意味着每行只需要4个采样（最坏情况为4而不是3，原因在于最坏情况为初始位置）。Y方向上的移动表示需要一个附加的采样列，从而需要5个采样。最后，写入操作需要一个附加的采样列。因此，每行共需6个采样。一个采样需要7行。为了生成每个x位置的3组RGB象素，沿y方向的最大移动将为4行（$3.125 = 2 \times 1.5625$）。移动X添加上一行采样和下一行采样。因此，共需13行。有关细节请参见10.2.10。
- 红色和蓝色平面具有的护照的Δ值为0.78125，表示在4个采样内可以包含4个位置。当正在读取剩余4个采样时，写入操作需要一个附加采样。每行共需5个采样，将采样数增加到6个采样以匹配绿色平面（用于启动）。为了适合y方向的移动需要6行。有关细节请参见10.2.10。

每个子缓冲器是一个带有译码功能的RAM，以便在每个周期内读写一个10比特的采样。表29概要表示子缓冲器，其消耗小于200字节。

表29. 子缓冲器概要

缓冲器	组成	位
红色缓冲器	6行×6采样/行×10比特/采样	360
蓝色缓冲器	6行×6采样/行×10比特/采样	360
绿色缓冲器	13行×6采样/行×10比特/采样	780
	总计	1500

10.2.10 重新采样和创建亮度信道

正如3.2.5中说明的那样，重新采样和创建亮度信道处理112通过对白平衡和范围扩展后的R、G、B平面图象重新采样，生成中分辨率空间中的RGB象素值。另外，必须生成给定RGB象素的亮度值，以及该RGB象素前后的象素的亮度值，以供稍后的锐化处理使用。

生成RGB值和3个L值的允许时间是75个周期。假设L是给定象素位置的R、G、B的最大值和最小值的平均值（参见3.2.6.1），必须有效生成3个象素（正在考虑的象素，以及上一个象素和下一个象素）坐标的RGB值。因此，我们在75个周期内计算3个中分辨率RGB采样及其对应的L值。

缓冲L值（因而缓冲RGB值）以避免重新计算需要大量存储器，总之，我们有足够时间生成RGB值。缓冲器4 117包含中分辨率象素，但是不能使用缓冲器4，因为它保持锐化后的CMY象素（而不是未经锐化的RGB象素）。

10.2.10.1 重新采样

可以将重新采样处理视为3组RGB生成，必须在25个周期内完成每组生成（共需75个周期）。可以将生成一个RGB值的过程视为3个并行执行的过程：对于给定的中分辨率象素坐标，计算R，计算G，计算B。可以在3.2.5中找到生成上述值的理论依据，但结果是在图象的信道上有效运行3个图象重建过滤器，每个信道上一个。在PCP的情况中，利用5个采样点执行图象重建，从而卷积内核需要4个系数（由于一个系数总是0，所以不需要该采样点）。

因此，通过在R数据上运行图象重建过滤器计算中分辨率的R象素。通过在G数据上运行图象重建过滤器计算中分辨率的G象素，通过在B数据上运行图象重建过滤器计算中分辨率的B象素。尽管内核在x和y方向上是对称的，但是每种颜色平面的内核并不相同。由于R和B的图象特征相似，所以其内核相同，但是对于G平面，由于其图象重建需要旋转操作，所以具有不同内核。图53表示该处理的高级视图。未示出地址生成。

只有正在生成的当前象素行的缓冲器1中具有足够象素，重新采样处理才能开始。即将4列数据写入缓冲器1 114中的各颜色平面后才开始处理。此前重新采样处理112必须停止。

为了计算给定颜色平面的中分辨率象素值，需要25个周期。为了将内核应用于4×4采样区域，在4行采样（每行4个输入采样）的每一行上应用一维内核（用x作索引）。然后在生成的4象素值上应用一维内核（用y作索引）。最终结果是输出重新采样后的象素。在每个周期内应用一个系数共需16个周期才能生成4个中间值，并且生成最终象素值需要4个周期，从而共需20个周期。

关于精度，输入象素为10比特（8:2），内核系数为12比特。在应用内核的4个步骤中，保持14比特的精度（8:6），但是结果只保存10比特（8:2）。因此，当沿x和y方向进行卷积运算时，可以使用相同的卷积引擎。最终输出（即R、G、B）为8比特。

重新采样处理的核心是图54所示的卷积装置150。

如图30所示，重新采样处理占用20个周期。请注意，“行1，象素1”指缓冲器1 114中的输入，并且采用以下寻址机制。

表30. 20个周期重新采样

周期	内核	将内核应用于	存储结果于
1	X[1]	行1，象素1	TMP
2	X[2]	行1，象素2	TMP
3	X[3]	行1，象素3	TMP

4	X[4]	行 1，象素 4	TMP，V1
5	X[1]	行 2，象素 1	TMP
6	X[2]	行 2，象素 2	TMP
7	X[3]	行 2，象素 3	TMP
8	X[4]	行 2，象素 4	TMP，V2
9	X[1]	行 3，象素 1	TMP
10	X[2]	行 3，象素 2	TMP
11	X[3]	行 3，象素 3	TMP
12	X[4]	行 3，象素 4	TMP，V3
13	X[1]	行 4，象素 1	TMP
14	X[2]	行 4，象素 2	TMP
15	X[3]	行 4，象素 3	TMP
16	X[4]	行 4，象素 4	TMP，V4
17	Y[1]	V1	TMP
18	Y[2]	V2	TMP
19	Y[3]	V3	TMP
20	Y[4]	V4	TMP（用于输出）

10.2.10.2 生成 L 8

正如 3.2.6.1 所述，必须从 RGB 转换 80 为 L，以便随后的锐化处理。考虑 CIE 1976 $L^*a^*b^*$ 颜色平面，其中感觉上 L 是一致的。为了从 RGB 转换到 L（亮度信道），按下式计算 R、G、B 的最大值和最小值的平均值:

$$L = \frac{MIN(R,G,B) + MAX(R,G,B)}{2}$$

并行生成给定象素的 R、G、B 值，该处理需要 20 个周期。此处说明的生成 L 的总时间是 4 个周期。因此生成 RGBL 的总时间是 24

个周期，有一个周期空闲（因为必须在25个周期内完成处理）。

因此，可以在第25个周期内将L的值安全写到缓冲器2 115内。以下说明地址生成。

如表31所示，一个8位比较器能够在3个周期内生成3个比特，随后使用这3个比特选择加法器的2个输入。可以在加法器内实现以2为除数的除法运算。

表31. 根据三次比较选择Min和Max

MIN	MAX	R>G	G>B	R>B
R	B	1	1	x[a]
R	G	1	0	1
G	R	0	1	0
G	B	0	1	1
B	G	0	0	x
B	R	1	0	0

a.不关心状态

由于只将最小值加到最大值中，所以顺序并不重要。因此，对于加法器的两个输入，输入1选择R和G，输入2选择G和B。该逻辑是表31的位组合模式的最小化。

10.2.10.3 缓冲器2的地址生成

重新采样装置的输出是一个RGB象素，和以RGB的垂直方向为中心的3个亮度（L）象素。将3个L值写到缓冲器2中，每25个周期一个。必须在周期45和周期50之前写入R、G、B值，因为生成的第二象素为必须保持其RGB值的中心象素。缓冲器2的地址包括一个2比特的列部分（表示写到4列中的哪一列），和一个表示L1、L2、L3、R、G、B的3比特值。每行的列号均从0开始，并且每75个周期（即，写出L3后）加1（环绕）。

10.2.10.4 内核查找表的地址生成

内核地址的计算方法与3.2.5的结尾部分说明的方法相同。内核的维数为1，查找表内有64项。使用当前内核空间中分数部分的6个最高有效位（截尾）作为内核系数表的索引。对于前16个周期，使用X坐标作为内核索引，而在下4个周期中，使用Y坐标。由于内核是对称的，所以X和Y可以使用相同的内核。

对于1280个重新采样值的每个值，需要生成3个象素 — 正在考虑的象素161，该象素之前的象素160和该象素之后的象素162。我们生成象素160，然后生成两个象素161和162，而不是生成中心象素，然后从中心象素向上、向下移动。接着，返回到原始行，生成下一个输出位值的下3个象素。这样，如图55所示，生成1280个位置的每个位置的象素。

因此，我们在内核空间中有一个当前位置。当在原始输入空间中沿x或y前进到下一象素时，向内核坐标添加适当的Δ值。考虑图56，可以看到旋转和不旋转输入空间两种情况。

考虑沿X方向的移动ΔX和沿Y方向的移动ΔY，其值取决于打印格式，因此mps（见3.2.5）也取决于打印格式。对于绿色信道，ΔX=ΔY=1/2mps。对于红色和蓝色信道，ΔX=1/mps，ΔY=0。有关ΔX和ΔY的适当值参见表9和表11。

现在，将ΔX和ΔY应用于内核内的移动。当沿X方向前进时，将ΔX加到X中，并从Y中减去ΔY。在不旋转情况中，只需要从Y中减去0。同样，当沿Y方向前进时，将ΔY加到X中，并将ΔX加到Y中。这么做是因为沿X和Y方向的移动相差90度。

内核查找表的地址生成假定由软件设置开始位置，以及相对于内核空间中沿Y方向移动的两个Δ值ΔX和ΔY。以下伪码表示地址生成逻辑。

```
ColumnKernelY = StartKernelY
```

```
ColumnKernelX = StartKernelX
Do NLines times (however many output lines there are to process)
   KernelX = ColumnKernelX
   KernelY = ColumnKernelY
   Do 1280 times
      GeneratePixel
      KernelX = KernelX + DeltaY (movement in Y)
      KernelY = KernelY + DeltaX (movement in Y)
      Generate Pixel
      KernelX = KernelX + DeltaY (movement in Y)
      KernelY = KernelY + DeltaX (movement in Y)
      GeneratePixel
      KernelX = ColumnKernelX + DeltaX (movement in X)
      KernelY = ColumnKernelY - DeltaY (movement in X)
   EndDo
   ColumnKernelY = ColumnKernelY + DeltaX (movement in Y)
   ColumnKernelX = ColumnKernelX + DeltaY (movement in Y)
EndDo
```

正如该伪码所示，3 个象素的生成出现 1280 次。与每个象素生成关联的是两次加法，在 GeneratePixel 的 25 个周期任务中执行加法。每个 GeneratePixel 任务为 25 个周期，包括 4 组通过 KernelX（系数 0、1、2、3）对内核进行索引的 4 个周期，然后是通过 KernelY（系数 0、1、2、3）对内核进行索引的 4 个周期，最后是 9 个等待周期。

请注意，所有值全是正的，并且仅为分数。将更新的 X 和 Y 内核值的两个进位输出到缓冲器 1 的地址生成（见 10.2.10.5）。进位标志仅仅表示算术运算期间内核的特定坐标是否环绕。环绕可大于 1 或小于 0，但是结果总是正的。

将两个进位发送到旋转/白平衡/范围扩展装置，用于确定图象的有

关输入行。

10.2.10.5 缓冲器 1 的地址生成

重新采样装置 112 从缓冲器 1 114 中读取数据，后者包括 3 个可独立寻址的缓冲器 145、146 和 147，每个颜色平面一个。可以在每个周期中读写各缓冲器。

$$L = \frac{MIN(R,G,B) + MAX(R,G,B)}{2}$$

将 75 个周期的读取处理分成 3 组，每组 25 个周期，25 个周期用于生成各象素。每 25 个周期包括读取缓冲器 1 的 16 个周期和没有访问操作的 9 个周期。在这 9 个周期中将数据写入缓冲器 1。将缓冲器 1 114 的 16 个读周期分成 4 组，每组 4 个读周期，从而与各颜色平面的每组 4 个读周期的 4 个分组一致。

地址生成包括生成计算第一象素的 16 个地址（然后是 9 个等待周期），生成计算第二象素的 16 个地址（然后是 9 个等待周期），和最终生成计算第三象素的 16 个地址（然后是 9 个等待周期）。

每个颜色平面具有其特有的启动缓冲器 1 的地址参数。在生成每行的 1280 个象素的每个象素的 3 组 16 个地址，并且当采样装置从每行 1280 个采样前进到下一行时，使用内核地址生成装置的两个进位来更新缓冲器 1 的地址参数。

10.2.10.6 绿色缓冲器 146

缓冲器 1 114 内的绿色子缓冲器 146 比红色子缓冲器 145 和蓝色子缓冲器 147 复杂，两个主要原因是:

- 绿色信道表示 CFA 中的交错排列格式。交错行仅包括奇数或偶数象素。为了对绿色信道进行重新采样，必须将其信道旋转 45 度。
- 绿色象素的数目为红色或蓝色象素的两倍。重新采样意味着要

在相同时间内读取更多采样 — 尽管生成中分辨率空间中的每个象素仍然读取16个采样，但是每次提前缓冲器的可能性更大。可能性取决于使用的比例系数。

然而，绿色信道仍然采用使用RAM作为循环缓冲器的相同概念。绿色子缓冲器是一个78项的RAM，逻辑排列为13行，每行6项。图57表示RAM地址和逻辑位置之间的关系。

缓冲器1 146中的采样表示CFA中的交错排列模式。因此，一行（如，0，13，26，39，52，65）中的采样可以表示奇数或偶数象素，这取决于整个图象内的当前行，以及是否将该图象旋转了90度。图58表示此种情况。

因而，当我们将一个4×4采样区域映射到缓冲器上时，采样的解释有两种可能性。因此，有两类寻址，取决于当前行是用奇数象素还是用偶数象素表示的。这意味着图象旋转0度的偶数行与图象旋转90度的奇数行的寻址相同，由于它们都保持奇数象素的缘故。同样，图象旋转0度的奇数行与图象旋转90度的偶数行的寻址相同，由于它们都保持偶数象素的缘故。表32概要表示此判定。

表32. 确定采样类型

旋转	当前行		象素	类型
0	偶数行	8	奇数	类型2
0	奇数行	8	偶数	类型1
90	偶数行	8	偶数	类型1
90	奇数行	8	奇数	类型2

实际上，我们采用4×4采样窗口将缓冲器旋转45度。正如3.2.5说明的那样，实际的重新采样需要45度旋转。

假设目前我们只需要生成单个重新采样，通过检查图59所示的两类4×4采样窗口实现缓冲器寻址。

尽管两类 4×4 采样窗口看起来很相似，其差别在于在平面图象中表示 4×4 映射的方式。图 60 表示将类型 1 的 4×4 采样映射到绿色子缓冲器。仅仅显示了前 7 行以及最右边的 4 列，因为该区域完全包含 4×4 采样区域。

正如在图 61 中看到的那样，对于类型 2 的采样处理，从缓冲器象素到采样行的映射是非常相似的。

在 16 个象素的类型 1 和类型 2 寻址中，处理一行采样的方式有两种。处理类型 1 的第 1 行和第 3 行与处理类型 2 的第 2 行和第 3 行相同（相对而言）。同样，处理类型 1 的第 2 行和第 4 行与处理类型 2 的第 1 行和第 3 行相同（相对而言）。如图 62 所示，将上述行寻址方法称为类型 A 170 和类型 B 171。

给定 4×4 窗口的起始位置（WindowStartAdr）和起始类型（WindowStartType），借助于一个 8 项查找表（用于遍历两组 4 个采样）生成 16 个采样的地址。当我们读取第一个采样值时，添加该表中的一个偏移以便到达下一个采样位置。该偏移取决于类型（A，B=0，1）。第四个采样的偏移值为到达下一行的第一个采样点需要的数量（必须考虑采样的列数）。在生成每行 4 个采样后，交换 TypeA 和 TypeB。以下伪码表示生成一组 16 个采样的地址的逻辑。以 78 为模数的加法适合循环缓冲器。

```
Adr = WindowStartAdr
TypeAB = WindowStartType
Do 4 times
   For N = 0 to 4
      Fetch Adr
      Adr = (Adr + Table[TypeAB,N]) mod 78
   EndFor
   TypeAB = NOT TypeAB
EndDo
```

查找表包括 8 项 — 4 项用于类型 A 170 的地址偏移生成，4 项用于类型 B 171 的地址偏移生成。偏移为相对于当前采样位置（Adr）的偏移。

表 33. 16 个采样地址生成的偏移值

TypeAB	N	偏移
0	0	14
0	1	1
0	2	14
0	3	37
1	0	1
1	1	14
1	2	1
1	3	37

在 16 个读取操作结束时，TypeAB 位与原始值相同（从 WindowStartType 加载）。

仅读取一组 16 个采样是不够的。必须读取 3 组 16 个采样（表示不旋转的输入空间中沿 Y 方向的 3 个不同位置）。在第一组和第二组 16 个采样结束时，利用内核地址生成器更新内核位置。使用上述更新的进位来设置下一组 16 个采样的窗口。两个进位作为包含一个偏移和 1 位标志的查找表的索引。将偏移添加到 WindowStartAdr 中，使用该标志确定是否倒置 WindowStartType。表 34 表示该查找表的值。

表 34. 更新 WindowStartAdr 和 WindowStartType

KernelX 进位	KernelX 进位	偏移	类型
0	0	0	不变

0	1	1	倒置
1	0	14	倒置
1	1	2	不变

在第三组16个采样结束时，更新内核位置以补偿不旋转的输入空间中沿X方向的前进。此时，生成不同的运动方向，从而使用不同的偏移/TypeAB修改表。不能将这些偏移添加到当前的WindowStartAdr值中，因为它们表示某个位置沿Y方向上的远离开始运动的两个运动。因此，我们从另一组变量中加载WindowStartAdr和WindowStartType: TopStartAdr，TopStartAdr表示当前行的1280个象素的第一项。将内核地址生成器的两个进位应用于查找表35，以确定添加到TopStartAdr的偏移，以及是否倒置TopStartType。如前所述，加法的模数是78（绿色RAM的大小）。将结果复制到WindowStartAdr和WindowStartType中，以便在生成下3组16个采样时使用。

表35. 更新TopStartAdr和TopStartType

KernelX 进位	KernelX 进位	偏移	类型
0	0	0	不变
0	1	12	倒置
1	0	13	倒置
1	1	14	不变

在处理1280个3组16个采样后，下一行的1280个象素开始。然而，必须确定下一行内位置0的第一个采样的地址。由于总是将象素加载到缓冲器1中的正确位置，所以总能从缓冲器1内的正确位置开始（即，可以从固定的Position0Adr中加载TopStartAdr）。然而，我们担心处理哪种类型，因为类型取决于我们前进的了多少。所以，

我们具有一个初始 Position0Type，必须根据内核地址生成器的进位标志更新后者。由于我们在不旋转的 Y 输入空间中前进，所以使用的逻辑与更新 WindowStartType 的逻辑相同。只是在 Position0Type 上执行。将 Position0Type 的新值复制到 TopStartType 中，并且 WindowStartAdr 开始新行的第一位置的采样。

只有旋转/白平衡/范围扩展装置在缓冲器 1 中放置了足够项目后，指定的 1280 个位置的采样处理才能开始。在开始各新行后，以上处理将存在 128 个周期（见 10.2.11）。

10.2.10.7 红色和蓝色缓冲器

缓冲器 1 的红色子缓冲器 145 和蓝色子缓冲器 147 只是两个作为循环缓冲器的 RAM。每个缓冲器为 30 字节，逻辑排列为 6 行，每行包含 6 项。图 63 表示 RAM 地址和逻辑位置之间的关系。

对于红色和蓝色，需要读取的前 16 项总是前 4×4 项。读取算法在本阶段并不访问采样的剩余两列。

前 16 项的地址生成只是一个起始位置（此时为 0），然后是 16 个模数为 36 的加法，如以下伪码所示：

```
ADR = StartADR
Do 4 times
  Do 4 times
     ADR = ADR + 6 MOD 36
  EndDo
  ADR = ADR + 13 MOD 36
EndDo
```

然而，以上地址生成机制与绿色信道的不同。可以将绿色寻址机制应用于红色和蓝色信道，并且只需在表中使用不同值，而不是设计两种寻址机制。从而降低了设计复杂性。其唯一区别在于，采用模数

为 36 的加法，而不是模数为 78 的加法。可以采用简单乘法器。

考虑绿色信道的各种地址生成表，并认为将其应用于红色和蓝色，显然不需要类型，因为不需要将红色和蓝色信道旋转 45 度。因此确实可以忽略类型值，表 36 所示的表 33 的红色/蓝色等价表具有两组完全相同的 4 项。

表 36. 16 个采样地址生成的偏移值（红/蓝）

TypeAB	N	偏移
0	0	6
0	1	6
0	2	6
0	3	13
1	0	6
1	1	6
1	2	6
1	3	13

关于绿色地址生成，在前进到 1280 项的下一项前，沿 Y 方向移动两次。对于红色和蓝色，内核空间中的移动和输入空间中的移动之间没有缩放。也没有旋转。当沿 Y 方向移动时，将ΔY=0 添加到 KernelX 中（见 10.2.10.4 中的地址生成）。因此，从不设置 KernelX 的进位。考虑表 34，唯一可能发生的是 KernelX/ KernelY=00 或 01。关于 00，绿色解决方法是不改变 WindowStartAdr 或 WindowStartType，这同样适用于红色和蓝色。关于 01，将 1 添加到 WindowStartAdr 中，而不关心 WindowStartType。因此，确实可以对红色和蓝色使用绿色值。最坏情况是两次地址都前进 1，从而导致图 65 所示的最坏的重叠情况。

在第三组 16 个采样结束时，必须更新 TopStartAdr 和 TopStartType。由于正沿 X 方向移动（并且将ΔY=0 添加到 KernelY 中），所以 KernelY 的进位总是 0。表 37 表示表 35 的红色/蓝色等价

表。请注意，没有类型列，由于类型对红色和蓝色不重要。

表 37. 更新 TopStartAdr 和 TopStartType（红/蓝）

KernelX 进位	KernelX 进位	偏移
0	0	0
0	1	-
1	0	6
1	1	-

从 1280 组 3 象素组成的一行前进到下一行的处理与用于绿色的处理相同。Position0Adr 与给定行（对于红色和蓝色，Position0Adr=0）的第一组 16 个采样的 Position0Adr 相同，而与类型无关。只有缓冲器 1 中具有足够采样，下一行的生成才能开始。红色和蓝色生成必须与绿色生成同时开始，因此，只有在新行开始 128 个周期后才能开始（见 10.2.11）。

