(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년04월07일
(11) 등록번호 10-1610610
(24) 등록일자 2016년04월01일

(51) 국제특허분류(Int. Cl.)
H03M 7/00 (2006.01) *H03M 7/30* (2006.01)
(52) CPC특허분류
H03M 7/00 (2013.01)
H03M 7/3068 (2013.01)
(21) 출원번호 10-2015-7027208
(22) 출원일자(국제) 2014년02월27일
심사청구일자 2015년11월06일
(85) 번역문제출일자 2015년10월01일
(65) 공개번호 10-2015-0126890
(43) 공개일자 2015년11월13일
(86) 국제출원번호 PCT/EP2014/000510
(87) 국제공개번호 WO 2014/131517
국제공개일자 2014년09월04일
(30) 우선권주장
1303661.1 2013년03월01일 영국(GB)
(56) 선행기술조사문헌
US5864650 A

(73) 특허권자
구루로직 마이크로시스템스 오이
핀란드 투르쿠 20100 린난카투 34
(72) 발명자
칼레보 오시
핀랜드 토이잘라 에프아이엔-37800 케툰한타 1
(74) 대리인
김태홍, 김진희

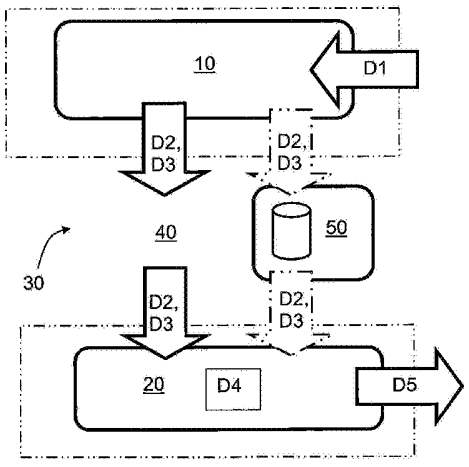
전체 청구항 수 : 총 34 항 심사관 : 조춘근

(54) 발명의 명칭 **인코더, 디코더 및 방법**

(57) 요 약

대응하는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 수치 값들의 시퀀스를 포함하는 입력 데이터(D1)를 인코딩하기 위한 인코더(10)가 제공되며, 인코더(10)는, 상기 입력 데이터(D1)에, 하나 이상의 대응하는 인코딩된 시퀀스를 생성하기 위하여 차 및/또는 합 인코딩의 형태를 적용하기 위한 데이터 프로세싱 장치를 포함하며, 상기 하나 이상의 대응하는 인코딩된 시퀀스는, 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여, 최대 값이 랩어라운드(wrap around)되고/되거나 최소 값이 랩어라운드되며, 상기 인코더(10)는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 사용되는 일련의 예측값에 대한 제1 디폴트 예측값을 사용하도록 동작될 수 있으며, 상기 인코딩된 데이터는 입력값, 예측값 및 인코딩 연산자를 이용하여 생성되는 것을 특징으로 한다.

대 표 도 - 도1



(52) CPC특허분류
H03M 7/3079 (2013.01)

명 세 서

청구범위

청구항 1

대응하는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 수치 값들의 시퀀스를 포함하는 입력 데이터(D1)를 인코딩하기 위한 인코더(10)에 있어서,

상기 입력 데이터(D1)에, 하나 이상의 대응하는 인코딩된 시퀀스를 생성하기 위하여 차, 합 또는 양자의 인코딩의 형태를 적용하기 위한 데이터 프로세싱 장치를 포함하며,

상기 하나 이상의 대응하는 인코딩된 시퀀스는, 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여, 최대 값이 랩어라운드(wrap around) 처리되거나, 최소 값이 랩어라운드 처리되거나 또는 양자 모두가 랩어라운드 처리되는 것을 특징으로 하는 인코더(10).

청구항 2

제1항에 있어서, 상기 데이터 프로세싱 장치는, 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하는데 사용하기 위한 상기 하나 이상의 대응하는 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋 값, 최소 또는 최대 값을 계산하기 위하여 상기 입력 데이터(D1), 상기 하나 이상의 대응하는 인코딩된 시퀀스 또는 양자 모두를 분석하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 3

제2항에 있어서, 상기 하나 이상의 오프셋 값은 “0” 값을 가지는 것을 특징으로 하는 인코더(10).

청구항 4

제1항에 있어서, 상기 인코더(10)는 하나 이상의 1-비트 값을 포함하는 수치 값을 프로세싱하도록 동작될 수 있고, 상기 인코더(10)는 비트-바이-비트 방식으로 상기 입력 데이터(D1)를 인코딩하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 5

제1항에 있어서, 상기 하나 이상의 대응하는 인코딩된 시퀀스는, 상기 입력 데이터(D1)의 순차적인 값들에서의 변화를 나타내는 것을 특징으로 하는 인코더(10).

청구항 6

제1항에 있어서, 상기 인코더(10)는 상기 입력 데이터(D1)를 개별적으로 인코딩되는 복수의 데이터 섹션으로 세분하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 7

제6항에 있어서, 상기 인코더(10)는, 데이터 압축이 상기 인코딩된 출력 데이터(D2 또는 D3)에서 달성될 수 있을 때에만 인코딩을 상기 데이터 섹션에 선택적으로 적용하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 8

제1항에 있어서, 상기 인코더(10)는, 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 사용되는 일련의 예측 값에 대하여 제1 디폴트 예측 값을 사용하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 9

제8항에 있어서, 상기 제1 디폴트 예측 값은, "0", 최소값(=lowValue), 최대값(=highValue), (최대값(highValue) + 최소값(lowValue) + 1) / 2 중 적어도 하나인 것을 특징으로 하는 인코더(10).

청구항 10

제6항에 있어서, 상기 인코더(10)는, 런-렝쓰 코딩(RLE), 호프만 코딩, 가변 길이 인코딩(VLE), 레인지 코딩 또는 산술 코딩을 이용하는 인코딩에 대하여 효율적인 상호 유사한 비트들의 런-렝쓰에 따라서 상기 입력 데이터(D1)를 복수의 섹션으로 세분하도록 동작될 수 있는 것을 특징으로 하는 인코더(10).

청구항 11

제1항 내지 제10항 중 어느 한 항에 있어서, 상기 프로세싱 장치는, 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 컴퓨터 프로그램을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 구현되는 것을 특징으로 하는 인코더(10).

청구항 12

대응하는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 수치 값의 시퀀스를 포함하는 입력 데이터(D1)를 인코딩하기 위한 인코더(10)를 이용하는 방법에 있어서,

(a) 하나 이상의 대응하는 인코딩된 시퀀스를 생성하기 위하여, 상기 입력 데이터(D1)에 차, 합 또는 양자의 인코딩의 형태를 적용하도록 데이터 프로세싱 장치를 이용하는 단계와,

(b) 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여, 상기 하나 이상의 대응하는 인코딩된 시퀀스가 최대 값이 랩어라운드 처리되거나, 최소 값이 랩어라운드 처리되거나, 또는 양자 모두가 랩어라운드 처리되어지도록 하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 13

제12항에 있어서, 상기 방법은 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하는데 사용하기 위한 상기 하나 이상의 대응하는 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋 값, 최소 또는 최대 값을 계산하기 위하여 상기 입력 데이터(D1), 상기 하나 이상의 대응하는 인코딩된 시퀀스, 또는 양자 모두를 분석하도록 상기 데이터 프로세싱 장치를 이용하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 14

제13항에 있어서, 상기 방법은, 하나 이상의 1-비트 값을 포함하는 수치 값을 프로세싱하는 단계를 포함하며, 상기 인코더(10)는 비트-바이-비트 방식으로 상기 입력 데이터(D1)를 인코딩하도록 동작될 수 있는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 15

제12항에 있어서, 상기 방법은 상기 입력 데이터(D1)를 개별적으로 인코딩되는 복수의 데이터 섹션으로 세분하도록 데이터 프로세싱 장치를 이용하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 16

제15항에 있어서, 상기 방법은 데이터 압축이 상기 인코딩된 출력 데이터(D2 또는 D3)에서 달성될 수 있을 때에만 인코딩의 형태를 상기 데이터 섹션에 선택적으로 적용하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 17

제12항에 있어서, 상기 방법은 상기 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 사용되는 일련의 예측 값에 대하여 제1 디폴트 예측 값을 사용하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 18

제12항 내지 제17항 중 어느 한 항에 있어서, 상기 방법은 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 컴퓨터 프로그램을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 상기 데이터 프로세싱 장치를 구현하는 단계를 포함하는 것을 특징으로 하는 인코더(10)를 이용하는 방법.

청구항 19

대응하는 디코딩된 출력 데이터(D5)를 생성하도록 인코딩된 데이터(D2 또는 D3)를 디코딩하는 디코더(20)에 있어서,

상기 디코더(20)는,

상기 인코딩된 데이터(D2 또는 D3)의 하나 이상의 부분을 프로세싱하기 위한 데이터 프로세싱 장치를 포함하며,

상기 데이터 프로세싱 장치는, 상기 하나 이상의 부분의 하나 이상의 대응하는 인코딩된 시퀀스에 차, 합 또는 양자의 디코딩의 형태를 적용하도록 동작될 수 있고, 디코딩된 출력 데이터(D5)를 생성하기 위하여, 상기 하나 이상의 인코딩된 시퀀스는, 최대 값이 랩어라운드 처리되거나, 최소값이 랩어라운드 처리되거나 또는 양자 모두가 랩어라운드 처리되는 것을 특징으로 하는 디코더(20).

