



(12)发明专利

(10)授权公告号 CN 105164923 B

(45)授权公告日 2018.10.23

(21)申请号 201480024777.X

(72)发明人 奥西·卡雷沃

(22)申请日 2014.03.01

(74)专利代理机构 北京英赛嘉华知识产权代理
有限责任公司 11204

(65)同一申请的已公布的文献号

申请公布号 CN 105164923 A

代理人 王达佐 王艳春

(43)申请公布日 2015.12.16

(51)Int.Cl.

H03M 7/30(2006.01)

(30)优先权数据

1303658.7 2013.03.01 GB

(56)对比文件

EP 0734126 A1, 1996.09.25,

GB 2301252 A, 1996.11.27,

CN 102210105 A, 2009.11.06,

CN 102474270 A, 2012.05.23,

EP 2131501 A1, 2009.12.09,

(85)PCT国际申请进入国家阶段日

2015.11.01

审查员 葛运滨

(86)PCT国际申请的申请数据

PCT/EP2014/000529 2014.03.01

(87)PCT国际申请的公布数据

W02014/131526 EN 2014.09.04

权利要求书5页 说明书23页 附图1页

(73)专利权人 古如罗技微系统公司

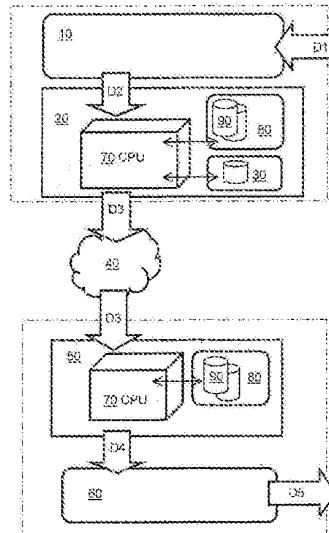
地址 芬兰土尔库市里南路34号,邮编20100

(54)发明名称

熵修正器及方法

(57)摘要

本发明提供了一种熵修正器(10、60)，该熵修正器用于对具有第一熵的输入数据比特流以一比特一比特的方式进行编码或者解码，以生成相应的具有第二熵的经熵修正的输出数据，其中，该熵修正器用于处理输入数据比特流以控制彼此类似比特组和第一比特值，以及彼此类似比特组的一个或多个最大行程长度。可选地，该熵修正器(10、60)用于通过使用至少一个转义码来控制彼此类似比特组的一个或多个最大行程长度。



1. 一种熵修正器(10),用于以一比特一比特的方式对具有第一熵的输入数据比特流(D1)进行编码,以生成相应的经熵修正的输出数据(D2),所述输出数据具有第二熵,其特征在于:

(a) 所述熵修正器(10)用于以一比特一比特的方式生成所述输出数据(D2),所述输出数据包括第一比特的信息,所述第一比特的信息作为包括在所述输出数据(D2)中的在所述第一比特之后的数据的参考值;

(b) 所述输出数据(D2)进一步包括指示存在于所述输入数据比特流(D1)中的彼此类似比特的行程长度的数据,其中,所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

(c) 通过包括在所述输出数据(D2)中的至少一个转义码来对在所述输入数据比特流(D1)中基本超过最大行程长度值(MaxRun)的一个或多个行程长度的出现进行标示,

其中,所述熵修正器(10)用作数据预处理器,用于将所述输入数据比特流格式化为所述输出数据(D2)以便在一个或多个编码装置(20)中进行后续压缩以生成压缩数据(D3)。

2. 根据权利要求1所述的熵修正器(10),其特征在于,通过多个转义码来控制彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

3. 根据权利要求1或2所述的熵修正器(10),其特征在于,所述熵修正器(10)用于以数据值“0”的方式在所述输出数据(D2)中实现至少一个转义码。

4. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)用于通过以下方式采用所述最大行程长度值(MaxRun)来处理所述输入数据比特流(D1),所述方式是所述最大行程长度值(MaxRun)作为所述输入数据比特流(D1)的特性的函数而动态变化。

5. 根据权利要求1所述的熵修正器(10),其特征在于,所述一个或多个编码装置包括下述一个或多个:ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

6. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)用于将所述输入数据比特流(D1)处理为多个部分,所述多个部分单独处理。

7. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。

8. 根据权利要求7所述的熵修正器(10),其特征在于,所述熵修正器(10)用于通过使用最大行程长度(MaxRun)来处理所述多个部分,一个或多个部分的最大行程长度(MaxRun)彼此不同。

9. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)被实现为作为一个用于压缩输入数据(D1)从而生成经熵修正的输出数据的编码器来工作。

10. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)可操作用于协助把输入数据(D1)的比特流转换为存在于经熵修正的输出数据之中的符号。

11. 根据权利要求1所述的熵修正器(10),其特征在于,所述熵修正器(10)包括计算硬件,其中,所述计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理所述输入数据比特流(D1)以生成所述输出数据(D2)。

12. 一种熵修正器(60),用于对具有第一熵的输入数据比特流(D3或D4)进行解码,以生成相应的经熵修正的输出数据(D5),所述输出数据具有第二熵,其特征在于:

(a) 所述熵修正器(60)用于通过使用输入数据(D3或D4)的第一比特的信息以一比特一比特的方式处理所述输入数据(D3或D4),所述第一比特的信息作为包括在所述输入数据(D3或D4)中的在所述第一比特之后的数据的参考值;

(b) 所述熵修正器(60)用于处理存在于所述输入数据(D3或D4)中的数据,所述数据用于指示存在于原始数据(D1)中的彼此类似比特的行程长度,其中,所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

(c) 所述熵修正器(60)用于检测在所述原始数据(D1)中基本超过最大行程长度值的一个或多个行程长度的出现,所述出现由包括在所述输入数据(D3或D4)中的至少一个转义码标示。

13. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

14. 根据权利要求13所述的熵修正器(60),其特征在于,通过使用多个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

15. 根据权利要求12或13或14所述的熵修正器(60),其特征在于,在输入数据(D3或D4)之中至少一个转义码被实现为“0”值。

16. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于通过以下方式使用所述最大行程长度值(MaxRun)来处理所述输入数据比特流(D3或D4),所述方式是所述最大行程长度值(MaxRun)作为所述输入数据比特流(D3或D4)的特性的函数而动态变化。

17. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于在所述输入数据比特流(D3或D4)中独立于数据比特序列单独处理第一比特的值的信息。

18. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于对存在于所述输入数据(D3或D4)中的数据元素进行解码,并应用逆熵修正以生成作为解码比特流的所述输出数据(D5)。

19. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于将所述第一比特的值处理为相对于一个数据值序列单独编码,所述数据值序列表示持续的类似比特量。

20. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)被利用于与一个或多个编码装置一起进行后处理,以处理所述输入数据比特流(D3或D4),其中,所述一个或多个编码装置包括下述一个或多个:ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

21. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于将输入数据比特流(D3或D4)处理为多个部分,所述多个部分被单独编码或解码。

22. 根据权利要求21所述的熵修正器(60),其特征在于,所述熵修正器(60)用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。

23. 根据权利要求21所述的熵修正器(60),其特征在于,所述熵修正器(60)用于通过使用最大行程长度值(MaxRun)来处理所述多个部分,一个或多个部分的最大行程长度值(MaxRun)彼此不同。

24. 根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)被实现为用

作解码器以对输入数据(D3或D4)进行解码以生成经熵修正的输出数据(D5)。

25.根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)用于将存在于所述输入数据(D3或D4)中的符号转换为存在于所述输出数据(D5)中的比特串。

26.根据权利要求12所述的熵修正器(60),其特征在于,所述熵修正器(60)包括计算硬件,其中,所述计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理所述输入数据比特流(D3或D4)以生成所述输出数据(D5)。

27.一种使用熵修正器(10)以一比特一比特方式对具有第一熵的输入数据比特流(D1)进行编码以生成相应的具有第二熵的经熵修正的输出数据(D2)的方法,其特征在于,所述方法包括:

(a)操作所述熵修正器(10)以一比特一比特的方式生成所述输出数据(D2),所述输出数据包括第一比特的信息,所述第一比特的信息作为包括在所述输出数据(D2)中的在所述第一比特之后的数据的参考值;

(b)所述输出数据(D2)进一步包括指示存在于所述输入数据比特(D1)中的彼此类似比特的行程长度的数据,其中,所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

(c)通过包括在所述输出数据(D2)中的至少一个转义码来对在所述输入数据(D1)中基本超过最大行程长度值的一个或多个行程长度的出现进行标示,

其中所述方法还包括,使用所述熵修正器(10)作为一个数据预处理器来工作,用于把输入数据(D1)格式化为输出数据(D2),以用于在一个或多个编码设备(20)中的后续的压缩从而生成压缩数据(D3)。

28.根据权利要求27所述的方法,其特征在于,所述方法包括使用多个转义码来控制彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

29.根据权利要求27或28所述的方法,其特征在于,所述方法包括,以“0”数据值的方式实现输出数据(D2)中的至少一个转义码。

30.根据权利要求27所述的方法,其特征在于,所述方法包括使用所述熵修正器(10)以通过以下方式使用所述最大行程长度值(MaxRun)来处理所述输入数据流(D1),所述方式是所述最大行程长度值(MaxRun)作为所述输入数据比特流(D1)的特性的函数而动态变化。

31.根据权利要求27所述的方法,其特征在于,所述一个或多个编码设备包括下列中的一个或多个:0Delta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

32.根据权利要求27所述的方法,其特征在于,所述方法包括使用所述熵修正器(10)将输入数据比特流(D1)处理为多个部分,所述多个部分被单独处理。

33.根据权利要求32所述的方法,其特征在于,所述方法包括使用所述熵修正器(10)以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。

34.根据权利要求32或33所述的方法,其特征在于,所述方法包括使用所述熵修正器(10)通过使用最大行程长度来处理所述多个部分,一个或多个部分的最大行程长度(MaxRun)值彼此不同。

35.根据权利要求27所述的方法,其特征在于,所述方法包括实现所述熵修正器(10)以用作编码器(10、20)以对所述输入数据(D1)进行压缩以生成所述经熵修正的输出数据(D2)、