10.2.11 旋转、白平衡和范围扩展 111

正如 3.2.3 和 3.2.4 中说明的那样，从图象 RAM 11 中加载缓冲器 1 114 的实际任务包括以下步骤：旋转、白平衡和范围扩展 111。必须快速生成缓冲器 1 的象素，以供重新采样处理 112 使用。这意味着在 75 个周期中，该装置必须能够读取、处理、并存储 6 个红色象素、6 个蓝色象素和 13 个绿色象素。

通过以适当顺序读取象素实现作为选择步骤的旋转处理。在读出图象存储器中适当平面的给定象素后，必须进行白平衡，并且根据 3.2.4 中定义的范围扩展计算调整其值。该处理仅包括一次减法（基数 0）和一次乘法（最大值 255），其中相对于颜色特有的常数进行减法和乘法。图 68 表示此装置的结构。

在利用图象直方图单元 8（见第 9 节）生成各颜色平面的直方图后，

由 CPU 10 确定红、绿、蓝低阈值 72，以及红、绿、蓝比例系数 73。

根据当前流水线中处理的象素是红色，还是绿色，抑或是蓝色，在加法器和乘法器中复用正确的低阈值和比例系数，其中将输出写到缓冲器 1 中的适当颜色平面。

加法器 172 从 8 比特图象 RAM 象素值中减去 8 比特低阈值，并具有基数 0。将 8 比特结果传送到专用 8×8 乘法器上，后者将 8 比特值乘以 8 比特的比例系数（8 比特的分数，整数=1）。只保存结果的前 10 比特，并提供 8 比特的整数和 2 比特的分数。乘法器 174 的结果的最大值为 255，因此，如果作为乘法结果设置了比特 7 之前的任何比特，则将整个 8 比特整数结果设置为 0，将分数部分设置为 0。

除减法器 172 和乘法器 174 之外，由地址生成器 175 执行此装置内的大部分工作，实际上，地址生成器 175 是该装置的状态机。地址生成受两个因素控制：在一个给定周期内，只能对图象 RAM 11 进行一次存取，并且在一个给定周期内，只能对缓冲器 1 114 进行一次存取。在 75 个可用周期中，使用 3 组 16 个周期读取缓冲器 1。实际使用 3 组 25 个周期，其中 16 个读周期，然后是 9 个等待周期。总共提供 27 个可用周期用于 25 个写操作（6 个红色，6 个兰色，6 个绿色）。这意味着如果写入缓冲器 1 的定时与重新采样装置 112 的等待周期一致，则满足两个约束条件。

10.2.11.1 缓冲器 1 的地址生成

当重新采样处理运行时，我们只关心在重新采样装置 112 不读取缓冲器 1 时，写入缓冲器 1。由于重新采样装置在每 75 个周期中有 3 组 16 个读操作，所以有 27 个写入周期。当重新采样装置未运行时，我们希望尽快加载缓冲器 1， 这意味着每个周期内缓冲器 1 114 的一次写操作。因此，缓冲器 1 的地址生成迅速写出状态机，后者考虑以上两种情况。每当从图象 RAM 11 中加载数值时，总是在随后的一个周期内将调整值写到缓冲器 1 内的适当颜色平面中。

因此，缓冲器 1 的地址生成包括分别用于红色、蓝色、绿色子缓

冲器的地址计数器。在各行开始时，RedAdr、BlueAdr、GreenAdr的初值为0，在写入缓冲器1后，地址加1，其中以36或78作为模数，模数取决于正在写入的缓冲器是红色、还是绿色抑或是蓝色。并不是在每75个周期内写入所有颜色。通常绿色需要的填充速度是红色或蓝色的两倍。

以下伪码表示其逻辑:

```
If the color to write is Red
    Write to Red Buffer1 at RedAdr
    RedAdr = RedAdr + 1 mod 36
Else
If the color to write is Blue
    Write to Blue Buffer1 at BlueAdr
    BlueAdr = BlueAdr + 1 mod 36
Else
If the color to write is Green
    Write to Green Buffer1 at GreenAdr
    GreenAdr = GreenAdr + 1 mod 78
EndIf
```

10.2.11.2 图象RAM的地址生成

可以沿两个方向（逆时针旋转0度或90度）中的一个方向读取各平面。从而能够在按行或按列读取平面图象时进行有效转换。另外，对于图象边界之外的读取允许边缘图象复制或固定颜色，并且对于护照31之类的打印格式，允许图象环绕。

在每个打印行的开始，必须尽快读取图象RAM 11以加载缓冲器1 114。这等价于每个周期访问一个采样。只有在加载5列后重新采样才能出现，这意味着5列的6、6和13个采样，总共125个周期。外加一个额外周期，用于从图象RAM 11加载后，将终值写出到缓冲器

1 114 中。为了简化计数，上舍入为 128 个周期。

在前 128 个周期之后，每 75 个周期检查一次是否需要加载 3 种颜色的下一列采样，并且在随后的 75 个周期中加载适当采样。然而，对于每种颜色，是否在第一组 75 个周期期间加载采样的初始设置总是 1。从而填充缓冲器 1 内每种颜色的第 6 列采样。

在 75 个周期结束后，检查重新采样装置 112 中内核地址生成器的每个颜色平面的 KernelXCarryout 标志，以确定是否读取下一列采样。同样，如果设置了 KernelYCarryout 标志，则 AdvanceLine 脉冲重新启动下一行的处理。

由于填充缓冲器 1 中一列的每个 “读操作”实际上为 6 个或 13 个读操作，所以我们保存一个起始位置以便前进到下一个 “读操作”。同时，我们保存一个坐标值，以允许生成越界坐标，从而支持边缘象素复制、固定颜色和图象环绕。

我们认为有效图象 180 在特定边界内，并且当坐标在实际区域之外时采取某些操作。按照行或象素，坐标可以在图象之前，在图象内，或者在图象后。图 67 表示此种情况，尽管为了清晰性而夸大了有效区域之外的空间。

请注意，由于我们使用(0,0)作为坐标生成的起点，所以 MaxPixel 和 MaxLine 为象素计数和行计数。然而，由于根据内核进位和 MJI 15 的 AdvanceLine 脉冲进行地址生成，所以不需要外限。象素行的地址生成继续直至收到 AdvanceLine 脉冲，并且可以包括边缘复制、越界地址的固定颜色，或图象象素环绕。

如果具有当前采样的地址(Adr)，并且希望移动到下一行或同一行上的下一采样，则采样的坐标将改变，但是地址改变方式取决于是否围绕实际图象环绕，并且在需要时必须进行边缘象素复制。

如果图象没有环绕(即，除护照 31 之外的所有打印格式)，则在行或象素中前进时执行表 38 中的操作。为了将图象旋转 90 度，CPU 10 只需交换ΔLine 和ΔPixel 值。

考虑表 38，当 PixelSense=0 时，ADR 改变ΔPixel，并且当

LineSense=0 时，ADR 改变ΔLine。根据上述原则，Adr 对边缘象素复制有效。当然，如果固定颜色对越界坐标合适，则可以选择该值代替适当地址中存储的值。

表 38. 在象素或行中前进时执行的操作

Line[a]	Pixel[b]	象素变化	行变化
–	–		
–	0	Adr=Adr+ΔPixel	
–	+		
0	–		Adr=Adr+ΔLine
0	0	Adr=Adr+ΔPixel	Adr=Adr+ΔLine
0	+		Adr=Adr+ΔLine
+	–		
+	0	Adr=Adr+ΔPixel	
+	+		

a. 比较当前 Line 坐标与 ActiveStartLine 和 ActiveEndLine。

如果 Line < ActiveStartLine，称该值为 “–”。

如果 ActiveStartLine ≤ Line < ActiveEndLine，称该值为 “0”。

如果 ActiveEndLine ≤ Line，称该值为 “+”。

b. 比较当前 Pixel 坐标与 ActiveStartPixel 和 ActiveEndPixel。

如果 Pixel < ActiveStartPixel，称该值为 “–”。

如果 ActiveStartPixel ≤ Pixel < ActiveEndPixel，称该值为 “0”。

如果 ActiveEndPixel ≤ Pixel，称该值为 “+”。

为了允许环绕，只需比较 Line 和 Pixel 的前一个读数（–，0，+）

与新读数。当读数为 “－”时，使用表 38 所示的增量，但是当坐标越界（即，从 0 移动到+）时，利用新值更新 Adr，而不是基于Δ。假设我们保存当前行的起始地址，从而在生成当前行后前进到下一行的起始地址，则我们能够进行以下处理:

- 如果 Pixel 改变，并且 Pixel 读数从 0 变为+（表示我们越过图象的边界），则利用 LineStartAdr 代替 Adr，并利用 ActiveStartPixel 代替 Pixel。Line 保持不变。
- 如果 Line 改变，并且 Line 读数从 0 变为+（表示我们越过图象的边界），则从 Adr 中减去 DeltaColumn，并利用 ActiveStartLine 代替 Line。Pixel 保持不变。DeltaColumn 为地址偏移，用于根据（Pixel,ActiveEndLine － 1）生成（Pixel,ActiveStartLine）的地址。

以下伪码表示加载采样数目（或者为 6，或者为 13，取决于颜色）的逻辑:

```
line = FirstSampleLine
pixel = FirstSamplePixel
adr = FirstSampleAdr
Do N times (6 or 13)
    oldPixelSense = PixelSense(pixel)
    oldLineSense = LineSense(gLine)
    inActive =((oldLineSense == InActive) AND (oldPixelSense == InActive))
    If ((NOT inActive) AND UseConstant)
        sample = ConstantColor
    else
        sample = Fetch(adr)
    EndIf
    line = line + 1
```

```
    If ((LineSense(line) == "+") AND wrapImage)
        adr = adr - DeltaColumn
        line = ActiveStartLine
    ElseIf ((LineSense(line) == "0") AND ((oldLineSense == "0"))
        adr = adr + DeltaLine
    EndIf
EndDo
```

诸如 FirstSampleLine、FirstSamplePixel 和 FirstSampleAdr 之类的变量的设置在地址生成器部分，地址生成器响应内核地址生成器的进位标志和 MJI 的 AdvanceLine 脉冲。以下伪码表示此部分地址生成的逻辑:

```
FirstSamplePixel =0
FirstSampleLine = 0

FirstSampleAdr =FirstSampleAdr = ActiveStartAddress
count = 0
Do Forever
    If ((KernelXCarryOut) OR (AdvanceLine AND KernelYCarryOut) OR (count
< 5))
        Do N Samples for this color plane （见以上伪码）
    EndIf
    oldPixelSense = PixelSense(FirstSamplePixel)
    oldLineSense = LineSense(FirstSampleLine)
    If ( AdvanceLine AND KernelYCarryout)
        count = 0
        FirstSampleLine = FirstSampleLine + 1
        FirstSamplePixel = 0
```

```
            If ((LineSense(FirstSampleLine) == "+") AND wrapImage)
                FirstLineSampleAdr = StartAddress
                FirstSampleLine = ActiveStartLine
            ElseIf ((LineSense(FirstSampleLine) == "0") AND (oldLineSense == "0"))
                FirstSampleLineAdr = FirstSampleLineAdr + DeltaLine
            EndIf
            FirstSampleAdr = FirstLineSampleAdr
        ElseIf ( KernelXCarryout OR (count < 5))
            FirstSamplePixel =FirstSamplePixel + 1
            count = count + 1
            If ((PixelSense(FirstSamplePixel) == "+") AND wrapImage)
                FirstSampleAdr = FirstLineSampleAdr
                FirstSamplePixel = ActiveStartPixel
            ElseIf ((PixelSense(FirstSamplePixel) == "0") AND (oldPixelSense ==
"0"))
                FirstSampleAdr = FirstSampleAdr + DeltaPixel
            EndIf
        EndIf
    EndDo
```

10.2.11.3 寄存器概述

在打印图象前，必须设置许多寄存器。表 39 概括总结这些寄存器。为了将图象旋转 90 度，只需交换 DeltaLine 和 DeltaPixel 值，并提供新的 DeltaColumn 值。

表 39. 在打印前需要由调用程序设置的寄存器

寄存器名称	描述
图象访问参数	
WrapImage	当越过图象边界时，平铺图象读数以复制图象

UseConstant	如果为 0，则当读取操作越过图象边界时，图象边缘复制或回绕出现。 如果为 1，则返回固定颜色。
红	
ActiveStartAddressR	ImageRAM 中红色采样（ActiveStartPixel，ActiveStartLine）的地址。
ActiveStartLineR	红色空间中该图象的第一有效行（相对于行 0）
ActiveEndLineR	红色空间中该图象的第一越界行
ActiveStartPixelR	红色空间中该图象的第一有效象素（相对于象素 0）
ActiveEndPixelR	红色空间中该图象的第一越界象素
DeltaLineR	在红色空间中，从某一行移动到下一行需要向当前地址添加的数量
DeltaPixelR	在红色空间中，从某一象素移动到同一行中的下一象素需要向当前地址添加的数量
DeltaColumnR	在红色空间中，从有效图象区域的最后一行中的某个象素移动到有效图象区域的第一行中同一象素需要向当前地址添加的数量
ConstantColorR	当地址越界且 UseConstant=1 时使用的红色值
绿	
ActiveStartAddressG	ImageRAM 中绿色采样（ActiveStartPixel，ActiveStartLine）的地址。
ActiveStartLineG	绿色空间中该图象的第一有效行（相对于行 0）
ActiveEndLineG	绿色空间中该图象的第一越界行
ActiveStartPixelG	绿色空间中该图象的第一有效象素（相对于象素 0）
ActiveEndPixelG	绿色空间中该图象的第一越界象素
DeltaLineG	在绿色空间中，从某一行移动到下一行需要向当前地址添加的数量
DeltaPixelG	在绿色空间中，从某一象素移动到同一行中的下一象素需要向当前地址添加的数量

DeltaColumnG	在绿色空间中，从有效图象区域的最后一行中的某个象素移动到有效图象区域的第一行中同一象素需要向当前地址添加的数量
ConstantColorG	当地址越界且 UseConstant=1 时使用的绿色值
蓝	
ActiveStartAddressB	ImageRAM 中蓝色采样（ActiveStartPixel，ActiveStartLine）的地址。
ActiveStartLineB	蓝色空间中该图象的第一有效行（相对于行 0）
ActiveEndLineB	蓝色空间中该图象的第一越界行
ActiveStartPixelB	蓝色空间中该图象的第一有效象素（相对于象素 0）
ActiveEndPixelB	蓝色空间中该图象的第一越界象素
DeltaLineB	在蓝色空间中，从某一行移动到下一行需要向当前地址添加的数量
DeltaPixelB	在蓝色空间中，从某一象素移动到同一行中的下一象素需要向当前地址添加的数量
DeltaColumnB	在蓝色空间中，从有效图象区域的最后一行中的某个象素移动到有效图象区域的第一行中同一象素需要向当前地址添加的数量
ConstantColorB	当地址越界且 UseConstant=1 时使用的蓝色值
白平衡和范围扩展参数	
RedLowThreshold	从红色输入值中减去的 8 位值
GreenLowThreshold	从绿色输入值中减去的 8 位值
BlueLowThreshold	从蓝色输入值中减去的 8 位值
RedScaleFactor	供红色象素范围扩展使用的 8 位比例系数
GreenScaleFactor	供绿色象素范围扩展使用的 8 位比例系数
BlueScaleFactor	供蓝色象素范围扩展使用的 8 位比例系数

11. 参考文献

[1] Silverbrook Research, 1998, *Authentication of Consumables.*

[2] Silverbrook Research, 1998, *Authentication chip.*

尽管参照特定示例说明本发明，但是熟练技术人员可以理解，能够以各种其他形式体现本发明。以下带有编号的段落向收件人提供本发明之范围的进一步说明，不过根据本文中的公开，其他创新和发明特征以及特征之组合也是显然的。

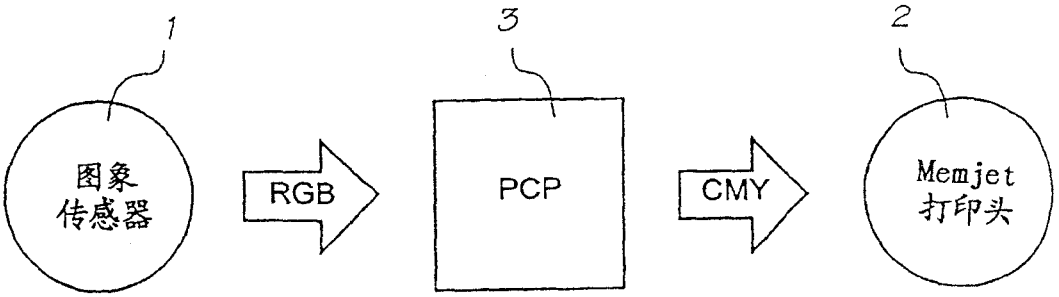


图1

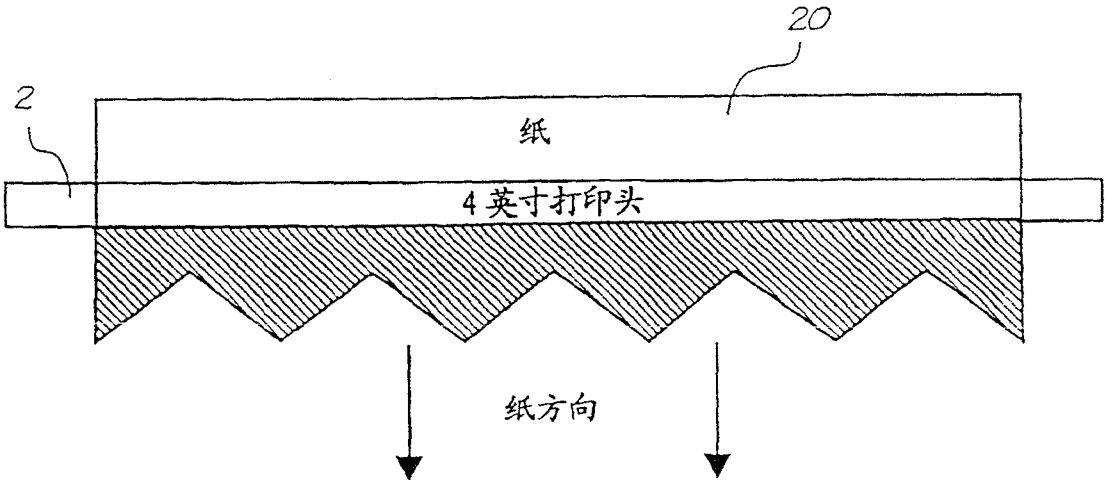


图4

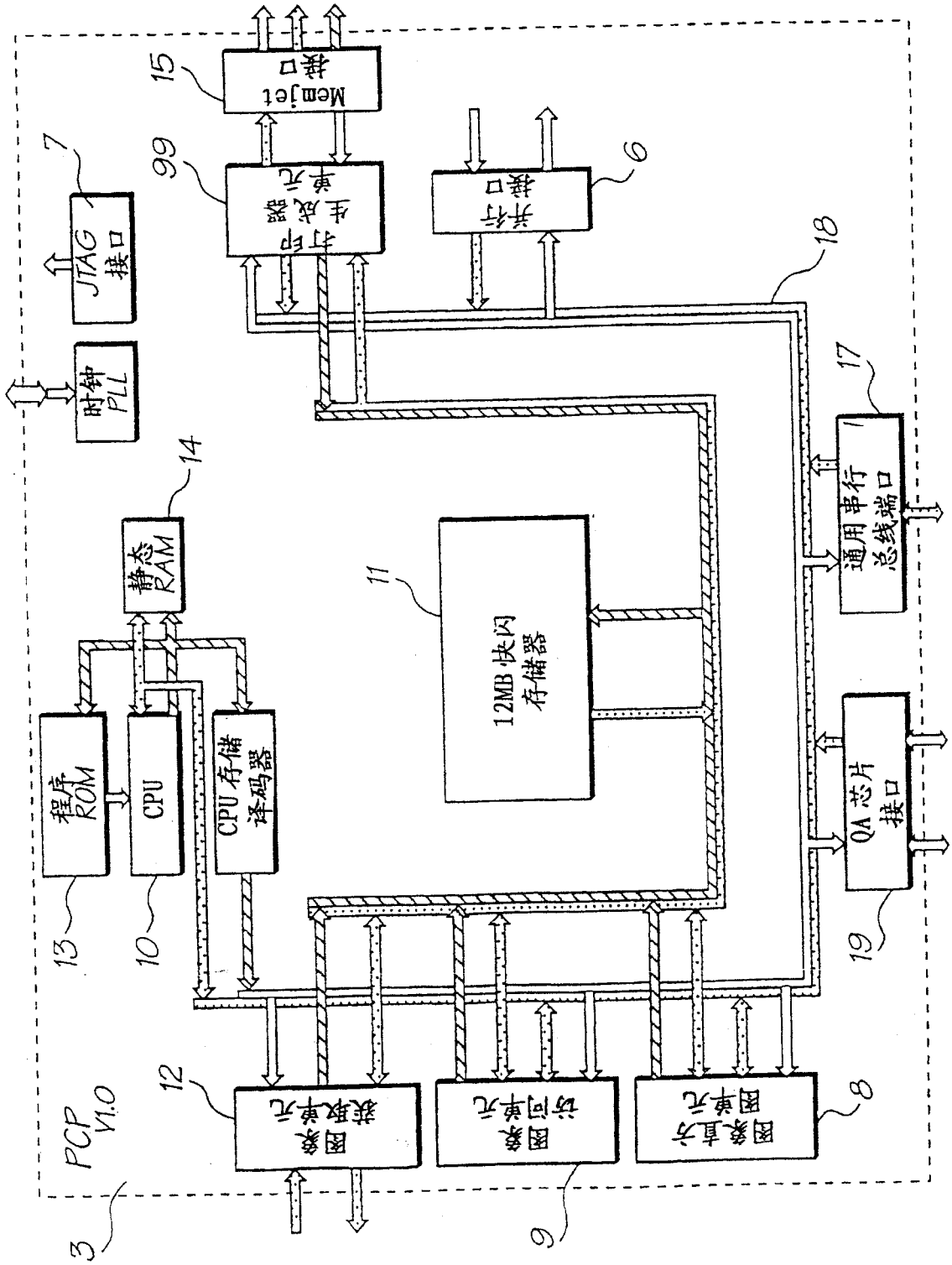

PCP V1.0
3
JTAG 接口
7
时钟 PLL
1
MemJet 接口
15
打印 生成器 单元
99
并行 接口
6
18
17
通用串行 总线端口
1
静态 RAM
14
12MB 快闪 存储器
11
程序 ROM
13
CPU
10
CPU 存储 译码器
QA 芯片 接口
19
图象 获取单元
12
图象 访问单元
9
图象直方 图单元
8