청구항 20

제19항에 있어서, 상기 디코더(20)는 하나 이상의 1-비트 값을 포함하는 인코딩된 데이터(D2 또는 D3)를 디코딩하도록 동작될 수 있고, 상기 디코더(20)는 비트-바이-비트 방식으로 상기 인코딩된 데이터(D2 또는 D3)를 디코딩하도록 동작될 수 있는 것을 특징으로 하는 디코더(20).

청구항 21

제19항에 있어서, 상기 데이터 프로세싱 장치는, 상기 디코딩된 출력 데이터(D5)를 생성하는데 사용하기 위한 상기 하나 이상의 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋 값, 최소 또는 최대 값을 수신하도록 동작될 수 있는 것을 특징으로 하는 디코더(20).

청구항 22

제21항에 있어서, 상기 하나 이상의 오프셋 값은 “0” 값을 가지는 것을 특징으로 하는 디코더(20).

청구항 23

제19항에 있어서, 상기 하나 이상의 대응하는 인코딩된 시퀀스는, 상기 인코딩된 데이터(D2 또는 D3)로 인코딩된 순차적인 값에서의 변화를 나타내는 것을 특징으로 하는 디코더(20).

청구항 24

제19항에 있어서, 상기 데이터 프로세싱 장치는, 디코딩되는 일련의 데이터에서의 제1 예측 값의 디폴트 값을 추정하도록 동작될 수 있는 것을 특징으로 하는 디코더(20).

청구항 25

제24항에 있어서, 상기 디폴트 값은 “0” 을 가지는 것을 특징으로 하는 디코더(20).

청구항 26

제19항 내지 제25항 중 어느 한 항에 있어서, 상기 데이터 프로세싱 장치는, 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 컴퓨터 프로그램을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 구현되는 것을 특징으로 하는 디코더(20).

청구항 27

대응하는 디코딩된 출력 데이터(D5)를 생성하도록 인코딩된 데이터(D2 또는 D3)를 디코딩하는 디코더(20)를 이용하는 방법에 있어서,

상기 방법은,

상기 인코딩된 데이터(D2 또는 D3)의 하나 이상의 부분을 프로세싱하기 위한 데이터 프로세싱 장치를 이용하는 단계를 포함하며,

상기 데이터 프로세싱 장치는, 상기 하나 이상의 부분의 하나 이상의 대응하는 인코딩된 시퀀스에 차, 합 또는

양자의 디코딩의 형태를 적용하도록 동작될 수 있고, 디코딩된 출력 데이터(D5)를 생성하기 위하여, 상기 하나 이상의 인코딩된 시퀀스는, 최대 값이 랩어라운드 처리되거나 최소값이 랩어라운드 처리되거나 또는 양자 모두가 랩어라운드 처리되는 것을 특징으로 하는 디코더(20)를 이용하는 방법.

청구항 28

제27항에 있어서, 상기 방법은, 하나 이상의 1-비트 값을 포함하는 인코딩된 데이터(D2 또는 D3)를 디코딩하는 디코더(20)를 동작시키는 단계를 포함하며, 상기 디코더(20)는 상기 인코딩된 데이터(D2 또는 D3)를 비트-바이-비트 방식으로 디코딩하도록 동작될 수 있는 것을 특징으로 하는 디코더(20)를 이용하는 방법.

청구항 29

제27항에 있어서, 상기 방법은 상기 디코딩된 출력 데이터(D5)를 생성하는데 사용하기 위한 상기 하나 이상의 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋 값, 최소 및 최대 값을 계산하도록 상기 데이터 프로세싱 장치를 동작시키는 단계를 포함하는 것을 특징으로 하는 디코더(20)를 이용하는 방법.

청구항 30

제27항에 있어서, 상기 방법은 디코딩되는 일련의 데이터에서의 제1 예측 값의 디폴트 값을 추정하도록 상기 데이터 프로세싱 장치를 동작시키는 단계를 포함하는 것을 특징으로 하는 디코더(20)를 이용하는 방법.

청구항 31

제27항 내지 제30항 중 어느 한 항에 있어서,

상기 방법은 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 컴퓨터 프로그램을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 상기 데이터 프로세싱 장치를 구현하는 단계를 포함하는 것을 특징으로 하는 디코더(20)를 이용하는 방법.

청구항 32

코덱(30)에 있어서,

대응하는 인코딩된 데이터(D2)를 생성하기 위하여 입력 데이터(D1)를 인코딩하는 제1항에 기재된 적어도 하나의 인코더(10)와,

대응하는 디코딩된 데이터(D5)를 생성하기 위하여 인코딩된 데이터(D2 또는 D3)를 디코딩하는 제19항에 기재된 적어도 하나의 디코더(20)를 포함하는 코덱(30).

청구항 33

컴퓨터 프로그램을 포함하는 컴퓨터 판독 가능한 기록 매체에 있어서,

상기 컴퓨터 프로그램은 제12항 내지 제17항 중 어느 한 항에 기재된 데이터 인코딩 방법을 실행하기 위한 컴퓨팅 하드웨어 상에서 실행될 수 있는 것을 특징으로 하는 컴퓨터 판독 가능한 기록 매체.

청구항 34

컴퓨터 프로그램을 포함하는 컴퓨터 판독 가능한 기록 매체에 있어서,

상기 컴퓨터 프로그램은 제27항 내지 제30항 중 어느 한 항에 기재된 데이터 디코딩 방법을 실행하기 위한 컴퓨팅 하드웨어 상에서 실행될 수 있는 것을 특징으로 하는 컴퓨터 판독 가능한 기록 매체.

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

발명의 설명

기 술 분 야

[0001] 본 개시는 인코더 예컨대 다이렉트 ODelta 연산자를 사용하도록 동작될 수 있는 인코더와 관련된다. 또한, 본 개시는 데이터를 인코딩하는 방법 예컨대, 다이렉트 ODelta 연산자를 사용함으로써 데이터를 인코딩하는 방법과 관련된다. 또한, 본 개시는 또한 인코딩된 데이터를 디코딩하는 디코더 예컨대, 역 ODelta 연산자를 적용하도록 동작될 수 있는 디코더와 관련된다. 부가적으로, 본 개시는 인코딩된 데이터를 디코딩하는 방법 예컨대, 역 ODelta 연산자를 적용함으로써 인코딩된 데이터를 디코딩하는 방법과 관련된다. 또한 부가적으로, 본 개시는 비순간적(비일시적) 머신 판독 가능한 데이터 저장 매체 상에 기록되는 소프트웨어 제품과 관련되며, 여기서 소프트웨어 제품은 전술한 방법을 구현하기 위한 컴퓨터 하드웨어 상에서 실행될 수 있다.

배 경 기 술

[0002] 클라우데 이 샤논(Claude E. Shannon)은 현대의 통신 시스템에 대한 기초를 제공한 통신의 수학적 이론은 제안하였다. 또한, 다양한 현대의 인코딩 방법들이 전술한 수학적 이론의 지식 내에서 발전되었다. 현대의 기술적 지식의 개요를 제공하는 정보 소스들의 리스트가 표 1에 제공된다.

표 1

[0003] 공지된 기술 문헌

초기 문헌	세부사항
P1	“가변 길이 코드”, 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Variable-length_code
P2	“런-렝쓰 인코딩”, 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Run-length_encoding
P3	“호프만 코딩”, 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Huffman_coding
P4	“산술 코딩(Arithmetic coding)”, 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Arithmetic_coding
P5	“통신의 수학적 이론(A Mathematic Theory of Communication)”, Shannon, Claude E. (1948) (2012년 11월 28일 액세스) URL: http://cm:bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
P6	“델타 인코딩”, 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Delta_coding
P7	샤논의 소스 코딩 이론; 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia.org/wiki/Source_coding_theorem
P8	“엔트로피” - 위키피디아 (2012년 11월 28일 액세스) URL: http://en.wikipedia/wiki/Entropy

[0004] 유럽 특허 EP 1 2376 974 B1("패킷 헤더 압축을 위한 방법 및 장치", Alcatel Lucent (FR))에는, 데이터 패킷의 송신 방법이 기재되어 있으며, 여기서 데이터 패킷은 압축된 값을 포함하는 필드(CF)를 구비한다. 압축된 값은 2개의 연속적인 데이터 패킷들 사이에서 전개되는 값을 나타내며 미리 정해진 간격(이하, “해석 간격”)을 포함한다. 이 방법은 이하의 단계 즉,

[0005] (i) 압축되는 값과 미리 정해진 랩 어라운드 경계 사이의 거리가 미리 정해진 임계값보다 작으면, 압축된 값에 부가적인 비트를 첨부하는 단계로서, 여기서 부가적인 비트는, 압축되는 값의 상대적인 위치를 랩 어라운드 경계에 명료하게 표시하는 것인 상기 첨부하는 단계와,

[0006] (ii) 수신기에서, 압축된 값의 모든 수신된 비트에 따라서 제1 해석 간격을 계산하는 단계와,

[0007] (iii) 제1 해석 간격의 하나의 값보다 큰 값이 수신되는 압축된 값과 일치하면 랩 어라운드를 시그널링하는 단계와,

[0008] (iv) 랩 어라운드의 시그널링시에, 부가적인 비트를 제외하고 압축된 값의 모든 수신된 비트에 따라서 제2 해석 간격을 계산하는 단계와,

[0009] (v) 부가적인 비트를 이용하여 제2 해석 간격 내의 압축해제된 값을 명확화하는 단계를 포함한다.

