(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1993976 B
(45)授权公告日 2010.09.29

(21)申请号 200580025638.X

(22)申请日 2005.07.26

(30)优先权数据
60/591,876 2004.07.29 US

(85)PCT申请进入国家阶段日
2007.01.29

(86)PCT申请的申请数据
PCT/EP2005/008723 2005.07.26

(87)PCT申请的公布数据
WO2006/010644 EN 2006.02.02

(73)专利权人 奥西-技术有限公司
地址 荷兰芬洛

(72)发明人 M·L·M·卢特默 P·索布卡克 J·马利克

(74)专利代理机构 中国专利代理(香港)有限公司 72001

代理人 刘春元 陈景峻

(51)Int.Cl.
H04N 1/417(2006.01)

(56)对比文件
US 20010036231 A1,2001.11.01, 第0022、0024-0026、0139、0148、0156、0157段,附图3和7B.
US 6292587 B1,2001.09.18, 全文.
US 5198898 A,1993.03.30, 第4栏第5至63行,第5栏第12行至第6栏第26行,说明书第3栏第13行值48行,附图3,摘要,权利要求.
CN 1405735 A,2003.03.26, 全文.
CN 1394429 A,2003.01.29, 全文.
CN 1167534 A,1997.12.10, 全文.

审查员 孟佳

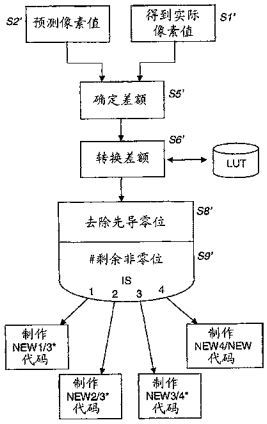
权利要求书 3 页 说明书 21 页 附图 7 页

(54)发明名称

用于利用熵编码的彩色图像数据的无损压缩的方法和设备

(57)摘要

一种压缩位于扫描行上的像素的数字连续色调图像的方法,每个像素的像素值由至少一个颜色通道值定义,颜色通道值具有B个位(B>1)的精度,该方法产生被压缩的图像数据并且包括:针对要被编码的当前像素,所述当前像素具有实际像素值,利用固定的规则,基于来自相同图像的至少一个先前处理过的像素的像素值预测一预测像素值;基于所述预测像素值和要被编码的所述当前像素的实际像素值的差值确定差别参数;并对于值等于零的不中断序列的最高阶位的存在,检查该差别参数;去除至少部分所述最高阶零位;并且,如果剩余在预定限制内的多个位,则生成具有预定的固定长度的压缩代码,所述代码表示剩余位的数量。为了以在实际压缩过程中能够更有效地压缩图像数据的方式来准备图像数据,公开了若干图像数据的预处理步骤。



CN 1993976 B

1. 一种压缩位于扫描行上的像素的数字连续色调图像的方法，每个像素的像素值由至少一个颜色通道值定义，颜色通道值具有 B 个位 (B ＞ 1) 的精度，该方法产生被压缩的图像数据并且包括，

对于要被编码的当前像素，所述当前像素具有实际像素值，

利用固定的规则，基于来自相同图像的至少一个先前处理过的像素的像素值预测一预测像素值，

基于所述预测像素值和要被编码的所述当前像素的实际像素值的差值来确定一差别参数，

该方法的特征在于，它还包括 ：

检查所述差别参数，以确定具有等于零的值的最高阶位的不中断序列的存在，

去除至少一部分所述具有等于零的值的最高阶位，并且，

如果剩余低于预定值的多个位，则为所述当前像素生成压缩代码，所述压缩代码具有预定的固定长度并表示剩余位的数量，以及

如果剩余等于或高于所述预定值的多个位，则基于所述当前像素的未压缩值来编码该当前像素。

2. 根据权利要求 1 所述的方法，其中，压缩代码包括代码类型名称符和多个剩余位名称符。

3. 根据权利要求 1 所述的方法，其中

该方法产生压缩图像数据信息块，这些压缩图像数据信息块包括 ：

代码块和数据块，代码块包含用于每个像素的具有预定的固定长度的压缩代码，且数据块包含附加数据，该附加数据包括所述剩余位。

4. 根据权利要求 1 所述的方法，其中

基于所述预测像素值与实际像素值的差值来确定差别参数的步骤包括 ：

将所述差值映射到所映射的差值上，所述所映射的差值对于每个可能的差值都是唯一的，并且具有随所述差值的增加的绝对值而增加的值。

5. 根据权利要求 3 所述的方法，其中

所述压缩代码和所述附加数据全部包括相等的、固定数量的位。

6. 根据权利要求 5 所述的方法，其中

所述压缩代码每个包括 4 个位。

7. 根据权利要求 3 所述的方法，其中

所述附加数据包括可变数量的位。

8. 根据权利要求 1 到 7 中的任何一个所述的方法，其中，所述像素值由至少 2 个颜色通道来定义，每个颜色通道具有 B 个位的精度，

其中，在确定色差参数的步骤中，

根据预定的方案首先通过去相关转换来转换每个像素值。

9. 根据权利要求 1 或 4 中的任何一个所述的方法，其中，所述像素值由至少 2 个颜色通道来定义，每个颜色通道具有 B 个位的精度，

其中，确定色差参数的步骤还包括 ：

根据预定的置换方案，通过连接每个颜色通道的所述差值的位，分别组合每个颜色通

道的差值或者每个颜色通道的转换后的像素值的差值。

10. 根据权利要求 1 所述的方法，其中

相同扫描行上的先前被编码的相邻像素的像素值被用作所述预测像素值。

11. 根据权利要求 1 所述的方法，还包括

检查要被编码的当前像素的实际像素值是否等于先前被编码的像素的实际像素值，该先前被编码的像素位于相同扫描行上并且与当前像素隔着具有不同于当前像素的像素值的像素值的一个像素，或者隔着都具有相等的、不同于当前像素的像素值的像素值的多个像素，并且如果这样，则为那个当前像素生成专用的通用代码。

12. 根据权利要求 1 所述的方法，还包括

行程长度编码步骤，其中，要被压缩的当前像素与其先前处理过的相邻像素中的特别预定的一个像素进行比较，并且如果所述像素的像素值相等，则针对下一个要被编码的像素重复所述比较步骤，从而最大化相同扫描行上的像素的行程长度，该相同扫描行上的像素的值分别等于它们的特别预定的相邻像素的值，并且

针对那些像素生成行程长度码，所述行程长度码包括一个或多个固定长度的位组，每个位组包括代码类型名称符和行程长度名称符，该行程长度名称符以位表示法详细说明所述行程长度编码步骤的行程长度。

13. 根据权利要求 12 所述的方法，其中，

当行程长度位表示法比一个行程长度名称符中能够容纳的长时，在连续位组的行程长度名称符上划分行程长度位表示法。

14. 一种解压根据权利要求 3 的被压缩的图像数据的方法，其包括

检查被包括在代码中的代码类型名称符，并且如果所述代码类型名称符将该代码命名为压缩代码，则针对要从与所述压缩代码有关的数据代码中读取的多个剩余位检查多个剩余位名称符，并在此基础上重新构建一个像素值。

15. 根据权利要求 14 所述的方法，还包括

检查被包括在当前代码中的代码类型名称符，并且如果所述代码类型名称符将该代码命名为行程长度码，则搜索具有与当前代码相同的代码类型名称符的所有紧随的代码，检查因此被发现的代码中所包括的行程长度名称符并通过连接所述行程长度名称符来形成行程长度。

16. 一种用于压缩位于扫描行上的像素的数字连续色调图像的设备，每个像素的像素值由至少一个颜色通道值来定义，颜色通道值具有 B 个位 ($B > 1$) 的精度，该设备产生被压缩的图像数据并且包括，

编码模块，用于编码当前像素，所述当前像素具有实际像素值，

预测器，用于利用固定的规则，基于至少一个来自相同图像的先前处理过的像素的像素值预测一预测像素值，

差别参数确定器，用于基于所述预测像素值和要被编码的所述当前像素的实际像素值的差值来确定差别参数，该设备的特征在于，它还包括：

检查模块，用于检查该差别参数以确定具有等于零的值的最高阶位的不中断序列的存在，

切断模块，用于去除至少一部分所述具有等于零的值的最高阶位，并且，

代码生成器，用于

如果剩余低于预定值的多个位，则为所述当前像素生成压缩代码，所述压缩代码具有预定的固定长度，并表示剩余位的数量，以及

如果剩余等于或高于所述预定值的多个位，则基于所述当前像素的未压缩值来编码该当前像素。

17. 根据权利要求 16 所述的设备，还包括

行程长度检查器，用于将要被压缩的当前像素与其先前处理过的相邻像素中的特别预定的一个像素进行比较，并且如果所述像素的像素值相等，则针对下一个要被编码的像素重复所述比较步骤，从而最大化相同扫描行上的像素的行程长度，这些相同扫描行上的像素的值分别等于它们的特别预定的相邻像素的值，以及

行程长度码生成器，用于为那些由所述行程长度检查器找到的像素生成行程长度码，所述行程长度码通过在所述行程长度码中包括一个或多个固定长度的位组来生成，每个位组包括代码类型名称符和行程长度名称符，该行程长度名称符以位表示法详细说明所述行程长度编码步骤的行程长度。

18. 根据权利要求 17 所述的设备，其中，

当行程长度位表示法比一个行程长度名称符中能够容纳的长时，所述行程长度码生成器在连续位组的行程长度名称符上划分行程长度位表示法。

用于利用熵编码的彩色图像数据的无损压缩的方法和设备

[0001] 本发明涉及连续色调图像的无损压缩。更特别地，本发明提供一种压缩位于扫描行上的像素的数字连续色调图像的方法，每个像素具有由至少一个颜色通道值定义的像素值，颜色通道值具有为 B 个位 (B ＞ 1) 的精度。应理解，颜色通道也可包括单个黑白通道。

[0002] 本发明还涉及一种计算机程序产品，用于当该计算机程序产品在其中执行该方法的计算机和设备、特别是打印控制器上运行时执行该方法。最后，本发明涉及一种用于解压缩已经利用本发明被压缩的图像数据的方法。

[0003] **背景技术**

[0004] 通常用页面描述语言 (PDL) 来定义被发送到打印机的页面。PDL 的实例是来自 Adobe 系统有限公司的 PostScript。在计算机上运行的 PDL 解释器解释该 PDL，从而为要被打印的每页产生数字页面图像。这些页面图像可以是能够直接被传到打印引擎的形式，或者这些页面图像能以数字连续色调图像的形式被存储，数字连续色调图像可在将它们传到打印引擎之前进一步被处理。例如当需要在 PDL 解释器中不可用的特殊的图像处理时，后者方法可被用来例如将连续色调图像转换为该打印引擎能够接受的格式。

[0005] 对于使用后者方法的彩色打印机，连续色调图像必须被存储并且可能被传送到图像处理硬件。然而，连续色调图像的数据量可能非常大。例如，对于 A4 大小的 600dpi 连续色调 (每个颜色通道 8 位)CMYK 图像来说，需要大约 140 兆字节。这意味着，花费相当大的存储量来存储该图像，并且花费相当长的时间量来传输该图像。这个问题的通常的解决方案是压缩该连续色调图像。

[0006] 例如，图 11 示出了打印控制器的示意图，该打印控制器也被称为“光栅图像处理器”(RIP) 或者“色彩服务器”。该打印控制器解释打印作业 (例如，用 PostScript 来定义的打印作业)，在磁盘上存储页面图像并且当需要时从那里将页面图像传输到打印机引擎。

[0007] 彩色打印控制器包含解释进入的打印作业并将该打印作业转换为未被压缩的连续色调彩色页面图像的 PDL 解释器 112。直接在生成页面图像之后，由本发明的压缩器 114 压缩这些页面图像。得到的被压缩的页面图像被存储在形成页面存储器的硬盘 116 上。当该引擎需要打印该作业时，该被压缩的页面图像从该硬盘被读取并通过打印机接口电缆 118(例如，火线) 被传输到该引擎。

[0008] 图 12 示出了通过打印机接口电缆 118 被连接到图 11 的打印控制器的打印机。

[0009] 在缓冲器 122 中接收来自该打印控制器的被压缩的页面图像。该被压缩的图像数据从缓冲器 122 被传到将被压缩的页面图像转换为未被压缩的图像的解压器 124。通过图像处理模块 126 传递未被压缩的图像，以将该未被压缩的图像转换为被用来驱动打印机引擎 128 的引擎专用位图。