图2

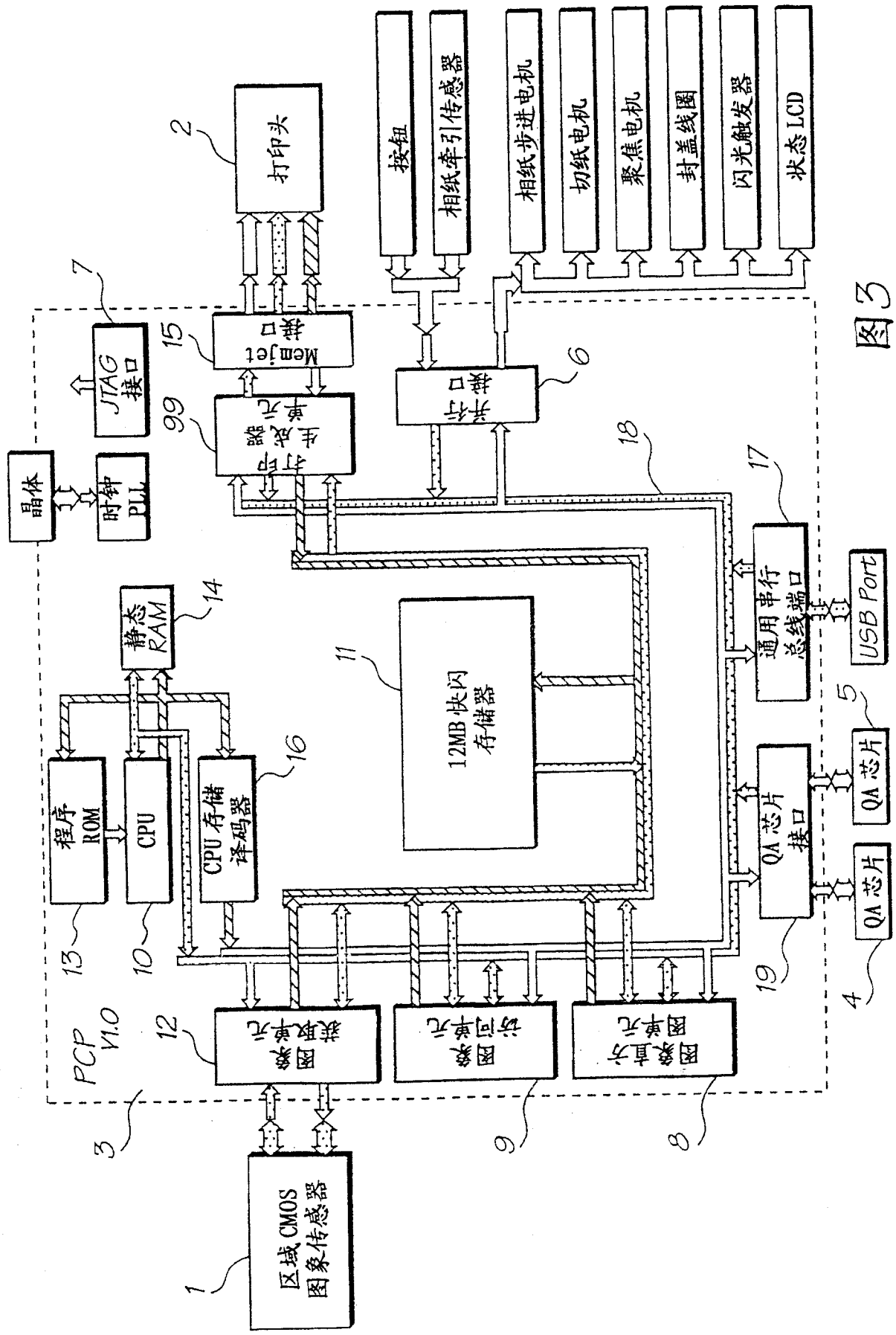

区域 CMOS 图象传感器
1
PCP V1.0
3
图象获取单元
12
图象访问单元
9
图象直方图单元
8
程序 ROM
13
CPU
10
CPU 存储译码器
16
静态 RAM
14
12MB 快闪存储器
11
通用串行总线端口
17
USB Port
5
QA 芯片
4
QA 芯片接口
19
并行接口
6
18
打印头
2
按钮
相纸牵引传感器
相纸步进电机
切纸电机
聚焦电机
封盖线圈
闪光触发器
状态 LCD
JTAG 接口
7
MemJet 接口
15
打印头单元控制器
99
晶体
时钟 PLL

图 3

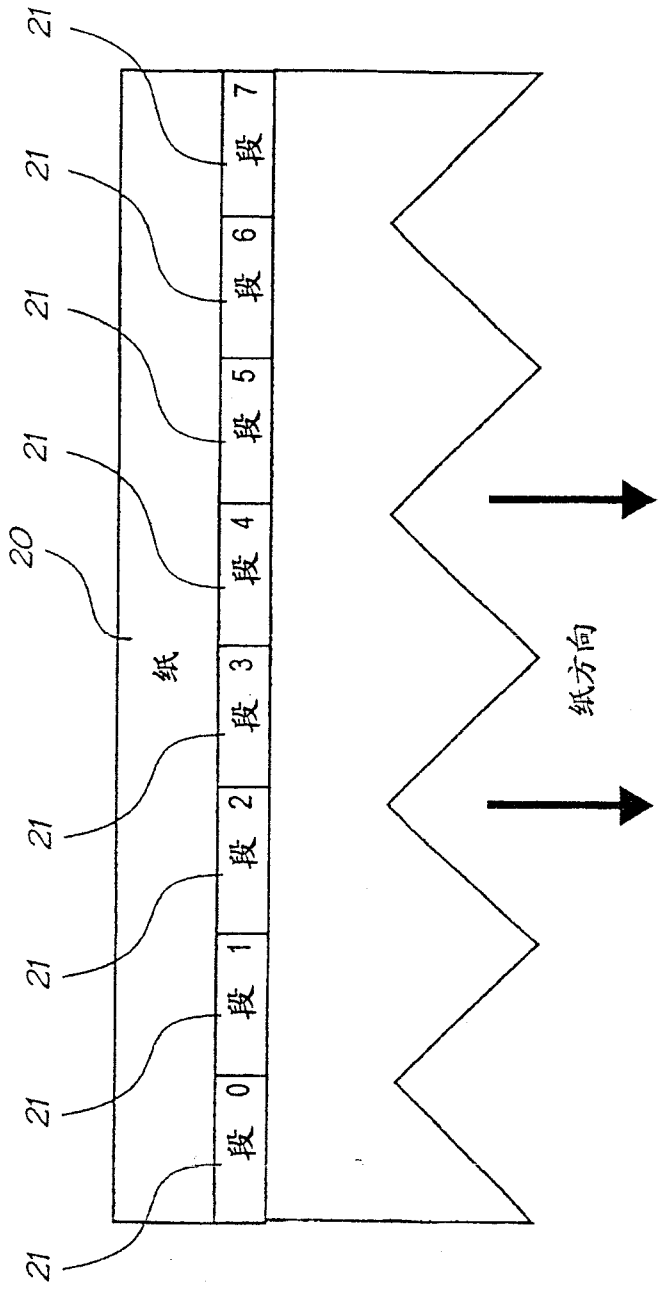

21
21
21
21
21
20
21
21
21
段 7
段 6
段 5
段 4
段 3
段 2
段 1
段 0
纸
纸方向
1 段
= 0.5 英寸
= 12700 μm

图5

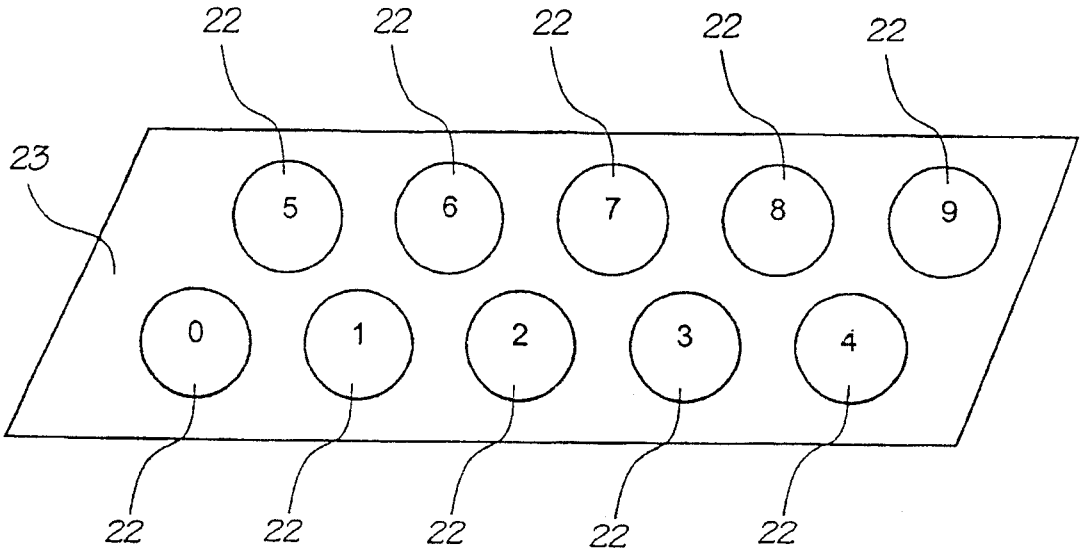

22
22
22
22
22
23
5
6
7
8
9
0
1
2
3
4
22
22
22
22
22

图6

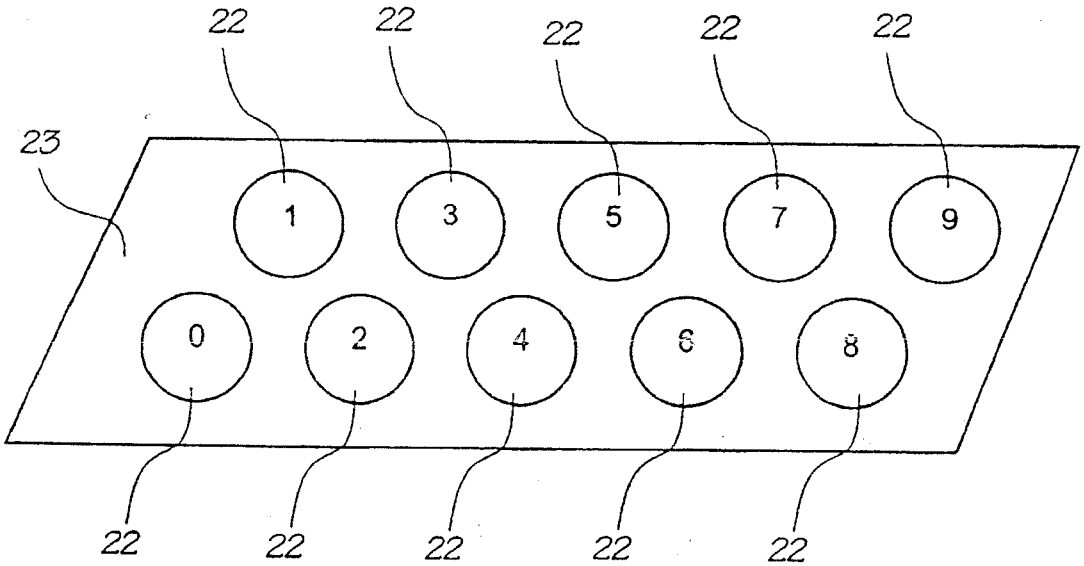

22
22
22
22
22
23
1
3
5
7
9
0
2
4
6
8
22
22
22
22
22

图7

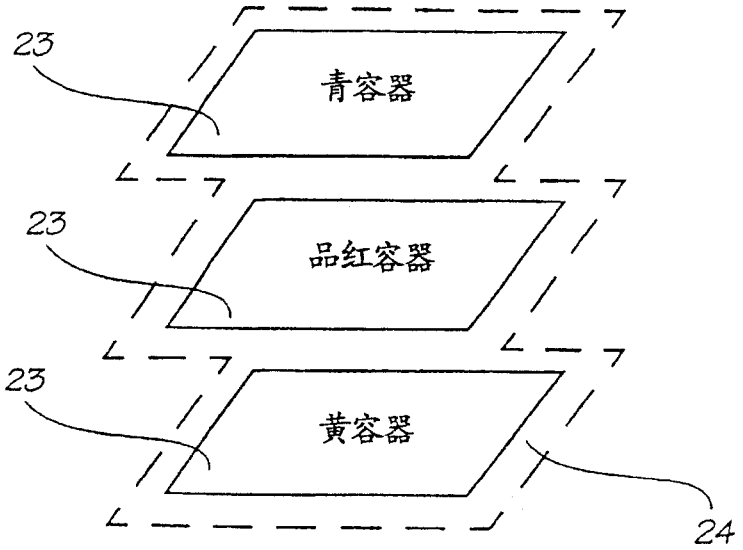
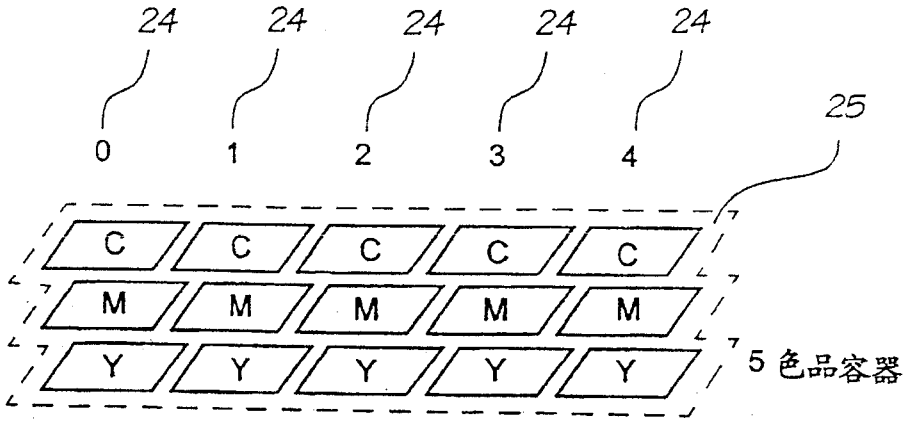


图8



C= 青
M= 品红
Y= 黄

图9

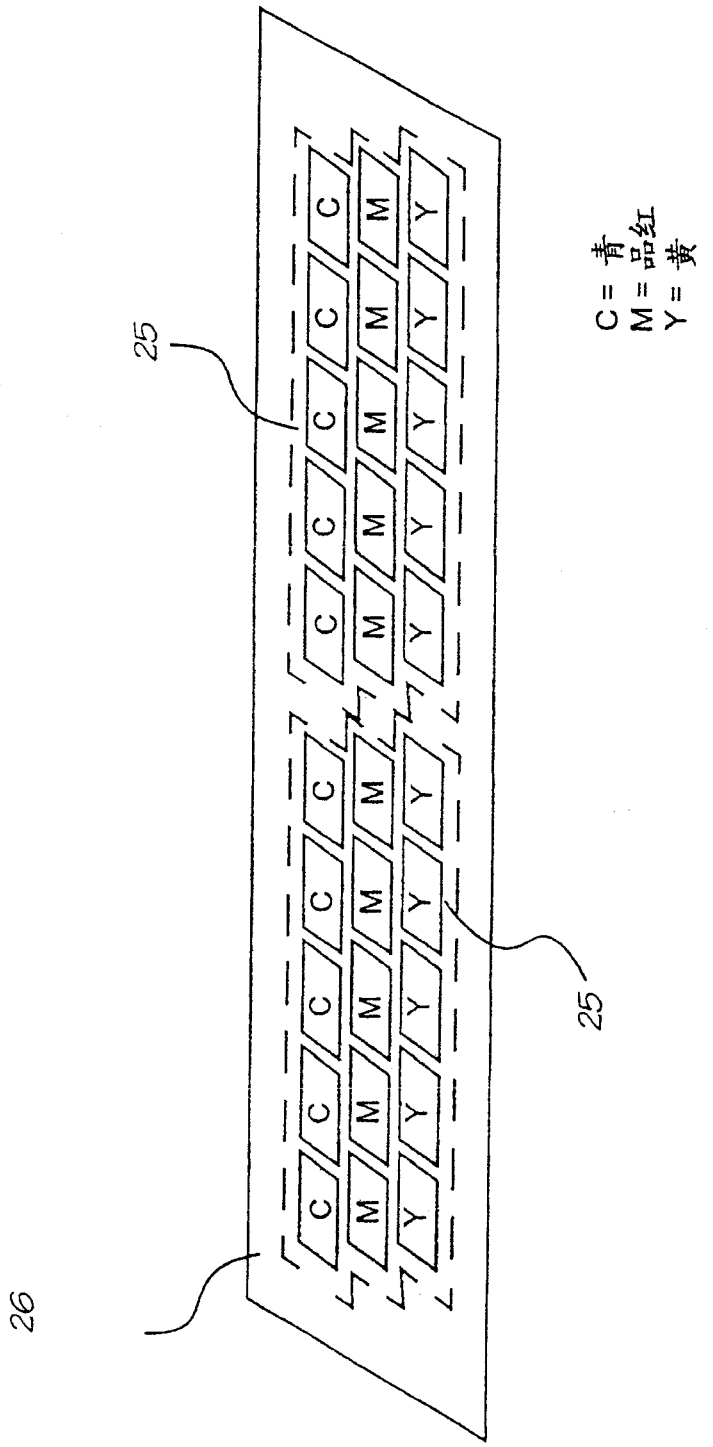


图10

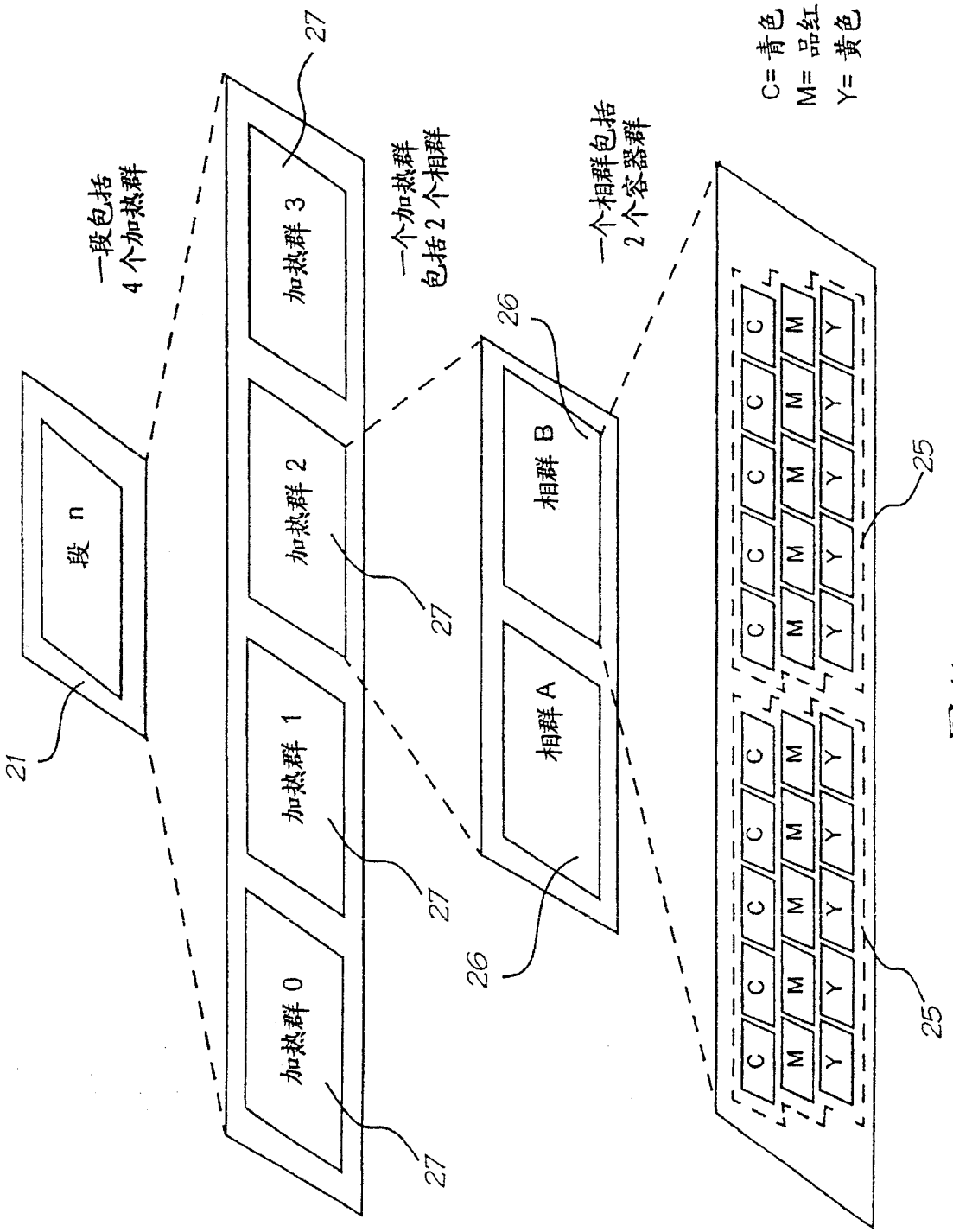

一段包括
4 个加热群
一个加热群
包括 2 个相群
一个相群包括
2 个容器群
C= 青色
M= 品红
Y= 黄色
段 n
加热群 3
加热群 2
加热群 1
加热群 0
相群 B
相群 A
21
27
26
25