[0010] 샤논 엔트로피의 정의는 표 1에 목록화된 문헌 P7 및 P8에 제공된다. 주어진 데이터에 존재하는 엔트로피를 압축하도록 동작될 수 있는 복수의 상이한 압축 방법이 존재하며, 이러한 방법들은 때때로 엔트로피를 변경하기 위하여, 예컨대 주어진 데이터에 대한 더 큰 무손실 압축비를 획득하기 위하여 사용되며; 이러한 엔트로피 변경 방법은, 예컨대 표 1의 문헌 P2에 기재된 바와 같은 런-렝쓰 인코딩(RLE), 표 1의 문헌 P1에 기재된 바와 같은 가변 길이 코딩(VLC), 표 1의 문헌 P3에 기재된 바와 같은 호프만 코딩, 표 1의 문헌 P6에 기재된 바와 같은 델타 코딩, 및 표 1의 문헌 P4에 기재된 바와 같은 산술 코딩 즉, 레인지 코딩을 포함한다. 이러한 방법들은 알파벳 문자, 숫자, 바이트 및 워드를 나타내는 데이터를 압축하도록 유익하게 사용된다. 그러나, 이러한 방법들은 비트 레벨에서 주어진 데이터를 압축하도록 잘 구성되지 않으며, 이러한 이유로, 비트-바이-비트로 변경될 수 있는 이러한 주어진 데이터를 압축하기에는 매우 적합하지 않다.

[0011] 예컨대 표 1의 문헌 P6에 기재된 델타 인코딩은, 포지티브 오리지널 데이터 값들로부터 포지티브 또는 네가티브 값일 수 있는 델타 값들을 생성하도록 동작될 수 있다. 또한, 사용된 데이터 요소들의 사이즈에 기초하여, 8-비트, 16-비트 또는 32-비트 랩 어라운드에 사용되는 공지된 델타 인코더의 구현이 존재한다. 그러나, 8-비트, 16-비트 또는 32-비트 값 이외의 레짐(regime)에 대하여 최적화되는 현대의 델타 인코더가 부족하다. 특히, 공지된 델타 인코더는 일반적으로 기인하는 3개의 상이한 값 즉, “-1”, “0” 및 “1”로서, 예컨대 비트-바이-

비트 인코딩에서와 같이 코딩 오리지널 비트 값들이 즉, "0" 과 "1" 인 경우에 특히 비효율적이다.

[0012] 모든 종류의 데이터는 저장 공간을 소모하며, 통신 시스템 전송 용량은, 이러한 데이터가 하나의 위치로부터 다른 위치로 이용될 때 필요하게 된다. 데이터의 양이, 예컨대 3-차원 비디오 콘텐츠와 같은 멀티미디어 발전의 결과로서 증가함에 따라서, 더 많은 저장 공간 및 전송 용량이 데이터를 처리하는데 필요하게 되며, 또한 더 많은 에너지가 데이터의 양이 증가함에 따라서 필요하게 된다. 전세계적으로, 통신되는 데이터의 양은 시간에 대하여 점진적으로 증가되며; 예컨대 인터넷은 거대한 양의 데이터를 포함하며, 이들의 일부는 복수의 카피로 저장된다. 또한, 예컨대 데이터의 사이즈를 감소시킬 때에 사용하기 위하여, 데이터와 연관된 엔트로피(E)를 압축하기 위하여 현재 이용가능한 방법들이 존재한다. 또한, 예컨대 델타 인코딩 및 런-렝쓰 인코딩(RLE)에 대하여 엔트로피를 변경하는 이용가능한 방법들이 또한 존재하지만, 현재 실현가능한 더욱 더 큰 데이터 압축을 제공하기 위해서 개선된 방법들이 요구된다.

발명의 내용

해결하려는 과제

[0013] 예컨대 비트-바이-비트 인코딩에서, 예컨대 오리지널 데이터 내의 데이터 요소들의 모든 값이 사용되지 않고/않거나 델타 인코딩 방법과 결합하여 사용되는 이전의 또는 후속의 인코딩 방법이, 디코딩되는 데이터에 대하여 원래 사용되는 비트 다이나믹보다 더 높은 비트 포맷을 요구하는 경우에, 오리지널 데이터의 더욱 빠르고 더욱 효율적인 인코딩을 달성할 수 있게 하는 공지된 델타 인코딩 방법의 최적 이용에 대한 필요성이 또한 존재한다.

과제의 해결 수단

[0014] 본 개시는 개선된 형태의 델타 인코더 즉, "다이렉트 ODelta 인코더" 를 제공하도록 시도하며, 이러한 인코더는, 데이터의 다른 값들뿐만 아니라 개별적인 비트들을 인코딩할 때 즉, 비트-바이-비트 인코딩에 있어서 더욱 효율적이다.

[0015] 또한, 본 개시는 데이터를 델타 인코딩하는 개선된 방법 즉, 데이터를 다이렉트 ODelta 인코딩하는 방법을 제공하도록 시도한다.

[0016] 또한, 본 개시는 인코딩된 데이터 예컨대 ODelta 인코딩된 데이터를 디코딩하는 개선된 디코더 즉, 역 ODelta 디코더를 제공하도록 시도한다.

[0017] 부가적으로, 본 개시는 인코딩된 데이터 예컨대 ODelta 인코딩 데이터를 디코딩하는 개선된 방법 즉, 데이터의 역 ODelta 디코딩 방법을 제공하도록 시도한다.

[0018] 제1 양태에 따르면, 첨부된 청구항 1에 청구된 인코더가 제공되는데, 대응하는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 수치 값들의 시퀀스를 포함하는 입력 데이터(D1)를 인코딩하기 위한 인코더가 제공되며, 상기 인코더는, 하나 이상의 대응하는 인코딩된 시퀀스를 생성하기 위하여 차 및/또는 합 인코딩의 형태를 입력 데이터(D1)에 적용하기 위한 데이터 프로세싱 장치를 포함하며, 상기 하나 이상의 대응하는 인코딩된 시퀀스는, 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여, 최대값이 랩 어라운드되고/되거나 최소값이 랩 어라운드되며, 상기 인코더(10)는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 사용되는 일련의 예측값에 대한 제1 디폴트 예측값을 사용하도록 동작될 수 있고, 상기 인코딩된 출력 데이터는 입력값, 예측값 및 인코딩 연산자를 이용하여 생성된다.

[0019] 유익하게, 랩 어라운드는, 인코더를 구현할 때 네가티브 차이 또는 너무 큰 값이 발생하는 것을 피하도록 사용된다. "수치 값" 이라는 용어는, 비트들의 그룹(바이너리보다 더 높은 차수)뿐만 아니라 개별적인 비트(바이너리)를 포함하도록 해석될 것이다.

[0020] 본 발명은, 차 및/또는 합 인코딩과 랩 어라운드의 조합이 인코딩된 데이터(D2)를 생성하기 위하여 입력 데이터(D1)의 유용한 엔트로피 변경을 제공할 수 있고, 이것이 인코딩된 데이터(D2)로부터 인코딩된 데이터(D3)를 생성할 때 주어진 엔트로피 인코더 내에서 개선된 데이터 압축을 달성되게 한다는 점에서 유리하다.

[0021] 옵션적으로, 인코더에 대하여, 데이터 프로세싱 장치는, 인코딩된 출력 데이터(D2 또는 D3)를 생성하는데 사용되는 하나 이상의 대응하는 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋, 최소 및/또는 최대 값을 계산하기 위하여 입력 데이터(D1) 및/또는 하나 이상의 대응하는 인코딩된 시퀀스를 분석하도록 동작된다. 또한 옵션적으로, 인코더에 대하여, 하나 이상의 오프셋 값은 "0" 값을 가진다.

[0022] 옵션적으로, 인코더는 하나 이상의 1-비트 값을 포함하는 수치 값을 프로세싱하도록 동작되며, 인코더는 비트-바이-비트 방식으로 입력 데이터(D1)를 인코딩하도록 동작된다.

[0023] 옵션적으로, 인코더에 있어서, 하나 이상의 대응하는 인코딩된 시퀀스는 입력 데이터(D1)의 순차적인 값에서의 변화를 나타낸다.