D3)。

36. 根据权利要求27所述的方法,其特征在于,所述方法包括使用所述熵修正器(10)以将存在于所述输入数据(D1)中的比特串转换为存在于所述经熵修正的输出数据(D2或D3)中的符号。

37. 根据权利要求27所述的方法,其特征在于,所述方法包括实现所述熵修正器(10)以包括计算硬件,其中,所述计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理所述输入数据比特流(D1)以生成所述输出数据(D2)。

38. 一种使用熵修正器(60)以对具有第一熵的输入数据比特流(D3或D4)进行解码以生成相应的具有第二熵的经熵修正的输出数据(D5)的方法,其特征在于,所述方法包括:

(a) 使用所述熵修正器(60)以通过使用输入数据(D3或D4)的第一比特的信息以一比特一比特的方式处理所述输入数据(D3或D4),所述第一比特的信息作为包括在所述输入数据(D3或D4)中的在所述第一比特之后的数据的参考值;

(b) 使用所述熵修正器(60)以处理存在于所述输入数据(D3或D4)中的数据,所述数据用于指示存在于原始数据(D1)中的彼此类似比特的行程长度,其中,所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

(c) 所述熵修正器(60)用于检测在所述原始数据(D1)中基本超过最大行程长度值的一个或多个行程长度的出现,所述出现由包括在所述输入数据(D3或D4)中的至少一个转义码标示。

39. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

40. 根据权利要求38所述的方法,其特征在于,所述方法包括,使用熵修正器(60),通过使用多个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

41. 根据权利要求38或39或40所述的方法,其特征在于,所述方法包括,在输入数据(D3或D4)之中至少一个转义码被实现为“0”值。

42. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以通过以下方式使用所述最大行程长度值(MaxRun)来处理所述输入数据比特流(D3或D4),所述方式是所述最大行程长度值作为所述输入数据比特流(D3或D4)的特性的函数而动态变化。

43. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)在所述输入数据比特流(D3或D4)中独立于数据比特序列单独处理第一比特的值的信息。

44. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以对存在于所述输入数据(D3或D4)中的数据元素进行解码,并应用逆熵修正以生成作为解码比特流的所述输出数据(D5)。

45. 根据权利要求43或44所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)将所述第一比特的值处理为相对于一个数据值序列单独编码,所述数据值序列表示持续的类似比特量。

46. 根据权利要求38所述的方法,其特征在于,所述方法包括将所述熵修正器(60)与一个或多个编码装置一起使用于后处理,以处理所述输入数据比特流(D3或D4),其中,所述一

个或多个编码装置包括下述一个或多个:ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

47. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)将输入数据比特流(D3或D4)处理为多个部分,所述多个部分被单独编码或解码。

48. 根据权利要求47所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。

49. 根据权利要求47或48所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以通过使用最大行程长度来处理所述多个部分,一个或多个部分的最大行程长度值(MaxRun)彼此不同。

50. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器以用作解码器以对所述输入数据(D3或D4)进行解压缩以生成所述经熵修正的输出数据(D5)。

51. 根据权利要求38所述的方法,其特征在于,所述方法包括使用所述熵修正器(60)以将存在于所述输入数据(D3或D4)中的符号转换为存在于所述输出数据(D5)中的比特串。

52. 根据权利要求38所述的方法,其特征在于,所述方法包括将所述熵修正器(60)实现为包括计算硬件,其中,所述计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理所述输入数据比特流(D3或D4)以生成所述输出数据(D5)。

53. 一种记录于机器可读数据存储介质上的程序模块,其特征在于,所述程序模块在计算硬件上执行以用于执行权利要求27中所述的方法。

54. 一种记录于机器可读数据存储介质上的程序模块,其特征在于,所述程序模块在计算硬件上执行以用于执行权利要求38中所述的方法。

熵修正器及方法

技术领域

[0001] 本发明涉及用于接收输入数据并生成相应的输出数据的熵修正器,例如为了数据压缩和/或数据解压缩的目的,相对于输入数据的熵,所述输出数据的熵被修正;该熵修正器可通过使用电子硬件和/或存储于机器可读数据存储介质上并可在计算硬件上执行的软件产品来实现。此外,本发明还涉及对输入数据的熵进行修正以生成相应的输出数据的方法,该输出数据具有相对于输入数据的熵有一定修正度的熵。更进一步地,本发明涉及记录于机器可读数据存储介质上的软件产品,其中,该软件产品可以在计算硬件上执行以执行上述方法。该熵修正器可以用作数据通信系统和数据供应系统的(例如媒体供应系统)的部件。

背景技术

[0002] 概括来说,在信息理论中,熵是随机变量的不确定性衡量度,与物理连续系统有关,例如,它用在热力学系统中,以及例如数据通信系统的信息系统中。此外,在数据通信系统的环境中,熵更恰当地是指香农熵(Shanon entropy);可以参阅特此纳入参考的Claude E. Shannon在1948年发表的科学论文“*A Mathematical Theory of Communication*(通信的数学理论)”,其中,香农熵用于量化包含在给定消息中的信息的期望值。更进一步地,香农熵可以以nats、bans、以及bits的方式表示。

[0003] 在给定的通信能够表示为独立同分布的随机变量的序列的前提下,香农熵提供了一种与所有通信的最佳可能的无损编码或压缩相关的绝对限制。此外,香农定理证明,对具有给定字母的给定消息进行编码的最短的可能表示的平均长度L为它们的熵E除以存在于字母中的符号数N的对数,如公式1(Eq. 1)所限定:

$$[0004] L = \frac{E}{\log_{10} N} \quad \text{Eq. 1}$$

[0005] 因此,熵E是信息内容的不可预知性的量度。在无损数据压缩方法中,对相应的输入数据应用该方法生成的压缩输出数据具有与输入数据类似的信息量,但与输入数据相比,输出数据包括更少的数据比特。因此,由于其中包含更少的信息冗余,压缩输出数据更不可预知。

[0006] 香农定理已经被现有多种数据编码装置的设计所考虑。例如,已公布的国际专利申请第W02010/050157A1号(PCT/JP2009/005548,“*Image encoding apparatus, image encoding method, and image encoding program*”,申请人是Thomson Licensing),其中描述了一种图像编码设备、图像编码方法以及图像编码程序,能够用于使整个图像的图像质量均匀而不降低编码效率,同时保持高的速度,并且还能够通过在不改变像条(slice)结构的情况下执行宏块移动(macroblob shuffling)来降低电路规模尺寸。此外,还提供了一种图像编码装置,包括:

- [0007] (i) 移动部分,其在图像中的各个位置收集和移动组成图像数据的多个宏块;
- [0008] (ii) 编码部分,其对由该移动部分收集和移动的多个宏块执行空间频率转换和熵

编码；以及

[0009] (iii) 比率控制部分，其控制编码部分在执行编码后调整多个宏块的比率。
[0010] 数据，无论何种类型的数据，均需要数据存储空间，并且在从一个空间位置移动至另一空间位置时，还需要通信网络容量中的带宽。这种带宽对应于现实中的基础设施的投资和能源的利用。由于人们预期要通信的数据量将增加，因此将需要更多的数据存储空间和更多的通信系统容量，以及通常将需要更多的能量。在目前的因特网中，存储了大量的数据，通常会有多份拷贝。因此，任何一种能够压缩数据的方法，尤其是当这种压缩是无损的，将具有潜在的巨大技术和经济上的效益。目前，已经有了若干种已知的用于降低数据集中的熵、压缩数据集的方法。此外，已经有了已知的修正存在于数据集中的熵的方法，例如德尔塔编码 (Delta coding) 和行程长度编码 (RLE)，但是仍然需要提供对数据更有效的数据压缩的方法。

[0011] 因此，有许多不同的压缩方法，通过降低存在于给定数据中的熵，压缩给定数据，例如，上述的RLE编码、上述的VLC编码、哈夫曼编码、德尔塔编码、算术编码以及距离编码。这些方法通常被设计用于压缩字母、数字、字节和词。然而，这些方法并不是特别合适压缩独立的比特，并且由于这个原因，并不是理想地适用于压缩那些能够以一比特一比特 (bit-by-bit) 的方式变化的数据，例如，数据比特流。

[0012] 至于传统的RLE编码，要么存储的是给定值，即，比特 (bit)，要么两次存储该值，即，比特 (bit)，然后多个类似的值，即，比特 (bits)，以此类推。例如，RLE能够选择性地应用，即它的编码能够被保留专门用于对某些行程进行编码，这些行程包括已知数量的比特，用于一些类似值。RLE的这种选择性的应用需要在每个新的行程中，将同一个值一次或两次放入给定的数据流中。因此，现在需要更好的数据压缩手段，以降低数据集中的熵，不管数据是哪种类型。

发明内容

[0013] 本发明的第一个目的是为提供一种改进的熵修正，例如提供改进的数据压缩。
[0014] 本发明的第二个目的是为提供一种改进的熵修正方法，例如提供改进的数据压缩。

[0015] 根据本发明的第一个方面，提供了一种熵修正器，用于以一比特一比特的方式对具有第一熵的输入数据比特流 (D1) 进行编码，以生成相应的经熵修正的输出数据 (D2)，该输出数据具有第二熵，其中

[0016] (a) 所述熵修正器 (10) 用于以一比特一比特的方式生成所述输出数据 (D2)，所述输出数据包括第一比特的信息，所述第一比特作为包括在所述输出数据 (D2) 中的在所述第一比特之后的数据的参考值；

[0017] (b) 所述输出数据 (D2) 进一步包括指示存在于所述输入数据比特 (D1) 中的彼此类似比特的行程长度的数据，其中，所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun)；以及

[0018] (c) 通过包括在所述输出数据 (D2) 中的至少一个转义码来对在所述输入数据 (D1) 中基本超过最大行程长度值 (MaxRun) 的一个或多个行程长度的出现进行标示。

[0019] 本发明的优势在于对表示数据的行程长度的选择使用能够提供增强的熵修正，例

如,增强的数据压缩。

[0020] 可选地,该熵修正器被实现以通过多个转义码来控制彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。