图 11

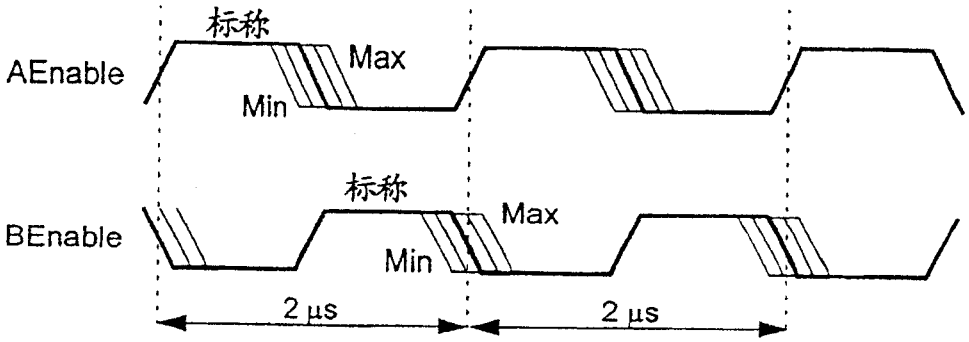

AEnable
标称
Max
Min
BEnable
标称
Max
Min
2 μs
2 μs

图12

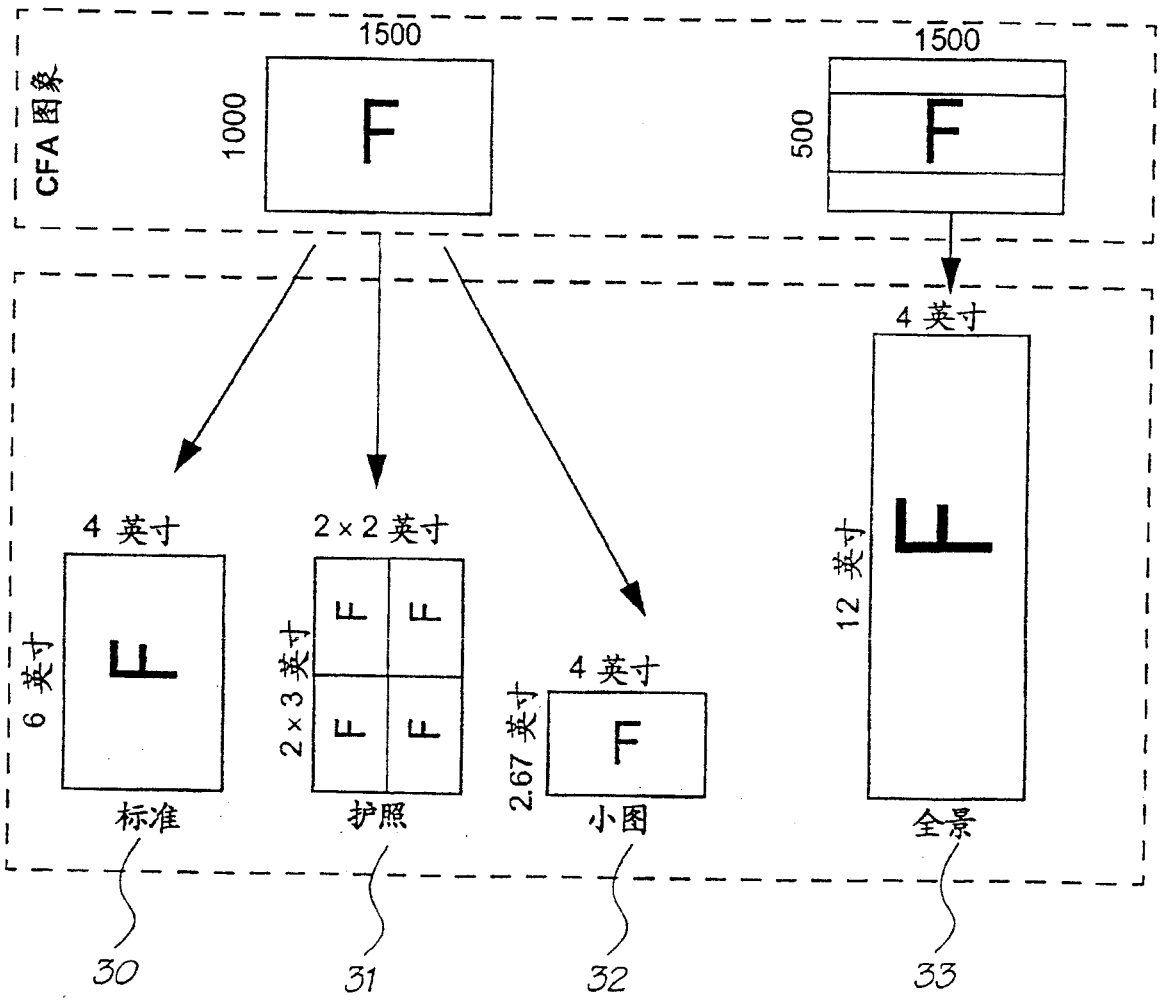

CFA 图象
1500
1000
F
1500
500
F
4 英寸
12 英寸
LL
4 英寸
6 英寸
LL
标准
2×2 英寸
2×3 英寸
F
F
F
F
护照
4 英寸
2.67 英寸
F
小图
全景
30
31
32
33

图13

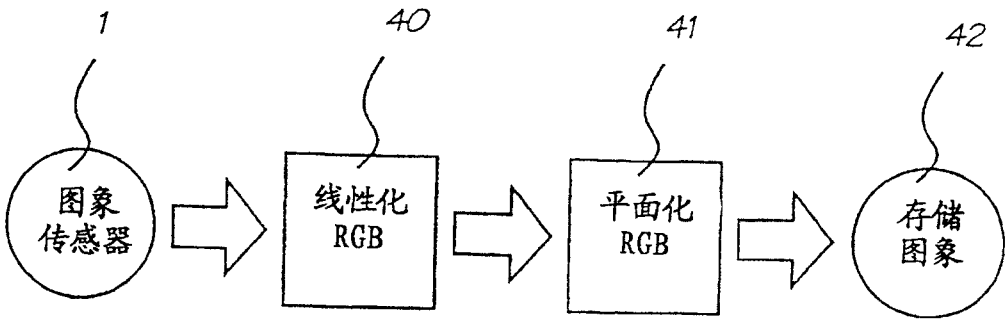

1
40
41
42
图象
传感器
线性化
RGB
平面化
RGB
存储
图象

图14

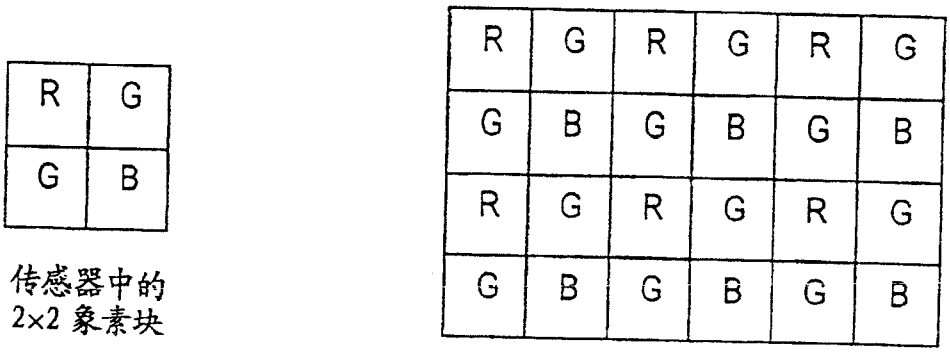

R G
G B
传感器中的
2×2 象素块
R G R G R G
G B G B G B
R G R G R G
G B G B G B
几个 2×2 象素块
的拜尔镶嵌面

图15

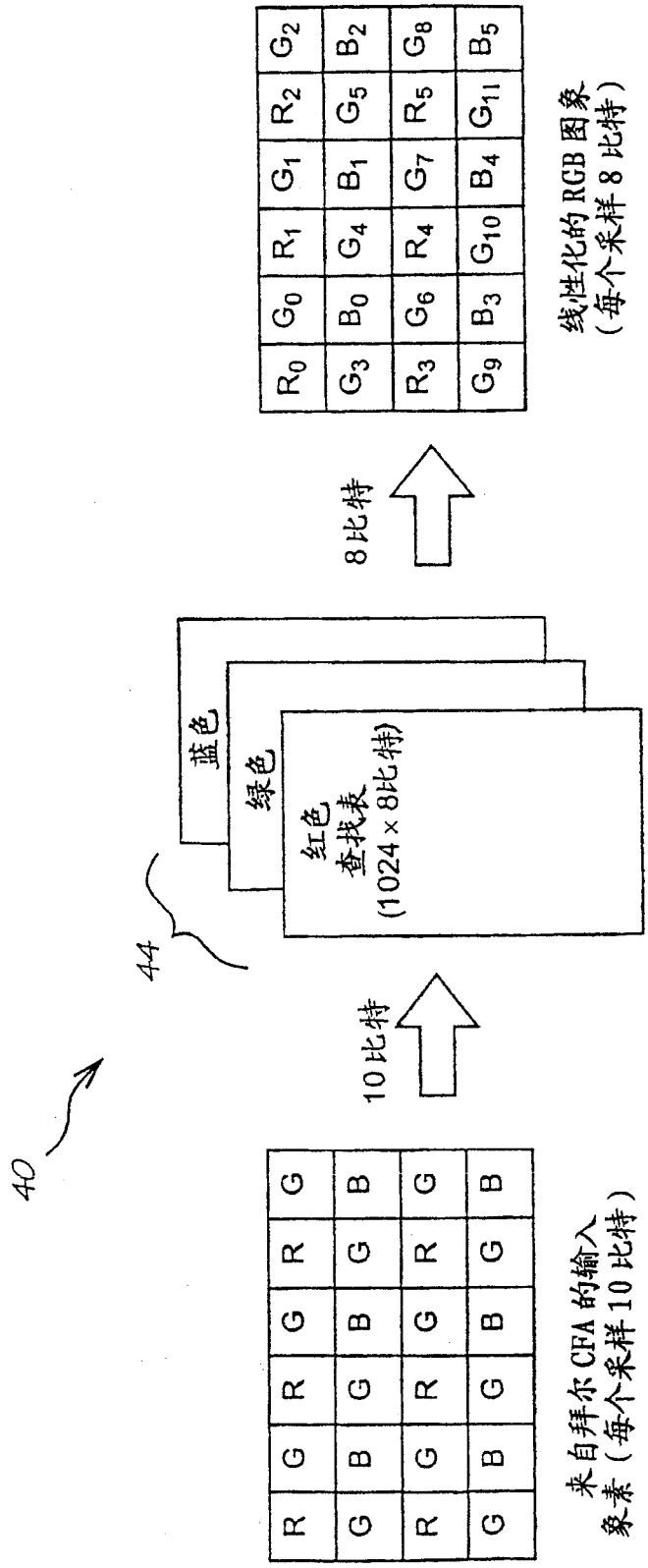

40
来自拜尔 CFA 的输入
象素（每个采样 10 比特）
10 比特
44
红色
查找表
(1024 × 8比特)
绿色
蓝色
8 比特
线性化的 RGB 图象
（每个采样 8 比特）

图16

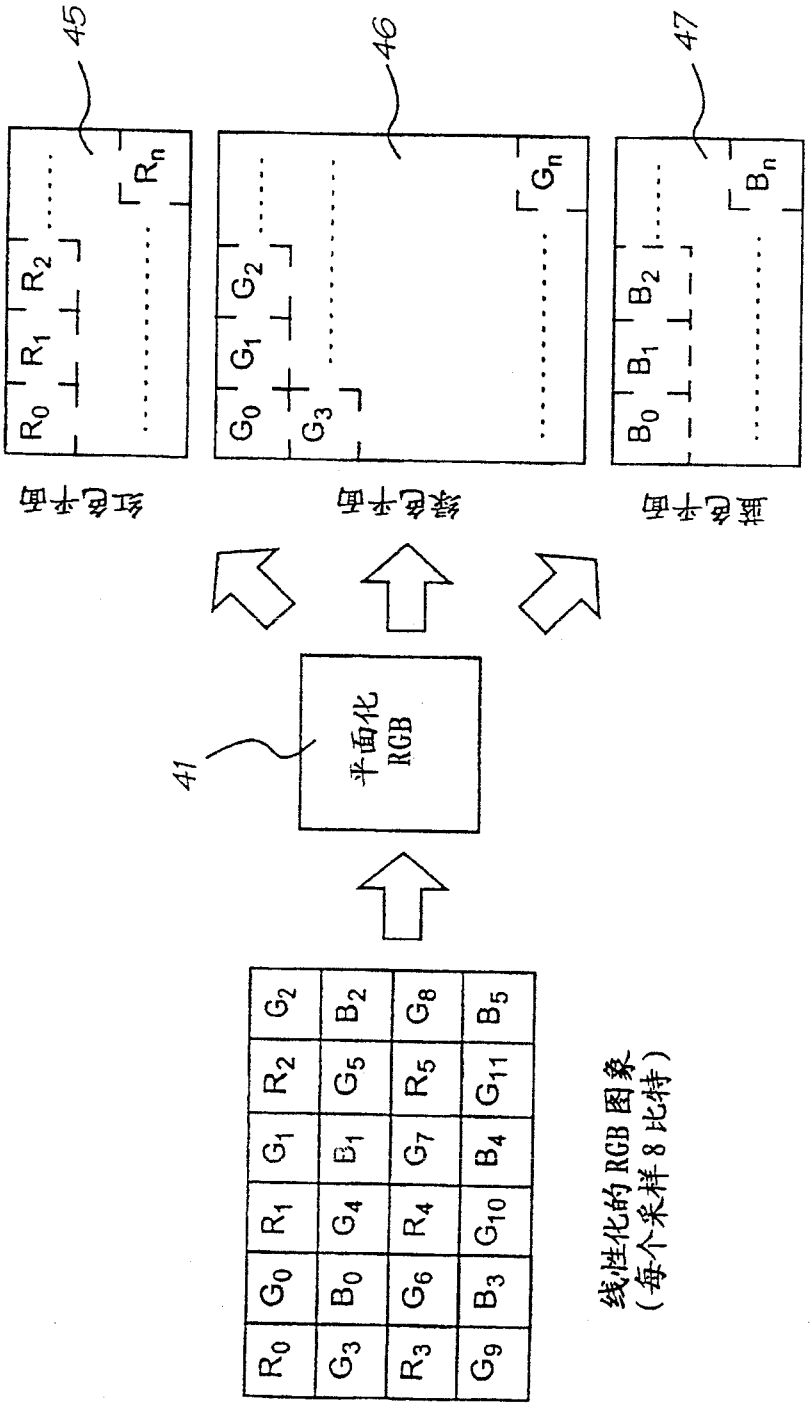

45
46
47
R_0 R_1 R_2 ..... R_n
G_0 G_1 G_2 G_3 ..... G_n
B_0 B_1 B_2 ..... B_n
红色平面
绿色平面
蓝色平面
41
平面化 RGB
R_0 G_0 R_1 G_1 R_2 G_2
G_3 B_0 G_4 B_1 G_5 B_2
R_3 G_6 R_4 G_7 R_5 G_8
G_9 B_3 G_{10} B_4 G_{11} B_5
线性化的 RGB 图象
(每个采样 8 比特)

图 17

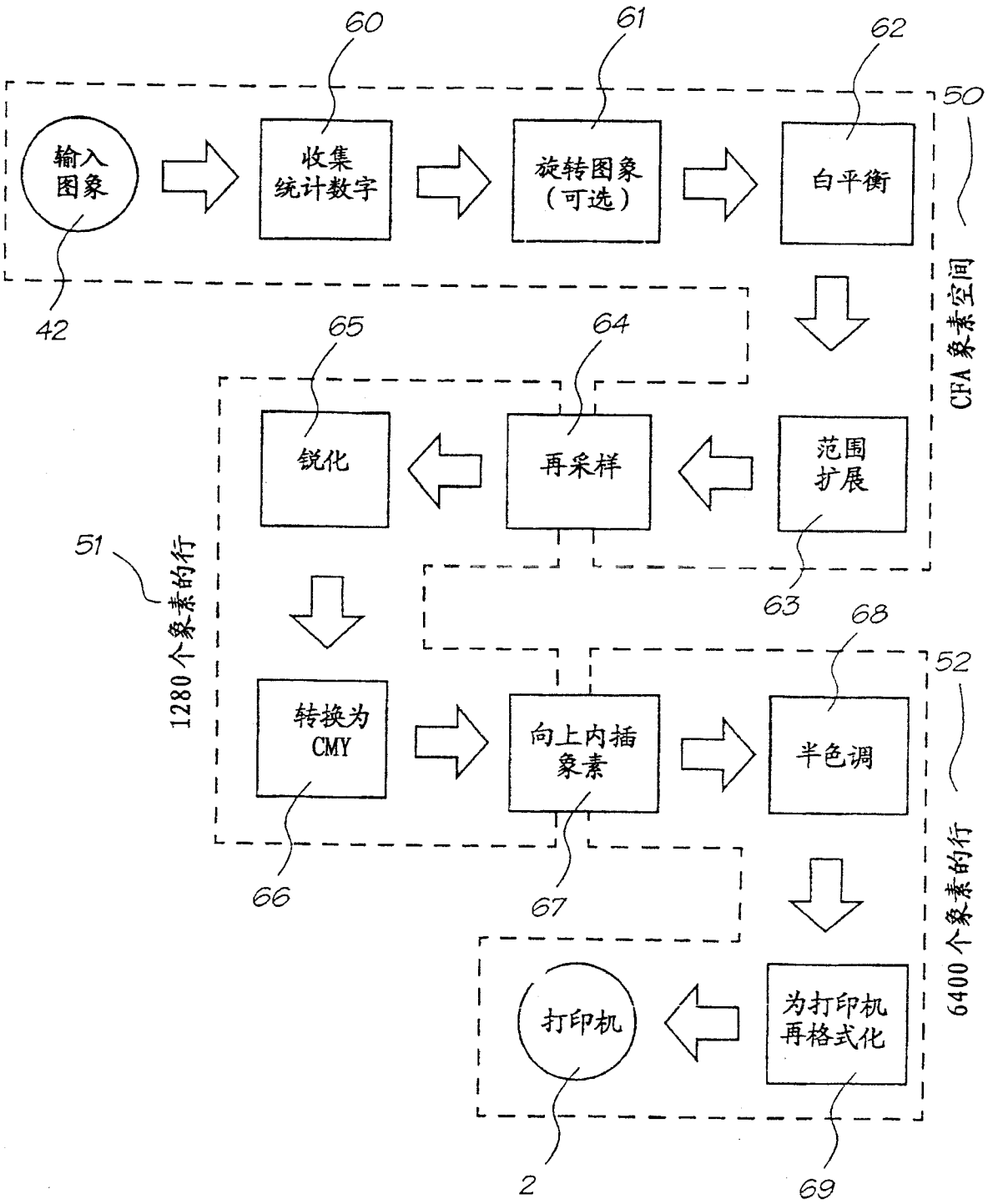

60
61
62
50
输入图象
收集统计数字
旋转图象（可选）
白平衡
42
CFA象素空间
65
64
锐化
再采样
范围扩展
51
1280个象素的行
63
68
52
转换为CMY
向上内插象素
半色调
66
67
6400个象素的行
打印机
为打印机再格式化
2
69