[0024] 옵션적으로, 인코더는, 랩어라운드 값(wrapValue)을 사용하도록 동작하며, 랩어라운드 값은 최대 값(highValue) - 최소 값(lowValue) + 1이다. 이러한 랩어라운드 값은, 인코딩된 출력 데이터(D2 또는 D3)를 생성할 때, 최소 값(lowValue), 종종 네가티브 값보다 더 작은, 또는 최대 값(highValue) 보다 더 높은 이러한 값들이 발생하는 것을 피하기 위하여, 유익하게 사용된다.

[0025] 옵션적으로, 인코더는 입력 데이터(D1)를 개별적으로 인코딩되는 복수의 데이터 섹션으로 세분(subdivide)하도록 동작될 수 있다. 또한 옵션적으로, 인코더는 데이터 압축이 인코딩된 출력 데이터(D2 또는 D3)에서 달성될 수 있을 때에만 데이터 섹션에 선택적으로 인코딩을 적용하도록 동작될 수 있다.

[0026] 또한 옵션적으로, 인코더에 있어서, 제1 디폴트 예측 값은 "0"이다. 또한 옵션적으로, 제1 예측 값은 또한 최소 값(lowValue), 최대 값(highValue), [최대 값(highValue) + 최소값(lowValue) +1]/2 등일 수도 있다.

[0027] 옵션적으로, 인코더는 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 부가적인 인코딩을 적용하도록 동작될 수 있으며, 여기서 부가적인 인코딩은, 런렝쓰 인코딩(run-length encoding; RLE), 스플릿 RLE, 엔트로피 모디파이어(entropy modifier; EM), 가변 길이 코딩(variable length coding; VLC), 호프만 코딩, 산술 코딩, 레인지 코딩 중 적어도 하나를 포함한다.

[0028] 옵션적으로, 인코더는, 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 호프만 코딩, 가변 길이 인코딩(variable length encoding; VLE), 레인지 코딩 및/또는 산술 코딩을 이용하는 인코딩에 효율적인 상호 유사한 비트들의 런-렝쓰에 따라서 입력 데이터(D1)를 복수의 섹션으로 세분하도록 동작될 수 있다.

[0029] 옵션적으로, 인코더에 있어서, 프로세싱 장치는, 비일시적 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 소프트웨어 제품을 실행하도록 동작되는 컴퓨팅 하드웨어를 이용하여 구현된다.

[0030] 제2 양태에 따르면, 인코딩된 대응하는 출력 데이터(D2 또는 D3)를 생성하기 위하여 수치 값들의 시퀀스를 포함하는 입력 데이터(D1)를 인코딩하는 인코더를 이용하는 방법이 제공되며, 이 방법은,

[0031] (a) 하나 이상의 대응하는 인코딩된 시퀀스를 생성하기 위하여 차 및/또는 합 인코딩의 형태를 입력 데이터(D1)에 적용하기 위한 인코더의 데이터 프로세싱 장치를 이용하는 단계와,

[0032] (b) 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여, 하나 이상의 대응하는 인코딩된 시퀀스에서 최대 값이 랩 어라운드되고/되거나 최소 값이 랩 어라운드되도록 데이터 프로세싱 장치를 이용하는 단계를 포함한다.

상기 방법은 인코딩된 출력 데이터(D2 또는 D3)를 생성하기 위하여 사용되는 일련의 예측값에 대한 제1 디폴트 예측값을 사용하는 단계를 포함하며, 상기 인코딩된 데이터는 입력값, 예측값 및 인코딩 연산자를 이용하여 생성된다.

[0033] 유익하게, 랩 어라운드는, 인코딩 방법을 구현할 때 네가티브 차, 또는 너무 큰 값이 발생하는 것을 피하기 위하여 사용된다. "수치 값"이라는 용어는, 비트들의 그룹(바이너리 보다 더 높은 차수)뿐만 아리라 개별적인 비트(바이너리)를 포함하도록 해석될 것이다.

[0034] 옵션적으로, 방법은 인코딩된 출력 데이터(D2 또는 D3)를 생성하는데 사용하기 위한 하나 이상의 대응하는 인코딩된 시퀀스에 적용하기 위한 하나 이상의 오프셋, 최소 및/또는 최대 값을 계산하기 위하여 입력 데이터(D1) 및/또는 하나 이상의 대응하는 인코딩된 시퀀스를 분석하는 데이터 프로세싱 장치를 이용하는 단계를 포함한다. 또한 옵션적으로, 방법은 하나 이상의 오프셋 값에 대하여 "0" 값을 이용하는 단계를 포함한다.

[0035] 옵션적으로, 방법은 하나 이상의 1-비트 값을 포함하는 수치 값을 프로세싱하는 단계와 입력 데이터(D1)를 비트-바이-비트 방식으로 인코딩하는 단계를 포함한다. 또한 옵션적으로, 방법은 하나 이상의 오프셋 값에 대하여 "0" 값을 이용하는 단계를 포함한다.

[0036] 옵션적으로, 방법을 구현할 때, 하나 이상의 대응하는 인코딩된 시퀀스는 입력 데이터(D1)의 순차적인 값들에서의 변화를 나타낸다.

[0037] 옵션적으로, 방법은 랩어라운드 값(wrapValue)을 사용하기 위하여 인코더를 동작시키는 단계를 포함하며, 여기

서 랩어라운드 값은 최대값(highValue) - 최소값(lowValue) + 1이다. 이러한 랩어라운드는, 인코딩된 출력 데이터(D2 또는 D3)를 생성할 때, 최소 값(lowValue), 종종 네가티브 값보다 작거나 또는 최대값(highValue)보다 더 높은 이러한 값들이 발생하는 것을 피하기 위하여, 유익하게 사용된다.

[0038] 옵션적으로, 방법은 입력 데이터(D1)를 개별적으로 인코딩되는 복수의 데이터 섹션으로 세분하는 단계를 포함한다. 또한 옵션적으로, 방법은 데이터 압축이 인코딩된 출력 데이터(D2 또는 D3)에서 달성될 수 있을 때에만 데이터의 섹션에 선택적으로 인코딩을 적용하는 단계를 포함한다.

[0039] 또한 옵션적으로, 방법에 있어서, 디폴드 제1 예측값은 "0"이다. 또한 옵션적으로, 제1 예측값은 또한 최소 값(lowValue), 최대 값(highValue), [최대 값(highValue) + 최소 값(lowValue) +1]/2 등일 수 있다.

[0040] 옵션적으로, 방법은 인코딩된 출력 데이터(D2)를 생성하기 위하여 부가적인 인코딩을 적용하는 단계를 포함하며, 상기 부가적인 인코딩은, 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 가변 길이 코딩(VLC), 호프만 코딩, 산술 코딩, 레인지 코딩 중 적어도 하나를 포함한다.

[0041] 또한 옵션적으로, 방법은, 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 호프만 코딩, 가변 길이 인코딩(VLE), 레인지 코딩 및/또는 산술 코딩을 이용하여 인코딩에 효율적인 상호 유사한 비트들의 런렝쓰에 따라서 입력 데이터(D1)를 복수의 섹션으로 세분하는 단계를 포함한다.

[0042] 또한 옵션적으로, 방법은 비일시적 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 소프트웨어 제품을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 프로세싱 장치를 구현하는 단계를 포함한다.

[0043] 제3 양태에 따르면, 디코딩된 출력 데이터(D5)를 생성하기 위하여 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하는 디코더가 제공되며, 상기 디코더는 인코딩된 데이터(D2, D3 또는 D4)의 하나 이상의 부분을 프로세싱하기 위한 데이터 프로세싱 장치를 포함하며, 데이터 프로세싱 장치는 상기 하나 이상의 부분의 하나 이상의 대응하는 인코딩된 시퀀스에 차 및/또는 합 디코딩의 형태를 적용하도록 동작될 수 있고, 하나 이상의 인코딩된 시퀀스는 디코딩된 출력 데이터(D5)를 생성하기 위하여, 최대 값이 랩 어라운드되고/되거나 최소 값이 랩 어라운드되며, 상기 데이터 프로세싱 장치는 디코딩되는 일련의 데이터에서의 제1 예측값의 디폴트값을 추정하도록 동작될 수 있다.

[0044] 옵션적으로, 디코더는, 하나 이상의 1-비트 값을 포함하는 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하도록 동작될 수 있고, 디코더는 비트-바이-비트 방식으로 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하도록 동작될 수 있다.

[0045] 옵션적으로, 디코더는 디코딩된 출력 데이터(D5)를 생성하는데 사용하기 위한 하나 이상의 인코딩된 시퀀스에 적용하기 위한 하나 이상의 값을 데이터 프로세싱 장치가 수신하게 동작될 수 있도록 배치된다. 또한 옵션적으로, 수신된 값들은 최대 값, 최소 값 및/또는 오프셋 값이다. 또한 옵션적으로, 디코더는 "0" 값을 가지는 하나 이상의 오프셋 값에 대한 함수에 대하여 동작될 수 있다.

[0046] 옵션적으로, 디코더는 인코딩된 데이터(D2, D3 또는 D4)로 인코딩된 순차적인 값들에서의 변화를 나타내는 하나 이상의 대응하는 인코딩된 시퀀스에 대한 함수에 대하여 동작될 수 있다.