[0010] 直接在 PDL 解释器之后压缩图像数据的事实节约了图像往返于硬盘传输时的带宽，从而允许使用比未被压缩的数据所需要的磁盘更慢的磁盘。还允许更小的磁盘。同样，降低了通过打印机接口电缆传输图像所需的带宽，因此可以使用具有较少可用带宽的接口。

[0011] 各种各样的无损图像压缩技术可用于压缩连续色调图像，这些技术例如 JPEG-LS。

[0012] JPEG-LS 基于 HP 的 LOCO-1 算法，并且已经被 ISO/ITU-T 选为连续色调图像的无损压缩标准。在 M.Weinberger、G.Seroussi、G.Saipiro 的“LOCO-1 :A Low Complexity, Context-Based,LosslessImage Compression Algorithm(低复杂度、基于上下文的无损图像压缩算法)” 中给出了该算法的说明，该文献在 www.hpl.hp.com/loco/HPL-98-193R1.pdf 上可得到。

[0013] 然而，提供合理压缩量的压缩技术压缩图像要花费比 PDL 解释器在相同计算机上执行时生成相同图像所花费的时间多得多的时间。这意味着，如果将这种压缩方法用于压缩页面图像，则打印机的性能会降低到无法接受的水平。

[0014] 足够快的现有技术（例如，简单的 1 维行程长度压缩）平均并没有提供足够的压缩。

发明内容

[0015] 本发明的目的是提供一种能够在通用处理器上有效地实现并且允许无损压缩 PDL 生成的图像的低复杂度的软件解决方案，该软件解决方案的执行时间远小于 PDL 解释器在相同的计算机上生成图像所花费的时间。

[0016] 这将允许压缩在与被用来运行 PDL 解释器相同的计算机上实现，而无需额外的硬件并且没有大的性能损失。

[0017] 本发明的另一目的是提供一种压缩系数至少与现有技术的无损图像压缩算法一样好的低复杂度的压缩器。

[0018] 本发明的另一目的是允许解压器的简单的硬件实现方案。

[0019] 通过权利要求 1 中所要求保护的压缩方法达到这些和其它目的。

[0020] 根据这种方法，像素值与基于相邻像素预测的预测像素值进行比较。由于在许多情况下像素值并不会在位置上快速变化，所以实际像素值与预测像素值之间的差通常很小，并且如果不多的话，将有几个前导零位。通过去掉这些前导零位，该数据被恢复为较少位，这有效地导致了压缩。

[0021] 在从属权利要求中陈述了更有利的方面。这些方面说明了被用来以这样的方式准备图像数据，使得实际像素值与预测像素值之间的最终的差变得甚至更小，从而导致更好的压缩。

[0022] 压缩针对 PDL 生成的图像被优化，但是该压缩也可有利地被用于其它种类的数字图像，例如用于扫描图像。

附图说明

[0023] 在下文中，将参考本发明的优选实施方式的公开内容，并且特别是参考附图，更加详细地论述本发明的这些和另外的特征、方面和优点，其中：

[0024] 图 1 示出由典型的 PDL 解释器缩放为双倍分辨率的图像像素；

[0025] 图 2 示出如根据本发明所生成的那样的针对扫描行的数据块和代码块的构造；

[0026] 图 3 示出根据本发明的行程长度码中的行程长度字段的解释的实例；

[0027] 图 4 示出在像素扫描行中的 PREV 和 LEFT 颜色的说明；

[0028] 图 5 示出预测器的实例；

[0029] 图 6 示出描述根据本发明的第一实施方式的颜色值的压缩方法的流程图；

[0030] 图 7A/B 示出第一实施方式中的颜色通道数据的组合的示意性表示；

[0031] 图 8 示出描述根据本发明的第一实施方式的完整的压缩方法的流程图；

[0032] 图 9 示出描述根据本发明的第二实施方式的完整的压缩方法的流程图；

[0033] 图 10 示出描述根据本发明的第二实施方式的灰度值压缩方法的流程图；

[0034] 图 11 示出打印控制器的示意图；和

[0035] 图 12 示出打印机的示意图。

具体实施方式

[0036] 根据本发明的压缩的概况

[0037] 本发明基本上打算供 PDL（页面描述语言，诸如 Postscript）解释器生成的页面图像的压缩使用。这种图像有特定的特性，这些特性将在下面描述。针对这些图像，这种压缩是最优的，但是这种压缩也可被用于其它种类的数字图像，例如，用于扫描图像。在后者情况下，扫描仪噪声消极地影响压缩比，但是试验已揭示，仍然可达到合理的压缩比。在这种情况下，噪声抑制预处理步骤将改善结果，因为该噪声抑制预处理步骤会增加行程长度压缩的机会，尽管随后压缩不再是无损的。

[0038] 此外，已经用软件实现了依照本发明的压缩器，以致压缩图像所需的执行时间平均远远小于 PDL 解释器在相同计算机上生成相同图像所花费的时间。这允许压缩器在运行 PDL 解释器的相同的计算机上被实现，而无需额外的硬件并且没有大的性能损失。

[0039] 压缩格式是简单的，以便用硬件简单地实现解压器。这尤其意味着，能够限制解压器所需的存储器数量。

[0040] 同样，已经设计了压缩器，以致硬件实现的解压器能够以固定的速率产生被解压的像素数据。这意味着，被压缩的数据的最大局部展开必定受到限制。

[0041] 要被压缩的图像的图像特性

[0042] 要被压缩的图像是优选地由 PDL 解释器生成的彩色或黑白连续色调图像。除非特别定义，否则在下面的说明中，术语“彩色”涉及天然色调和黑白浅灰。PDL 生成的图像有某些典型特性：

[0043] - 在颜色不变的区域中（例如，许多页面中的白色背景），没有噪声。这意味着，为了压缩颜色不变的区域，可以使用象行程长度编码这样的简单机制来压缩。

[0044] - 为了将所采样的图像缩放到设备分辨率，大多数 PDL（象 PostScript）使用最邻近插值法。最邻近插值法给装置像素提供最接近于该装置像素的初始像素的颜色。例如，如图 1 中所示，对于在 600dpi 的装置上所打印的 PostScript 作业中的 300-dpi 图像来说，每个初始像素被映射到 2×2 个装置像素上。

[0045] 现在将描述本发明的第一实施方式，该第一实施方式是用于有色图像的压缩方法。

[0046] 第一实施方式中的压缩图像格式

[0047] 被压缩的图像格式由连续字节组成，每个字节包含两个 4 位半字节。每个字节中的第一半字节是该字节的高阶 4 位；每个字节中的第二半字节是该字节的低阶 4 位。在这个第一实施方式中，压缩图像格式中的数据的最小单位是半字节。用半字节作为数据的最

小单位而不是用位运算的原因是因为在通用处理器上半字节比单个位更有效地被处理。由多个半字节组成的所有数用最高阶的头半字节和最低阶的尾半字节表示。

[0048] 被压缩的图像格式以扫描行在该页面上出现的顺序被组织为连续的信息块，每个信息块描述一个扫描行。选择扫描行式的构造（与页面式的构造相对）的原因是扫描行式的构造允许基于扫描行处理，而无需必须存储完整的被压缩的页面图像。

[0049] 图 2 示出了扫描行的信息块的基本形式的实例。每个信息块由代码块紧跟其后的数据块组成。

[0050] 数据块和代码块中的前 4 个半字节每个形成了长度字段，该长度字段以多个 8 字节来说明该数据 / 代码块的长度。紧跟着这个长度字段的是实际的代码 / 数据半字节。块末端的任何未使用的半字节值被设为 0。数据 / 代码块最多可长 (2 ^ 16-1)*8 = 524288 个字节。长度字段允许基于扫描行浏览被压缩的数据（快速跳过扫描行数据）。

[0051] 代码块包含连续的代码半字节，这些代码半字节是解压器的指令，说明要被生成的像素，从扫描行上的第一像素开始并朝着最后一个像素运算。

[0052] 代码块中的某些代码需要附加数据。这个附加数据以与相对应代码在相同的扫描行的代码块中出现的顺序相同的顺序被存储在数据块中。

[0053] 第一实施方式中所使用的压缩代码

[0054] 给出了 PDL 生成的图像的属性（无噪声的颜色不变的较大区域），2 维行程长度编码将很好地压缩多个 PDL 生成的图像。然而，包含被采样的图像的页面（例如，照片）通常有许多不同的颜色，而许多像素的颜色与其先前被处理的相邻像素不同，所以行程长度编码将不会很好地执行。对于这种情况，附加代码是必需的。本发明提供一种在上述两种情况下都很好地执行的方法。

[0055] 首先，将描述根据本发明的具有专用代码的基本 2 维行程长度压缩方法。

[0056] LEFT 代码

[0057] 定义代码 LEFT，该代码 LEFT 表示当前像素与其左边的相邻像素颜色相同。给这个代码加上行程长度，以表示该代码应该被重复若干次。这是第一维上的行程长度编码。作为第一个在扫描行上运行的代码 LEFT 表示相对应的像素具有白色。

[0058] TOP 代码

[0059] 为了在 2 维进行行程长度编码，定义代码 TOP，该代码 TOP 表示当前像素与上面的像素颜色相同。给这个代码加上行程长度，以表示该代码应该被重复若干次。这是第二维上的行程长度编码。第一扫描行上的代码 TOP 表示相对应的像素为白色。

[0060] 注意，这个代码非常有助于以例如设备分辨率的一半压缩包含被采样的图像的页面。由于 PostScript/PDF 的最邻近插值法，在缩放到设备分辨率之后，对这样一个图像，每隔一个扫描行与上一个扫描行相同（见图 1）。这意味着，在这种情况下，这样一个图像内的每隔一个扫描行可使用 TOP 行程 (run) 来压缩。

[0061] NEW 代码

[0062] 由于不能使用代码 TOP 或者 LEFT 来编码每个像素，所以不得不增加产生新颜色的像素的代码，该代码在被压缩的数据中被规定。代码 NEW 表示当前像素具有在该数据块中用高阶头半字节以 C、M、Y、K 的顺序被规定为未被压缩的颜色值的颜色。对于四个颜色通道中的每一个来说，值 0 对应于无油墨覆盖，而值 255 对应于全油墨覆盖。

[0063] 为了减少压缩格式的最不利情况的展开（以允许简单的硬件解压器），给这个代码增加行程长度，以表示该代码应被重复若干次。

[0064] 增加行程长度

[0065] 如上所述，上面的代码 LEFT、TOP 和 NEW 中的每个都有增加的行程长度。行程长度值的概率分布大致是负指数分布的。

[0066] 一种增加行程长度的简单方式是要增加说明行程长度的额外的半字节。然而，这将意味着，每个代码将变为 2 个半字节长（第一半字节用于代码而第二半字节用于行程长度）。还意味着，由于，例如后面跟着行程长度为 1 的代码 TOP 的行程长度为 1 的代码 TOP 将与行程长度为 2 的代码 TOP 意义相同，所以该代码中有相当多的冗余。

[0067] 为了防止这些问题，将相同代码的连续出现解释为一个行程。在代码 LEFT、TOP 和 NEW 的每个代码值中，为行程长度保留 1 位，以被称为 L。

[0068] 通过连接每个代码中的行程长度位来形成二进制数，共同解释几个连续的代码 LEFT、TOP 或 NEW。给该数增加取决于连续代码数量的偏移值，以形成实际的行程长度值。

[0069] 结果是，用单个代码半字节能规定长度为 1 或 2 的 TOP、LEFT 或 NEW 行程。用 2 个半字节能规定长度为 3、4、5 或 6 的行程，等等。

[0070] 下表 1 示出所连接的行程长度位的解释。

[0071]

半字节 的编号	所连接的 L 位的值	得到的长度
1	0-1	1-2
2	00-11	3-6
3	000-111	7-14
4	0000-1111	15-30
5	00000-11111	31-62
6	000000-111111	63-126
7	0000000-1111111	127-254
8	00000000-11111111	255-510
9	000000000-111111111	511-1022
10	0000000000-1111111111	1023-2046
11	00000000000-11111111111	2047-4094
12	000000000000-111111111111	4095-8190
13	0000000000000-1111111111111	8191-16382
14	00000000000000-11111111111111	16383-32766
15	000000000000000 - 111111111111111	32767-65534
16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 - 1111111111111111	65535-131070