图18

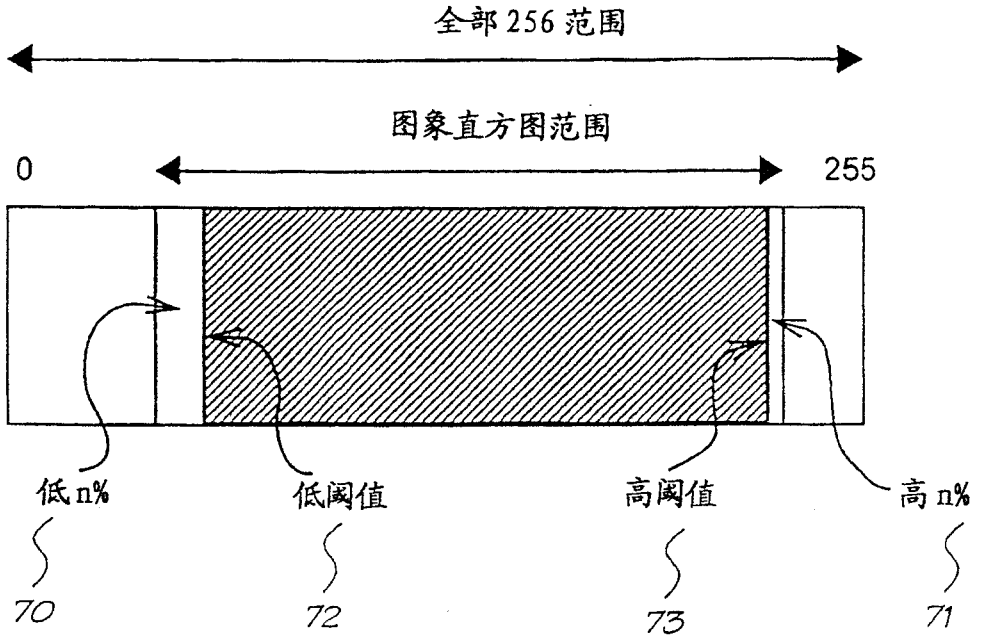

全部256范围
图象直方图范围
0
255
低n%
低阈值
高阈值
高n%
70
72
73
71

图19

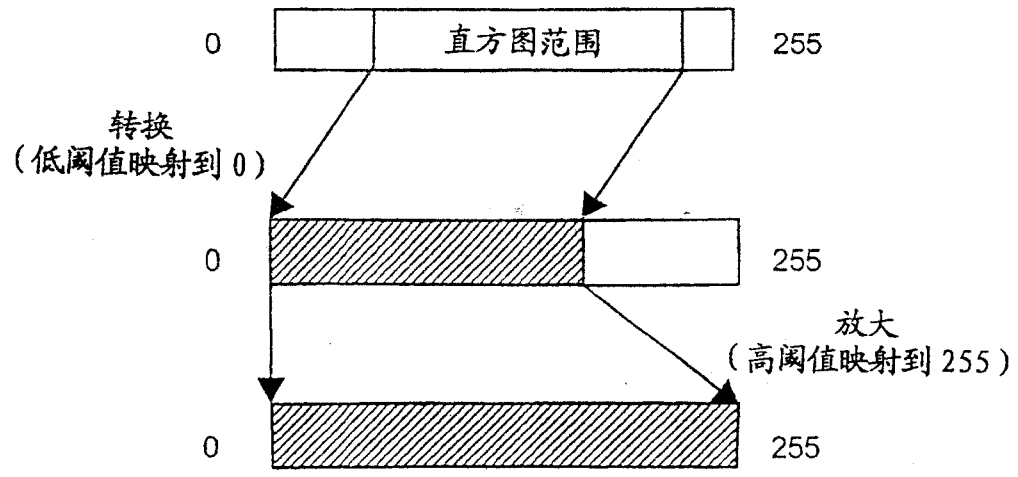

0
直方图范围
255
转换
(低阈值映射到0)
0
255
放大
(高阈值映射到255)
0
255

图20

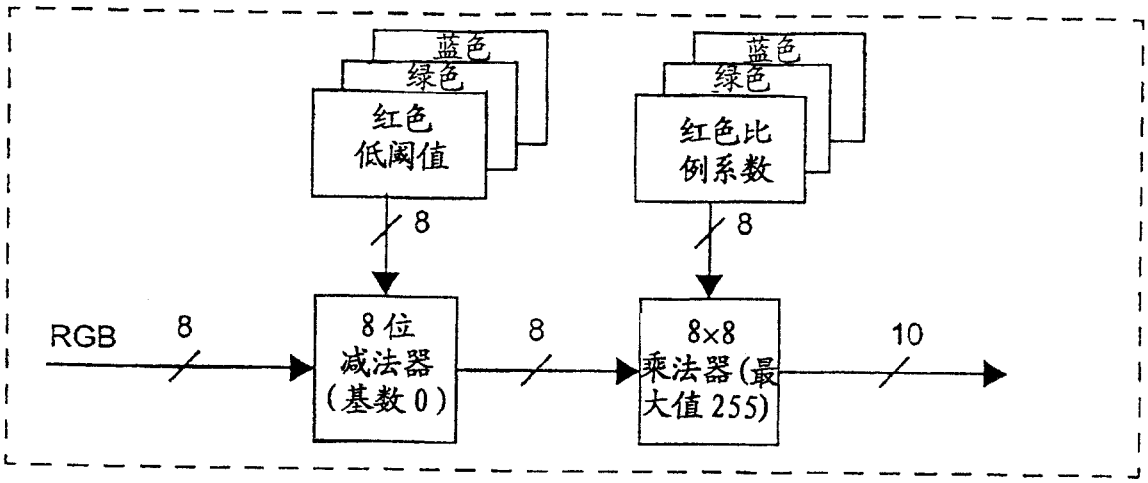

蓝色
绿色
红色
低阈值
8
蓝色
绿色
红色比
例系数
8
RGB
8
8位
减法器
(基数0)
8
8×8
乘法器(最
大值255)
10

图21

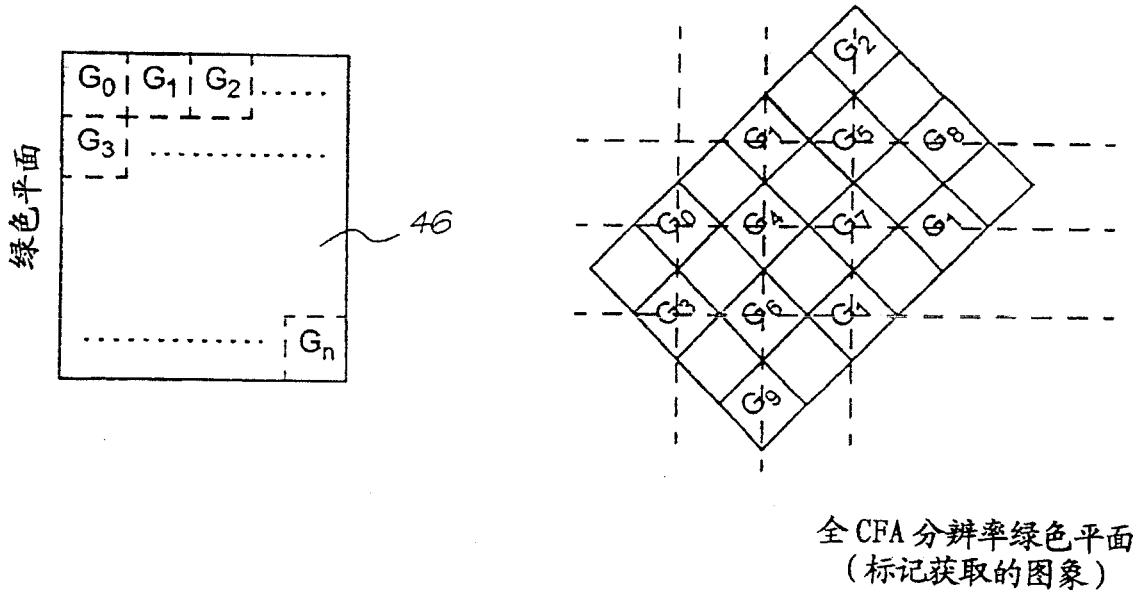
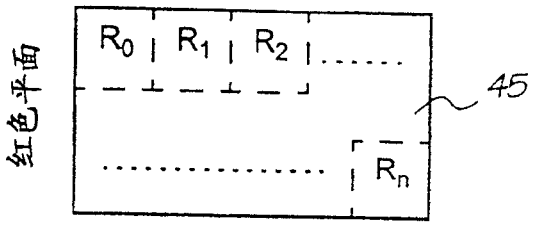
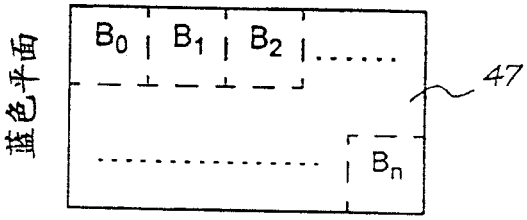

绿色平面
G0 G1 G2 .....
G3 .....
46
..... Gn
全CFA分辨率绿色平面
(标记获取的图象)