[0021] 采用的一类转义码的方案使用最大行程长度值和持续的类似比特量的指示。在该类方案中,所有的符号可以编码为一个数据流。另一类转义码使得利用多种独立于编码的符号的不同的方案成为可能,并且该类转义码表示大于最大行程长度值的持续的类似比特值量。该类方案能够作为两路单独的流进行处理,或者该方案能够被设计为表示数字的所有值均不大于最大行程,因此其也能够与相同的其他符号流一起编码。可选地,也可以采用其他转义方法。

[0022] 修正器 (10) 用于以数据值“0”的方式在所述输出数据 (D2) 中实现至少一个转义码。

[0023] 可选地,熵修正器用于通过使用最大行程长度来处理所述输入数据流 (D1),所述使用的方式是最大行程长度值 (MaxRun) 作为输入数据比特流 (D1) 的特性的函数而动态变化。

[0024] 可选地,熵修正器用作数据预处理器,用于将输入数据 (D1) 格式化为输出数据 (D2) 以便在一个或多个编码装置中进行后续压缩以生成压缩数据 (D3)。

[0025] 可选地,一个或多个编码装置包括下述一个或多个:ODelta 编码器、RLE 编码器、VLC 编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0026] 可选地,熵修正器用于将输入数据比特流 (D1) 处理为多个部分,该多个部分单独处理。更可选地,熵修正器用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。更可选地,熵修正器用于通过使用最大行程长度值来处理多个部分,一个或多个部分的最大行程长度值 (MaxRun) 彼此不同。

[0027] 应当理解,第一比特信息能够被编码至其自身的数据流,或者编码至与行程长度信息相同的数据流,或者第一比特的信息能够被所使用的编码方法的信息所包括。

[0028] 可选地,熵修正器被实现以用作对输入数据流 (D1) 进行压缩以生成经熵修正的输出数据 (D2 或 D3) 的编码器。

[0029] 可选地,熵修正器用于将存在于输入数据 (D1) 中的比特串转换为存在于经熵修正的输出数据 (D2 或 D3) 中的符号。

[0030] 可选地,熵修正器包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流 (D1) 以生成输出数据 (D2)。

[0031] 根据本发明的第二个方面,提供了一种熵修正器,该熵修正器用于对具有第一熵的输入数据比特流 (D3 或 D4) 进行解码,以生成相应的经熵修正的输出数据 (D5),所述输出数据具有第二熵,其中:

[0032] (a) 熵修正器用于通过使用输入数据 (D3 或 D4) 的第一比特的信息以一比特一比特的方式处理输入数据 (D3 或 D4),第一比特作为包括在输入数据 (D3 或 D4) 中的在第一比特之后的数据的参考值;

[0033] (b) 熵修正器用于处理存在于输入数据 (D3 或 D4) 中的数据,该数据用于指示存在于原始数据 (D1) 中的彼此类似比特的行程长度,其中,彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun);以及

[0034] (c) 熵修正器用于检测在原始数据 (D1) 中基本超过最大行程长度值的一个或多个行程长度的出现,该出现由包括在输入数据 (D3或D4) 中的至少一个转义码标示。

[0035] 可选地,熵修正器用于通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。更可选地,熵修正器被实现以使用多个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。

[0036] 在输入数据 (D3或D4) 之中至少一个转义码被实现为“0”值。

[0037] 采用的一类转义码方案使用最大行程长度值和对类似比特量持续的指示。在该类方案中,所有的符号可以编码为一个数据流,类似地作为一个解码数据流。其他类的转义码使得利用多种独立于编码的符号的不同的方案成为可能,并且该类转义码表示大于最大行程长度值的持续的类似比特值量。该类方案能够作为两路单独的流进行处理,或者该方案能够被设计为表示数字的任一值均不大于最大行程,因此其也能够与相同的其他符号流一起编码。可选地,也可以采用其他转义方法。

[0038] 可选地,熵修正器用于通过使用所述最大行程长度值 (MaxRun) 来控制输入数据比特流 (D3或D4),所述使用的方式是最大行程长度作为输入数据比特流 (D3或D4) 的特性的函数而动态变化。

[0039] 可选地,熵修正器用于在所述输入数据流 (D3或D4) 中独立于数据比特序列单独处理第一比特的值的信息。更可选地,熵修正器用于将所述第一比特的值处理为相对于所述数据值序列单独编码,数据值序列表示持续的类似比特量。

[0040] 可选地,熵修正器用于对存在于输入数据 (D3或D4) 中的数据元素进行解码,并应用逆熵修正以生成作为解码比特流的输出数据 (D5)。

[0041] 可选地,熵修正器被利用于与一个或多个编码装置一起进行后处理,以控制输入数据比特流 (D3或D4),其中,该一个或多个编码装置包括下述一个或多个:0Delta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0042] 可选地,熵修正器用于将输入数据比特流 (D3或D4) 处理为多个部分,所述多个部分被单独编码或解码。更可选地,熵修正器用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。更可选地,熵修正器用于通过使用最大行程长度值 (MaxRun) 来处理多个部分,一个或多个部分的最大行程长度彼此不同。

[0043] 可选地,熵修正器被实现于为用作解码器以对输入数据 (D3或D4) 进行解码以生成经熵修正的输出数据 (D5)。

[0044] 可选地,熵修正器用于将存在于输入数据 (D3或D4) 中的符号转换为存在于输出数据 (D5) 中的比特串。

[0045] 可选地,熵修正器包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流 (D3或D4) 以生成输出数据 (D5)。

[0046] 根据本发明的第三个方面,提供了一种使用熵修正器以一比特一比特的方式对具有第一熵的输入数据比特流 (D1) 进行编码以生成相应的具有第二熵的经熵修正的输出数据 (D2) 的方法,其中,该方法包括:

[0047] (a) 操作熵修正器 (10) 以一比特一比特的方式生成输出数据 (D2),输出数据包括第一比特的信息,该第一比特作为包括在输出数据 (D2) 中的第一比特之后的数据的参考

值；

[0048] (b) 输出数据 (D2) 中进一步包括指示存在于输入数据比特 (D1) 中的彼此类似比特的行程长度的数据，其中，彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun)；以及

[0049] (c) 通过包括在输出数据 (D2) 中的至少一个转义码来对在输入数据 (D1) 中基本超过最大行程长度值的一个或多个行程长度的出现进行标示。

[0050] 可选地，该方法包括使用多个转义码来控制彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。更可选地，该方法包括使用熵修正器以通过使用最大行程长度来控制数据比特流 (D1)，所述使用的方式是最大行程长度值 (MaxRun) 作为输入数据比特流 (D1) 的特性的函数而动态变化。

[0051] 可选地，该方法包括，在熵修正器中，将至少一个转义码实现为值“0”。

[0052] 可选地，所述方法包括，使用所述熵修正器 (10) 作为一个数据预处理器来工作，用于把输入数据 (D1) 格式化为输出数据 (D2)，以用于在一个或多个编码设备 (20) 中的后续的压缩从而生成压缩数据 (D3)。所述一个或多个编码设备包括下列中的一个或多个：ODelta 编码器、RLE 编码器、VLC 编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0053] 可选地，该方法包括使用熵修正器将输入数据比特流 (D1) 处理为多个部分，该多个部分被单独处理。

[0054] 可选地，该方法包括使用熵修正器以时间上并行的方式，即，以并行执行的方式，处理所述多个部分。

[0055] 可选地，该方法包括使用熵修正器通过使用最大行程长度值 (MaxRun) 来处理该多个部分，一个或多个部分的最大行程长度彼此不同。

[0056] 可选地，该方法包括实现熵修正器以用作编码器以对输入数据流 (D1) 进行压缩以生成经熵修正的输出数据 (D2 或 D3)。更可选地，该方法包括使用熵修正器以将存在于输入数据 (D1) 中的比特串转换为存在于经熵修正的输出数据 (D2 或 D3) 中的符号。更可选地，该方法包括实现熵修正器以包括计算硬件，其中，该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品，以处理输入数据比特流 (D1) 以生成输出数据 (D2)。

[0057] 根据本发明的第四个方面，提供了一种使用熵修正器以对具有第一熵的输入数据比特流 (D3 或 D4) 进行解码以生成相应的具有第二熵的经熵修正的输出数据 (D5) 的方法，其中，该方法包括：

[0058] (a) 使用所述熵修正器以通过使用输入数据 (D3 或 D4) 的第一比特的信息以一比特一比特的方式处理输入数据 (D3 或 D4)，该第一比特作为包括在输入数据 (D3 或 D4) 中的在一比特之后的数据的参考值；

[0059] (b) 使用熵修正器以处理存在于输入数据 (D3 或 D4) 中的数据，该数据用于指示存在于原始数据 (D1) 中的彼此类似比特的行程长度，其中，彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun)；以及

[0060] (c) 熵修正器用于检测在原始数据 (D1) 中基本超过最大行程长度值的一个或多个行程长度的出现，该出现由包括在输入数据 (D3 或 D4) 中的至少一个转义码标示。

[0061] 可选地，该方法包括使用熵修正器以通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。更可选地，该方法包括通过使用多个转

义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。

[0062] 所述方法包括,在输入数据 (D3或D4) 之中至少一个转义码被实现为“0”值。

[0063] 可选地,该方法包括使用熵修正器以通过采用最大行程长度值 (MaxRun) 来控制输入数据比特流 (D3或D4),所述使用的方式是最大行程长度作为输入数据比特流 (D3或D4) 的特性的函数而动态变化。

[0064] 可选地,该方法包括使用熵修正器在所述输入数据比特流 (D3或D4) 中独立于数据比特序列单独处理第一比特的值的信息。更可选地,该方法包括使用熵修正器将所述第一比特的值处理为相对于一个数据值序列单独编码,该数据值序列表示持续的类似比特量。

[0065] 可选地,该方法包括使用熵修正器以对存在于输入数据 (D3或D4) 中的数据元素进行解码,并应用逆熵修正以生成作为解码比特流的输出数据 (D5)。

[0066] 可选地,该方法包括将熵修正器与一个或多个编码装置一起使用以进行后处理,以控制输入数据比特流 (D3或D4),其中,该一个或多个编码装置包括下述一个或多个: ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0067] 可选地,该方法包括使用熵修正器将输入数据比特流 (D3或D4) 处理为多个部分,该多个部分被单独编码或解码。更可选地,该方法包括使用熵修正器以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。所述方法包括使用所述熵修正器 (60) 以通过使用最大行程长度来处理所述多个部分,一个或多个部分的最大行程长度值 (MaxRun) 彼此不同。