[0072] 表 1 :所连接的行程长度位的解释

[0073] 在图 3 中，示出了如何解释 NEW、LEFT 和 TOP 代码序列及其行程长度的实例。作为实例，解码十六进制串 EDCCDDBA。

[0074] 下面，描述除了上述 TOP、LEFT 和 NEW 代码之外可以使用的非行程长度码。

[0075] PREV 代码

[0076] 为了允许有效压缩页面上全分辨率二进制采样的图像，增加了产生具有如下所述的 PREV 颜色的一个像素的代码 PREV。

[0077] 每个扫描行的开头，PREV 颜色被设为 K- 黑色 (C = 0，M = 0，Y = 0，K = 255)。

[0078] 每次 LEFT 颜色改变为不同的颜色时，就将 PREV 颜色设为 LEFT 颜色的旧值。

[0079] 图 4 示出了示例性扫描行，该扫描行的前 5 个像素具有黄色，中间 5 个像素具有青色，而后 5 个像素具有品红色。那么在第一像素，PREV 颜色是 K- 黑色而 LEFT 颜色是白色。

在第二直至第 6 像素，PREV 颜色是白色而 LEFT 颜色是黄色。在第 7 像素直至第 11 像素，PREV 颜色是黄色而 LEFT 颜色是青色。在第 12 直至第 16 像素，PREV 颜色是青色而 LEFT 颜色是品红色。

[0080] 如果扫描行上的颜色在两个颜色（例如，黑色和白色）之间交替，则 PREV 代码可被用来在扫描行上为每个颜色转变规定颜色。这个代码产生 1 个像素，所以不需要行程长度。

[0081] TOPL、TOPR 代码

[0082] 代码 TOPL 规定当前像素与其西北方的相邻像素颜色相同。代码 TOPR 规定当前像素与其东北方的相邻像素颜色相同。这些代码产生 1 个像素，所以不需要行程长度。

[0083] 下面的表 2 示出了到目前为止已经定义的代码。编号是任意的并反映代码的最终编号。代码 1 到 7（“0001”到“0111”）仍然可用。

[0084]

代码	名称	说明
0000	PREV	本扫描行上先前的颜色，1 个像素
1000	TOPL	西北方像素的颜色，1 个像素
1001	TOPR	东北方像素的颜色，1 个像素
101L	TOP	上方颜色（N 个像素）
110L	LEFT	左方颜色（N 个像素）
111L	NEW	新颜色值（N 个像素）

[0085] 表 2：针对 2 维行程长度编码所定义的代码

[0086] 用于压缩颜色值的附加代码

[0087] 到目前为止已经定义的代码允许压缩大多数简单页面。然而，包含被采样的图像（例如，照片）的页面包含许多颜色不同于其先前被处理的相邻像素的像素，所以要使用 NEW 代码来表示这些像素，这导致展开而不是压缩。由于被采样的图像中的连续颜色值经常

类似，所以可能压缩那些颜色值。

[0088] 现在将参考图 6 说明一种根据本发明的用于压缩高度关联的颜色值的方法。

[0089] 基本策略由下列步骤组成：

[0090] 1) 基于前面的值预测值。

[0091] 2) 确定预测值与实际值之间的差额。

[0092] 3) 针对经常出现（小差额）的值用短代码来编码该差额，而针对较不经常出现的值用较长的代码来编码该差额。

[0093] 不同的无损图像压缩算法（象 JPEG 无损、JPEG-LS 等等）使用类似的策略。差别大多数在预测器的结构和差值的编码上。

[0094] 由于将被打印的每个页面都必须经过压缩器，所以压缩时间必须短。这意味着，不得不使用能够在通用处理器上有效实现的简单的预测器和差值的简单编码。

[0095] 由于关于附近的像素值编码了考虑中的像素的灰度值，所以不仅访问了当前像素的值（图 6 中的步骤 S1），而且访问了之前已经被处理的预定相邻像素的值（这些像素值在解压步骤中也可用）。在预测步骤中，那些相邻像素值被用来准备预测像素值。

[0096] 预测步骤（图 6 :S2）

[0097] 在文献中已经提出了基于像素 X 的先前被解码的北方、西方和西北方像素值来预测像素 X 的值的各种各样的方法。在图 5 中，示出了 JPEG 无损中所使用的预测器。其它算法使用更复杂的（非线性的）预测器（例如，JPEG-LS）或者当图像正被压缩时而被修改的自适应预测器。

[0098] 由于需要一种快速的算法，所以在本实施方式中使用一种简单的预测器：像素 X 左边的像素的颜色（或者如果 X 是该扫描行上第一像素则是白色）。也可选择更高级的预测器。应根据情况作出选择，因为通常这种预测器将增加压缩系数，但压缩器的执行时间也会增加。

[0099] 确定色差的步骤

[0100] 大多数无损压缩算法是针对灰度图像定义的，而差额是预测值与实际像素值之间的灰度级的差。为了压缩彩色图像，该算法独立地被用于每个颜色通道。这样一种方法的缺点是，由于颜色通道通常强烈相关，所以压缩不是最优的。

[0101] 在根据本发明的编码方法中，与所有颜色通道值一起工作。基本上，在 CMYK 图像的情况下，将确定 4 个颜色差值，并将这些差值组合成一个单个颜色差值。注意，对于具有 N 个颜色通道的图像的压缩来说，要确定 N 个颜色通道差值并将这些差值组合成单个颜色差值。

[0102] 在低油墨覆盖值时，对许多 CMYK 打印机来说，K 通道经常为 0。为了最优使用这一点，已经定义了 2 种模式用于说明色差值：CMY 模式和 CMYK 模式。

[0103] 在 CMYK 模式，在所有 4 个颜色通道上确定色差。

[0104] CMY 模式可以在 K 通道值并不变化时使用。在 CMY 模式，在 3 个通道（K 与所预测的 K 值相同）上确定色差。

[0105] 去相关（图 6 :S3，S4）

[0106] 由于颜色通道的值通常相关，将不确定初始通道值的色差值，但是将首先如下转换这些差值：

[0107] 在 CMYK 模式：

[0108] C1 = M

[0109] C2 = (K-M+128)mod 256

[0110] C3 = (C-M+128)mod 256

[0111] C4 = (Y-M+128)mod 256

[0112] 在 CMY 模式：

[0113] C1 = M

[0114] C2 = (Y-M+128)mod 256

[0115] C3 = (C-M+128)mod 256

[0116] 这个去相关步骤改善了压缩而没有大的性能损失。其它转换也能对颜色通道进行去相关。

[0117] 确定差额

[0118] 将去相关步骤 S3 用于所预测的颜色值 C^*、M^*、Y^*、K^*，并输送已去相关的预测值 $C1^*$、$C2^*$、$C3^*$ 和 $C4^*$。同样，将去相关步骤 S4 用于要被压缩的像素的实际颜色值 C、M、Y、K，并输送已去相关的预测值 C1、C2、C3 和 C4。

[0119] 然后，在最终的已去相关的颜色值之间确定差值 ΔC1、ΔC2、ΔC3 和 ΔC4。如将被理解的那样，在 CMY 模式仅确定三个差值 ΔC1、ΔC2、ΔC3。

[0120] 编码色差的步骤

[0121] 目标是将已去相关的颜色值的通道差 ΔC1、ΔC2、ΔC3（和 ΔC4）编码为通常小的单个组合数 ΔC（在 CMY 模式为 24 位而在 CMYK 模式为 32 位）。然后，通过从这个数去掉先导零半字节能够有效地编码该数。

[0122] 差值的变换（图 6 :S6）

[0123] 通过将每个通道差值 ΔC1、ΔC2、ΔC3 和 ΔC4 转换为相应的字节值 ΔC1'、ΔC2'、ΔC3' 和 ΔC4' 来开始，这些字节值能够表示任何可能的色差值并且对于小色差来说是小值。

[0124] 每个通道的差值可以是正的或者是负的。为了将这些值组合成单个色差值，将这些值映射到正值上，当差额的绝对值小时，该正值小。这可以通过用跳过不可能的差值的这种方式将差值 0、1、-1、2、-2 等等映射到值 0、1、2、3、4、5 等等来完成。例如，当去相关预测器通道 $C1^*$、$C2^*$、$C3^*$ 和 $C4^*$ 中的一个通道的值为 2 时，不映射低于 -2 的负差值 ΔC1、ΔC2、ΔC3 和 ΔC4 以及高于 253 的正差值，因为这些值不可能出现（所有颜色通道值在 0 到 255 的范围内）。

[0125] 这个映射可使用下列公式完成：

[0126] 设 P 为去相关预测器的颜色通道的字节值，而设 A 为去相关实际颜色通道的相对应的颜色通道的字节值。那么下面的 C 语言表达式执行上面所描述的映射：

[0127] (A == P ? 0:(P<128 ?

[0128] (A>2*P ? A:(A>P ? 2*(A-P)-1:2*(P-A))):

[0129] (A<= 2*P-255 ? 255-A:(A>P ? 2*(A-P)-1 : 2*(P-A)))))

[0130] 在实际的压缩器软件中，这些值被预先计算好并被存储在 256×256 字节的查询表中。

[0131] 可替换地，可以使用将差值0、-1、1、-2、2等等映射到值0、1、2、3、4、5等等的映射。也可能是其它类似的映射，例如，基于从实际文件中收集并被存储在256×256的查询表中的统计量的映射。

[0132] 当开始解码时，解压器可以如下反转这个转换。

[0133] 设P为去相关预测器的颜色通道的字节值，而设C是被转换的色差值，则下面的C语言函数可被用来确定已去相关的实际颜色通道值的颜色通道值：

[0134]
```
unsigned char color(unsigned char P,unsigned char C)
```

[0135]
```
{ int M0 = (C&1) ? -1:0;              // 布线 (wiring)
```

[0136]
```
 int M7 = (P&128) ? -1:0;            // 线
```

[0137]
```
 int C0 = M0 ^ (C>>1);              // “异或”
```

[0138]
```
 int P7 = (M7 ^ P)<<1;             // 异或”
```

[0139]
```
 int C7 = M7 ^ C;                 // “异或”
```

[0140]
```
 int PP = P7-C;                   // 加法器
```

[0141]
```
 int PC = P-C0;                   // 加法器
```

[0142]
```
 int res = (PP&256) ? C7:PC; // 多路转换 (mux)
```

[0143]
```
 return res;
```

[0144]
```
}
```

[0145] 这个函数反映了能够用硬件实现转换的方式。

[0146] 通过将该转换用于颜色通道的差额（ΔC1，ΔC2，ΔC3和ΔC4），得到3个（在CMY模式中）或4个（在CMYK模式中）色差字节值ΔC1’、ΔC2’、ΔC3’（和ΔC4’），这些色差字节值说明色差并且通常很小。

[0147] 组合颜色通道（图6:S7）

[0148] 在下一步骤S7中，将转换后的差值ΔC1’、ΔC2’、ΔC3’（和ΔC4’）组合为通常小的单个数ΔC。为了这样，合并单独值的位，如图7A(CMYK)和7B(CMY)中所示。

[0149] 如在图7A中能够看到的那样，在CMYK模式，简单地交织4个色差值ΔC1’、ΔC2’、ΔC3’和ΔC4’的位。在CMY模式（图7B），并不简单地交织每个差值的位。原因是ΔC1’通道的值通常大于其它通道的值。针对一组有代表性的测试图像用实验确定这些位的最优排列次序。在CMYK模式，这也可作为对交织的替换方案来完成。

[0150] 合并的结果是单个24或32位数ΔC，该数ΔC表示色差并且针对小色差来说具有小值。

[0151] 本发明范围内的可替换的方法是使用C维查询表，其中C是颜色通道的数量。该查询表用C个色差值来编索引，而且该表中的每个条目包含唯一的号码，以致经常出现的色差具有小值而很少出现的色差具有较大的值。可以基于从测试图像所收集的统计量来填充该表。

[0152] 编码差值（图6:S8/S9）

[0153] 最后的步骤是通过从ΔC中去除高阶0半字节(S8)并针对剩余值生成代码(S9)来以压缩格式有效地编码数ΔC。在编码步骤S9中，生成一个代码半字节，并将该代码半字节存储在代码块（图2）中，而且生成整数个相关的数据半字节，并将这些数据半字节存储在数据块中。