[0047] 옵션적으로, 디코더는 랩어라운드 값(wrapValue)을 사용하도록 동작될 수 있으며, 여기서 랩어라운드 값은 최대 값(highValue) - 최소 값(lowValue) + 1과 동일하다. 이러한 랩어라운드는, 디코딩된 출력 데이터(D5)를 생성할 때, 최소 값(lowValue), 종종 네가티브 값보다 작거나 또는 최대 값(highValue)보다 더 높은 이러한 값들이 발생하는 것을 피하기 위하여 사용된다.

[0048] 옵션적으로, 디코더는, 데이터 프로세싱 장치가 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 가변 길이 코딩(VLC), 호프만 코딩, 산술 코딩, 레인지 코딩 중 적어도 하나의 역을 통하여 프로세싱되는 데이터에 적용하게 동작될 수 있도록 동작된다.

[0049] 또한 옵션적으로, 디폴트 예측값은 "0" 값을 가진다. 그러나, 예측 값은 또한 전술한 바와 같이, 최소 값(lowValue), 최대 값(highValue), [최대값(highValue) + 최소 값(lowValue) +1]/2 등일 수 있다.

[0050] 옵션적으로, 디코더에 있어서, 프로세싱 장치는, 비일시적 머신 판독 가능한 데이터 저장 매체 상에 기록된 하나 이상의 소프트웨어 제품을 실행하도록 동작가능한 컴퓨팅 하드웨어를 이용하여 구현된다.

[0051] 제4 양태에 있어서, 대응하는 디코딩된 출력 데이터(D5)를 생성하기 위하여 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하는 디코더를 사용하는 방법이 제공되며, 이 방법은, 인코딩된 데이터(D2, D3 또는 D4)의 하나 이상의

부분들을 프로세싱하기 위한 데이터 프로세싱 장치를 이용하는 단계를 포함하며, 상기 데이터 프로세싱 장치는, 하나 이상의 부분들의 하나 이상의 대응하는 인코딩된 시퀀스에 차 및/또는 합 디코딩의 형태를 적용하도록 동작될 수 있고, 상기 하나 이상의 인코딩된 시퀀스는, 디코딩된 출력 데이터(D5)를 생성하기 위하여, 최대 값이 랩 어라운드되고/되거나 최소 값이 랩 어라운드되며, 상기 데이터 프로세싱 장치는 디코딩되는 일련의 데이터에서의 제1 예측값의 디폴트값을 추정하기 위하여 동작될 수 있다.

[0052] 옵션적으로, 방법은 하나 이상의 1-비트 값을 포함하는 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하는 단계와, 인코딩된 데이터(D2, D3 또는 D4)를 비트-바이-비트 방식으로 디코딩하는 디코더를 이용하는 단계를 포함한다. 또한 옵션적으로, 방법에 있어서, 하나 이상의 오프셋 값은 "0" 값을 가진다.

[0053] 옵션적으로, 방법은, 디코딩된 출력 데이터(D5)를 생성하는데 사용하기 위한 하나 이상의 대응하는 인코딩된 시퀀스에 적용하기 위한 입력 데이터(D1) 및/또는 하나 이상의 대응하는 인코딩된 시퀀스(D2, D3 또는 D4)의 값을 수신하는 데이터 프로세싱 장치를 이용하는 단계를 포함한다.

[0054] 옵션적으로, 방법에 있어서, 하나 이상의 대응하는 인코딩된 시퀀스는 인코딩된 데이터(D2, D3 또는 D4)로 인코딩된 순차적인 값들에서의 변화를 나타낸다.

[0055] 옵션적으로, 방법은 디코딩된 출력 데이터(D5)를 생성할 때 네가티브 차이 또는 너무 큰 값이 발생하는 것을 피하기 위하여 최대 값을 랩 어라운드하거나 또는 최소 값을 랩 어라운드하는 데이터 프로세싱 장치를 이용하는 단계를 포함한다.

[0056] 옵션적으로, 방법에 있어서, 데이터 프로세싱 장치는, 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 가변 길이 코딩(VLC), 호프만 코딩, 산술 코딩, 레인지 코딩 중 적어도 하나의 역을 통하여 프로세싱되는 데이터에 적용하도록 동작될 수 있다.

[0057] 또한 옵션적으로, 방법에 있어서, 디폴트 예측 값은 "0" 값을 가진다.

[0058] 옵션적으로, 방법에 있어서, 프로세싱 장치는 비일시적 머신 판독 가능한 데이터 저장 매체 상에 기록되는 하나 이상의 소프트웨어 제품을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어를 이용하여 구현된다.

[0059] 제5 양태에 따르면, 대응하는 디코딩된 출력 데이터(D5)를 생성하도록 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하는 디코더를 이용하는 방법이 제공되며, 이 방법은,

[0060] (a) 인코딩된 데이터(D2, D3 또는 D4)가 변형된 데이터의 순차적인 값에서의 변화를 나타내는 적어도 하나의 인코딩된 시퀀스를 포함하며, 최대 값을 랩 어라운딩하거나 또는 최소값을 랩 어라운드하는 것을 고려하여, 인코딩된 데이터(D2, D3 또는 D4)의 하나 이상의 부분들을 디코딩하기 위하여 인코딩된 데이터(D2, D3 또는 D4)를 프로세싱하기 위한 데이터 프로세싱 장치를 이용하는 단계와,

[0061] (b) 대응하는 프로세싱된 데이터를 생성하고, 디코딩된 출력 데이터(D5)를 생성하기 위하여 적어도 하나의 프리-오프셋 값 및/또는 포스트-오프셋 값을 이용하여 하나 이상의 부분들을 변형하는 데이터 프로세싱 장치를 이용하는 단계를 포함한다.

[0062] 전술한 양태를 참조하면, 오프셋 값에 대하여, 옵션적으로, 디코더에 있어서, 데이터 프로세싱 장치는 인코딩된 데이터(D2, D3 또는 D4) 내의 값들의 범위를 수신하고, 이로부터 적어도 하나의 프리-오프셋 및/또는 포스트-오프셋 값을 유도하도록 동작될 수 있다. 포스트-오프셋 값은, 역 연산자, 예컨대 다이렉트 역 ODelta 연산자의 사용 이전에 변형된 데이터를 생성하는데 사용될 수 있고, 프리-오프셋 값은 역 연산자 이후에 데이터를 변형시키는데 사용된다. 적어도 하나의 오프셋 값은 디코딩된 출력 데이터(D5)를 생성하기 위하여 프로세싱된 데이터와 옵션적으로 결합된다.

[0063] 오프셋은 옵션적으로 디코더에서 지원된다. 오프셋은 이것이 인코더에서도 사용될 때 사용된다. 레인지(lowValue 내지 highValue)를 사용하는 파라미터(wrapValue) 내의 랩어라운드, 역 연산자(합 대(vs.) 차이) 및 역 예측자(입력 값 대 출력 값)는 디코더의 중요한 요소들이다.

[0064] 옵션적으로, 디코더에 있어서, 데이터 프로세싱 장치는, 런렝쓰 인코딩(RLE), 스플릿 RLE, 엔트로피 모디파이어(EM), 가변 길이 코딩(VLC), 호프만 코딩, 산술 코딩, 레인지 코딩 중 적어도 하나의 역을 통하여 프로세싱되는 데이터에 적용하도록 동작될 수 있다. 이 프로세싱은 데이터(D3)로부터 데이터(D4)를 생성하기 위하여 실행된다.

[0065] 디코더에 있어서, 데이터 프로세싱 장치는, 역 디코딩 예컨대 역 ODelta 디코딩의 형태로 구현될 때 레인지

(lowValue 내지 highValue)를 이용하여 파라미터(wrapValue) 내의 랩어라운드를 사용하도록 동작될 수 있다.

[0066] 제6 양태에 따르면, 대응하는 인코딩된 데이터(D2 또는 D3)를 생성하기 위하여 입력 데이터(D1)를 인코딩하는 제1 양태에 따른 적어도 하나의 인코더, 그리고 대응하는 디코딩된 데이터(D5)를 생성하기 위하여 인코딩된 데이터(D2, D3 또는 D4)를 디코딩하는 제3 양태에 따른 적어도 하나의 디코더를 포함하는 코덱이 제공된다.

[0067] 제7 양태에 따르면, 비순간적(비일시적) 머신 판독 가능한 데이터 저장 매체 상에 기록되는 소프트웨어 제품이 제공되며, 이 소프트웨어 제품은 제2 양태에 따른 데이터 인코딩 방법을 실행하기 위하여 컴퓨팅 하드웨어 상에서 실행될 수 있는 것을 특징으로 한다.

[0068] 제8 양태에 따르면, 비순간적(비일시적) 머신 판독 가능한 데이터 저장 매체 상에 기록되는 소프트웨어 제품이 제공되며, 이 소프트웨어 제품은 제4 양태에 따른 데이터 디코딩 방법을 실행하기 위하여 컴퓨팅 하드웨어 상에서 실행될 수 있는 것을 특징으로 한다.