[0068] 可以理解,解码器从其自身的数据流中获取第一比特的值的信息,或者从与行程长度信息相同的数据流中获取该第一比特的信息,或者可以将该第一比特的信息包括在所使用的编码装置的信息中。

[0069] 可选地,该方法包括使用熵修正器以用作解码器以对输入数据 (D3或D4) 进行解压缩以生成经熵修正的输出数据 (D5)。

[0070] 可选地,该方法包括使用熵修正器以将存在于输入数据 (D3或D4) 中的符号转换为存在于输出数据 (D5) 中的比特串。

[0071] 可选地,该方法包括将熵修正器 (60) 实现为包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流 (D3或D4) 以生成输出数据 (D5)。

[0072] 根据本发明的第五个方面,提供了一种记录于机器可读数据存储介质上的软件产品,其中,该软件产品在计算硬件上执行以执行根据本发明第三方面的方法。

[0073] 根据本发明的第六个方面,提供了一种记录于机器可读数据存储介质上的软件产品,其中,该软件产品在计算硬件上执行以执行根据本发明第四方面的方法。

[0074] 在下文中,结合附图和本发明的实施例的详细描述,本发明的其它方面、优点、目的将会更清楚地被理解。

[0075] 应当理解,在不背离本发明由所附权利要求限定的范围的情况下,本发明的特征还可以做各种组合。

附图说明

[0076] 接下来将以示例的方式,参照下图对本发明的实施方式进行描述,其中:

[0077] 图1为包括根据本发明的熵修正器的编解码系统的示意图。

[0078] 在附图中,使用带下划线的数字来代表项目,所代表的项目位于下划线数据所处的位置或接近的位置。无下划线数字所表示的项目通过指示线与无下划线数字连接。

具体实施方式

[0079] 在第一个方面,本发明的实施例提供了一种熵修正器,用于以一比特一比特的方式对具有第一熵的输入数据比特流(D1)进行编码,以生成相应的经熵修正的输出数据(D2),该输出数据具有第二熵,其中

[0080] (a) 所述熵修正器(10)用于以一比特一比特的方式生成所述输出数据(D2),所述输出数据包括第一比特的信息,所述第一比特作为包括在所述输出数据(D2)中的在所述第一比特之后的数据的参考值;

[0081] (b) 所述输出数据(D2)进一步包括指示存在于所述输入数据比特(D1)中的彼此类似比特的行程长度的数据,其中,所述彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

[0082] (c) 通过包括在所述输出数据(D2)中的至少一个转义码来对在所述输入数据(D1)中基本超过最大行程长度值(MaxRun)的一个或多个行程长度的出现进行标示。

[0083] 本发明的优势在于对表示数据的行程长度的选择使用能够提供增强的熵修正,例如,增强的数据压缩。

[0084] 可选地,该熵修正器被实现以通过多个转义码来控制彼此类似比特组的一个或多个最大行程长度值(MaxRun)。

[0085] 采用的一类转义码的方案使用最大行程长度值和持续的类似比特量的指示。在该类方案中,所有的符号可以编码为一个数据流。另一类转义码使得利用多种独立于编码的符号的不同的方案成为可能,并且该类转义码表示大于最大行程长度值的持续的类似比特值量。该类方案能够作为两路单独的流进行处理,或者该方案能够被设计为表示数字的所有值均不大于最大行程,因此其也能够与相同的其他符号流一起编码。可选地,也可以采用其他转义方法。

[0086] 修正器(10)用于以数据值“0”的方式在所述输出数据(D2)中实现至少一个转义码。

[0087] 可选地,熵修正器用于通过使用最大行程长度来处理所述输入数据流(D1),所述使用的方式是最大行程长度值(MaxRun)作为输入数据比特流(D1)的特性的函数而动态变化。

[0088] 可选地,熵修正器用作数据预处理器,用于将输入数据(D1)格式化为输出数据(D2)以便在一个或多个编码装置中进行后续压缩以生成压缩数据(D3)。

[0089] 可选地,一个或多个编码装置包括下述一个或多个:0Delta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0090] 可选地,熵修正器用于将输入数据比特流(D1)处理为多个部分,该多个部分单独处理。更可选地,熵修正器用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。更可选地,熵修正器用于通过使用最大行程长度值来处理多个部分,一个或多个部分

的最大行程长度值 (MaxRun) 彼此不同。

[0091] 应当理解,第一比特信息能够被编码至其自身的数据流,或者编码至与行程长度信息相同的数据流,或者第一比特的信息能够被所使用的编码方法的信息所包括。

[0092] 可选地,熵修正器被实现以用作对输入数据流 (D1) 进行压缩以生成经熵修正的输出数据 (D2或D3) 的编码器。

[0093] 可选地,熵修正器用于将存在于输入数据 (D1) 中的比特串转换为存在于经熵修正的输出数据 (D2或D3) 中的符号。

[0094] 可选地,熵修正器包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流 (D1) 以生成输出数据 (D2)。

[0095] 在本发明的第二个方面,提供了一种熵修正器,该熵修正器用于对具有第一熵的输入数据比特流 (D3或D4) 进行解码,以生成相应的经熵修正的输出数据 (D5),所述输出数据具有第二熵,其中:

[0096] (a) 熵修正器用于通过使用输入数据 (D3或D4) 的第一比特的信息以一比特一比特的方式处理输入数据 (D3或D4),第一比特作为包括在输入数据 (D3或D4) 中的在第一比特之后的数据的参考值;

[0097] (b) 熵修正器用于处理存在于输入数据 (D3或D4) 中的数据,该数据用于指示存在于原始数据 (D1) 中的彼此类似比特的行程长度,其中,彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun);以及

[0098] (c) 熵修正器用于检测在原始数据 (D1) 中基本超过最大行程长度值的一个或多个行程长度的出现,该出现由包括在输入数据 (D3或D4) 中的至少一个转义码标示。

[0099] 可选地,熵修正器用于通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。更可选地,熵修正器被实现以使用多个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。

[0100] 在输入数据 (D3或D4) 之中至少一个转义码被实现为“0”值。

[0101] 采用的一类转义码方案使用最大行程长度值和对类似比特量持续的指示。在该类方案中,所有的符号可以编码为一个数据流,类似地作为一个解码数据流。其他类的转义码使得利用多种独立于编码的符号的不同的方案成为可能,并且该类转义码表示大于最大行程长度值的持续的类似比特值量。该类方案能够作为两路单独的流进行处理,或者该方案能够被设计为表示数字的任一值均不大于最大行程,因此其也能够与相同的其他符号流一起编码。可选地,也可以采用其他转义方法。

[0102] 可选地,熵修正器用于通过使用所述最大行程长度值 (MaxRun) 来控制输入数据比特流 (D3或D4),所述使用的方式是最大行程长度作为输入数据比特流 (D3或D4) 的特性的函数而动态变化。

[0103] 可选地,熵修正器用于在所述输入数据流 (D3或D4) 中独立于数据比特序列单独处理第一比特的值的信息。更可选地,熵修正器用于将所述第一比特的值处理为相对于所述数据值序列单独编码,数据值序列表示持续的类似比特量。

[0104] 可选地,熵修正器用于对存在于输入数据 (D3或D4) 中的数据元素进行解码,并应用逆熵修正以生成作为解码比特流的输出数据 (D5)。

[0105] 可选地,熵修正器被利用于与一个或多个编码装置一起进行后处理,以控制输入

数据比特流(D3或D4),其中,该一个或多个编码装置包括下述一个或多个:ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0106] 可选地,熵修正器用于将输入数据比特流(D3或D4)处理为多个部分,所述多个部分被单独编码或解码。更可选地,熵修正器用于以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。更可选地,熵修正器用于通过使用最大行程长度值(MaxRun)来处理多个部分,一个或多个部分的最大行程长度彼此不同。

[0107] 可选地,熵修正器被实现于为用作解码器以对输入数据(D3或D4)进行解码以生成经熵修正的输出数据(D5)。

[0108] 可选地,熵修正器用于将存在于输入数据(D3或D4)中的符号转换为存在于输出数据(D5)中的比特串。

[0109] 可选地,熵修正器包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流(D3或D4)以生成输出数据(D5)。

[0110] 在第三个方面,本发明的实施例提供了一种使用熵修正器以一比特一比特的方式对具有第一熵的输入数据比特流(D1)进行编码以生成相应的具有第二熵的经熵修正的输出数据(D2)的方法,其中,该方法包括:

[0111] (a) 操作熵修正器(10)以一比特一比特的方式生成输出数据(D2),输出数据包括第一比特的信息,该第一比特作为包括在输出数据(D2)中的在第一比特之后的数据的参考值;

[0112] (b) 输出数据(D2)中进一步包括指示存在于输入数据比特(D1)中的彼此类似比特的行程长度的数据,其中,彼此类似比特的行程长度大小基本限制于不超过最大行程长度值(MaxRun);以及

[0113] (c) 通过包括在输出数据(D2)中的至少一个转义码来对在输入数据(D1)中基本超过最大行程长度值的一个或多个行程长度的出现进行标示。

[0114] 可选地,该方法包括使用多个转义码来控制彼此类似比特组的一个或多个最大行程长度值(MaxRun)。更可选地,该方法包括使用熵修正器以通过使用最大行程长度来控制数据比特流(D1),所述使用的方式是最大行程长度值(MaxRun)作为输入数据比特流(D1)的特性的函数而动态变化。

[0115] 可选地,该方法包括,在熵修正器中,将至少一个转义码实现为值“0”。

[0116] 可选地,所述方法包括,使用所述熵修正器(10)作为一个数据预处理器来工作,用于把输入数据(D1)格式化为输出数据(D2),以用于在一个或多个编码设备(20)中的后续的压缩从而生成压缩数据(D3)。所述一个或多个编码设备包括下列中的一个或多个:ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0117] 可选地,该方法包括使用熵修正器将输入数据比特流(D1)处理为多个部分,该多个部分被单独处理。