[0154] 通过使用 6 个代码半字节值以使用 1、2、3、4、5 或 6 个额外的数据半字节来表示色差 ΔC，可以编码色差值并实现压缩。如果，在 CMYK 模式，色差 ΔC 不能使用 6 个数据半字节来表示，则用初始代码 NEW 来表示未被压缩的颜色值（将 8 个数据半字节用于颜色值，即每个颜色通道 2 个数据半字节）。

[0155] 这 6 个新的代码将被称为 NEW1 到 NEW6 并且具有代码值 1 到 6，其中，NEW1 具有 1 个数据半字节而 NEW6 具有 6 个数据半字节，以表示色差值 ΔC。

[0156] 第 7 个自由代码半字节被用于新的代码 NEM6，该新的代码 NEM6 与 NEW6 作用相同但是也在 CMY 与 CMYK 模式之间切换模式。

[0157] 由新代码生成的像素数量

[0158] NEW1 到 NEW6 代码中的每一个和代码 NEM6 能够给每个代码生成 1 个像素。然而，页面上的大多数被采样的图像的分辨率低于设备分辨率。

[0159] PostScript 解释器将最邻近插值法用于按比例放大图像。这导致行中的几个像素具有完全相同的值（参见图 1）。该压缩器可以使用 LEFT 代码来压缩这种情况，但是，如果允许 NEW1 到 NEW6 和 NEM6 代码生成 N 个相同像素的行程，则更有效。每个图像的开头 N 固有地被设为 1。

[0160] 这意味着，需要增加允许改变 N 的代码。这并不经常发生，所以并不需要短代码来改变 N。

[0161] 为了使其成为可能，利用了以下事实，即上面定义的 NEW1 到 NEW6 代码的一些被编码的色差值将不出现。

[0162] 第一半字节 0（代码 NEWi*）：$N \rightarrow 1$

[0163] 永远不会出现的自变量值是具有为零的第一半字节的自变量值（由于在将其编码之前去掉了该值的先导零半字节）。使用那些自变量值如下改变半字节代码值 2 到 6 的意义：

[0164] 如果该自变量值的第一半字节为零，则代码 2 到 6 将被解释为代码 NEW2* 到 NEW6*。这些代码与代码 NEW2 到 NEW6 意义相同，除了这些代码总是只产生 1 个像素而不是 N 个像素（当前的 N 值），并且还将 N 设为 1。

[0165] 代码 INC：$N \rightarrow N+1$

[0166] 永远不会出现的另一种组合是代码半字节 1 与零自变量值，因为这意味着该像素的颜色与其西方的相邻像素的颜色相同，而使用代码 LEFT 来编码那种情况。使用这种组合来定义新的代码 INC，该新的代码 INC 产生颜色与左边像素相同的像素并且 N 的增量为 1。

[0167] 代码 NEW

[0168] 如果不能仅仅用 6 个半字节来表达该颜色，则压缩器不可能总是使用 NEW1* 到 NEW6* 代码之一来将 N 复位为 1。因此，具有大于 1 个新颜色值（即，L>1）的代码 NEW 也将 N 设为 1。

[0169] 下表 3 示出了所有代码半字节及其解释：

[0170]

代码	名称	数据半字节	说明
0000	PREV	无	扫描行上的先前颜色，1 个像素
0001	NEW1	1 个，非零	新颜色值，N 个像素的行程
	INC	1 个，零	左边的颜色，1 个像素，N=N+1
0010	NEW2	2 个，第 1 个非零	新颜色值，N 个像素的行程
	NEW2*	2 个，第 1 个为零	新颜色值，1 个像素，N=1

[0171]

0011	NEW3	3 个，第 1 个非零	新颜色值，N 个像素的行程
	NEW3*	3 个，第 1 个为零	新颜色值，1 个像素，N=1
0100	NEW4	4 个，第 1 个非零	新颜色值，N 个像素的行程
	NEW4*	4 个，第 1 个为零	新颜色值，1 个像素，N=1
0101	NEW5	5 个，第 1 个非零	新颜色值，N 个像素的行程
	NEW5*	5 个，第 1 个为零	新颜色值，1 个像素，N=1
0110	NEW6	6 个，第 1 个非零	新颜色值，N 个像素的行程
	NEW6*	6 个，第 1 个为零	新颜色值，1 个像素，N=1
0111	NEM6	6 个	新颜色值，N 个像素的行程+模式改变（CMY<->CMYK）
1000	TOPL	无	西北方像素的颜色，1 个像素
1001	TOPR	无	东北方像素的颜色，1 个像素
101L	TOP	无	上方颜色。对于 L 位的解释，参见表 1
110L	LEFT	无	左方颜色。对于 L 位的解释，参见表 1
111L	NEW	新颜色的数量*8	新颜色。对于 L 位的解释，参见表 1

[0172] 表 3：代码半字节及其解释的概要

[0173] 根据本发明的压缩方法

[0174] 既然已经定义了被压缩的图像格式，将解释根据本发明的示例性压缩器可用来产生这种格式的方法。参考图 8。

[0175] 像素处理顺序

[0176] 该示例性压缩器起始于图像的第一扫描行来逐个扫描行地处理图像，并且继续直

到已处理所有扫描行，(步骤 S26 到 S29)。对于每个扫描行来说，该压缩器起始于该扫描行上的最左边的像素来处理所有像素，直到该扫描行上的所有像素已被处理。新扫描行一开始，就在初始化步骤 S11 将 PREV 和 LEFT 像素设为“白色”。

[0177] 每个像素的步骤

[0178] 该压缩器试图通过设法将不同代码用于以下面的固定顺序压缩像素来压缩每下一个像素：

[0179] 1) 该像素与 TOP 颜色进行比较 (S12)。这是北方相邻像素的颜色，或者如果当前像素在第一扫描行上，则颜色为白色。如果发现匹配，则该压缩器通过生成一个或多个 TOP 代码来生成 TOP 行程，并通过给这个行程加上所有后面能用 TOP 行程编码的像素来使这个行程的长度最大化 (S13)。然后，该过程从步骤 1 对下一个未被压缩的像素重新开始 (S26，S27)。

[0180] 2) 如果前面的步骤并不允许编码该像素，则该像素与 LEFT 颜色进行比较 (S14)。这是西方相邻像素的颜色，或者如果当前像素在一扫描行的开始处，则颜色为白色。如果发现匹配，则该压缩器通过生成一个或多个 LEFT 代码来生成 LEFT 行程，并通过给这个行程加上所有后面能用 LEFT 行程编码的像素来使这个行程的长度最大化 (S15)。然后，该过程从步骤 1 对下一个未被压缩的像素重新开始 (S26，S27)。

[0181] 3) 如果前面的步骤并不允许编码该像素，则该像素与 PREV 颜色进行比较 (S16)。此前已经解释了 PREV 颜色的值。如果发现匹配，则该压缩器生成 PREV 代码 (S17)。然后，该过程从步骤 1 对下一个未被压缩的像素重新开始 (S26，S27)。

[0182] 4) 如果前面的步骤并不允许编码该像素，则该像素与 TOPL 颜色进行比较 (S18)。这是西北方相邻像素的颜色，或者如果该西北方相邻像素不存在，则颜色为白色。如果发现匹配，则该压缩器生成 TOPL 代码 (S19)。然后，该过程从步骤 1 对下一个未被压缩的像素重新开始 (S26，S27)。

[0183] 5) 如果前面的步骤并不允许编码该像素，则该像素与 TOPR 颜色进行比较 (S20)。这是东北方相邻像素的颜色，或者如果该东北方相邻像素不存在，则颜色为白色。如果发现匹配，则该压缩器生成 TOPR 代码 (S21)。然后，该过程从步骤 1 对下一个未被压缩的像素重新开始 (S26，S27)。

[0184] 6) 如果前面的步骤并不允许编码该像素，则应用此前在“用于压缩颜色值的附加代码”部分中所描述的用于压缩颜色值的方法 (S22)。如果 K 通道值与所预测的 K 通道值相同，则 CMY 模式被用来压缩该颜色，否则就使用 CMYK 模式。然后检查 (S23) 是否产生可接受的 NEWi 代码，即，模式不变的 6 个或更少个半字节的色差值。如果是这样，则与色差半字节一起生成 NEW1 到 NEW6 代码中的适当的一个代码 (S24)。如果不是这种情况，则生成代码 NEW，而新的颜色值不被压缩地被写到数据块中 (S25)。

[0185] 如果前面的模式不同于被用来压缩该像素的模式，并且如果该色差值的长度为 6 个半字节或更少，则与该色差半字节一起生成代码 NEM6。如果该色差值比 6 个半字节长，则生成代码 NEW，并且新的颜色值未被压缩地被写到数据块。这些步骤没有在图 8 中明确地示出，但是这些步骤可被认为形成步骤 S24 和 S25 的部分。

[0186] 如果 N 的值大于具有那个颜色的连续像素的数量，而色差值的长度为 5 个半字节或更少，则与之前加上一空的半字节的色差半字节一起生成 NEW2* 到 NEW6* 代码中的适当的

一个代码。如果这不可能，则为当前像素和下一个像素生成代码 NEW，并将当前像素和下一像素的颜色值未被压缩地写到数据块。这些步骤没有在图 8 中明确地示出，但是可被认为形成步骤 S24 和 S25 的部分。

[0187] 然后，该过程从步骤 S11 对下一个未被压缩的像素重新开始 (S26，S27)。

[0188] 行程长度至少为 2 的代码 NEW 使得 N 被复位为 1。

[0189] 为了该压缩器在适当的时候增加 N，每次 LEFT 行程跟随在范围 1 到 6 中的代码（表示新的颜色），该压缩器都存储其状态（除非一状态之前已经被存储了少于 32 个像素）。从那一点开始，记住在扫描行上具有相同的颜色的最少 M 个连续像素。

[0190] 如果压缩器确定增加 N 将允许更好的压缩，则压缩器将其状态恢复到先前存储的状态（从而放弃存储那一状态之后它生成的任何输出）并且通过生成一个或多个 INC 代码来对 M 的值增加 N。接着，压缩在用所存储的状态表示的点重新开始。

[0191] 本发明范围内的可能的简化的实例

[0192] 压缩步骤 3、4 和 5 可以以不同的顺序完成，而没有显著的压缩损耗。

[0193] 压缩步骤 3、4 和 5 可以不考虑压缩器，而对大多数图像来说没有显著的压缩损耗。然后，可以从该压缩格式中去除相对应的代码 PREV、TOPL 和 TOPR。

[0194] 压缩步骤 1 和 2 可被交换，而压缩没有显著的变化。

[0195] 不是具有 CMYK 和 CMY 模式，而是只有 CMYK 模式工作，虽然有一些压缩损耗。利用一种模式，可以从压缩格式中去除代码 NEM6。

[0196] N 可以被固定为 1，虽然有一些压缩损耗。然后，可以从压缩格式中去除代码 INC 和 NEW2* 到 NEW5*。

[0197] 代码 NEW 的行程长度降低了压缩格式的最大展开。如果不需要最大展开，则代码 NEW 不需要行程长度。

[0198] 根据本发明的解压方法

[0199] 对于每个连续的扫描行，从图像的第一扫描行开始并且继续到最后一个扫描行，解压器处理来自代码块的代码半字节，并从数据块中取出相对应的数据半字节。从最左边的像素开始给扫描行上的连续像素分配解压缩后的颜色值。

[0200] 实际上，给每个像素一次分配一颜色。

[0201] - 当该解压器遇到代码 PREV 时，该解压器给当前像素分配（该解压器记住的）PREV 颜色。上文解释了 PREV 颜色的值。

[0202] - 当该解压器遇到代码值 1（十六进制为 0001）到 6（十六进制为 0110）之一时，该解压器检查数据块中的下一个数据半字节的值。

[0203] 如果该数据半字节为非零，则该代码被解释为代码 NEW1 到 NEW6 之一。在那种情况下，通过反转压缩器所应用的转换步骤，该解压器根据当前模式结合数据块中的差值存储而根据预测器的颜色值重建下一个像素的颜色值。这个颜色值随后被分配给从当前的像素开始的 N 个连续像素。