[0069] 본 발명의 특징은 첨부된 청구범위에 의해 규정되는 바와 같이 본 발명의 범위로부터 벗어나지 않고 여러 가지 조합으로 조합될 수 있음이 인식될 것이다.

도면의 간단한 설명

[0070] 이제, 본 개시의 실시형태를 하기의 도면을 참조하면서 단지 예로서 설명한다.

도 1은 본 개시에 따른 기능에 대하여 구현된 인코더 및 디코더를 포함하는 코덱을 예시하는 도면이다.

도 2는 도 1의 인코더에서 실행되는 바와 같이 데이터 인코딩 방법의 단계들을 예시하는 도면이다.

도 3은 도 1의 디코더에서 실행되는 바와 같이 데이터 디코딩 방법의 단계들을 예시하는 도면이다.

첨부 도면에 있어서, 밑줄 친 숫자는, 밑줄 친 숫자가 위에 있는 항목 또는 밑줄 친 숫자가 인접해 있는 항목을 나타내기 위해 사용된다. 밑줄 없는 숫자는 밑줄 없는 숫자를 항목에 연결하는 선에 의해 식별되는 항목과 관련된다. 숫자가 밑줄이 없고 연관된 화살표를 수반할 경우, 그 밑줄 없는 숫자는 화살표가 지시하는 일반적인 항목을 식별하는데 사용된다.

발명을 실시하기 위한 구체적인 내용

[0071] 본 개시의 실시형태를 설명할 때, 표 2에 제공되는 바와 같이, 이하의 약어 및 정의가 사용된다.

표 2

[0072] 약어 및 정의

약어	설명
ADC	아날로그/디지털 컨버터
코덱(Codec)	디지털 데이터에 대한 인코더 및 대응하는 디코더
DAC	디지털/아날로그 컨버터
DB	랜덤 액세스 메모리(RAM) 또는 판독 전용 메모리(ROM)에서의 데이터베이스
DC	주어진 영상의 DC 성분, 즉 영상의 수단, 즉 평균 휘도에 대응, 영상의 최저 공간 주파수 성분을 나타냄
RLE	런-렝쓰 인코딩
ROI	관심 있는 영역
ROM	판독 전용 메모리
VLC	가변 길이 코드

[0073] 개요에 있어서, 도 1을 참조하면, 본 개시는 인코더(10) 및 그 연관된 동작 방법에 관련되며, 유익하게도, 인코더(10)는 다이렉트 ODelta 인코더로서 구현된다. 또한, 본 개시는 또한 대응하는 디코더(20)와 관련되며, 유익하게도, 디코더(20)는 역 ODelta 디코더로서 구현된다. 본 개시의 실시형태는 기타 데이터의 레인지 최적화된 버전뿐만 아니라 전술한 공지된 델타(Delta) 인코딩 방법의 비트-최적화된 버전인 다이렉트 ODelta 연산자를 유

익하게 사용한다. ODelta 인코딩은 가변-길이 데이터 워드 예컨대 8/16/32/64 비트를 사용하거나 및/또는 오리지널 값이 1 내지 64 비트의 범위로 표현되는 8/16/32/64 비트 데이터 요소의 가변-길이 인코딩을 사용하는 컴퓨팅 하드웨어 또는 전용 디지털 하드웨어에 사용되며, 대응하는 인코딩된 값은 1 내지 64 비트로 생성된다. 물론, 인코더(10) 및 디코더(20)는, 어떤 경우에, 어떤 종류의 숫자 값들이 데이터(D1) 예컨대 오리지널 데이터에 포함되어 있다고 알고 있으므로, 그 정의 또는 전송은 여기서는 추가로 설명되지 않는다. 숫자 범위(MIN 및 MAX)가 알려져 있고, 데이터(D1)가 이용될 수 있음이 단순히 가정된다.

[0074] 공지된 델타 코딩 방법은, 오리지널(MIN 내지 MAX)로부터 결과(MIN-MAX 내지 MAX-MIN)까지의 값의 범위를 증가시킨다. 이는 오리지널 데이터가 포지티브 값만을 포함할 때 이 방법이 네가티브 값을 또한 생성한다는 것을 의미한다. 본 개시에 따른 ODelta 연산자는 대응하는 오리지널 값의 범위에 있지 않는 값을 결코 생성하지 않고, 따라서 이 연산자는 사용되는 데이터 범위를 증가시키지 않고, 이에 따라서 실행시에, 예컨대 엔트로피 감소 및 연관된 데이터 압축시에 유익하게 사용된다. 예컨대, 공지된 델타 인코딩 방법은, 이러한 델타 인코딩 방법에 의해 생성된 데이터 값이 -31 내지 +31의 범위 즉, 6(즉, 사인 비트 + 5 비트) 비트를 이용하여 실질적으로 표현될 수 있는 63개 값에 있도록, 5-비트 데이터의 스트림 즉, 0 내지 31의 값의 범위로 동작되며, 이와 대조적으로, 다이렉트 ODelta 생성된 값들은, 전술한 5-비트 데이터의 스트림으로부터 생성될 때, 계속해서 0 내지 31의 범위에 있다. 또한, 델타 인코딩의 공지된 방법들은 재귀적으로 구현되도록 허용되지 않는 반면에, 다이렉트 또는 역으로, 본 개시에 따른 ODelta 연산자는 재귀적으로 구현되도록 허용되지만, 이 연산자는 사용된 범위의 값들을 계속해서 유지한다. 값들의 범위는 정확한 비트(bit-exact)일 필요는 없는데 예컨대, 0 내지 31의 값이 5비트에 의해 규정되고, ODelta 연산자는 임의의 범위의 값들 예컨대 0 내지 25의 값의 범위를 사용할 수 있는 반면에 여전히 적절히 동작된다.

[0075] 원칙적으로, 여기에 설명된 ODelta 방법은 항상 기존 데이터 범위에 기초하여 직접 기능할 수 있으며, 이는 이하에 예로서 주어진다. ODelta 방법은 또한 데이터 내의 최소 발생 숫자 값(“lowValue”) 및 데이터 내의 최대 발생 숫자 값(“highValue”)을 나타내는 정보를 전달함으로써 개선될 수도 있다. lowValue >= MIN이고 highValue <= MAX이고 이들 값이 옵션적임에 주목해야 한다.

[0076] 이하 본 개시에 따른 다이렉트 및 역 ODelta 연산자들의 2개의 예를 설명한다. 다이렉트 및 역 ODelta 연산자들의 제1 예는, 예컨대 비순간적(비일시적) 머신 판독 가능한 데이터 저장 매체 상에 기록된 하나 이상의 소프트웨어 제품을 실행하도록 동작될 수 있는 전자식 하드웨어 및/또는 컴퓨팅 하드웨어로 구현하기에 효율적이며 비교적 단순하다.

[0077] 본 개시에 따른 다이렉트 또는 역 ODelta 연산자를 구현할 때, 유익하게 데이터 값들의 오리지널 시퀀스의 전체는 포지티브이며, 최저값은 0이다. 옵션적으로, 일부 오프셋 값 즉, 프리-오프셋 값 또는 포스트-오프셋 값은 이들이 값에 있어서 모두 포지티브를 가지며 최저값이 “0” 이 되도록 데이터 값들을 시프트하는데 사용될 수 있다. 본 개시에 따른 ODelta 연산자는 다이렉트 방식으로 모든 타입의 데이터에 대하여 사용될 수 있으며; 이는 일반적으로 데이터 압축을 제공할 수 있으며 즉, 통신된 데이터 레이트를 감소시킬 수 있으며, 그 이유는, 오프셋 값이 모든 값들에 부가되거나 또는 모든 값들로부터 감산될 때, 데이터 값들의 범위가 소수의 비트를 이용하여 규정될 수 있기 때문이다. 예컨대, 다이렉트 또는 역 ODelta 연산자의 애플리케이션 이전에, 오리지널 데이터 값들은, -11 내지 +18의 범위 내에 있고, 이러한 범위는 +11의 오프셋 값을 이용하여 0 내지 29의 범위로 변형될 수 있고, 변형된 범위는 이후에 5-비트로 설명된다. 이러한 프리-오프셋 또는 포스트-오프셋 값이 사용되지 않을 때, 오리지널 데이터 값은 이들을 기재하기 위하여 적어도 6 비트를 요구하며, 종종, 실제로는, 풀 8-비트 사인 바이트가 편의를 위하여 사용된다.

[0078] 데이터 범위의 유사한 최적화는 일반화된 다이렉트 또는 역 ODelta 연산자를 이용할 때, 또한 가능하다. 이에 따라서, 다이렉트 또는 역 ODelta 연산자, 또는 일부 다른 방법은, 풀 레인지의 값보다 작은 오프셋 값이 제공될 수 있는 데이터 값을 생성하면, 그 후 그 범위의 최적화는 ODelta 인코딩 방법에서 어떤 위상으로 구현될 수 있다. 오프셋 값, 부호에 있어서 네가티브 또는 포지티브가 사용될 때, 이는 도 1, 도 2 및 도 3을 참조하여 이후에 설명되는 바와 같이, 인코더(10)로부터 디코더(20)까지 또한 전달되어야 한다.