图23



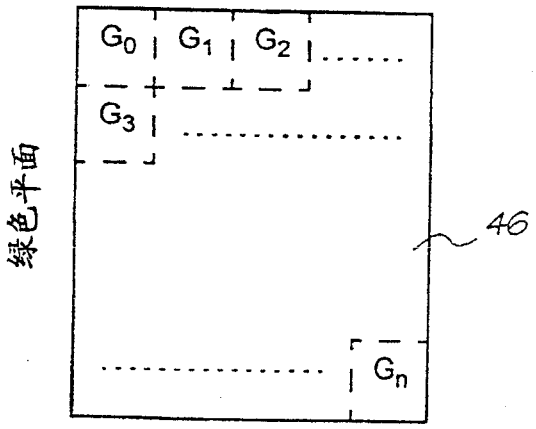
R_0		R_1		R_2	
R_3		R_4		R_5	

全 CFA 分辨率红色平面
（标记获取的象素）



	B_0		B_1		B_2
	B_3		B_4		B_5

全 CFA 分辨率蓝色平面
（标记获取的象素）



	G_0		G_1		G_2
G_3		G_4		G_5	
	G_6		G_7		G_8
G_9		G_{10}		G_{11}	

全 CFA 分辨率绿色平面
（标记获取的象素）

图22

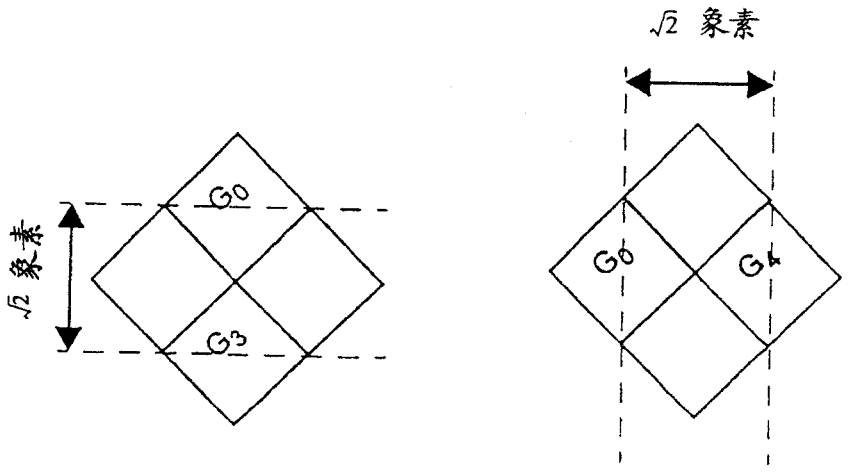


图24

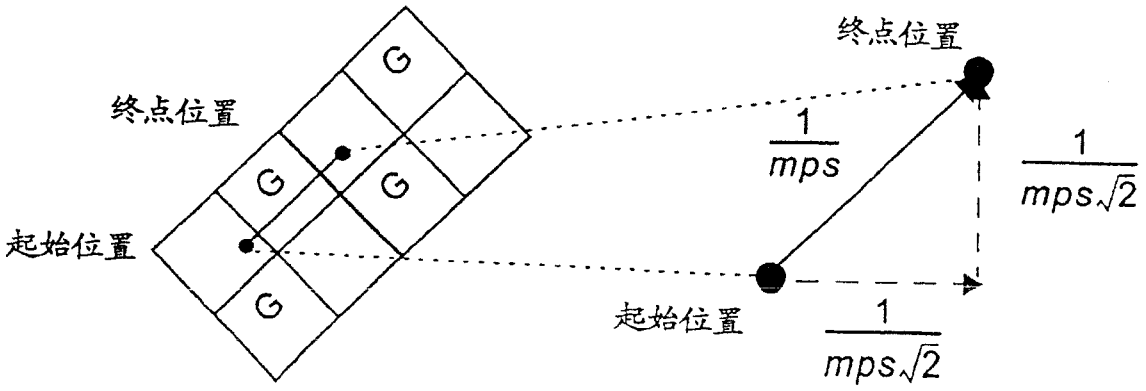


图25

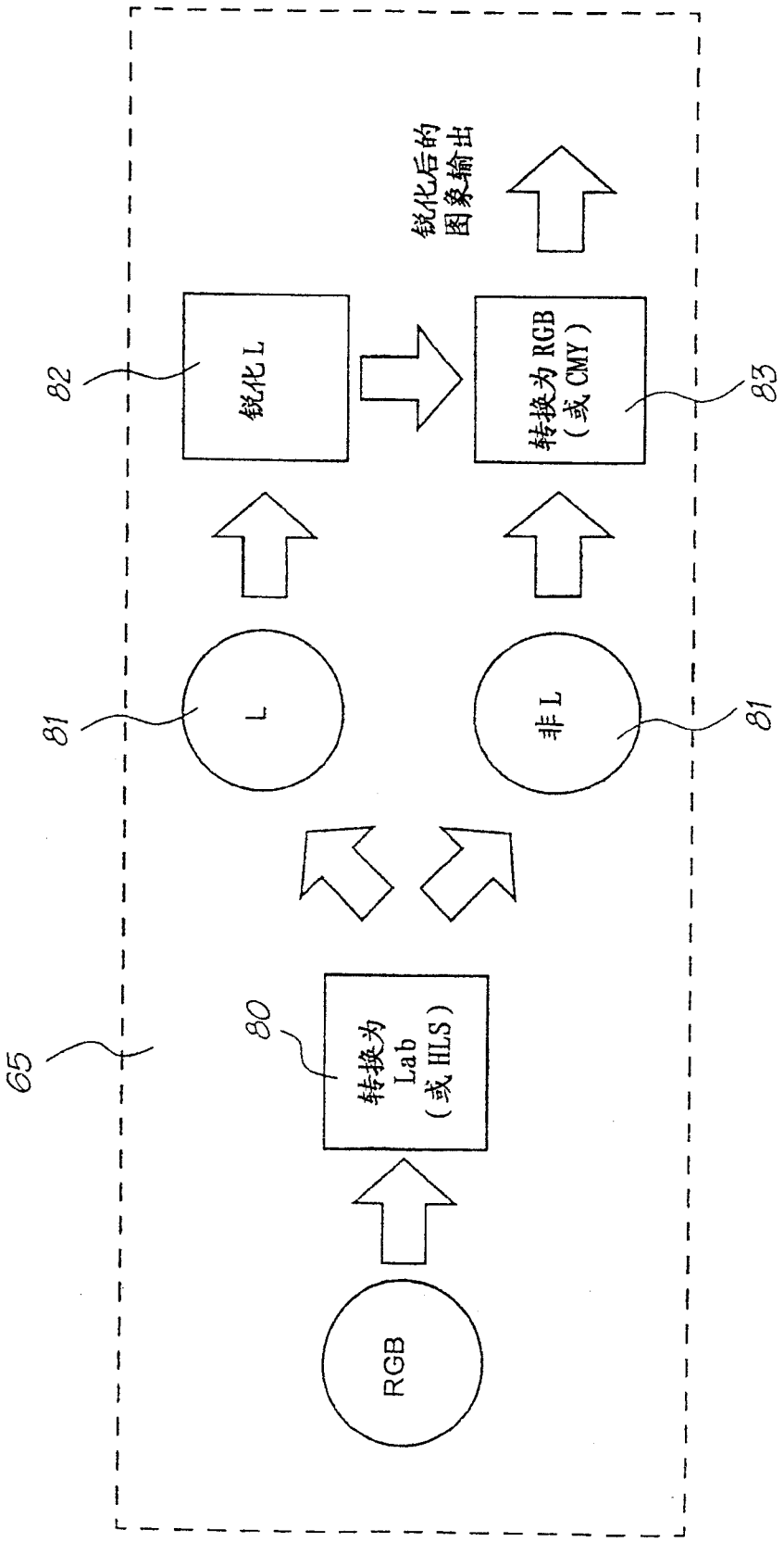


图26

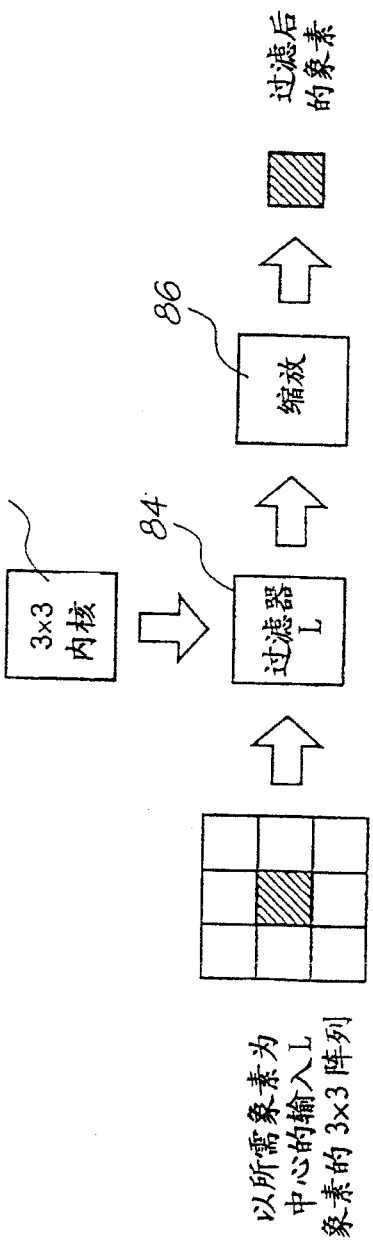

以所需象素为
中心的输入L
象素的3x3阵列
过滤器
L
84
3x3
内核
85
缩放
86
过滤后
的象素

图27

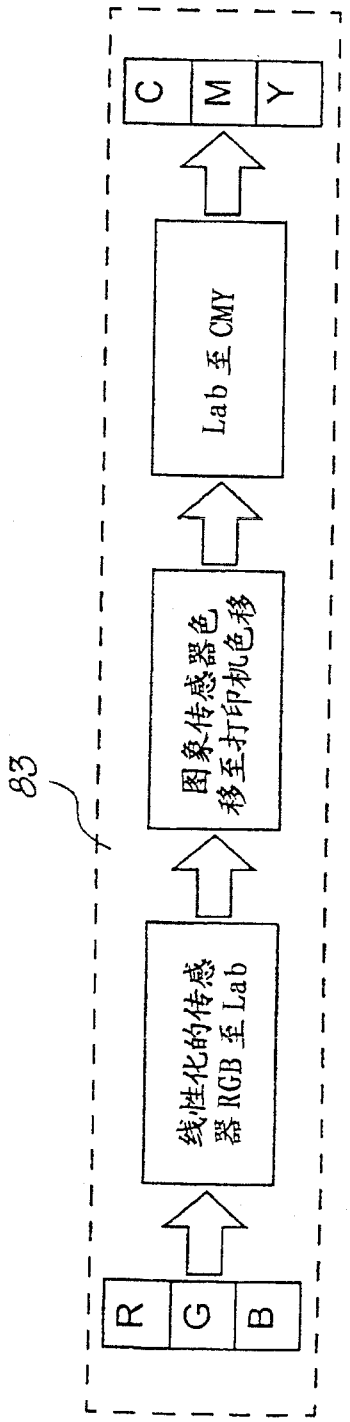

R
G
B
线性化的传感
器RGB至Lab
图象传感器色
移至打印机色移
Lab至CMY
C
M
Y
83

图28

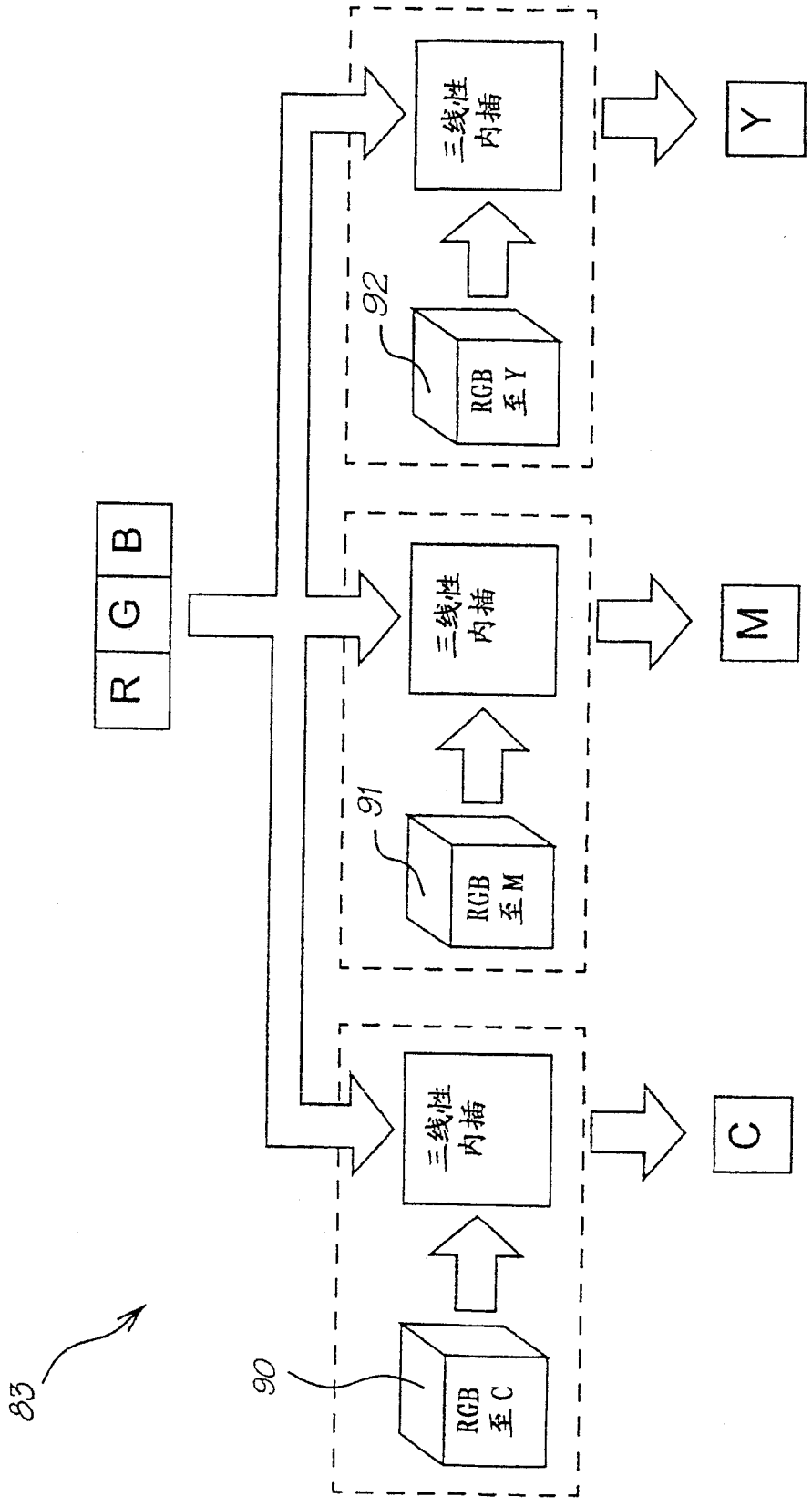


图29

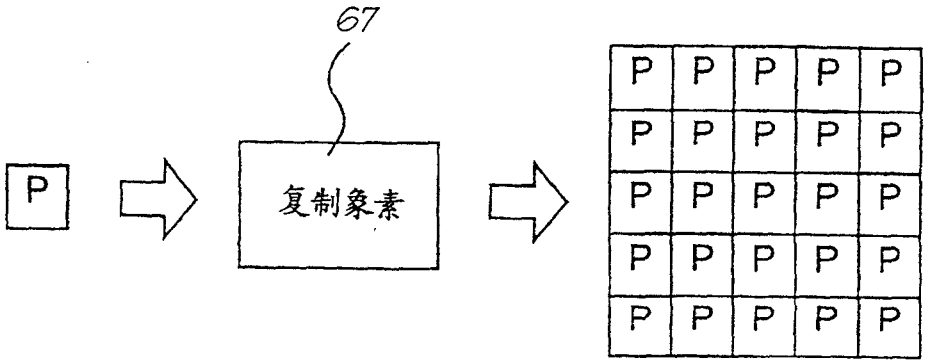

67
P
复制象素
P P P P P
P P P P P
P P P P P
P P P P P
P P P P P

图30

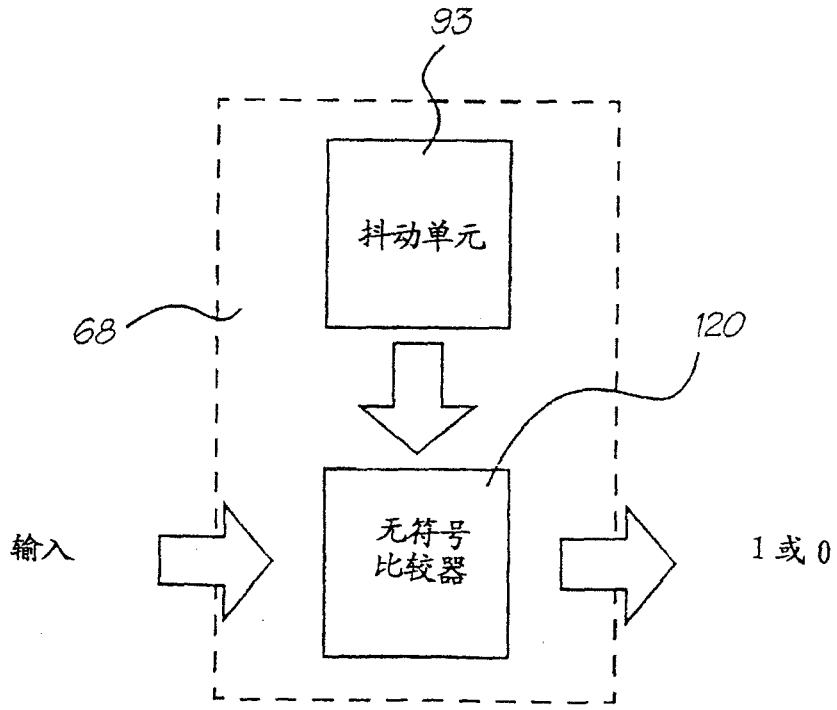

93
抖动单元
68
120
输入
无符号
比较器
1或0

图31

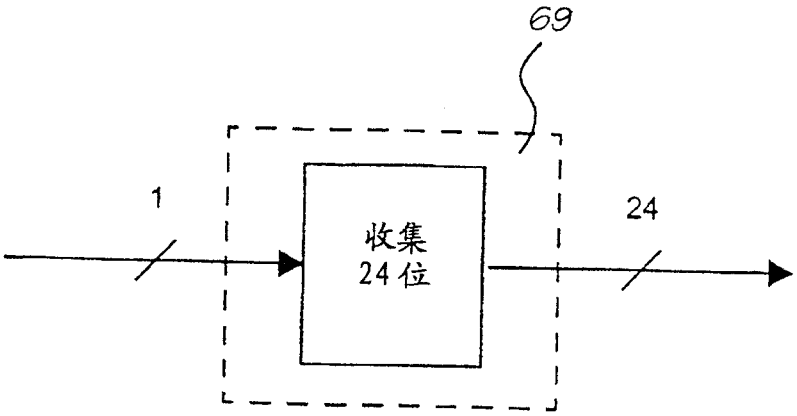

69
1
收集
24位
24

图32

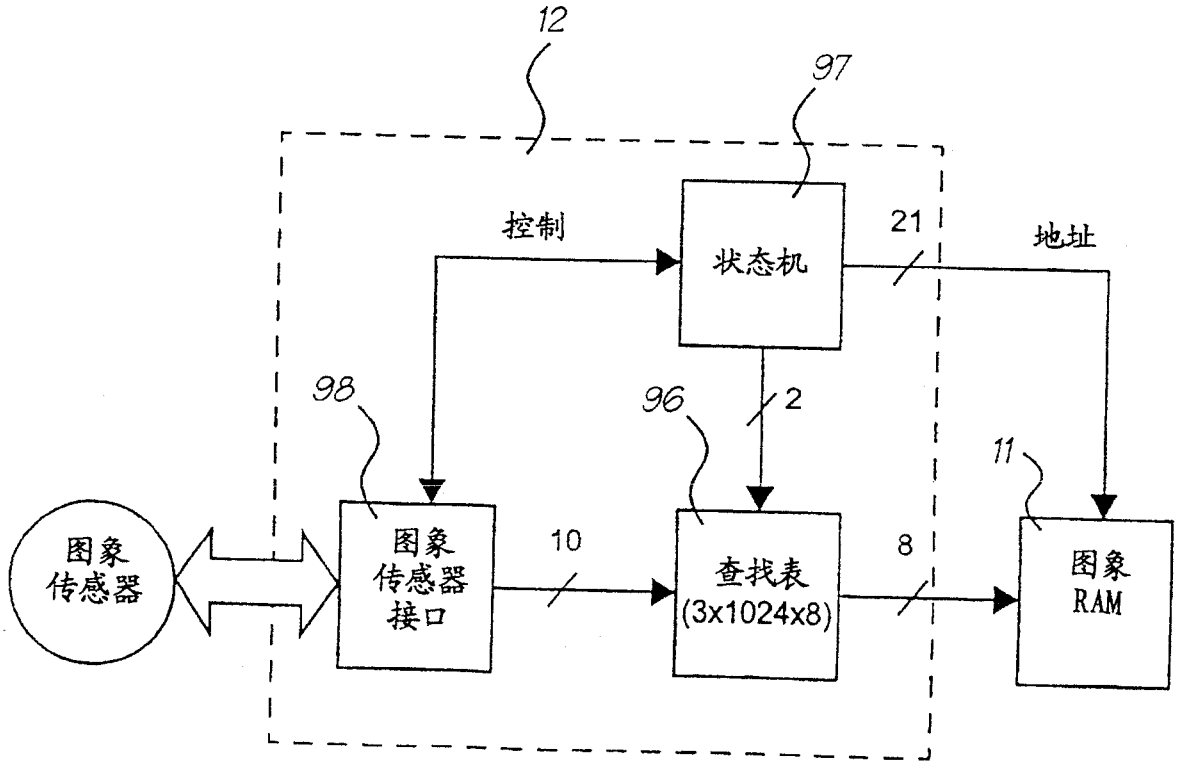

12
97
控制
状态机
21
地址
98
96
2
11
图象
传感器
图象
传感器
接口
10
查找表
(3x1024x8)
8
图象
RAM

图33

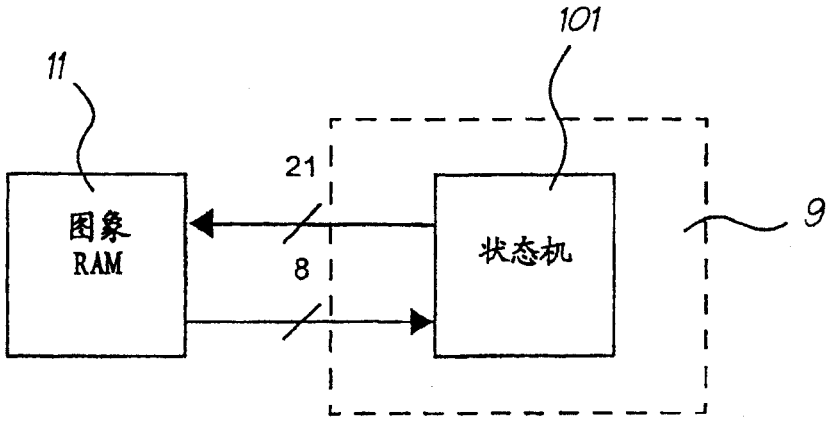


图34

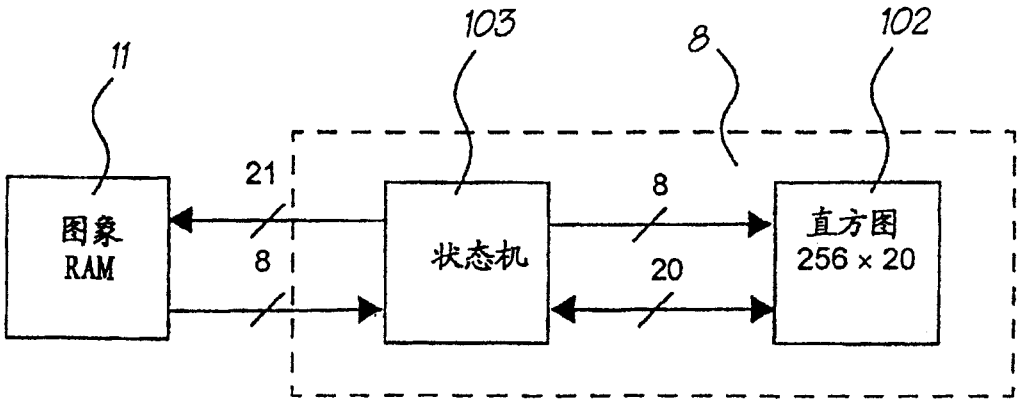


图35

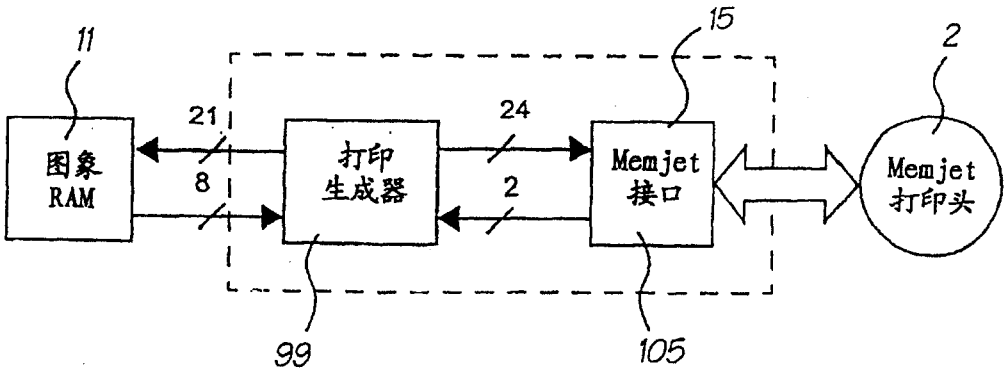

11
15
2
21
24
图象
RAM
打印
生成器
Memjet
接口
Memjet
打印头
8
2
99
105

图36

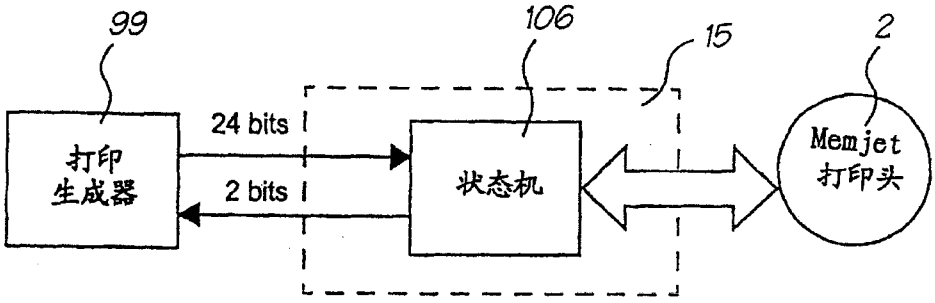

99
106
15
2
24 bits
打印
生成器
状态机
Memjet
打印头
2 bits

图37

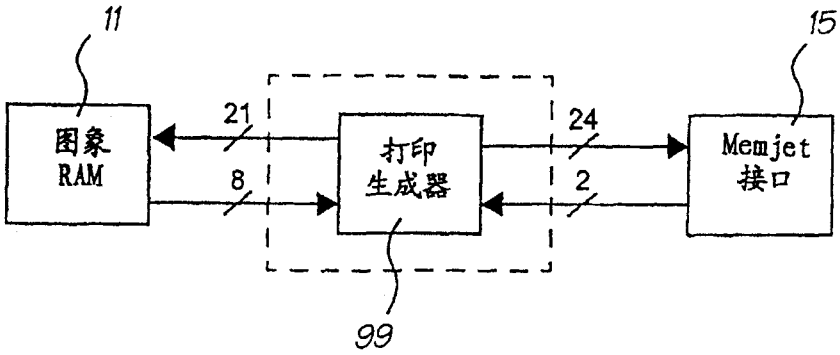

11
15
21
24
图象
RAM
打印
生成器
Memjet
接口
8
2
99

图40

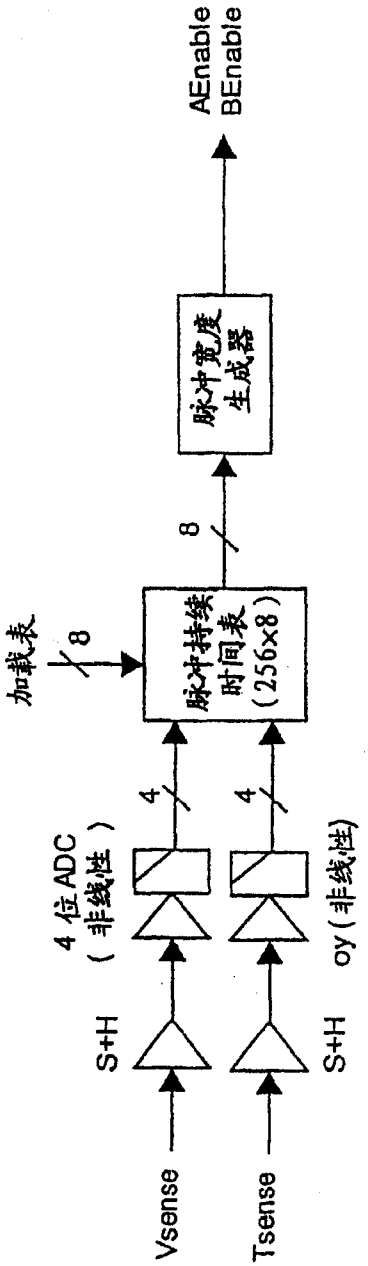

AEnable
BEnable
加载表
8
脉冲持续时间表 (256x8)
脉冲宽度生成器
8
4
4
4 位 ADC (非线性)
S+H
S+H
Vsense
Tsense
oy (非线性)

图38

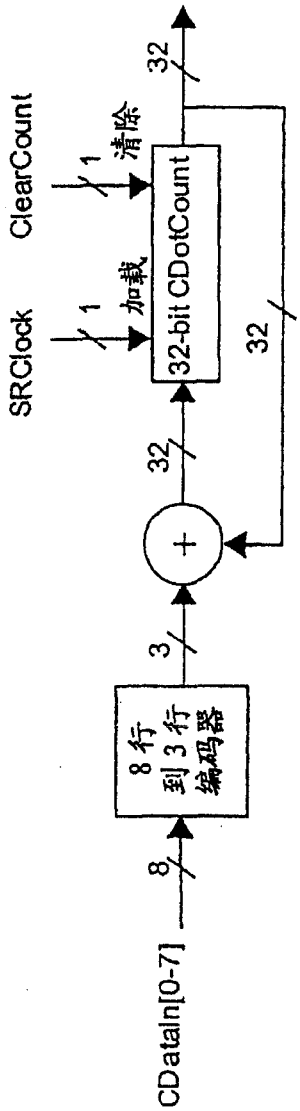

SRClock
ClearCount
1
加载
1
清除
32-bit CDotCount
32
32
32
+
3
8 行到 3 行编码器
8
CDataIn[0-7]

图39

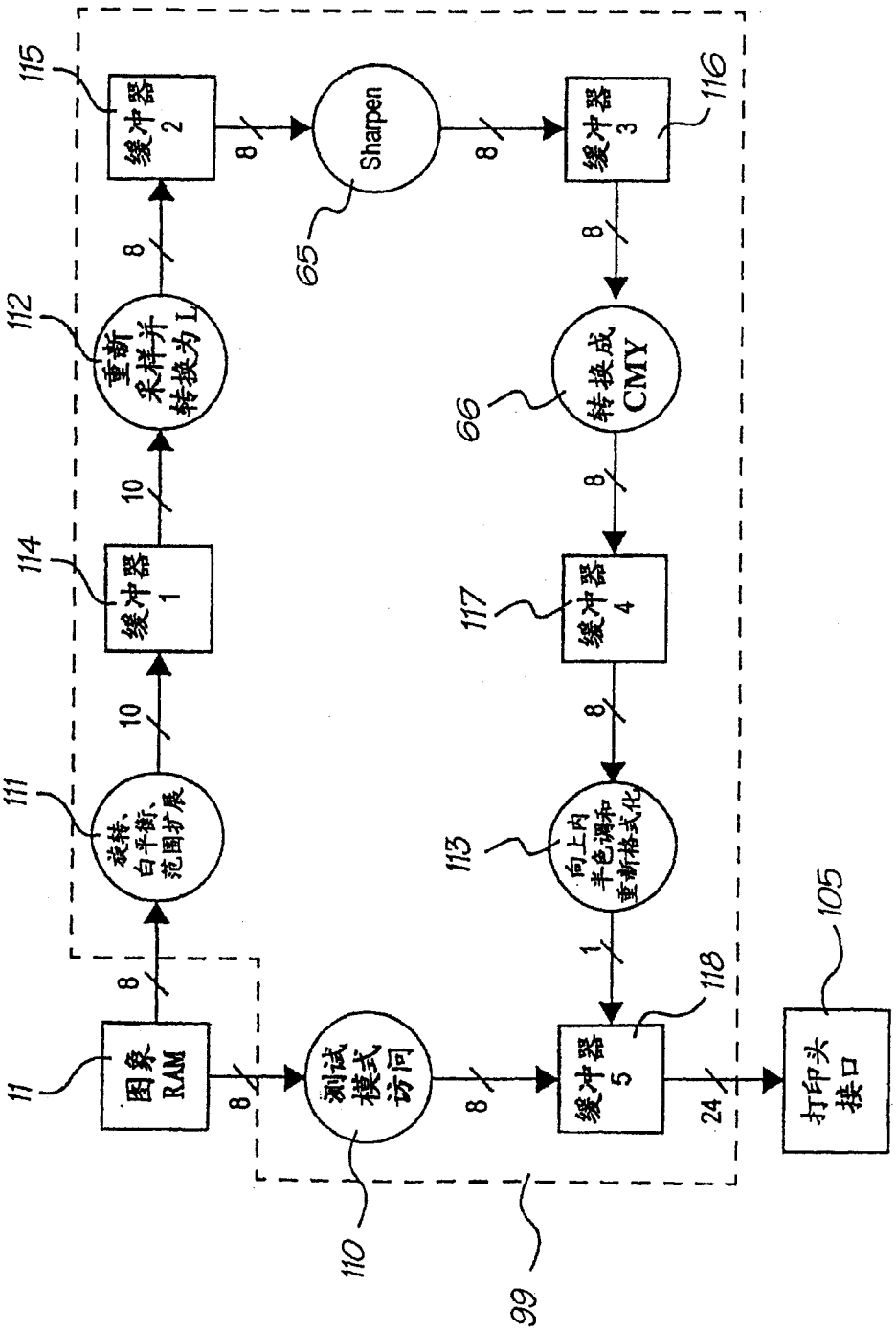

图象 RAM
11
8
测试模式访问
110
8
缓冲器 5
118
24
打印头接口
105
99
1
向上内半色调和重新格式化
113
8
缓冲器 4
117
8
转换成 CMY
66
8
缓冲器 3
116
8
Sharpen
65
8
缓冲器 2
115
8
重新采样并转换为 L
112
10
缓冲器 1
114
10
减特、白平衡、范围扩展
111
8

图41

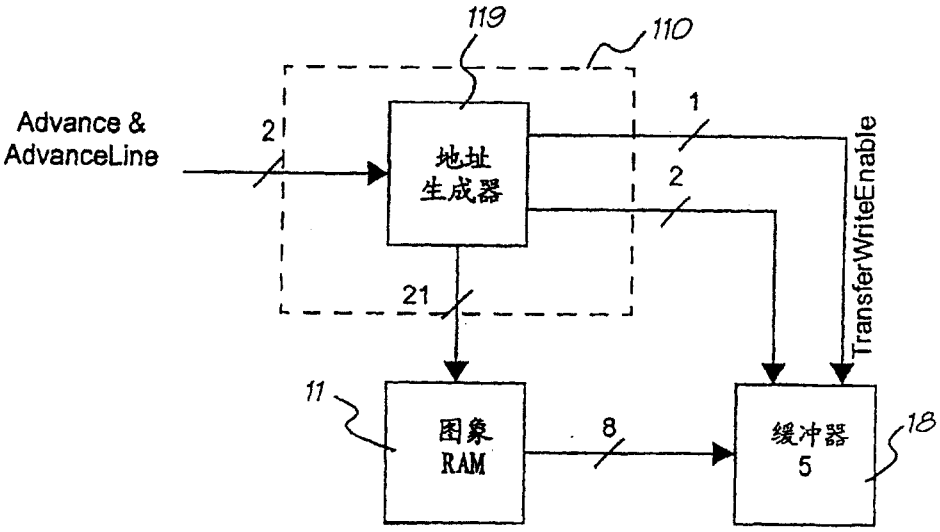

119
110
Advance &
AdvanceLine
2
地址
生成器
1
2
TransferWriteEnable
21
11
图象
RAM
8
缓冲器
5
18

图42

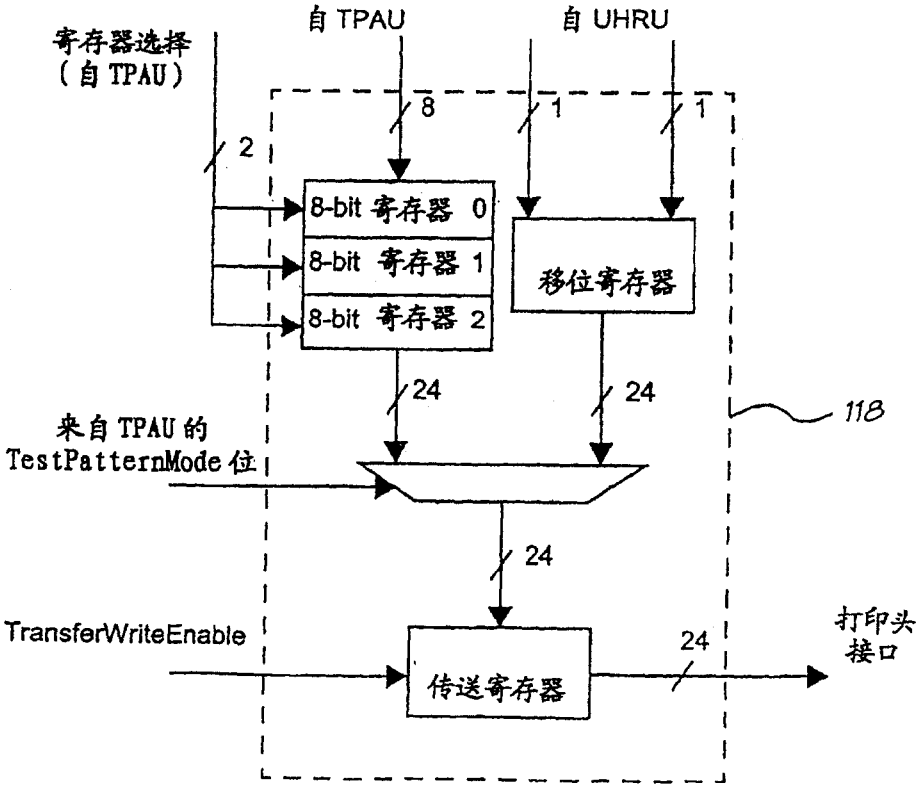

寄存器选择
(自 TPAU)
自 TPAU
自 UHRU
8
1
1
2
8-bit 寄存器 0
8-bit 寄存器 1
8-bit 寄存器 2
移位寄存器
24
24
118
来自 TPAU 的
TestPatternMode 位
24
TransferWriteEnable
传送寄存器
24
打印头
接口