[0204] 预测器的颜色值是扫描行上的先前被解压的像素的颜色，或者如果当前像素是该扫描行上的第一个，则为白色。

[0205] 如果该数据半字节为零，而该代码值大于 1，则该代码被解释为代码 NEW2* 到 NEW6* 之一。在那种情况下，通过反转压缩器所应用的转换步骤，该解压器根据当前模式结合数据

块中的差值存储而根据预测器的颜色值重建颜色值。这个颜色值随后被分配给当前像素。另外，该解压器将 N 设为 1。

[0206] 如果该数据半字节为零，而该代码值为 1，则该代码被解释为代码 INC。在那种情况下，该解压器将先前被解压的像素的颜色（或者如果当前像素是该扫描行上的第一像素，则颜色为白色）分配给当前像素。另外，该解压器将 N 增加 1。

[0207] - 当该解压器遇到代码 NEM6（代码值 0111）时，通过反转压缩器所应用的转换步骤，该解压器切换当前模式，并根据当前模式结合数据块中的差值存储而根据预测器的颜色值重建颜色值。这个颜色值随后被分配给从当前的像素开始的 N 个连续像素。

[0208] - 当该解压器遇到代码 TOPL（代码值 1000）时，该解压器复制来自先前被解压的西北方相邻像素的颜色，并将该颜色分配给当前像素。如果没有这样的相邻像素，则该解压器给当前像素分配白色。

[0209] - 当该解压器遇到代码 TOPR（代码值 1001）时，该解压器复制来自先前被解压的东北方相邻像素的颜色，并将该颜色分配给当前像素。如果没有这样的相邻像素，则该解压器给当前像素分配白色。

[0210] - 当该解压器遇到代码 TOP（代码值 101L）时，该解压器收集来自数据块的所有紧随的 TOP 代码，并利用上面描述的方法确定行程长度 R。然后，给从当前像素开始的 R 个像素的每一个分配来自其北方的相邻像素的颜色。如果该解压器正在第一扫描行上进行解压，则所有 R 个像素都分配为白色。

[0211] - 当该解压器遇到代码 LEFT（代码值 110L）时，该解压器收集来自代码块的所有紧随的 LEFT 代码并利用上面描述的方法确定行程长度 R。然后，给从当前像素开始的 R 个像素分配来自其左边的相邻像素的颜色。如果当前像素是扫描行上的第一像素，则给所有 R 个像素都分配白色。

[0212] - 当该解压器遇到代码 NEW（代码值 111L）时，该解压器收集来自代码块的所有紧随的 NEW 代码，并利用上面描述的方法确定行程长度 R。然后，该解压器从该数据块中取出 R 个未被压缩的颜色值，并将这些颜色值分配给从当前像素开始的随后的 R 个像素。如果行程长度 R 大于 1，则该解压器将 N 设为 1。

[0213] 压缩方法的评估

[0214] 为了评估压缩方法的性能，使用压缩器来压缩一系列图像，并将结果（压缩时间和压缩系数）与用 JPEG-LS 压缩相同图像的结果进行比较。也将压缩时间与采用 Adobe PostScript 解释器生成这些页面的时间进行比较。

[0215] 所有测试是在 2.4GHz 的 Pentium4 上完成的。在 AdobePostScript 解释器上运行 PostScript 测试文件，并压缩得到的 600dpi 连续色调 CMYK 位图。该位图的大小是 A4(4958×7040 像素)，得到未被压缩的大小为 136345 千字节。在 PostScript 解释器上，DeviceRGB 的解释被设为 sRGB，而 DeviceCMYK 的解释被设为欧洲地区墨水标准(Euroscale)。使用了欧洲地区墨水标准输出简档。

[0216] 所用的 PostScript 测试文件包含复杂但逼真的页面描述。已经包括了特别构造的 PostScript 文件 (chrisA4.ps)，该 PostScript 文件包含带有许多非常精细的细节（像片镶嵌页面，其包含由许多非常小的图像组成的页面大小的图像）的极为复杂的页面。这不是非常逼真的页面，但该页面被用来探究可压缩性的边界。

[0217] 为了将新压缩器的压缩性能与无损图像压缩中的现有技术进行比较，将本发明的结果与 JPEG-LS 产生的那些结果进行比较。

[0218] JPEG-LS 基于 HP 的 LOCO-1 算法，并且已经由 ISO/ITU-T 选为连续色调图像的无损压缩的标准。在 M.Weinberger、G.Seroussi、G.Saipiro 的“LOCO-1 :A Low Complexity, Context-Based, Lossless Image Compression Algorithm”中给出了该算法的说明，该文献可在 www.hpl.hp.com/loco/HPL-98-193R1.pdf 上得到。

[0219] 与其它无损连续色调图像压缩算法（无损 JPEG、CALIC、FELICS）相比，JPEG-LS 相对简单而且快速，并且 JPEG-LS 产生更好的压缩。JPEG-LS 的压缩性能比 JPEG2000 无损（参见 Diego Santa Cruz、Touradj Ebrahimi 的“An analitical study of JPEG 2000functionalities(JPEG 2000 功能性的分析性研究）”，其可在 www.jpeg.org/public/wgln1815.pdf 上得到）稍好一些。

[0220] 已经为 Adobe photoshop 选择了 JPEG-LS 插件程序来测试 JPEG-LS。这是报告页面需要的压缩时间的有效实现方案。该 JPEG-LS 插件程序可在 http://www.hpl.hp.com/loco 上得到。注意，JPEG-LS 已经被设计为通用的无损连续色调图像压缩技术，不专用于压缩 PDL 生成的连续色调图像。

[0221] 在下面的表 4 中呈现了详细的测量结果。

[0222]

	PS-RIP	本发明的压缩		JPEG-LS	
文件	时间（秒）	时间（秒）	大小（千字节）	时间（秒）	大小（千字节）
breakfast.ps	3.51	0.61	17712	11.69	28233
chqf.ps	1.18	0.19	3711	5.01	6678
chrisA4.ps	8.79	2.22	80413	17.42	67916
gatf1.ps	1.69	0.13	1002	4.35	4118
gatf2.ps	1.34	0.31	6802	6.47	15657
gatf3.ps	1.09	0.34	7284	6.77	14979
gatf4.ps	0.94	0.27	6283	6.31	13713
oilplatform.ps	3.70	0.63	17857	11.05	25637
model-puzzle.ps	6.70	1.44	43832	11.12	35080
printwork1.ps	1.89	0.11	434	2.82	1870
printwork2.ps	2.80	0.09	122	4.40	1123
printwork4.ps	3.27	0.56	17317	7.46	18944
printwork5.ps	3.59	0.98	31423	10.85	32585
printwork6.ps	1.97	0.58	19396	8.00	21758
平均:	3.03	0.60	18113	8.12	20592

[0223] 表 4 :测量结果

[0224] 第一列列出了测试页的名称。名称没有特殊的意义,除了这些名

[0225] 称正好是文件以其被存储的名称。第二列表示现有技术的 AdobePostScript 解释器将页面的 PostScript 描述转换为要被压缩的图像的执行时间。第三列表示新的压缩方

法的执行时间，而第四列表示得到的被压缩数据量。第五列表示 JPEG-LS 压缩器的执行时间，而第六列表示得到的被压缩数据量。该表的最后一行表示所有测试图像的平均值。

[0226] 新方法的平均压缩时间为 0.6 秒，该平均压缩时间是 PostScript 解释器生成页面的执行时间的大约 20％。JPEG-LS 的平均压缩时间是 8.12 秒，该平均压缩时间比新压缩方法慢大约 13.5 倍。从这些结果中也清楚，由于与 PDL 解释时间相比太慢，所以 JPEG-LS 并没有满足我们的要求。

[0227] 通过新压缩所产生的被压缩数据量平均小于 JPEG-LS 所产生的量。对于包含分辨率等于设备分辨率的样本图像的一些非常复杂的页面（“chrisA4.ps”和“model-puzzle.ps”）来说，JPEG-LS 比新方法产生更少的被压缩数据，但是这种页面在实践中很少。

[0228] 用硬件实现

[0229] 该压缩格式有允许用硬件简单地实现解压器的若干属性。

[0230] 当长度为 3 的 NEW 代码后面直接跟着 NEW6 或者 NEM6 代码时，该格式的最大局部展开发生。该 NEW 代码占用 2 个代码半字节 +3*8 个数据半字节。该 NEW6 或者 NEM6 代码占用 1 个代码半字节 +6 个数据半字节。这意味着，在这种情况下产生 4 个 CMYK 像素值总共需要 33 个半字节，因此最坏情况的局部展开为 1/32 ＝ 3.1％。这意味着，必须处理未被压缩的数据的速度最多比产生未被压缩像素的速度高 3.1％。

[0231] 已经以每个代码产生至少一个像素的这种方式构造了该压缩格式。

[0232] 对于需要整体解释连续出现的代码的代码 LEFT、TOP 和 NEW，仅仅特殊的情况发生。当已经解释了这些代码之一的第一代码半字节时，明确了连续颜色值的来源：

[0233] - 对于代码 LEFT：所有代码与左边颜色相同

[0234] - 对于代码 TOP：所有代码来自目标像素之上的像素

[0235] - 对于代码 NEW：所有代码来自被压缩的数据流。

[0236] 在那一点，可以开始将颜色转移到新像素，并且可以与代码解释并行地进行。代码解释器可进行组合连续代码，直到遇到不同的代码。然后知道了有多少像素必须被生成。由于由这些代码所产生的像素的数量增加得比需要编码这个数量的半字节的数量增加得快，所以该代码解释器将总是在已经转移了那个数量的新颜色值之前，结束确定像素数量。因此，在已经产生正确数量的像素之后，该代码解释器停止将颜色值转移到新像素。

[0237] 现在将描述本发明的第二实施方式，该第二实施方式是用于连续色调黑白（灰度值）图像的压缩方法。

[0238] 在第二实施方式中所使用的机制与上述第一实施方式的那些机制相同，即：

[0239] - 一行接一行地完成压缩。对于每一行来说，被压缩的数据由 2 个部分组成，这两个部分即“数据块”和“代码块”。

[0240] - 对于每个像素或者每组相同像素来说，在代码块中生成了代码或一系列代码。对于某些代码（NEWi 代码）来说，在数据块中存储的附加数据是必要的。在每个扫描行的末端，把数据块和代码块放在一起，以生成当前行的被压缩的数据。

[0241] 第二实施方式使用减少的代码组并且是面向位的（与面向半字节的第一实施方式相对）。更特别地，代码总是长 3 个位，但数据块中的相关联的部分具有可变的长度（位的数量）。在第二实施方式中所使用的代码如下被定义。

[0242] - 代码 TOP、LEFT 和 PREV 具有与第一实施方式中的意义相同的意义：这些代码分

别将像素的行程编码为与上方、左边像素相同的颜色或先前颜色。

[0243] - 代码 NEW 编码新的颜色；尽管行程长度定义机制不同于第一实施方式的行程长度定义机制，这个代码仍能够定义相同像素的行程，如稍后将解释的那样。

[0244] - 代码 NEW1 到 NEW4 和 NEW3* 以及 NEW4* 分别编码灰度值与预测值（在本实例中为左边的像素）的灰度值相差 1 到 4 个位或者 2 个或 3 个位的量的像素或者相同像素的行程（一次转换为正整数）。

[0245] 代码字长 3 个位，而且可以采用下表 5 中所列出的任意值。

[0246]

代码	名称
000	NEW1/PREV
001	NEW2/NEW
010	NEW3/NEW3*
011	NEW4/NEW4*/INC
10L	LEFT
11L	TOP

[0247] 表 5：代码概要

[0248] 代码的定义如下：

[0249] -LEFT 和 TOP 代码：没有相关联的数据部分。一起解释几个相同的代码，而“L”位被用来以与第一实施方式几乎相同的方式编码相对应像素的数。

[0250] -PREV 代码：没有相关联的数据部分。该 PREV 代码生成具有当前先前颜色的像素。

[0251] -NEW1/NEW2/NEW3/NEW4 代码：当要被编码的当前像素不能由 LEFT、TOP 或者 PREV 代码表达时，计算当前像素和左边像素的颜色值之间的差额，并随后（通过与第一实施方式中相同的机制）将该差额转换为正整数。如果这个差额（8 位整数）在截断先导零位之后能够适合 1、2、3、或者 4 位，则选择相对应的 NEWi 代码。另外，形成相关联的数据部分，该相关联的数据部分对应于该差额并因此分别长 1、2、3 或 4 个位。代码 NEW1 到 NEW4 产生像素的行程，在处理文件期间维持该像素行程的长度（称为 N）。