[0079] 다이렉트 ODelta 연산자는 1-비트 방식으로 구현될 수 있고, 예컨대 비트-바이-비트 방식으로 오리지널 데이터(D1)를 인코딩할 수 있고, 이하에 더욱 상세히 설명되는 바와 같이, 이러한 1-비트 방식에서, 방법 1 및 방법 3은 도 1에서 오리지널 데이터(D1)의 비트 값에서의 변화가 없을 때 “0” 값을 생성하고, 오리지널 데이터(D1)의 비트 값에서의 변화가 발생할 때 “1” 값을 생성한다. 오리지널 데이터에서의 제1 비트에 대한 예측은 옵션적으로 “0” 값이며, 따라서 오리지널 데이터(D1)에서의 제1 비트의 값이 유지된다. 오리지널 데이터 내의 예측된

값의 제1 비트를 사용하는 것이 또한 대안적으로 옵션적으로 가능하지만, 이러한 선택은 어떤 코딩 이점을 제공하지 않는데, 그 이유는, 예측이 옵션적으로 1-비트 데이터에 대한 디폴트에 의해, "0" 값이 되도록 항상 가정될 때 선택은 전달될 필요가 없고, 즉 미리 정해된 값 "0" 이 인코더(10)와 디코더(20)에 의해 사용되고, 이에 의해 통신되는 이러한 예측에 대한 필요성을 피하고, 이에 따라서 데이터 압축을 개선시킨다.

[0080] 본 개시에 따른 다이렉트 ODelta 인코딩의 예를 이제 설명한다. 예시적인 오리지널 비트 시퀀스, 17개의 "1" 및 20개의 "0" 을 포함하는 즉 27 비트가 아래와 같이 식 1(Eq. 1)에 제공된다.

[0081] **010101100100010100000000000011111111111** 식 1

[0082] 이것의 엔트로피(E)는 식 2(Eq.2)으로부터 계산될 수 있다.

[0083] $$E = 17 * \log_{10}\left(\frac{37}{17}\right) + 20 * \left(\frac{37}{20}\right) = 11.08523$$ 식 2

[0084] 식 2(Eq. 2)에서 엔트로피(E)를 코딩하는데 필요한 비트 수 즉, Min_bit

[0085] 는. 전술한 문헌 D7 및 D8에 기재된 바와 같이, 샤논의 소스 코딩 이론으로부터 계산될 수 있고, 식 3(Eq. 3)에 제공된다.

[0086] $$Min\_bits = \frac{E}{\log_{10}(2)} = 36.82 \quad \text{비트}$$ 식 3

[0087] 비트의 오리지널 시퀀스는, 전술한 바와 같이 즉 방법 1 및 방법 3에 의해 다이렉트 ODelta 연산자 처리될 때, 13개의 "1" 및 24개의 "0" 이 존재하는 37 비트을 포함하여, 아래와 같이 비트의 시퀀스가 생성된다.

[0088] **0111110101100111100000000001000000000** 식 4

[0089] 이것의 엔트로피(E)는 식 5(Eq. 5)로부터 계산될 수 있다.

[0090] $$E = 13 * \log_{10}\left(\frac{37}{13}\right) + 20 * \left(\frac{37}{24}\right) = 10.41713$$ 식 5

[0091] 이는 식 6(Eq. 6)에 따라서, 최소 비트 수 즉, Min_bits로 표현될 수 있다.

[0092] $$Min\_bits = \frac{E}{\log_{10}(2)} = 34.60 \quad \text{비트}$$ 식 6

[0093] 식 4(Eq. 4)에서의 비트의 시퀀스는 유익하게 예컨대, 런렝Tm 인코딩(RLE), 호프만 코딩, 산술 인코딩, 레인지 인코딩, 엔트로피 모디파이어 인코딩 또는 SRLE 인코딩 중 적어도 하나를 이용하여 데이터 압축을 달성하도록 추가로 인코딩된다.

[0094] ODelta 연산자는 그 연관된 엔트로피 코딩 방법이 적용될 때, 예컨대 식 1(Eq. 1)에서와 같이, 오리지널 데이터 대신에, 예컨대 식 4(Eq. 4)에서와 같이, 예컨대 RLE 또는 SRLE가 동작된 데이터에 대하여 사용될 때, 오리지널 데이터(D1)를 나타내도록 요구되는 비트양을 감소시키며, 이러한 1-비트 다이렉트 ODelta 연산자 즉 방법 1 및 방법 3은, 식 1(Eq. 1)에서의 비트들의 오리지널 시퀀스에서의 많은 변화가 존재할 때 "1" 을 생성하고, 상기 연산자는 식 1(Eq. 1)에서 비트의 오리지널 시퀀스에서의 상호 유사한 비트의 긴 시퀀스가 존재할 때 "0" 을 생성한다.

[0095] ODelta 연산자의 역 버전 즉, 방법 1 및 방법 3의 역은, 적절하게 "0" 값으로부터 "1" 값으로 또는 "1" 값으로부터 "0" 값으로 비트 값을 변경하며, 데이터의 인코딩된 스트림에서 즉, 데이터(D2)에서 "1" 값이 존재할 때, 인코딩된 데이터의 스트림에서 즉 데이터(D2)에서 "1" 값이 존재할 때, 데이터(D2)의 인코딩된 스트림에서 "0" 값이 존재할 때 비트 값을 변경하지 않는다. 이러한 ODelta 연산이 데이터(D2)의 다이렉트 ODelta-연산된 비트 스트림에 대하여 실행될 때, 데이터(D1)의 오리지널 스트림은, 디코딩된 데이터(D5)로서 생성되지만, 전술한 바와 같이, VLC 또는 호프만 코딩과 같은 부가적인 코딩이 유익하게 사용되며, 이러한 코딩은 또한 고려될 필요가 있고, 이는 데이터(D3)가 엔트로피 인코더의 포워딩 동작을 이용하여 데이터(D2)로부터 생성되고, 데이터(D4)는 엔트로피 디코더의 역 동작을 이용하여 데이터(D3)로부터 생성된다는 것을 의미한다.

[0096] 유익하게, 데이터(D1)의 오리지널 스트림은 인코딩이 적용되기 이전에 2 개 이상의 섹션들로 세분된다. 이러한

세분은 데이터(D1)의 오리지널 스트림을 인코딩할 때 사용되는 더 많은 수의 최적화에 대한 기회를 제공한다. 예컨대, 이러한 세분은, 데이터(D1)에서의 변경가능한 시퀀스가 직접 ODelta 인코딩 즉, 방법 1 및 방법 3을 이용할 때 더 많은 "1"을 생성하기 때문에 유익한 반면에 평평한 변경가능하지 않은 시퀀스 즉 "평평한(flat)" 시퀀스는, 예컨대 후속의 VRL 인코딩 또는 호프만 인코딩에 대하여 바람직하게 더 많은 "0"을 생성하므로, 엔트로피(E)는 전술한 바와 같이 데이터(D1)를 개별적으로 인코딩될 수 있는 복수의 섹션으로 분할함으로써 상기 데이터(D1)를 구성하는 전체 비트 스트림에 대하여 감소될 수 있다.

[0097] 서로 개별적으로 인코딩되는 복수의 섹션이 사용될 때, 본 발명에 따른 다이렉트 ODelta 인코딩의 예를 다음에 설명한다. 오리지널 단일 비트의 시퀀스를 포함하는 제1 섹션은, 아래와 같이 식 7(Eq. 7)에서, 전체로 16 비트 즉, 7개의 "1"과 9개의 "0"을 포함한다.

[0098] **0101011001000101** 식 7

[0099] 여기서 H(x) = 4.7621이고 B = 15.82이며; "H"는 엔트로피를 나타내며, "B"는 Max_bit를 나타낸다. 오리지널 비트의 식 7(Eq. 7) 시퀀스는 다이렉트 ODelta 연산자 처리될 때, 대응하는 변환된 비트의 시퀀스가 식 8(Eq. 8)에서와 같이 제공된다.

[0100] **0111110101100111** 식 8

[0101] 여기서, H(X) = 4.3158이고, B = 14.34이다.

[0102] 오리지널 싱글 비트의 시퀀스를 포함하는 제2 섹션은 아래와 같이 식 9(Eq. 9)에 포함된다.

[0103] **000000000001111111111** 식 9

[0104] 여기서, H(X) = 6.3113이고, B = 20.97이다. 오리지널 비트의 식 9(Eq. 9) 시퀀스는, 다이렉트 ODelta 연산자 처리될 때, 대응하는 변환된 비트의 시퀀스는 식 10(Eq. 10)에서와 같이 제공된다.

[0105] **000000000001000000000** 식 10

[0106] 여기서, H(X) = 1.7460이고 B = 5.80이다. 이들 예에 있어서, 전술한 바와 같이, H(X)는 엔트로피 E를 나타내며, B는 코딩에 필요한 최소 비트 수를 나타낸다.