[0118] 可选地,该方法包括使用熵修正器以时间上并行的方式,即,以并行执行的方式,处理所述多个部分。

[0119] 可选地,该方法包括使用熵修正器通过使用最大行程长度值(MaxRun)来处理该多个部分,一个或多个部分的最大行程长度彼此不同。

[0120] 可选地，该方法包括实现熵修正器以用作编码器以对输入数据流 (D1) 进行压缩以生成经熵修正的输出数据 (D2或D3)。更可选地，该方法包括使用熵修正器以将存在于输入数据 (D1) 中的比特串转换为存在于经熵修正的输出数据 (D2或D3) 中的符号。更可选地，该方法包括实现熵修正器以包括计算硬件，其中，该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品，以处理输入数据比特流 (D1) 以生成输出数据 (D2)。

[0121] 在第四个方面，本发明的实施例提供了一种使用熵修正器以对具有第一熵的输入数据比特流 (D3或D4) 进行解码以生成相应的具有第二熵的经熵修正的输出数据 (D5) 的方法，其中，该方法包括：

[0122] (a) 使用所述熵修正器以通过使用输入数据 (D3或D4) 的第一比特的信息以一比特一比特的方式处理输入数据 (D3或D4)，该第一比特作为包括在输入数据 (D3或D4) 中的在第一比特之后的数据的参考值；

[0123] (b) 使用熵修正器以处理存在于输入数据 (D3或D4) 中的数据，该数据用于指示存在于原始数据 (D1) 中的彼此类似比特的行程长度，其中，彼此类似比特的行程长度大小基本限制于不超过最大行程长度值 (MaxRun)；以及

[0124] (c) 熵修正器用于检测在原始数据 (D1) 中基本超过最大行程长度值的一个或多个行程长度的出现，该出现由包括在输入数据 (D3或D4) 中的至少一个转义码标示。

[0125] 可选地，该方法包括使用熵修正器以通过使用至少一个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。更可选地，该方法包括通过使用多个转义码来处理多个彼此类似比特组的一个或多个最大行程长度值 (MaxRun)。

[0126] 所述方法包括，在输入数据 (D3或D4) 之中至少一个转义码被实现为“0”值。

[0127] 可选地，该方法包括使用熵修正器以通过采用最大行程长度值 (MaxRun) 来控制输入数据比特流 (D3或D4)，所述使用的方式是最大行程长度作为输入数据比特流 (D3或D4) 的特性的函数而动态变化。

[0128] 可选地，该方法包括使用熵修正器在所述输入数据比特流 (D3或D4) 中独立于数据比特序列单独处理第一比特的值的信息。更可选地，该方法包括使用熵修正器将所述第一比特的值处理为相对于一个数据值序列单独编码，该数据值序列表示持续的类似比特量。

[0129] 可选地，该方法包括使用熵修正器以对存在于输入数据 (D3或D4) 中的数据元素进行解码，并应用逆熵修正以生成作为解码比特流的输出数据 (D5)。

[0130] 可选地，该方法包括将熵修正器与一个或多个编码装置一起使用以进行后处理，以控制输入数据比特流 (D3或D4)，其中，该一个或多个编码装置包括下述一个或多个：ODelta编码器、RLE编码器、VLC编码器、哈夫曼编码器、德尔塔编码器、算术编码器、距离编码器。

[0131] 可选地，该方法包括使用熵修正器将输入数据比特流 (D3或D4) 处理为多个部分，该多个部分被单独编码或解码。更可选地，该方法包括使用熵修正器以时间上并行的方式，即，以并行执行的方式，处理所述多个部分。所述方法包括使用所述熵修正器 (60) 以通过使用最大行程长度来处理所述多个部分，一个或多个部分的最大行程长度值 (MaxRun) 彼此不同。

[0132] 可以理解，解码器从其自身的数据流中获取第一比特的值的信息，或者从与行程长度信息相同的数据流中获取该第一比特的信息，或者可以将该第一比特的信息包括在所

使用的编码装置的信息中。

[0133] 可选地,该方法包括使用熵修正器以用作解码器以对输入数据(D3或D4)进行解压缩以生成经熵修正的输出数据(D5)。

[0134] 可选地,该方法包括使用熵修正器以将存在于输入数据(D3或D4)中的符号转换为存在于输出数据(D5)中的比特串。

[0135] 可选地,该方法包括将熵修正器(60)实现为包括计算硬件,其中,该计算硬件用于执行一个或多个记录在机器可读数据存储介质上的软件产品,以处理输入数据比特流(D3或D4)以生成输出数据(D5)。

[0136] 在第五个方面,本发明的实施例提供了一种记录于机器可读数据存储介质上的软件产品,其中,该软件产品在计算硬件上执行以执行根据本发明第三方面的方法。

[0137] 在第六个方面,本发明的实施例提供了一种记录于机器可读数据存储介质上的软件产品,其中,该软件产品在计算硬件上执行以执行根据本发明第四方面的方法。

[0138] 概括来说,本发明涉及一种熵修正器。参照图1,将输入数据D1提供至第一熵修正器10,该第一熵修正器对输入数据进行转换以生成相应的经转换数据D2;可选地,相对于输入数据D1,根据第一熵修正器10中采用的转换的性质,经转换的数据D2的大小彼此不同,并且相对于输入数据D1,经转换的数据D2的熵降低。可选地,进一步通过编码级20对经转换的数据D2进行编码以生成最终的编码输出数据D3;当未采用编码级20时,数据D2和输出数据D3彼此类似。将编码输出数据D3存储于数据存储器30中和/或通过数据通信网络40进行传输。此外,直接将输出数据D3传送至第二熵修正器60,或者可选地,当采用上述编码级20时将输出数据D3通过解码级50后,再传送至第二熵修正器60,以使得数据D4与数据D2基本类似。第二熵修正器60生成解码输出数据D5,该解码输出数据D5与输入数据D1基本类似。因此,第一熵修正器10和第二熵修正器60执行彼此相反的熵修正操作,就像在编解码结构中一样。

[0139] 可选地,第一熵修正器10和第二熵修正器60可用于处理作为多个并行的数据流通过其的数据,以达到更快的熵修正,例如,更快的编码和/或解码。在处理大的数据量,例如,来自数据服务器(例如,来自提供有线电视服务的基于因特网的服务器)的流视频内容时,这种使用多个并行数据流的更快的处理方法是有利的。

[0140] 因此,第一熵修正器10消耗存在于输入数据D1中的比特流,并从这些比特流中创建数据元素,与输入数据D1的熵相比,这些数据元素的熵降低。可选地,通过采用诸如变长编码(VLC)、哈夫曼编码、算术编码、距离编码、行程长度编码(RLC)等等的编码方法,进一步在编码级20中对这些经修正的数据元素进行编码。在第一熵编码器10中执行的熵修正时可逆的,并且是有益地无损耗的,如作适当变动,第二熵修正器60中亦然。此外,在每次比特值改变后,在第一熵修正器10中执行的熵修正至少存储一个新的数据元素,该新的数据元素包含连续的大量类似比特。

[0141] 当执行如上述的熵降低时,应当提供任一给定序列中的第一个比特的标示。此外,还必须提供一段转义码,当数据元素要被压缩时,该转义码使得利用有限数量的码来表示数据元素,即,最大行程长度值(MaxRun),成为可能。可选地,转义码由数字“0”来表示;转义码指示了数据元素能够表示的最高值,即,最大行程长度值(MaxRun),该转义码还传递了比特值是连续的这一信息。有益地,在本发明的具体实施方式中,与比特的最大行程相对应的

一个或多个码将随数据及其特性而变化；换言之，与最大行程相对应的一个或多个码将随要压缩的数据而适应性变化。可选地，熵修正器10、60采用预处理和/或后处理方法，例如，0Delta算子，该算子执行1比特数据的运算。然而，对于熵修正器10、60来说，通过其进行处理的数据可能是任何类型的，例如，图像数据、视频数据、音频数据、参考数据、屏蔽数据(Masks)、分割(split)比特、符号比特、以及压缩数据，即，能够以比特的方式进行处理的任何类型的数据。可选地，任何数据的字节或词均能够插入包括比特的流中，并且能够通过熵修正器10和60进行处理。在下文中，“熵修正器”和“逆熵修正器”将分别缩写为EM和IEM，即，分别用于提供数据压缩和数据解压缩，即，分别通过第一熵修正器10和第二熵修正器60来执行数据压缩和数据解压缩。

[0142] 可选地，通过使用计算硬件(CPU)70来实现熵修正器10、60，该计算硬件用于执行一个或多个软件产品80，该软件产品记录在机器可读数据存储介质90上以执行根据本发明的方法。

[0143] 现在将对本发明的具体实施方式进行更详细地描述。熵修正器10、60采用一个参数来定义存在于数据比特流中的数字0或1的最大行程。此外，熵修正器10、60还采用转义元素，该转义元素在比特流中的比特的行程大于比特流的最大行程时使用。有益地，最大行程长度被设置为一个序列中的一个或多个：

$$L = 2^n - 1 \quad \text{Eq. 2}$$

[0145] 其中，n为整数，其值为n=2,3,……

[0146] 这对应于一个序列L=3,7,15,31,……，而不是任意大于1的数字。由于在实现本发明时，不可能出现多个与零类似的比特，因此最好将转义元素选择为值“0”。应当理解，如果类似比特的数量如上述地减少了一个计数，则转义元素最好是最大行程。当需要对比特流进行压缩时，熵修正器10需要知道流中的第一个比特，以使得在熵修正器60中，相应的压缩比特流能够被顺序解码，即，可逆地进行解压缩。

[0147] 为进一步地阐明本发明的实施方式，将对一个示例进行描述。下面示出了第一个示例，其中，根据本发明的熵编码在最大行程长度为3比特和转义码为“0”的条件下发挥作用：

[0148] 原始比特流如下：

[0149] 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1

[0150] 其中在流中总共包括26个比特，在该26个比特中，包括16个“1”和10个“0”。

[0151] 可以如下计算与该原始比特流相关的熵E：

$$E = 16 * \log_{10} \left(\frac{26}{16} \right) + 10 * \log_{10} \left(\frac{26}{10} \right) = 7.52339 \quad \text{Eq. 3}$$