[0252] -INC/NEW3*/NEW4* 代码：当前行程长度值 N 开始被设为 1，并且利用 INC 代码来增加 1，该 INC 代码也产生左边颜色的像素。当该值太高时（即，当一组 N 个像素不适合输入数据时），N 可被复位为 1：为此，如以初始格式使用 NEWi* 代码。由于 NEWi* 代码具有以 0 开始的相关联的数据部分作为 MSB（最高有效位），所以解码器将识别出这些代码。

[0253] -NEW 代码：当不能利用先前的代码之一来编码新的颜色值时，使用 NEW 代码。该 NEW 代码的数据部分包含新的灰度值（8 位），但是也可以如下面将说明的那样包含行程长度信息。最后，连续的至少两个 NEW 代码以与第一实施方式中的方式相同的方式将 N 的值复位为 1。

[0254] 这些代码利用其数据部分来区分：

[0255] 代码 000：NEW1/PREV

[0256] 与这个代码相关联的数据长 1 个位。这意味着，当前像素与其左边像素之间的差额仅用 1 个位来表达。数据位被设为 0 的情况是不可能的，因为这将意味着两个像素之间没有差别。从而，这种可能性被用来编码 PREV 代码。

[0257] 代码 001：NEW2/NEW

[0258] 与这个代码相关联的数据长 2 个位。这意味着，当前像素与其左边像素之间的差额可用 2 个位表达。数据部分的 MSB（最高有效位）为 0 的情况被用于 NEW 代码。在那种情况下，2 位长的数据部分的 LSB（最低有效位）是被用来以与 LEFT 和 TOP 代码中的 L 位几乎相同的方式（尽管在那些代码中，该 L 位位于代码本身中）将若干个 NEW 代码聚集在一起的行程长度位。在被聚集在一起的若干个 NEW 代码的情况下，聚集所有 NEW 代码的这 2 个位的数据部分，并将这些数据部分置于数据块中，以及然后将所有 NEW 颜色值（每个 8 位）置入数据块中。

[0259] 然而，注意，取决于该 NEW 颜色值，跟随 NEW 代码的代码 NEW2 可能导致被解码器误解。因此，防止代码 NEW2 跟随任何 NEW 代码。

[0260] 代码 010 :NEW3/NEW3*

[0261] 这个代码的相关联的数据部分长 3 个位。当在当前像素与其左边像素之间的差额用 3 个位来表达时，使用这个代码。NEW3* 代码是具有其 MSB 被设为 0 的数据部分的 NEW3 代码。该 NEW3* 代码被用来将 N 值复位为 1，并利用这 2 个附加数据位编码差额。由于在这种实施方式中，NEW2* 代码不可用，该 NEW3* 也被用于能够用 1 个位表达的差额。

[0262] 代码 011 :NEW4/NEW4*/INC

[0263] 这个代码的相关联的数据部分长 4 个位。当在当前像素与其左边像素之间的差额用 4 个位来表达时，使用这个代码。NEW4* 代码是具有其 MSB 被设为 0 的数据部分的 NEW4 代码。该 NEW4* 代码被用来将 N 值复位为 1，并用这 3 个附加数据位来编码差额。INC 代码是具有等于 0（所有 4 个位）的数据部分的 NEW4 代码。该 INC 代码被用来将 N 值增加 1，并生成颜色与左边像素的颜色相同的像素。

[0264] 代码 10L :LEFT

[0265] 没有与这个代码相关联的数据。当要编码的像素与其左边像素颜色相同时，使用该代码。“L”位是被用来将若干个 LEFT 代码聚集在一起的行程长度位（与第一实施方式中的机制相同）。

[0266] 代码 11L :TOP

[0267] 没有与这个代码相关联的数据。当要编码的像素与其上方像素颜色相同时，使用该代码。“L”位是被用来将若干个 TOP 代码聚集在一起的行程长度位（与第一实施方式中的机制相同）。

[0268] 图 9 示出了根据本发明的第二实施方式的通用编码过程的流程图。尽管由于使用较少的代码而丢失了某些步骤，这个过程还是很大程度上与参考图 8 所描述的第一实施方式相类似。由于在上文中对代码说明的基础上很容易理解，所以这里省略了图 9 的完整说明。

[0269] 该过程的单独步骤已经根据图 8 中的相对应步骤被编号。已经给图 9 中的参考编号设置了撇号。

[0270] 图 10 描述了根据第二实施方式形成过程的基本 NEWi 代码，如图 9 中用参考编号 S22’ 指出的那样。由于这个过程仅在一个颜色通道上工作，所以第一实施方式的若干步骤是多余的。剩下的那些步骤已经用如图 6 中的相对应的参考编号来进行编号（设置了撇号），并在下面列出。对于每个步骤的详细解释，读者可以查阅上文中图 6 的说明以及对第二实施方式的代码的说明。

[0271] 在步骤 S1’中访问考虑中的像素实际值，而在步骤 S2’中根据相邻像素（例如，左边的像素）确定预测值。然后，在步骤 S5’中确定实际值与预测值的差额，而在步骤 S6’中将这个值转换为针对小差额小而针对较大差额较大的正整数。实际的转换可以基于可能在用通用的或专用的转换数据组之前已经填充好了的查询表 (LUT) 的使用。

[0272] 然后，通过切断先导零位来将得到的差值截断 (S8’)，而且在步骤 S9’中将剩余的非零位编码到 NEWi 代码中，如上文中所述的那样。

[0273] 上面已经详细描述了其两种实施方式的本发明为无损压缩提供了新的压缩方法和格式，并为（优选地）PDL 生成的连续色调页面图像数据提供了解压缩方法。

[0274] 与象 JPEG-LS 那样的现有技术无损图像压缩算法相比，如针对一组有代表性的测试文件所测量的那样，该新方法平均更快。这允许在与被用来运行 PDL 解释器相同的计算机上实现压缩，而无需额外的硬件并且没有大的性能损失。

[0275] 本发明是如此描述的，显而易见可以多种方式改变。对本领域的技术人员来说是显而易见的这种变型和修改意图被包括在随后的权利要求的范围内。

[0276] 例如，尽管该方法针对 PDL 生成的图像被优化，该方法也可被用来压缩来自其它来源的图像。

[0277] 此外，上面的说明集中于压缩每个颜色通道 8 位的 CMYK 彩色图像和 8 位连续色调黑白图像。但是本领域的任何技术人员将意识到，可以修改该方法来压缩多种其它的连续色调图像，诸如 RGB 连续色调彩色图像、每个颜色值超过 4 个通道的连续色调彩色图像或者针对颜色通道的精度不同于 8 位的图像，或者其它连续色调单色（灰阶）图像。

[0278] 同样地，可以用其它的、类似的代码代替如第一实施方式和第二实施方式中所描述的实际代码形式，而且第一实施方式的那些代码也可被用于第二实施方式中，并且反之亦然。

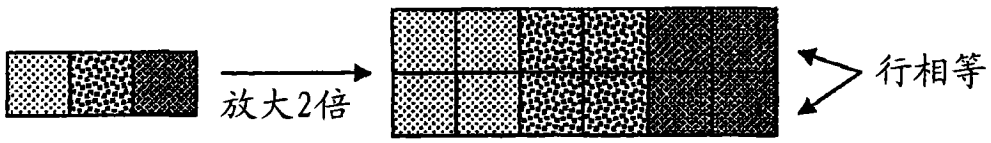

放大2倍
行相等

图 1

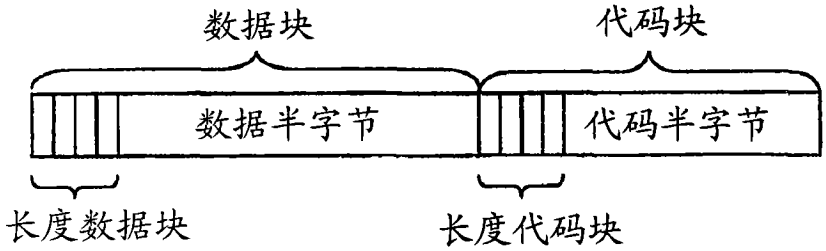

数据块
代码块
数据半字节
代码半字节
长度数据块
长度代码块

图 2

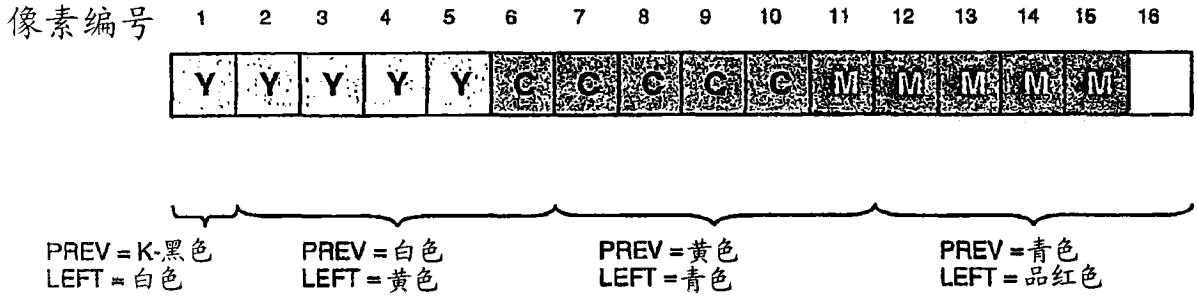

像素编号
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Y Y Y Y Y C C C C C M M M M M
PREV = K-黑色
LEFT = 白色
PREV = 白色
LEFT = 黄色
PREV = 黄色
LEFT = 青色
PREV = 青色
LEFT = 品红色

图 4

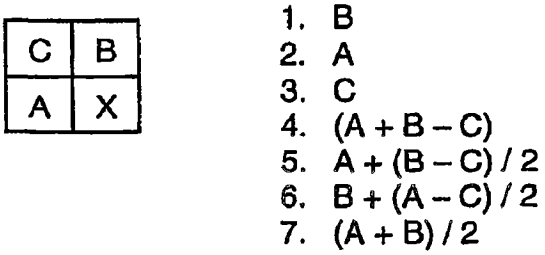

C B
A X
1. B
2. A
3. C
4. (A + B – C)
5. A + (B – C) / 2
6. B + (A – C) / 2
7. (A + B) / 2

图 5

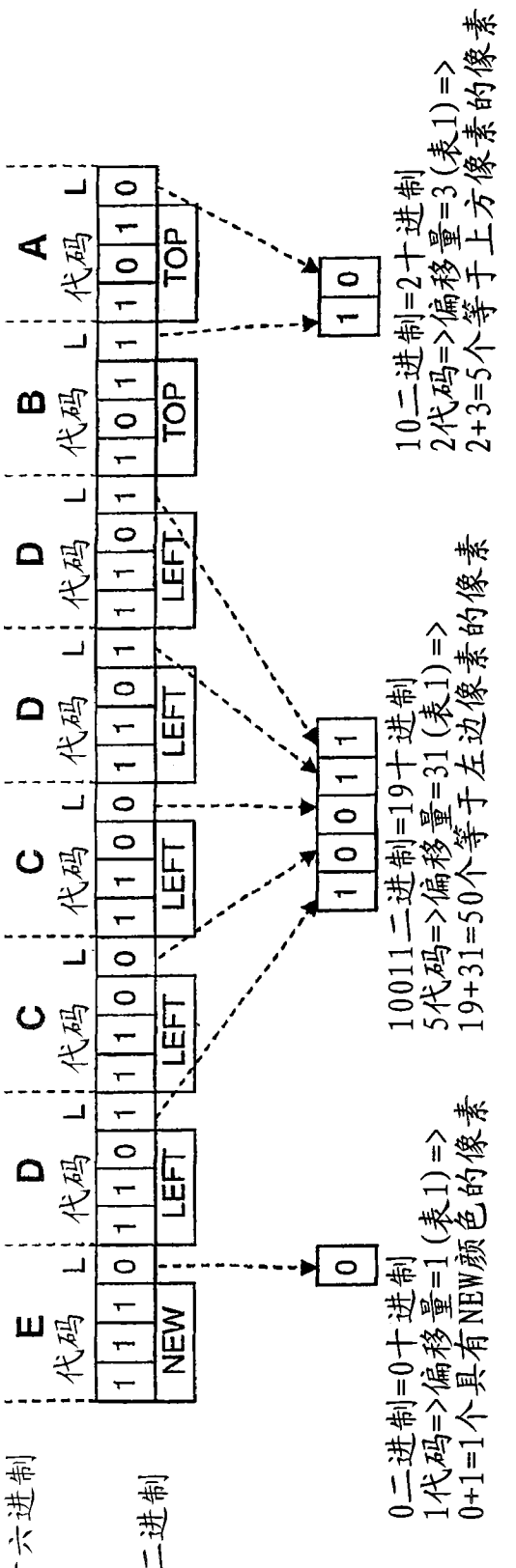

十六进制
二进制
E 代码 L
D 代码 L
C 代码 L
D 代码 L
B 代码 L
A 代码 L
NEW
LEFT
LEFT
LEFT
LEFT
TOP
TOP
0二进制=0十进制
1代码=>偏移量=1(表1)=>
0+1=1个具有NEW颜色的像素
10011二进制=19十进制
5代码=>偏移量=31(表1)=>
19+31=50个等于左边像素的像素
10二进制=2十进制
2代码=>偏移量=3(表1)=>
2+3=5个等于上方像素的像素