图43

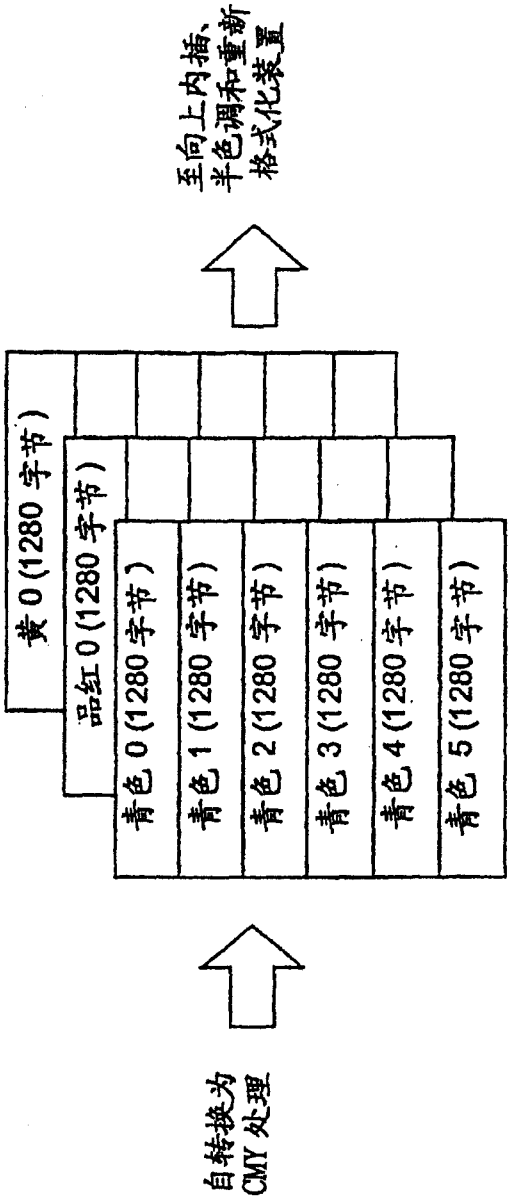

自转换为
CMY 处理
青色 0 (1280 字节)
青色 1 (1280 字节)
青色 2 (1280 字节)
青色 3 (1280 字节)
青色 4 (1280 字节)
青色 5 (1280 字节)
品红 0 (1280 字节)
黄 0 (1280 字节)
至向上内插、
半色调和重新
格式化装置

图44

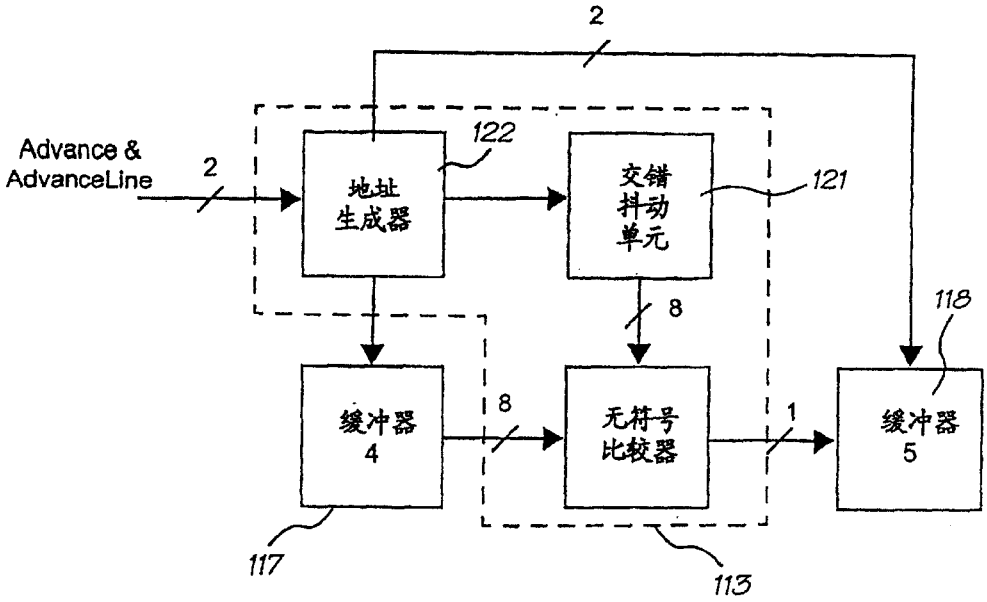

2
Advance &
AdvanceLine
2
地址
生成器
122
交错
抖动
单元
121
8
118
缓冲器
4
8
无符号
比较器
1
缓冲器
5
117
113

图45

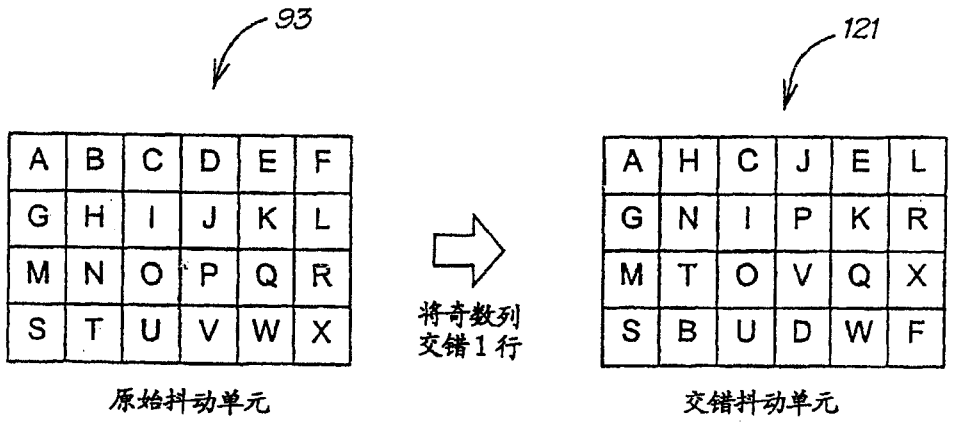

93
121
A B C D E F
G H I J K L
M N O P Q R
S T U V W X
原始抖动单元
将奇数列
交错1行
A H C J E L
G N I P K R
M T O V Q X
S B U D W F
交错抖动单元

图46

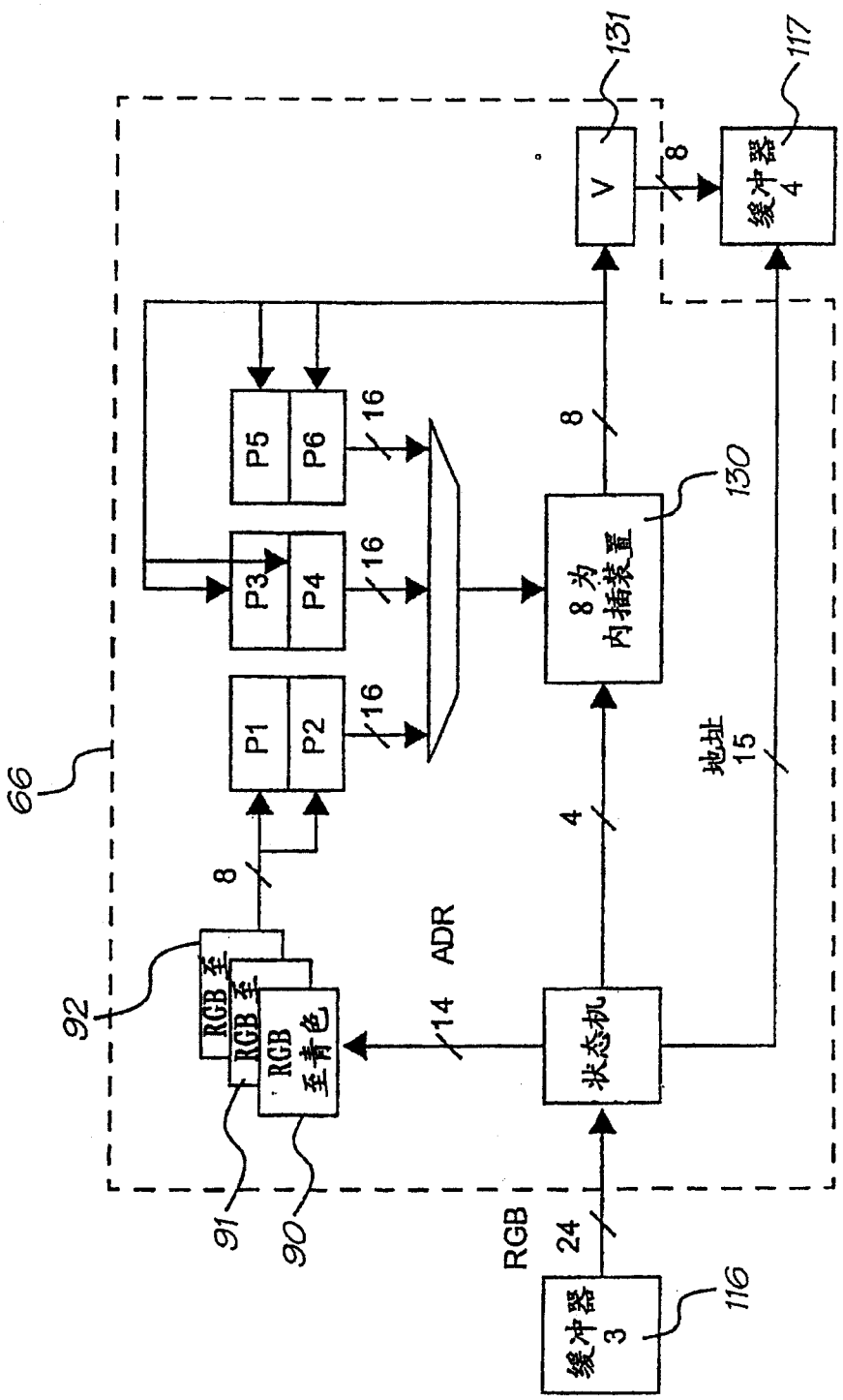

66
92
RGB 至
RGB 至
RGB
至青色
91
90
8
P1
P2
P3
P4
P5
P6
16
16
16
8 为
内插装置
130
8
V
131
缓冲器
4
117
8
4
ADR
14
状态机
地址
15
RGB
24
缓冲器
3
116

图47

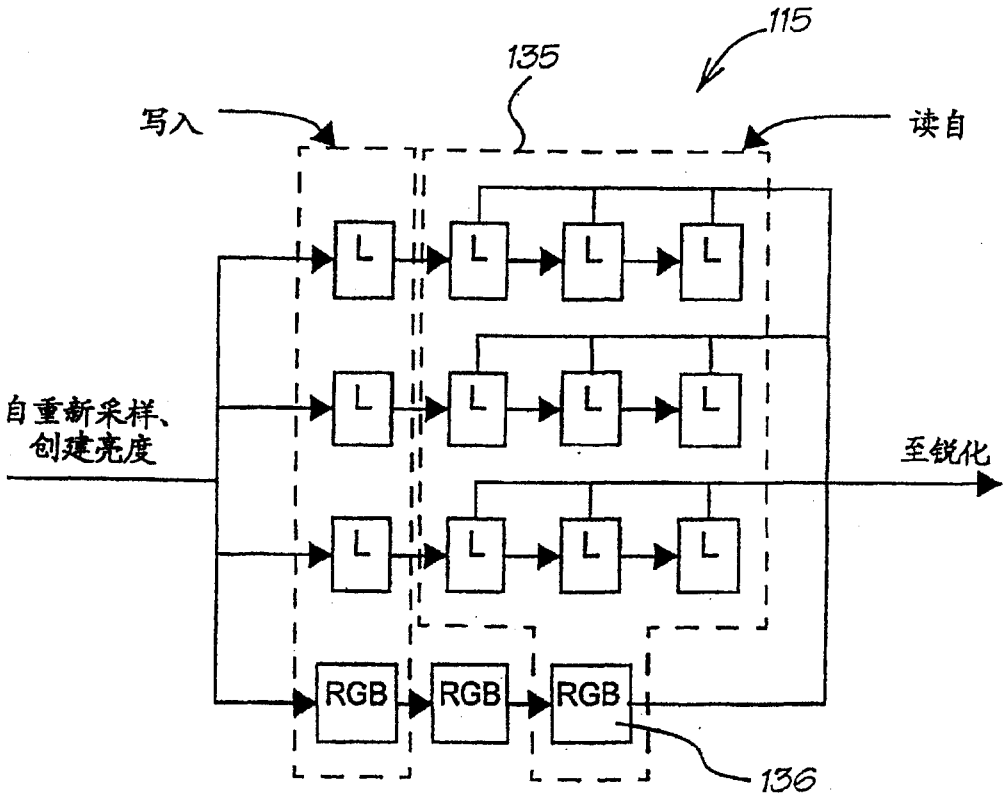


图48

$\frac{1}{9}\times$

-1	-1	-1
-1	8	-1
-1	-1	-1

图49

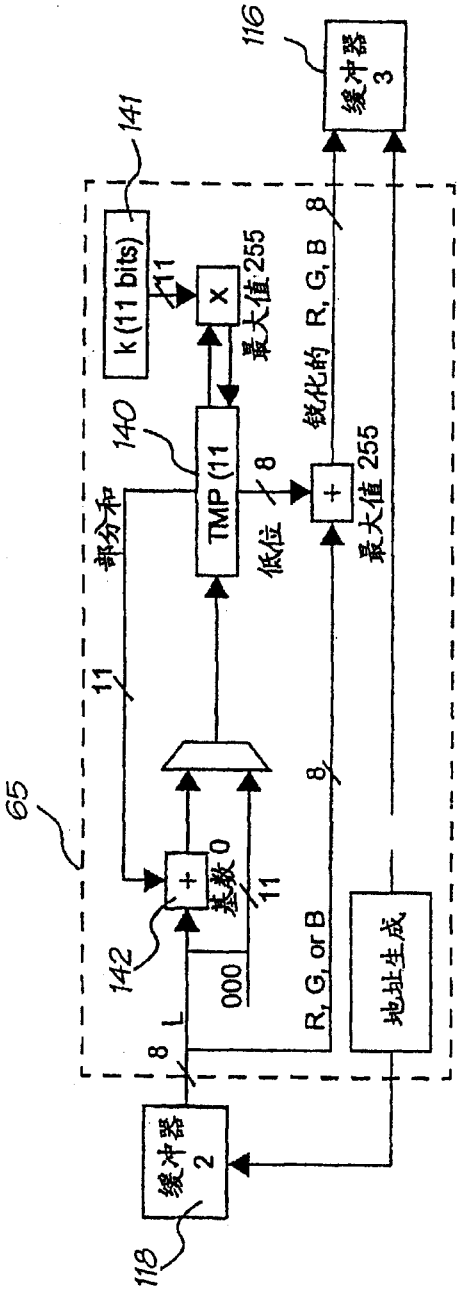

65
缓冲器 2
118
缓冲器 3
116
141
k (11 bits)
11
X
最大值255
140
TMP (11
8
低位
锐化的 R, G, B
8
部分和
11
+
最大值 255
142
基数 0
11
000
R, G, or B
8
L
8
地址生成

图50

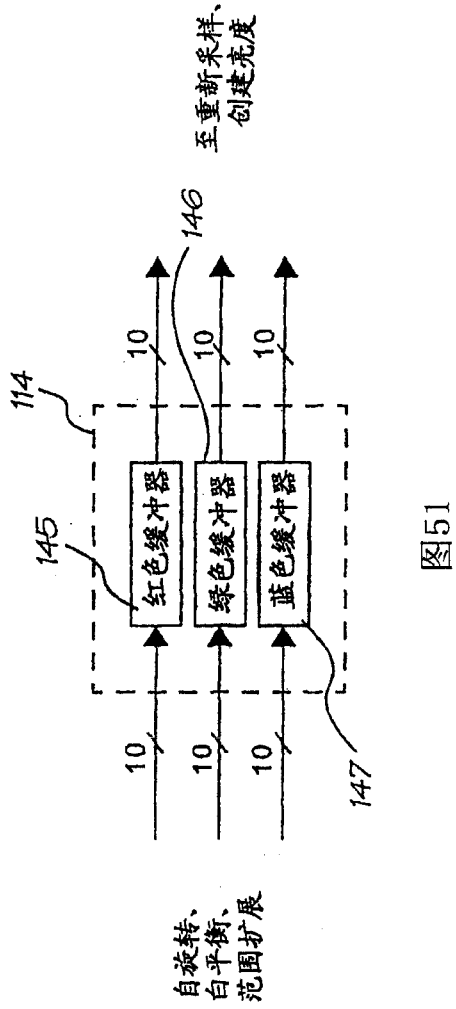

114
145
红色缓冲器
绿色缓冲器
蓝色缓冲器
10
10
10
146
至重新采样、创建亮度
自旋转、白平衡、范围扩展
147

图51

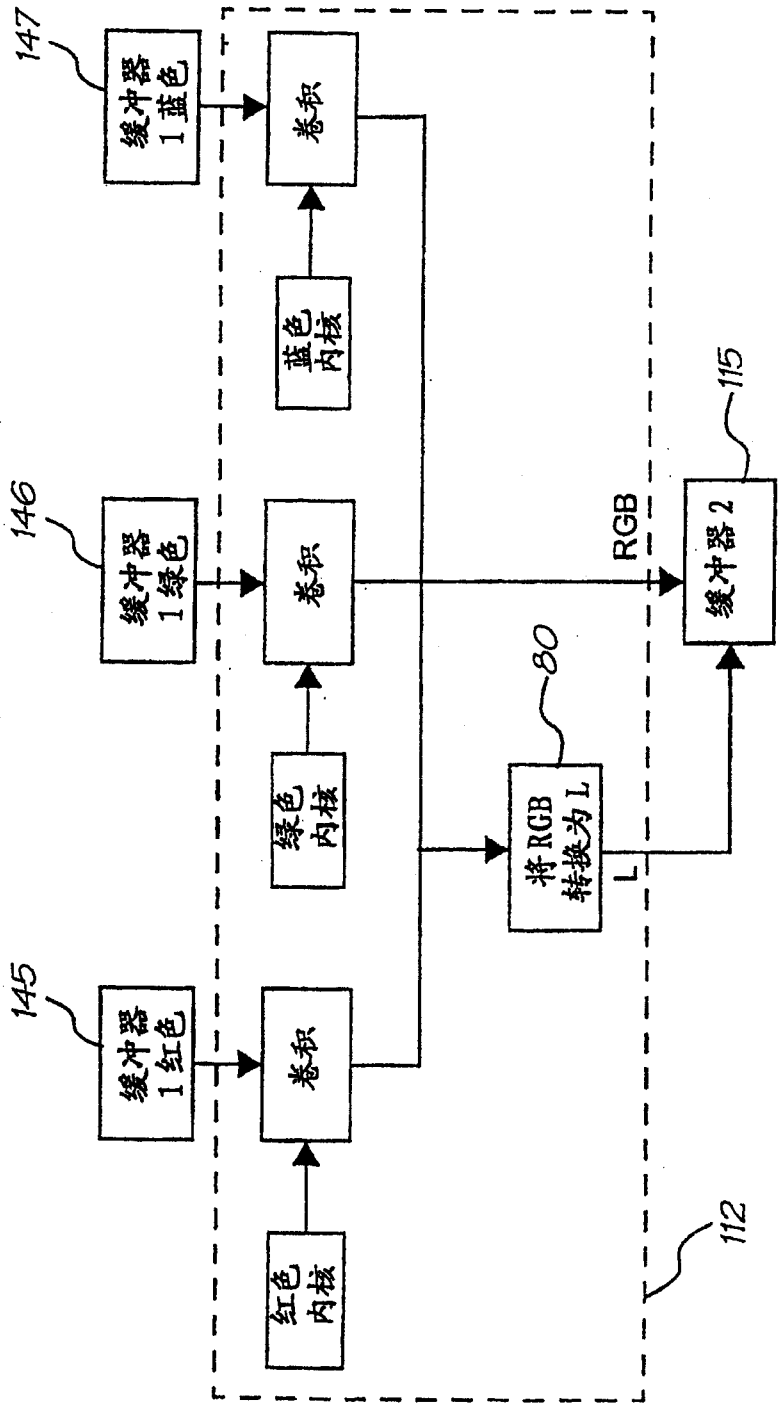

147
缓冲器1蓝色
卷积
蓝色内核
146
缓冲器1绿色
卷积
绿色内核
115
缓冲器2
RGB
80
将RGB转换为L
L
145
缓冲器1红色
卷积
红色内核
112