[0153] 通过公式3(Eq. 3)，根据信源编码原理，可以如下计算得到对具有这样的相关熵E的原始比特流进行编码需要的最小比特数M_B：

$$M_B = \frac{E}{\log_{10}(2)} = 24.99125 \quad \text{比特} \quad \text{Eq. 4}$$

[0155] 通过运用根据本发明的方法可对上述原始比特流进行编码，以生成相应的码序列：

[0156] 第一比特值为“0”以及连续的大量类似比特(=熵修正符号)为:

[0157] 1 2 2 7 4 1 3 6

[0158] 当使用转义符号“0”来将不同符号的数量限制到从0(转义)至3(最大行程长度)的4个不同的值,并且将第一比特值增加为流的第一个符号时,从而将得到:

[0159] 0 1 2 2 0 0 1 0 1 1 3 0 3

[0160] 其中在流中总共包括13个码,在该13个码中,包括5个“0”、4个“1”、2个“2”和2个“3”。

[0161] 可以如下计算关于这个码流的熵E:

$$[0162] E = 5 * \log_{10}\left(\frac{13}{5}\right) + 4 * \left(\frac{13}{4}\right) + 2 * 2 * \log_{10}\left(\frac{13}{2}\right) = 7.37405 \quad \text{Eq. 5}$$

[0163] 其中,可以如下计算以无损方式表示该熵E的最小比特数M_B:

$$[0164] M_B = \frac{E}{\log_{10}(2)} = 24.49608 \quad \text{比特} \quad \text{Eq. 6}$$

[0165] 从该示例中可以看出,在码序列中的熵少于原始比特流的熵,并且需要用来表示码序列的比特更少。可选地,以表1中概括的方式生成经熵修正的码序列:

[0166] 表1:经熵修正的码的翻译

[0167]

码	指示
0	第一个比特值
1	第一种比特的数量: 1个“0”
2	随后改变的比特的数量: 2个“1”
2	随后改变的比特的数量: 2个“0”
0	转义码代表 4个或更多彼此类似的比特, 例如, 3个“1”
0	转义码代表 4个或更多彼此类似的比特, 例如, 3个“1”
1	随后改变的比特的数量: (7)-2*(转义比特), 其中, 转义比特为3个, 即, 1个“1”
0	转义码代表 4个或更多彼此类似的比特, 例如, 3个“0”
1	随后改变的比特的数量: (4)-1*(转义比特), 其中, 转义比特为3个, 即, 1个“0”
1	随后改变的比特的数量: 1个“1”
3	随后改变的比特的数量: 3个“0”
0	转义码代表 4个或更多彼此类似的比特: 3个“1”
3	随后改变的比特的数量: (6)-1*(转义比特), 其中, 转义比特为3个, 即, 3个“1”

[0168] 表1仅为示例,本发明的实施方式能够生成多个可选地示例。例如,可以采用其他

最大行程长度值和其他转义码。此外，可选地，可以独立地插入第一比特，其并不对采用的实际码及其相关压缩造成任何实质程度上的影响。

[0169] 可选地，在某些情况下，熵修正器10、60可用于增加熵，从而能够有益于使用例如信号监视装置(signal monitoring arrangement)对熵修正器10、60的输出数据进行监视，以确定采用的编码方法是否提供了所期望的数据压缩程度。在对将要传送的给定类型数据的熵进行增加的事件中，可以采用信号监视装置以一种适应性的方式控制熵修正器10、60的运算，以确保在运算中熵通过其达到所期望的修正。例如，对于给定类型的数据，当开始对数据流进行编码时，期望将熵修正器10用于运用一种或多种测试编码方法，以搜索最大行程的最优值，该最大行程的最优值用于获得最优数据压缩；该最大行程可对熵E有重大影响。另外，当给定数据流的长度相对较短时，与给定数据流中的第一比特值相关的熵E对熵E具有重大影响；例如，有益地，第一比特值与用于表示给定数据流的码字的其他码字分开放送。

[0170] 如图1所示，熵修正器10、60与熵编码方法结合使用是有益的，该熵编码方法可以是诸如可变长度编码(VLC)、哈夫曼编码、行程长度编码(RLE)、算术编码和距离编码。可选地，最好是结合诸如德尔塔编码的其他类型的熵修正器使用。最好在熵修正器10之后以及在熵修正器60之前使用这些熵编码方法；可选地，或者附加地，最好在熵修正器10之前以及熵修正器60之后使用这些熵编码方法；例如，在将要编码的原始数据为字母数字、数字、字节或字的事件中，结合表2中提供的附加编码对原始数据进行有效编码：

[0171] 表2：根据本发明的熵修正器采用的附加编码

[0172]

附加编码方法	使用情况
RLE	当给定数据流包括相同数据值的连续流时
德尔塔	当给定数据流包括类似或彼此基本类似或具有类似变化的数据值时
VLC/哈夫曼/ 算术/距离	当给定数据流包括多个彼此类似的数据值时

[0173] 可选地，以递归的方式使用熵修正器10、60。

[0174] 如上所述，最好是联合其他类型的熵修正器使用根据本发明的熵修正器，其中，该其他类型的熵修正器可选地为已知类型。已知类型包括例如德尔塔编码。典型地，德尔塔编码用于处理以字节或字的格式提供的数值。接下来将要对本发明与德尔塔编码结合使用的熵修正器的示例进行描述。

[0175] 为进一步阐释本发明，将描述两个示例，一个示例未使用根据本发明的熵修正器，以及另一个利用了根据本发明的熵修正器。在示例中，采用的7比特的最大行程长度以及相关的转义码为“0”。当采用1比特的ODelta类型算子时，已编码序列的第一个比特与位于被编码的原始数据流中的相应原始数据比特相同。可选地，该1比特的ODelta运算可以递归运行。

[0176] 示例1：

[0177] 在原始数据比特流中,总共包括37个比特,即,17个“1”和20个“0”:

[0178] 0 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1

[0179] 其中,如下计算相应的相关熵E:

$$[0180] E = 17 * \log_{10} \left(\frac{37}{17} \right) + 20 * \log_{10} \left(\frac{37}{20} \right) = 11.08523 \quad \text{Eq. 7}$$

[0181] 相应地计算得到的M_B为36.82比特,即,大概为上述提供的37比特。应当理解,公式7(Eq.7)对应于原始比特流中的比特量。

[0182] 以与生成表1所采用地、大体类似的方式生成的相当的熵修正(EM)编码如下:

[0183] 0 1 1 1 1 1 2 2 1 3 1 1 1 0 4 0 3

[0184] 其中,相关的熵E可以如下计算得到:

$$[0185] E = 3 * \log_{10} \left(\frac{17}{3} \right) + 9 * \log_{10} \left(\frac{17}{9} \right) + 2 * 2 * \log_{10} \left(\frac{17}{2} \right) + 1 * \log_{10} \left(\frac{17}{1} \right) = 9.69397 \quad \text{Eq. 8}$$

[0186] 应当理解,公式8(Eq.8)对应于熵修正后的比特量。此外,应当理解,该比特量少于公式7(Eq.7)对应的比特量。

[0187] 但是,当对上述的原始比特流采用ODelta运算时,将生成相应的修正的比特流,该比特流包括37比特,其中,13个“1”和24个“0”:

[0188] 0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0

[0189] 其中,相关的熵E可以如下计算得到:

$$[0190] E = 13 * \log_{10} \left(\frac{37}{13} \right) + 24 * \log_{10} \left(\frac{37}{24} \right) = 10.41713 \quad \text{Eq. 9}$$

[0191] 即,小于原始比特流相关的熵E(Eq.7)。当对该经ODelta处理后的上述数据比特流采用根据本发明的熵修正时,从而获得相应的被编码的编码序列,诸如:

[0192] 0 1 5 1 1 1 2 2 4 0 3 1 0 2

[0193] 其中,相关的熵E可以如下计算得到:

$$[0194] E = 2 * 3 * \log_{10} \left(\frac{14}{3} \right) + 5 * \log_{10} \left(\frac{14}{5} \right) + 3 * 1 * \log_{10} \left(\frac{14}{1} \right) = 9.68822 \quad \text{Eq. 10}$$

[0195] 其少于总共包括37个比特的原始数据比特流的熵E(Eq.7)。此外,应当理解,该比特量少于与公式8(Eq.8)或公式9(Eq.9)相关的比特量。

[0196] 该示例表明,当首先采用1比特ODelta算子对原始数据流进行压缩,再采用根据本发明的熵修正器进行处理时,能够对原始数据比特流进行有效压缩。

[0197] 示例2:

[0198] 上述示例1中的原始比特流被分为两个部分,并且通过选择该两部分数据流的结果的最佳组合来优化根据本发明的熵修正的收益。熵修正器的第一个比特值被单独传输,并且未被插入作为结果的、熵被降低的码序列中。可获得的第一个熵修正结果类似于示例1,其中,该第一个比特也被设置作为码字“1”。如表3所提供的,采用了类似的7比特的最大行程,并且采用“0”作为转义码:

[0199] 表3:示例2熵修正

[0200]

	比特流	熵, E	编码的最大比特数, MB
原始比特第1部分 (16 比特, 包括 7 个“1”, 9 个“0”):	0 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1	4.7621	15.82 比特
经修正的 熵 (EM)	0 1 1 1 1 1 2 2 1 3 1 1 1	5.2910	17.58 比特
经修正的 熵 (EM)	(0) 1 1 1 1 1 2 2 1 3 1 1 1	3.7599	13.49 比特
ODelta 修正	0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1	4.3158	14.34 比特
EM	(0) 1 5 1 1 1 2 2 3	4.2144	15.00 比特
原始比特第2部分 (21 比特, 包括 11 个“1”, 10 个“0”):	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1	6.3113	20.97 比特
经修正的 熵 (EM)	(0) 0 4 0 3	1.08062	6.00 / 7.00 比特
ODelta	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	1.7460	5.80 比特
EM	(0) 0 4 1 0 2	2.8928	9.61 / 10.61 比特