图 3

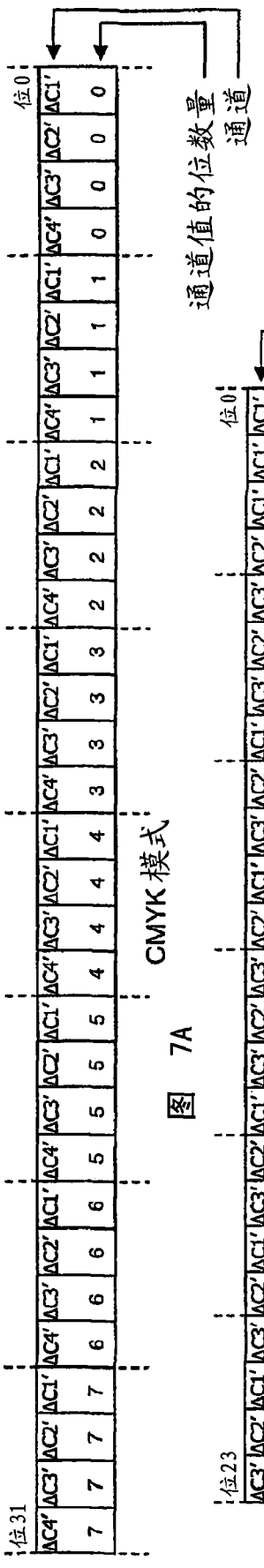

位31
位0
CMYK模式
通道值的位数量
通道

图 7A

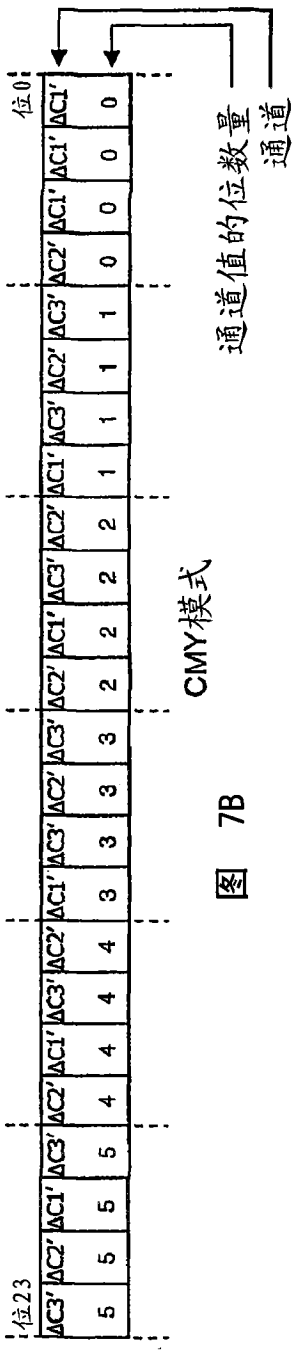

位23
位0
CMY模式
通道值的位数量
通道

图 7B

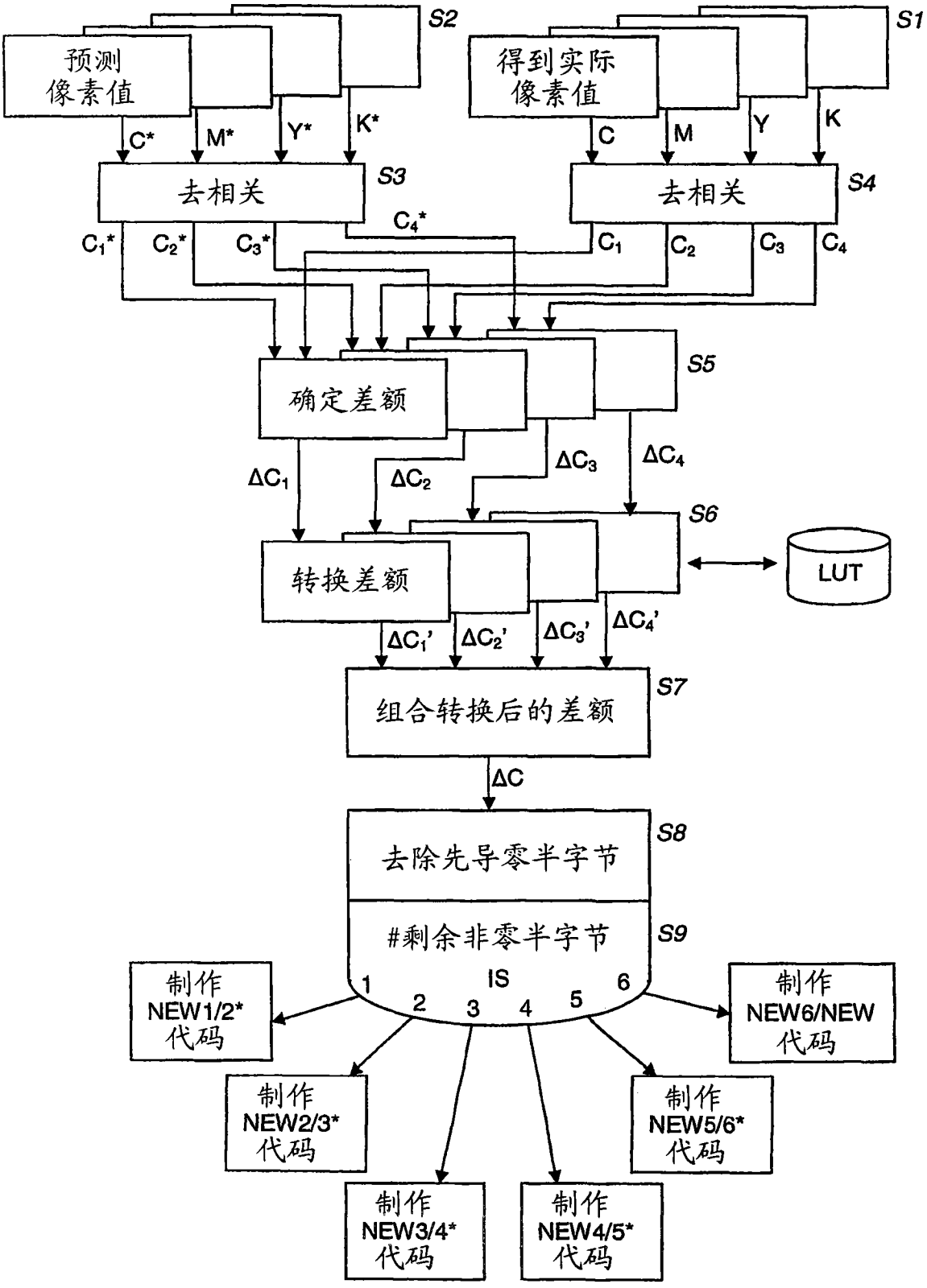

S2
预测
像素值
C*
M*
Y*
K*
S1
得到实际
像素值
C
M
Y
K
S3
去相关
S4
去相关
C4*
C1*
C2*
C3*
C1
C2
C3
C4
S5
确定差额
ΔC1
ΔC2
ΔC3
ΔC4
S6
转换差额
LUT
ΔC1'
ΔC2'
ΔC3'
ΔC4'
S7
组合转换后的差额
ΔC
S8
去除先导零半字节
S9
#剩余非零半字节
IS
1
2
3
4
5
6
制作
NEW1/2*
代码
制作
NEW6/NEW
代码
制作
NEW2/3*
代码
制作
NEW5/6*
代码
制作
NEW3/4*
代码
制作
NEW4/5*
代码

图 6

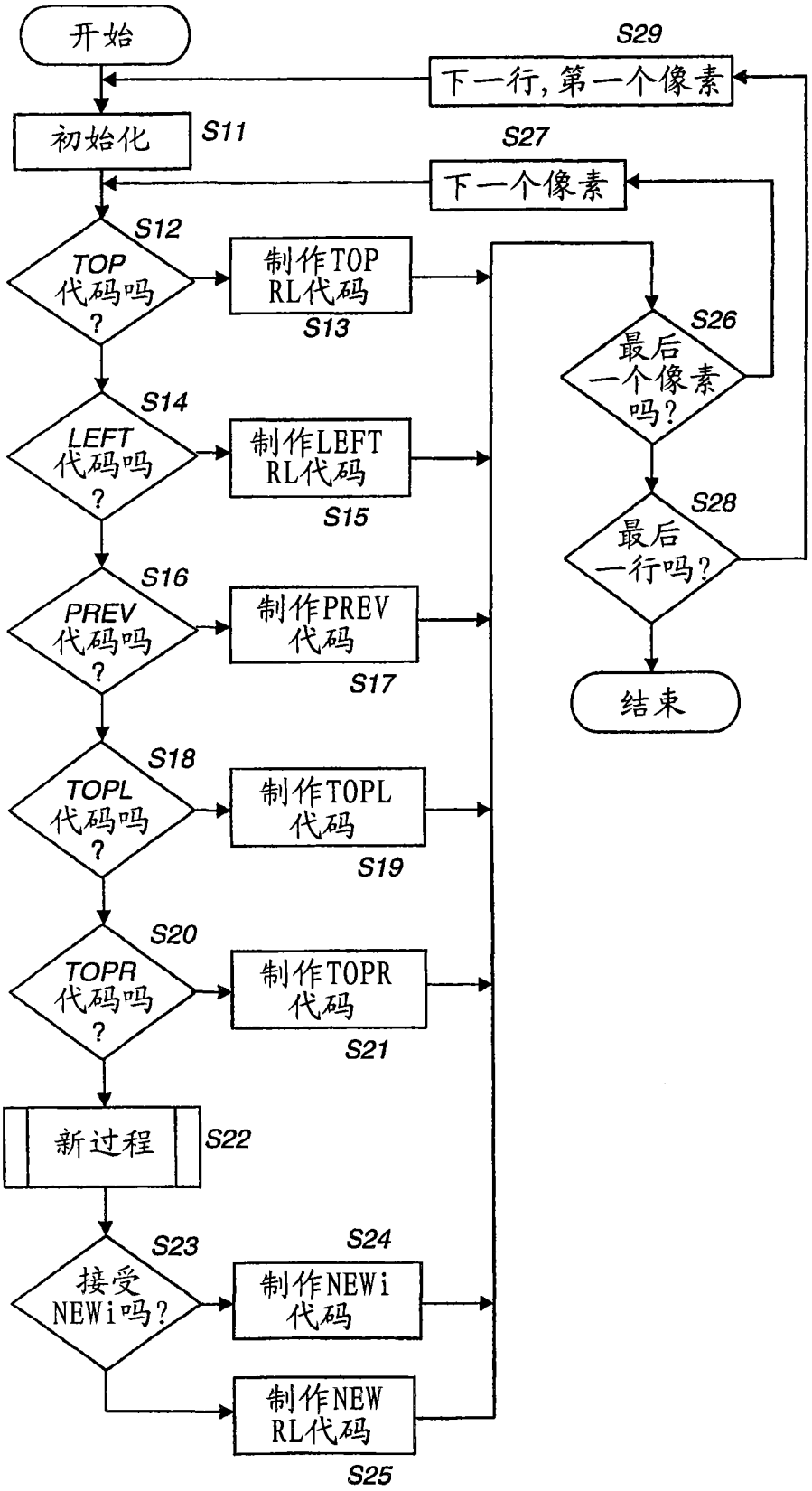

开始
初始化
S11
S12
TOP 代码吗 ?
制作TOP RL代码
S13
S14
LEFT 代码吗 ?
制作LEFT RL代码
S15
S16
PREV 代码吗 ?
制作PREV 代码
S17
S18
TOPL 代码吗 ?
制作TOPL 代码
S19
S20
TOPR 代码吗 ?
制作TOPR 代码
S21
新过程
S22
S23
接受 NEWi吗?
S24
制作NEWi 代码
制作NEW RL代码
S25
S26
最后 一个像素 吗?
S27
下一个像素
S28
最后 一行吗?
S29
下一行,第一个像素
结束