图52

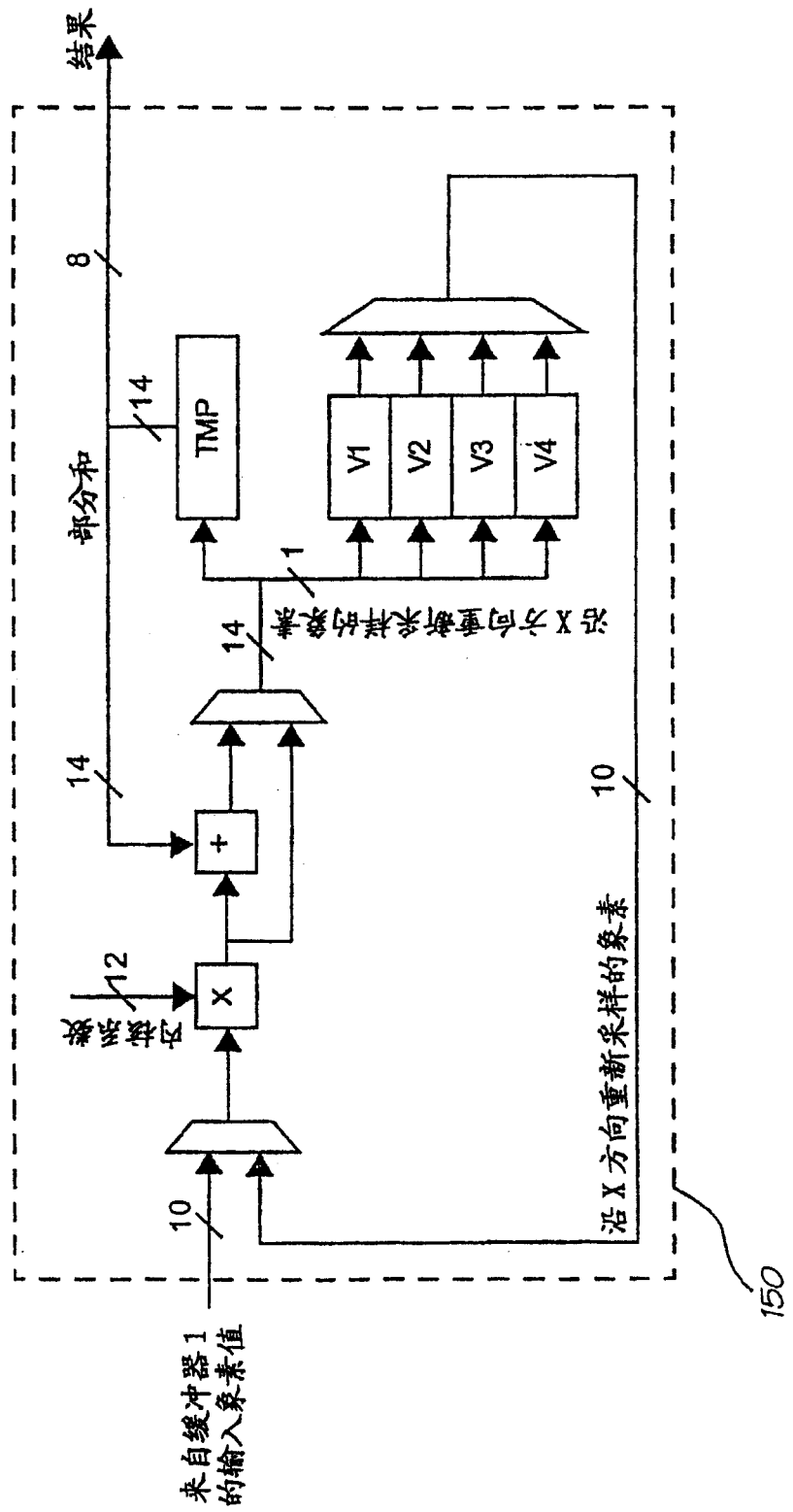

结果
8
部分和
14
TMP
V1
V2
V3
V4
1
沿 X 方向重新采样的象素
14
14
+
10
沿 X 方向采样
12
X
10
来自缓冲器 1
的输入象素值
沿 X 方向重新采样的象素
150

图53

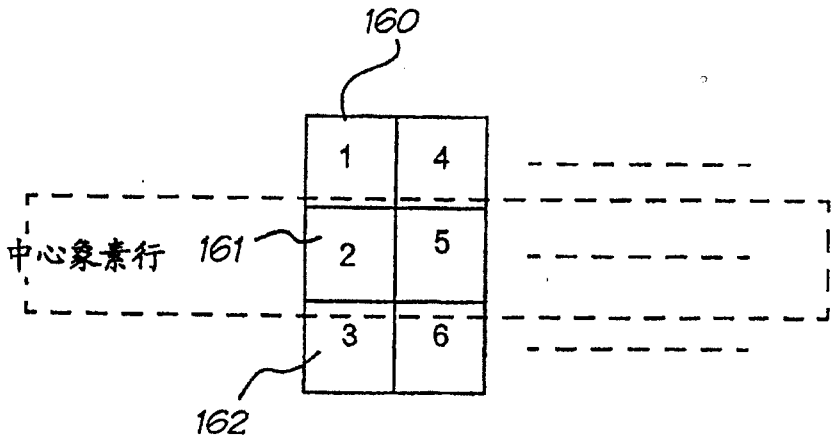

160
1
4
中心象素行
161
2
5
3
6
162

图54

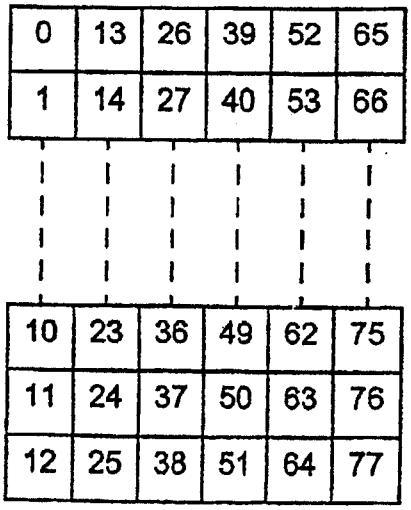

0 13 26 39 52 65
1 14 27 40 53 66
10 23 36 49 62 75
11 24 37 50 63 76
12 25 38 51 64 77

图56

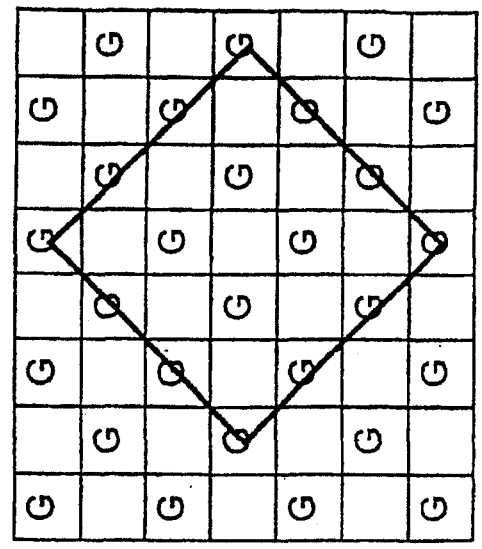
沿X方向的移动使得沿X方向增加但沿Y方向不变
沿Y方向的移动使得沿X方向增加但沿X方向不变
不旋转
沿Y方向的移动使得沿X方向增加
沿X方向的移动使得沿X方向增加沿Y方向减少
Rotated

图55

G
G
偶数行具有基数象素的采样
奇数行具有偶数象素的采样
基础2×2单元（旋转=0）

G
G
偶数行具有偶数象素的采样
奇数行具有奇数象素的采样
基础2×2单元（旋转=90）

图57



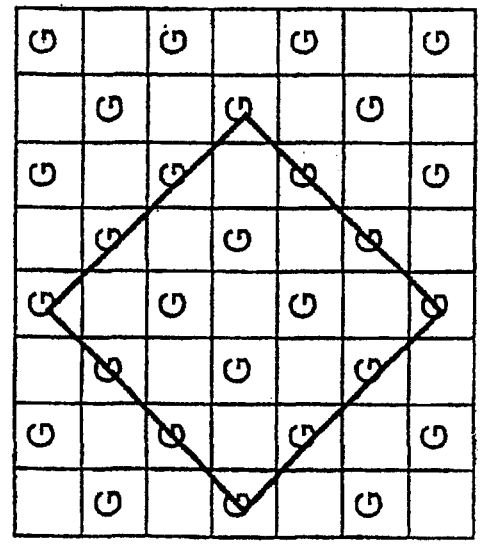


图58

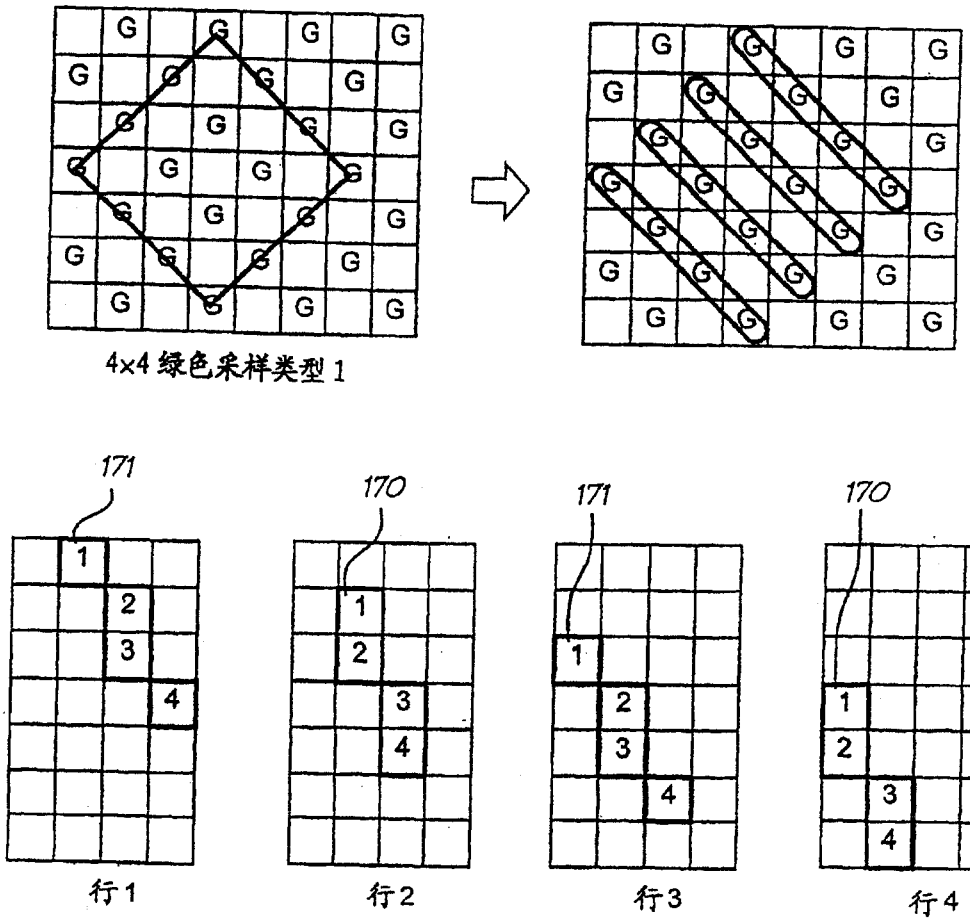
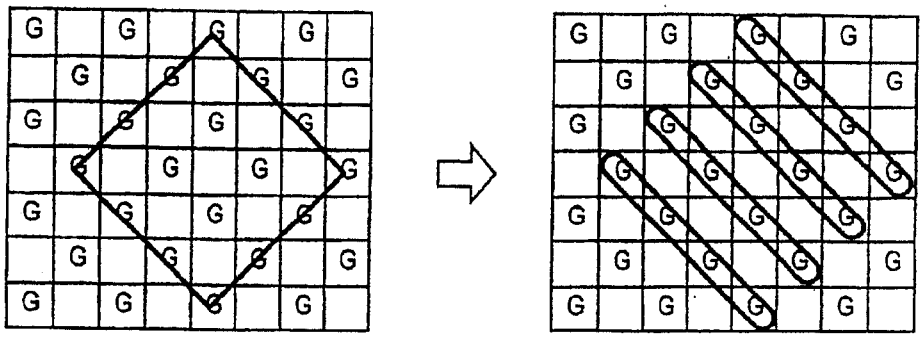


图59



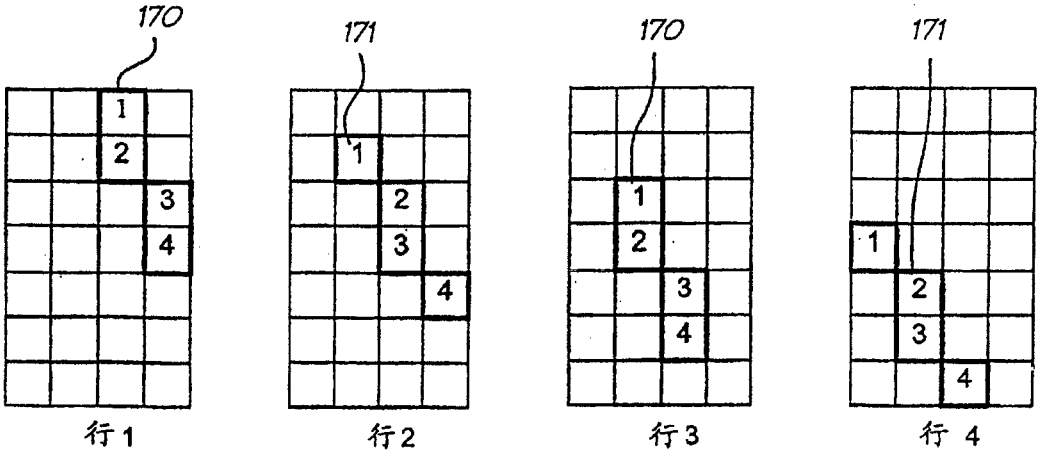


图60

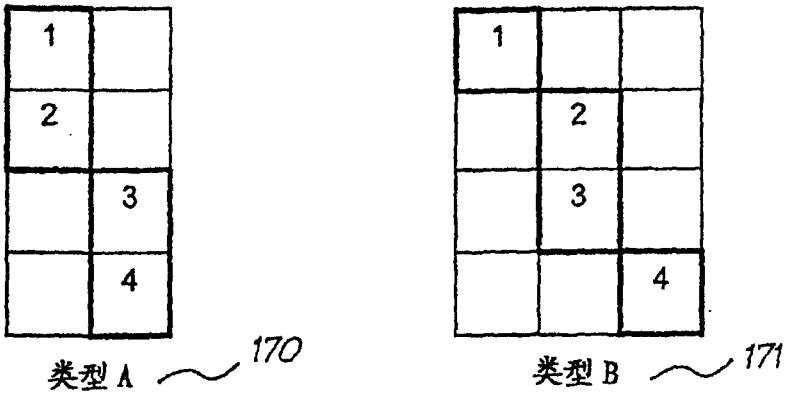


图61

0	6	12	18	24	30
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

图62

1	2	3	4		
5	6	7	8		
9	10	11	12		
13	14	15	16		

图63

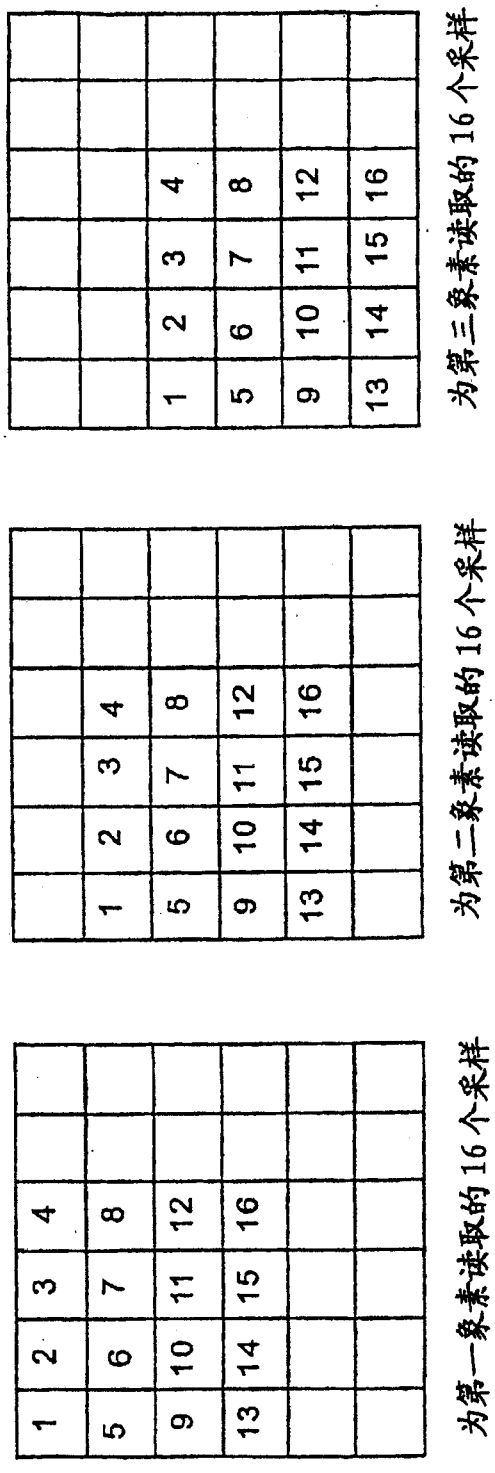


图64

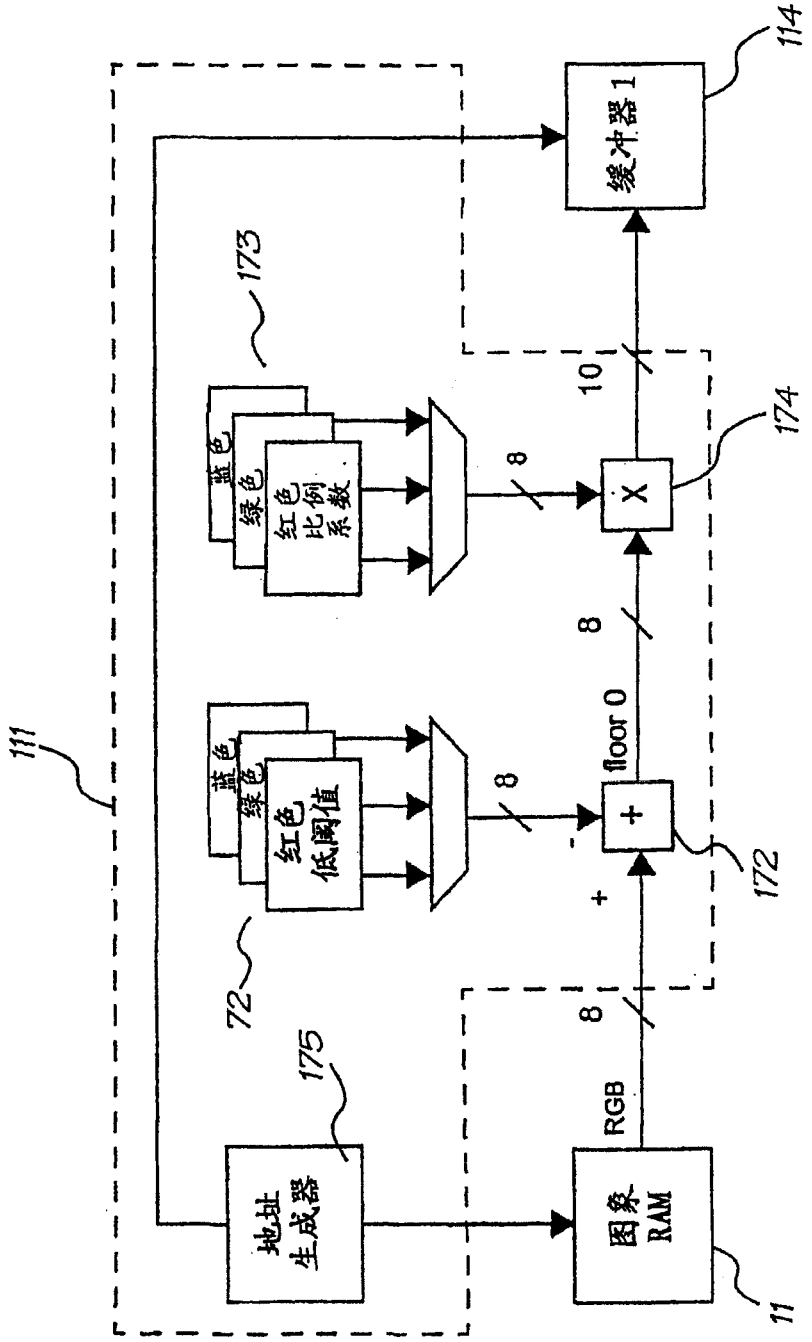

111
114
缓冲器1
173
蓝色
绿色
红色比例系数
8
10
174
X
8
floor 0
+
-
172
蓝色
绿色
红色低阈值
8
72
+
8
175
RGB
地址生成器
图象RAM
11

图65

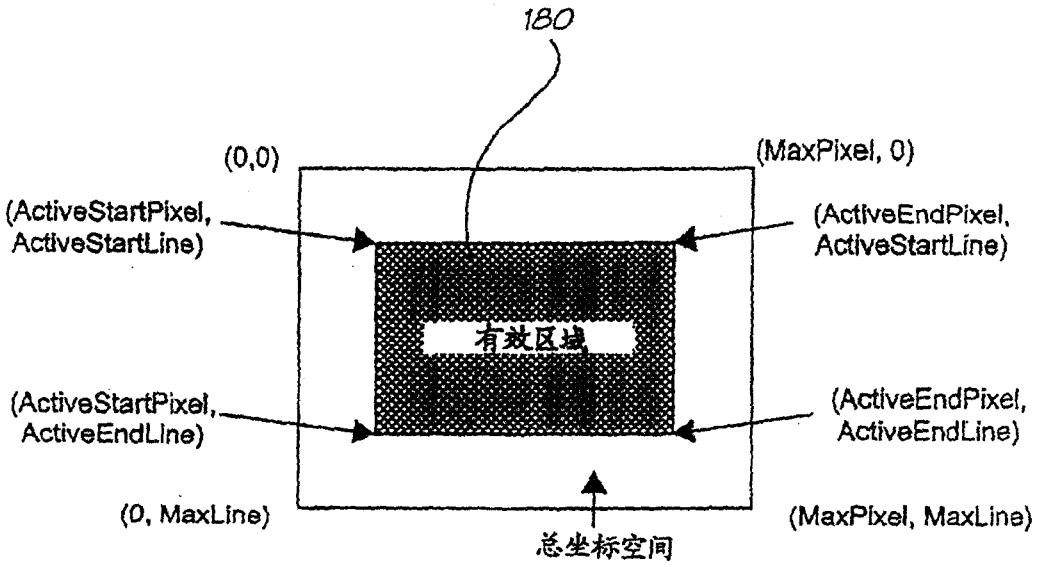

180
(0,0)
(MaxPixel, 0)
(ActiveStartPixel, ActiveStartLine)
(ActiveEndPixel, ActiveStartLine)
有效区域
(ActiveStartPixel, ActiveEndLine)
(ActiveEndPixel, ActiveEndLine)
(0, MaxLine)
(MaxPixel, MaxLine)
总坐标空间

图66