[0201] 从表3中可以看出,受益于数据压缩的实现,实现了熵的降低。当对原始数据流中生成的数据的第一部分数据流(该数据被进行熵修正)采用熵编码时,即,使用13.49比特表示第一部分,且通过1比特ODelta编码对第二部分进行熵修正,即,使用5.80比特表示第二部分,即,可使用19.29比特表示时,与原始的需要37比特相比,获得了最好的数据压缩;这代表了相当大的数据压缩。在对第二数据流进行编码时,由于可以基于第一数据流获知第二数据流的第一比特,因此不需要发送第一比特的值。因此,表3中示出了由“/”划分的两个不同的值;该第一个值是连续的情况(即,没有发送第一比特的值)以及该第二个值是数据流开始于此类比特组合的情况。应当理解,在该示例中,尽管ODelta编码独自降低了数据熵,即,大部分在第二部分数据降低;但是与熵修正器10、60的组合仍然更有效地降低了熵。此处,经熵修正的原始数据的熵小于首先以ODelta编码处理的经熵修正数据的熵,即6.00比特<9.61比特。在实际的编码条件下,尽管ODelta编码的数据比较小,即,5.80比特<6.00比特,但对经熵修正的原始数据进行压缩比对ODelta编码数据的压缩更有效;尽管在

编码算法中,也采用算术编码来压缩1比特数据,但1比特数据并不实际用于熵编码器的实现。可选地,为了实现数据压缩,在被执行熵修正处理之前,原始比特流被切分(split)为超过两个部分的多个部分。

[0202] 可选地,可以通过采用以下方法来执行将原始数据比特流切分为一个或多个流(部分),该方法包括:

[0203] (a) 分析与原始数据比特流相关的熵;

[0204] (b) 根据分析得到的原始数据比特流的熵的函数,将原始数据流切分为两个或多个部分;

[0205] (c) 对该两个或多个部分应用多个熵修正运算中的一个,以生成一个或多个相应的经熵修正的部分;以及

[0206] (d) 将该一个或多个经熵修正的部分组合到一起,以生成压缩编码输出数据。

[0207] 本发明还涉及一种对编码输出数据的解码方法,其中,利用了步骤(a)到(d)的逆过程。为实现增强的压缩率和/或实现基本无损耗的压缩,该方法可选地,包括对存在于原始比特流中的数据块进行切分或组合以更有效地实施编码。

[0208] 可选地,步骤(b)以粗粒度(coarse)的方式实现,即,在原始数据比特流中遇到多个长的行程部分时,仅基于原始数据比特流进行分割,即,在原始数据比特流的具有足够大的数据比特流范围的区域之后或区域之间进行分割,该范围内的数据比特以快速的方式顺序变化。

[0209] 可选地,如图1所示,为生成根据本发明的编码比特流,通过例如应用VLC、算术或距离编码和/或RLE对原始数据比特流进行处理以提供中间编码数据,接下来,对该中间编码数据采用根据本发明的熵修正,以生成被压缩且熵降低的输出编码数据。从而实现了数据压缩过程中相当大的速度优化,从而对应于在通过一个或多个软件产品的方式实现本发明时需要更多的计算能力,该一个或多个软件产品记录在机器可读数据存储介质上,其中,该一个或多个软件产品可在计算硬件上执行以实现熵修正编码器及经适当修改的熵修正解码器。可选地,首先使用一些其他方法对原始数据比特流中的原始比特、字母、数字、字节、字进行编码,以及然后,应用根据本发明的熵修正以生成编码输出数据以实现熵优化。

[0210] 在当代已知的行程长度编码(RLE)方法中,首先生成给定的值,然后生成表示该给定值的数字的出现次数的对应数据,接下来是其他值。只要待编码的输入数据存在,这种RLE方法就一直持续,即,一个值接一个值的顺序编码。作为对比的,本发明源自于注意到,如果仅采用了1比特的值,那么就没有必要在待编码的数据流中增加交替依次出现的数据值“0”和“1”;因此,只将给定序列中的第一值表示为编码数据进行传输是足够的。因此,本发明与RLE编码类似,但是采用如上述的1比特数据流。

[0211] 当实现本发明的具体实施例时,采用附加的转义码,该转义码使得实际的编码方法的使用更加有效。有利地,本发明的具体实施例包括从编码器发送一张或多张编码表至相应的解码器,以有助于对来自于编码器且被解码器接收的编码数据进行解码。在2014年2月20号递交的专利申请GB1403039.9中示例性的公开了传递这种码表的一些非常好的方法,将该专利申请结合于此作为引证。在2014年2月20号递交的专利申请GB1403038.1中还公开了组合熵编码和传递码表的其他方法,也将该专利申请结合于此作为引证。上述公开的方法均能够较好地与本发明实施例结合使用,并且有助于使用于熵编码的附加编码数据最小

化。

[0212] 采用于实现本发明实施例的转义码倾向于以多种不同的方式实现。例如,在持续为“0”值的行程长度符号或数字的情况下,实现令人满意的编码。但是,当这些值的量较小,或者相反,偶尔出现非常大的值时,潜在的情况可能会出现;在这种情况下,转义码最好设置为下一个没有采用的值,或者被分配作为码字使用。例如,采用的最大行程长度为7比特或元素,转义码为“0”,那么在生成编码数据流期间使用码字“8”;该新的转义码仅增加一个新的码字,并且在此之后,实际的符号字可以使用任何方法例如,以1、2和/或3字节,或使用半字节(即nibbles)表示。可选地,可以不压缩地传输实际的符号字。可选地,将实际的符号字与最大行程长度值组合,并且在熵压缩器(即,熵修正器)中与其他数据值一起压缩。

[0213] 当选择了适当的转义值以用于本发明实施例时,在调用该转义值之后,立即表示相邻元素的总数,并且与转义值的使用不同,随后相同比特不再持续。这种新的终止形式提供了一种转义,即,码字,该码字可用于附加到上述持续的转义方法;可选地,这种新的转义码可仅用作替换,并且,在这种情况下,其码字优选设置为值“0”。类似地,如果该终止码的给定数据值通常较小,但是偶尔较大,那么以替代的字节数或半字节数来表示数据中存在的连续类似符号数是有益的。有益地,这种字节的一个比特,例如,最低有效位(LSB)或最高有效位(MSB)被保留用于指示数字(例如,比特串)的终止,并且因此,在值结束的情况下,将其码字设置为值“0”或“1”,在其他情况下,该值持续。

[0214] 为了对上述内容进行阐释,接下来将在利用两种类型的转义的情况下对转义码的功能的实例进行描述。在该示例中,采用的最大行程为14个元素,具有以15表示的用于持续转义的码字,其中,用于终止转义的码字为0。此外,连续类似符号或比特的数量的值为:

[0215] 9,1024,16,9,12,2000,7,20,21,6,8,120,12,...

[0216] 此外,示例中利用半字节(即,nibble)来表示数字(即,比特串)的终止,该半字节采用值为“1”的最高有效位(MSB)。有益地,如果采用终止转义,则从数字(即,比特流)中减去下述是可行的:

[0217] (bit string) - (MAXRUN+1)

[0218] 例如:

[0219] 1024_{10} 为底 = >

[0220] 1024_{10} 为底 - (14_{10} 为底 + 1) = > 1009_{10} 为底 = > 1 111 110 001 表示为半字节数字 10 为底 = >

[0221] 1+8(终止) 761)。因此,该示例可以编码为:

[0222] 9 0 1 6 7 9 15 2 9 12 0 1 0 7 11 7 15 6 15 7 6 8 0 1 5 9 12。

[0223] 在该示例中,熵修正器10、60需要能够对具有范围0到15内的值的码字进行翻译。通过仅使用单个转义码来表示该示例是可行的,其中,码字“0”对应于转义码,该转义码也指示终止转义;在该单一类型的转义码的可替换示例中,未使用持续转义,并且其他参数同上,即:

[0224] 9 0 1 6 7 9 0 9 9 12 0 1 0 7 11 7 0 13 0 14 6 8 0 1 5 9 12,

[0225] 其中,熵修正器10、60需要识别范围0到14内的码字。可选地,通过仅使用由码字“0”表示的持续转义且采用最大行程长度14来表示该可替换的示例,那么结果是下述形式的编码序列:

[0226] 9 73x0 2 0 2 9 12 142x0 12 7 0 6 7 6 8 8x0 8 12。

- [0227] 可以基于该三种解决方案各自计算得到的熵来选择最佳的编码解决方案。
- [0228] 接下来,将对提供熵修正和熵的逆修正的本发明的实施例进行描述。然而,其他实施例也可以包括在本发明的范围之内,例如,对应的基于数字硬件(例如,可变状态机、ASIC等等)的熵修正器。下述示例中的“GetBit”、“SetBit”以及“ClearBit”指令通常用于更新“HeaderBits”值。此外,当下一比特将在下一字节中时,“HeaderIndex”值也将被更新。在下面基于软件产品的示例中,第一比特被存储为一个码。可选地,在更佳的实施例中,如上述(例如,参考表3)所言,第一比特值与码分离。有益地,通过将多个比特一起写入目的字节来优化相应的解码器,该目的字节用于解码输出数据。应当理解,下述示例中提供的“MaxRun”类似于(最大行程长度+1);例如,当采用7比特的最大行程长度时,将值8给作下述实施例中的函数参数“AMaxRun”。

[0229]

```

function EncodeBnRLE1u(APtrSrc : PByte; ASrcBnLen : Cardinal; APtDst : PByte; var ADst3BitOffset : 
Cardinal; AMaxRun : Cardinal) : Boolean;
var
  iHeaderIndex, iHeaderBits, iRunLength, Index : Cardinal;
  bBit : Byte;
  bLastBit : Boolean;
begin
  // Reset offsets
  ADst3BitOffset := 0;
  iHeaderIndex := 0;
  iHeaderBits := 0;

  // Read first bit, write it to destination and initialize run
  bLastBit := GetBit(APtrSrc, (@iHeaderIndex, (@iHeaderBits));
  APtDst[(ADst3BitOffset + 7) div 8] := Byte(bLastBit);
  Inc(ADst3BitOffset, 8);
  iRunLength := 1;

  // Go through bits
  for Index := 1 to ASrcBnLen-1 do
  begin
    // Read bit
    bBit := GetBit(APtrSrc, (@iHeaderIndex, (@iHeaderBits));

    // Same bit as previous
    if (bBit = bLastBit) then
    begin
      // increment run of bits
      Inc(iRunLength, 1);

      // Escape (same bits continuous over maximum run)
      if (iRunLength = AMaxRun) then
      begin
        // Write escape code to destination and initialize run
        APtDst[(ADst3BitOffset + 7) div 8] := 0;
        Inc(ADst3BitOffset, 8);
        iRunLength := 1;
      end
    end
  end;

  // Different bit as previous
end;