图 8

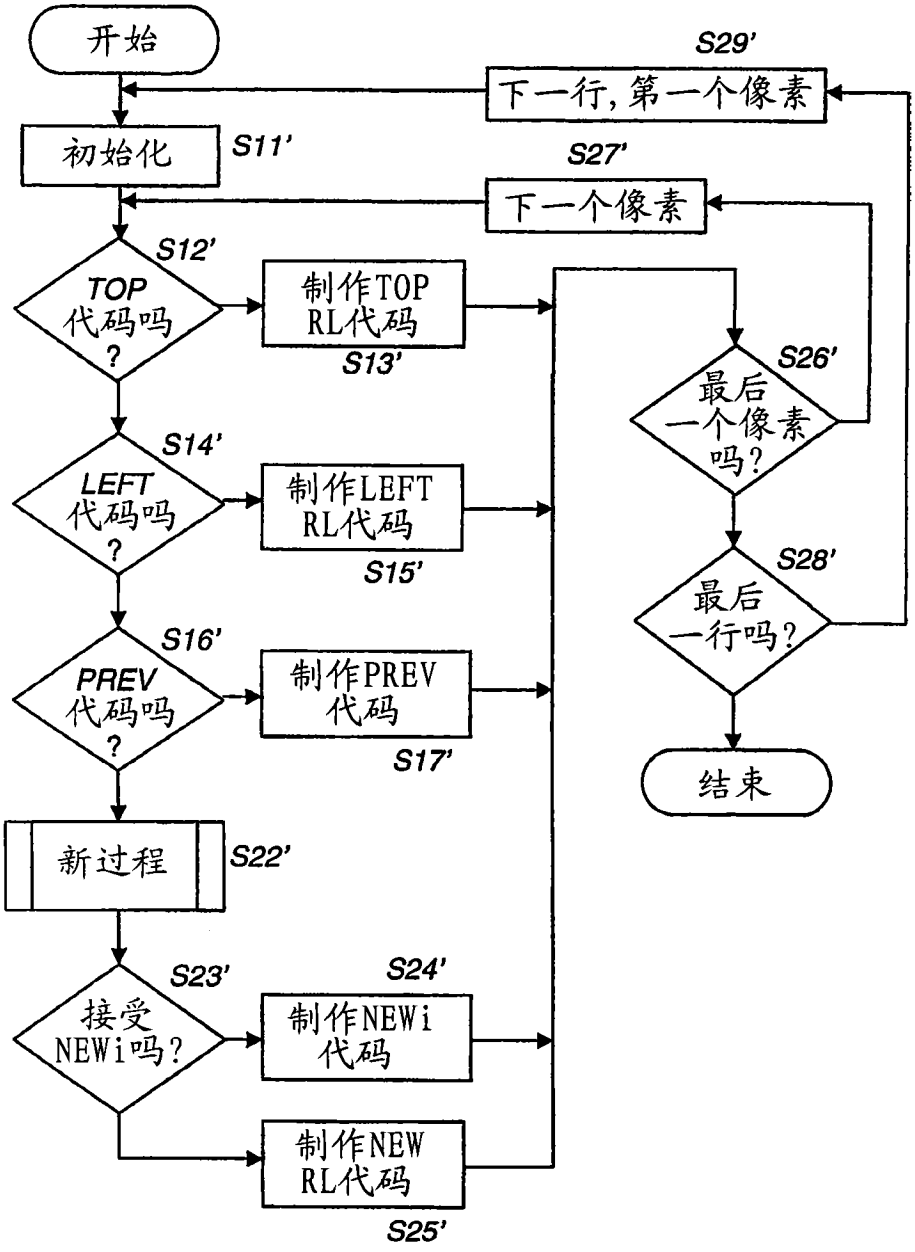

开始
初始化
S11'
S12'
TOP 代码吗 ?
制作TOP RL代码
S13'
S14'
LEFT 代码吗 ?
制作LEFT RL代码
S15'
S16'
PREV 代码吗 ?
制作PREV 代码
S17'
新过程
S22'
S23'
接受 NEWi吗?
S24'
制作NEWi 代码
制作NEW RL代码
S25'
S29'
下一行,第一个像素
S27'
下一个像素
S26'
最后 一个像素 吗?
S28'
最后 一行吗?
结束

图 9

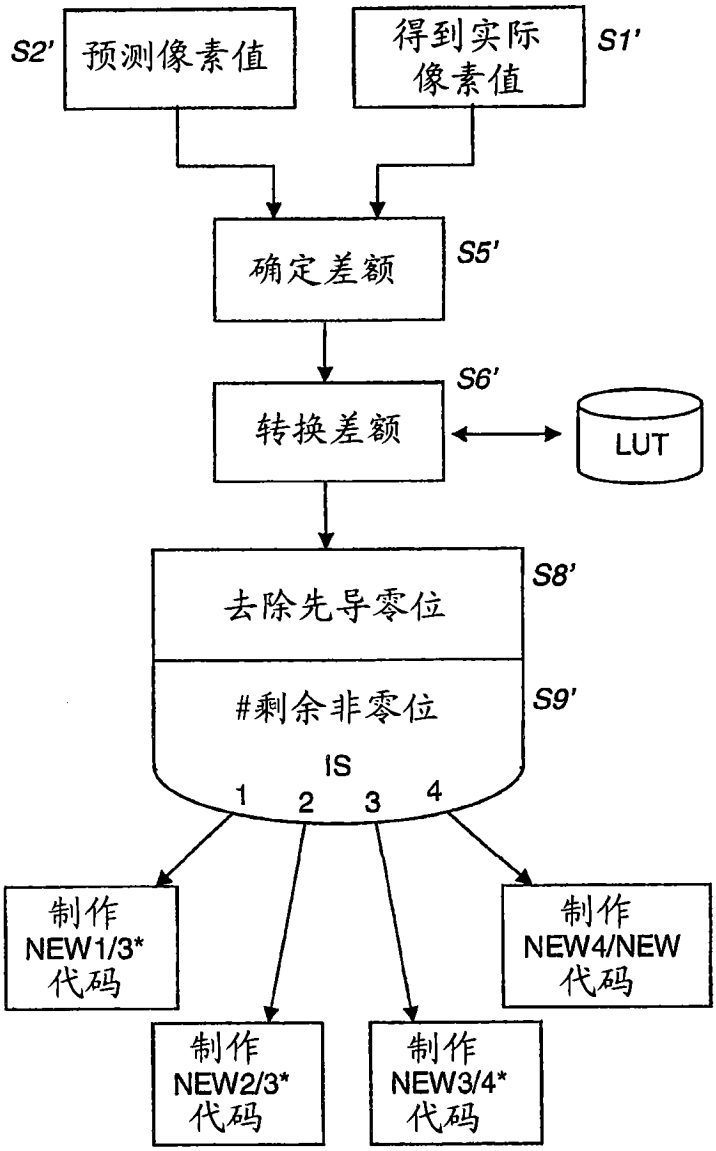

S2'
预测像素值
得到实际像素值
S1'
确定差额
S5'
转换差额
S6'
LUT
去除先导零位
S8'
#剩余非零位
S9'
IS
1
2
3
4
制作 NEW1/3* 代码
制作 NEW2/3* 代码
制作 NEW3/4* 代码
制作 NEW4/NEW 代码

图 10

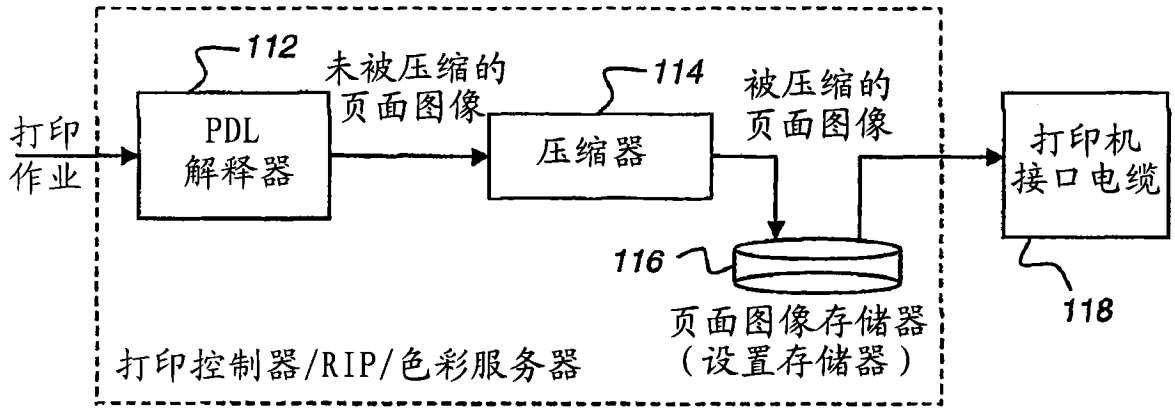

112
未被压缩的页面图像
114
被压缩的页面图像
打印作业
PDL 解释器
压缩器
打印机接口电缆
116
118
页面图像存储器（设置存储器）
打印控制器/RIP/色彩服务器

图 11

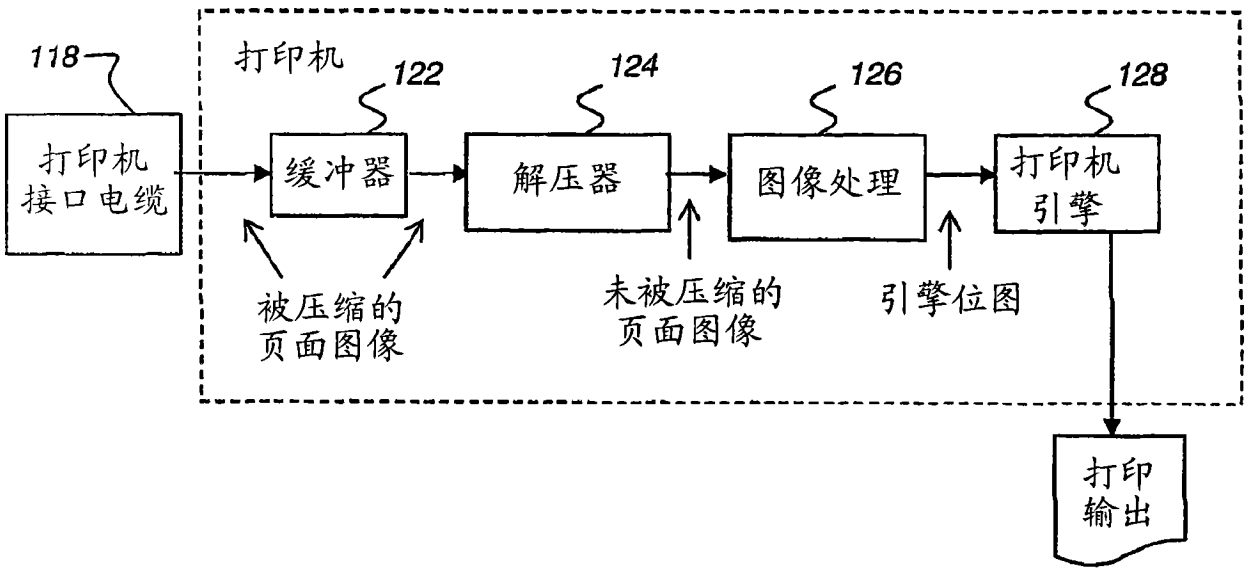

118
打印机
122
124
126
128
打印机
接口电缆
缓冲器
解压器
图像处理
打印机
引擎
被压缩的
页面图像
未被压缩的
页面图像
引擎位图
打印
输出

图 12