```

```

procedure RLE1u(APtrSrc : PByte; ASrcBitOffset : Cardinal; APtrDat : PByte; var ADstBitLen : Cardinal; AMaxRun : Cardinal); Boolean;
var
  bBit : Boolean;
  Value : Byte;
  iSrcBitOffset, nIndex, iRunLength, iHeaderIndex, iHeaderBits, iDstBitOffset : Cardinal;
begin
  // Reset srcbitoffset, dstbitoffset and clear first destination byte
  iSrcBitOffset := 0;
  iDstBitOffset := 0;
  iHeaderIndex := 0;
  iHeaderBits := 0;
  APtrDst[0] := 0;

  // Read first bit value (= byte) and update srcbitoffset
  bBit := (APtrSrc[iSrcBitOffset + 7] div 8) > 0;
  inc(iSrcBitOffset, 8);

  // Go through all Bytes until source is finished or destination buffer is full
  while ((iSrcBitOffset < ASrcBitOffset) and (iDstBitOffset < ADstBitLen)) do
  begin
    // Read runlength and update srcbitoffset
    iRunLength := APtrSrc[iSrcBitOffset + 7] div 8;
    inc(iSrcBitOffset, 8);

    // Process escape code

```

[0230]

```

if (iRunLength == 0) then
begin
  // Set bits for escape
  for iIndex := 1 to AMaxRun - 1 do
  begin
    // Set bit if destination buffer is not full
    if (iDstBitOffset < ADstBitLen) then
    begin
      // Set bit and/or go to next bit
      if (bBit) then
        SetBit(APtrDst, @iHeaderIndex, @iHeaderBits)
      else ClearBit(APtrDst, @iHeaderIndex, @iHeaderBits);
      Inc(iDstBitOffset, 1);
    end;
  end;
end;

// Process runlength
else
begin
  // Set bits for run
  for iIndex := 0 to iRunLength - 1 do
  begin
    // Set bit if destination buffer is not full
    if (iDstBitOffset < ADstBitLen) then
    begin
      // Set bit and/or go to next bit
      if (bBit) then
        SetBit(APtrDst, @iHeaderIndex, @iHeaderBits)
      else ClearBit(APtrDst, @iHeaderIndex, @iHeaderBits);
      Inc(iDstBitOffset, 1);
    end;
  end;

  // Change bit value
  if (bBit) then
    bBit := False
  else bBit := True;
end;

```

[0231] 尽管对于例如VLC、哈夫曼编码、算术编码、距离编码、RLE编码等等的熵编码方法来说，在不使用其他预测形式或其他附加信息的情况下，即，在不发生信息损失的情况下，

将数据压缩至小于其熵允许的大小是不可能的,但是并不意味着,在所有的情况下,实践中的最大程度压缩数据对应于最小的熵;换言之,在实践中,熵和数据压缩并不是完全相同的概念。将编码数据解码回至其相应的原始形式通常需要一些附加信息,在被压缩的实际数据之外,这些附加信息的传输有时需要不合理的额外的信息量。当待处理的数据量较大时,最小的熵通常对应于最佳的数据压缩,但是当待处理的处理量较小时,对数据的压缩并不总是如此简单;例如,1比特数据通常难于有效压缩,并且因此,可用于处理1比特信息的熵修正器10、60是潜在地、商业地、非常有价值的发明。此外,与应用格式处理之前的对应的原始数据相比,对具有更容易压缩的格式的数据进行处理通常导致格式数据的熵的降低。

[0233] 如上述,如果将代表编码数据的码字序列中的第一比特值与表示持续符号或比特的那些值分开传输,通常会更有效。但是,这种独立传输并不需要对应于完全地独立通信。在接下来的示例中,尽管将独立处理第一比特,但仍然需要将第一比特量增加到连续元素的总量中,因为否则在不将第一元素量数字设置为0的情况下,将无法区分下述两个示例:

[0234] 0 0 1 and 0 1 或

[0235] 1 1 0 and 1 0,

[0236] 在用于实现前述转义码时,并不需要将其设置为0,该转义码在连续的类似信号数超出示例中的最大行程长度时使用。

[0237] 在图1中,尽管数据D1和D2包括彼此类似的信息,但数据D2的大小并不是必须小于数据D1的大小的,但D2的熵小于D1。数据D2的大小依赖于采用的行程长度。例如,在数据D1中,数据元素的大小为1比特,而数据D2的大小依赖于采用的行程长度的大小。例如,如果数据D1具有104比特的大小,并且熵E=80,可以通过熵修正器10通过使用数据D2中的7个元素的最大行程长度对数据D1进行转换,数据D2具有40个元素,即,该40个元素为具有0到7范围内的值的3比特元素,且熵E=50,那么数据D2具有 40×3 比特=120比特的大小;在该示例中,数据D2在大小上大于数据D1。编码级20有益地采用VLC转换,该转换能够将数据D2压缩为52比特以提供数据D3。因此,应当理解,熵修正器10、60并不是通常各自提供数据压缩和解压缩,而是在与其他编码级一起使用时,具有提供数据压缩和解压缩的能力。

[0238] 本发明的实施例有益地被包括作为通信系统、视听消费产品,科技图像处理设备、计算装置,诸如智能手机的无线通信装置、数码相机、数字监视设备、互动式电脑游戏(仅为提及的几个示例)等的组成部分。增加的数据压缩能够降低数据存储需求,节省能量并实现更快的通信,上述均将被认为是工业上采用的有益技术效果。此外,实施例可以在软件产品、硬件、或者其组合上实现。可选地,该软件产品可以作为软件应用下载,即,“软件app”。

[0239] 在不背离本发明范围的情况下,可以对所述的实施例进行修改。本文所用的术语“包括”、“包含”、“由……组成”应当被理解为非排它的方式,即,允许未被明示描述的项目、部件、或元素的存在。单数或“一个”的使用应当被理解为与复数有关。括号内的序号的使用是为了帮助理解内容,不应当被当作限定的作用。

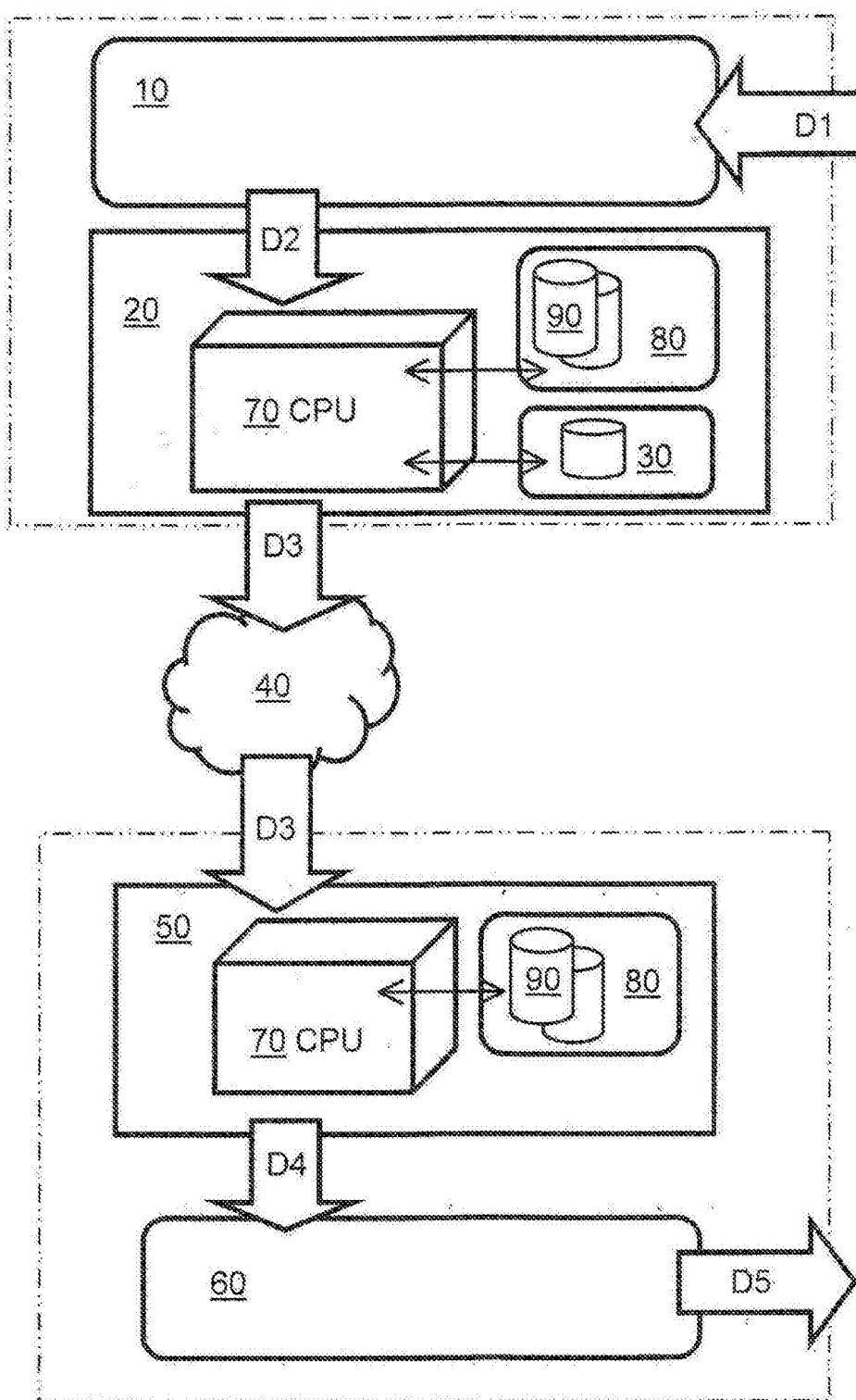


图1