[0107] 이 예에서의 식 7(Eq. 7) 및 식 10(Eq. 10)으로부터의 최적 압축은, 양쪽 섹션이 개별적으로 다이렉트 ODelta 연산자 처리될 때 달성되며(즉, 14.34 비트 + 5.80 비트 = 총 20.14 비트에 대한 인코딩); 이러한 압축은 오리지널로 요구되는 36.82 비트 보다 더 적은 비트 즉, 34.60 비트를 요구하는 다이렉트 ODelta-연산된 비트 또는 분할 이후에 요구되는 오리지널 비트 수(= 15.82 비트 + 20.97 비트 = 36.79 비트)를 요구한다. 유익하게, 데이터(D1) 내의 비트들의 오리지널 스트림을 섹션들로 분할하는 것은, 즉 피스-바이-피스로 데이터(D2) 내에 포함되는 바와 같이, 오리지널 데이터(D1)의 엔트로피(E) 및 변경된 데이터의 대응하는 엔트로피(H)를 분석함으로써 자동적으로 실행된다.

[0108] 데이터 압축은, 데이터의 큰 충분한 영역이 존재하면, 데이터(D1)에서 이용가능한 복수의 긴 런 섹션이 존재할 때, 단순히 데이터(D1)의 부분들을 인코딩될 새로운 섹션으로 분할함으로써 조악한 방식으로 옵션적으로 구현되며, 여기서 비트 값들은 시퀀스에 따라서 급속하게 변한다. 옵션적으로, 데이터(D1)의 일부 섹션은 다이렉트 ODelta 연산자를 사용하지 않고 인코딩되는데, 예컨대 비교적 적은 개별적인 차이 비트를 가진 상호 유사한 비트의 긴 런이 존재하면; 이러한 경우, 다이렉트 ODelta 연산자는 데이터 압축 목적을 위하여 현저한 이점을 제공하지 않는다.

[0109] 데이터(D1)를 더 작은 섹션들로 분할하는 것은, 데이터를 인코딩된 데이터(D2)로 만드는데 기여하는 부가적인 오버헤드를 생성하는 단점을 가진다. 이러한 오버헤드는, 예컨대 매 새로운 섹션과 연관되는 데이터 비트 또는 데이터 바이트의 양을 나타내는 정보를 포함한다. 그러나, 적어도 어떤 량의 오버헤드 데이터 값을 송신할 필요가 있음이 항상 발견되며, 이에 따라서 주어진 데이터가 데이터의 2개의 섹션으로 분할될 때 하나의 추가(extra) 오버헤드 데이터 값만이 존재한다.

[0110] 이후에 디코딩될 수 있는 인코딩된 비트 스트림을 달성하기 위하여, 엔트로피 인코딩은 다이렉트 ODelta 연산자, 예컨대, VLC, 호프만 코딩, 산술 코딩, 레인지 코딩, RLE, SRLE, EM 및 유사 코딩 이후에 유익하게 구현된다. 실제 데이터 인코딩과 비교할 때 계산된 엔트로피(E) 및 최소 비트 추정 값에 기초하여 최적화 계산을

실행하는 것이 더 쉽고 계산상으로 더욱 효율적이다. 이러한 실행 순서는 상당한 속도 최적화를 가능하게 하고, 종종 인코딩된 데이터(D2)에서의 최적의 데이터 압축 결과를 달성한다. 대안적으로, 즉 데이터(D1) 내의 오리지널 비트, 알파벳, 숫자, 바이트 및 워드 데이터가, 엔트로피-최적화된 비트 스트림을 생성하기 위하여 일부 다른 방법으로 먼저 코딩되는 방식으로 엔트로피 최적화를 실행하는 것이 실현될 수 있고, 그 후, 다이렉트 ODelta 연산자는 대응하는 인코딩된 데이터 즉, 데이터(D2)를 제공하기 위하여 엔트로피-최적화된 비트 스트림을 변경하는데 사용된다. 또한, 이러한 ODelta 연산된 데이터는, 데이터(D3)를 생성하기 위하여 데이터(D2)로부터 다른 인코딩 방법으로 계속해서 인코딩될 수 있다.

[0111] 일반화된 다이렉트 ODelta 연산자는, 데이터(D1) 내에 사용되는 값들의 범위를 기술하는 파라미터 즉, 그 값들을 나타내는데 필요한 비트의 값 또는 비트 수를 사용한다. 또한, ODelta 연산자는, 포지티브 및 네가티브 오프셋 값 다시 말해서 포지티브 및 네가티브 "페디스털(pedestal)" 값을 사용할 수 있는 방법에서 사용된다. 예컨대, 데이터(D1)에 7 비트가 제공되면, 즉 지원되는 "0" 내지 "127"의 값을 갖지만, 이 데이터가 "60" 내지 "115"의 범위의 값들만을 포함하고, 그 후, -60의 오프셋 값이 데이터(D1)에 적용될 때, 이에 의해 단지 6 비트를 포함하는 값으로써 표현될 수도 있는 "0" 내지 "55"의 범위의 값을 가지는 생성되어진 변환된 데이터가 존재하며, 이에 의해 즉 달성하기 위한 데이터 압축의 정도가 실현될 수 있다. 이에 따라서, 일반화된 다이렉트 ODelta 연산자는, 풀 레인지의 데이터 값이 데이터(D1)에 제공되는데, 즉 7비트로 표현되며, 종래에는 8-비트 바이트로 표현된다.

[0112] 본 발명에 따라서, 다이렉트 ODelta 값, 즉 방법 1은, 단지 포지티브 값을 가진 데이터에 대하여 아래와 같이 예시적인 소프트웨어 코드를 제외하고 설명되는 바와 같이 절차를 이용하여 계산될 수 있다(lowValue = MIN = 0 및 highValue = MAX = 127, wrapValue = 127 - 0 + 1 = 128).

```
wrapValue = power(2, bits) = power(2, 7) = 128
prediction Value = (lowValue + highValue + 1) div 2 = (wrapValue + 1) div 2 + lowValue =
64
for all pixels

begin
    if(originalValue >= predictionValue) then
        ODeltaValue = originalValue – predictionValue
    else
        ODeltaValue = wrapValue + originalValue – predictionValue
    predictionValue = originalValue
end
```

[0113]

[0114] 이제, 전술된 ODelta 연산자를 추가로 설명하기 위한 예가 제공될 것이다. 값들의 오리지널 시퀀스는 아래와 같이 식 11(Eq. 11)에 제공된다.

[0115] **65, 80, 126, 1, 62, 45, 89, 54, 66** 식 11

[0116] 대응하는 델타 코딩 값은 아래와 같이 식 12(Eq. 12)에 제공된다.

[0117] **65, 15, 46, -125, 61, -17, 44, -35, 12** 식 12

[0118] 대응하는 다이렉트 ODelta 코딩 값들은, 아래와 같이 식 13(Eq. 13)에 제공된다.

[0119] **1, 15, 46, 3, 61, 111, 44, 93, 12** 식 13

[0120] 여기서, 파라미터 내의 랩어라운드(wrapValue)가 사용된다.

[0121] 역 ODelta 연산자, 즉 방법 1은, 예컨대 아래와 같이 예시적인 소프트웨어 코드에 의해 구현되는 바와 같이, 역 ODelta 값을 생성하는데 사용될 수 있다.

```
wrapValue = power(2, bits) = power(2, 7) = 128
predictionValue = (wrapValue + 1) div 2 + lowValue = 64
for all pixels

begin
    ODeltaValue = originalValue + predictionValue
    if (ODeltaValue >= wrapValue) then
        ODeltaValue = ODeltaValue – wrapValue
    predictionValue = ODeltaValue
end
```

[0122]

[0123] 이 소프트웨어 코드가 실행되어 식 13(Eq. 13)에 적용될 때, 이러한 소프트웨어는 식 14(Eq. 14)에 제공된 바와 같이 값들을 생성한다.

[0124] **65, 80, 126, 1, 62, 45, 89, 54, 66** 식 14

[0125] 이 예는 2의 거듭제곱 값으로서 wrapValue을 이용한다. 이는 의무사항이 아니며, wrapValue는, 만일 네가티브 값들이 또한 이용가능하거나 또는 레인지가 데이터의 주어진 시퀀스에서 프리-오프셋에 의해 변경되면, 가장 큰 데이터 값보다 더 큰 임의의 값이거나 사용된 범위보다 더 넓은 값일 수도 있다. 이후에 이러한 특징을 나타내는 추가적인 예가 존재한다.

[0126] 도 1을 참조하여 전술한 내용을 요약하기 위하여, 본 개시는 인코더(10)와 디코더(20)와 관련된다. 옵션적으로, 인코더(10) 및 디코더(20)는 도면 번호 30에 의해 일반적으로 표시되는 코덱으로서 결합되어 사용된다. 인코더(10)는 예컨대, 대응하는 인코딩된 데이터(D2 또는 D3)를 생성하기 위하여 다이렉트 ODelta 방법을 이용하여 인코딩되는 오리지널 입력 데이터(D1)를 수신하도록 동작될 수 있다. 인코딩된 데이터(D2 또는 D3)는 옵션적으로 통신 네트워크(40)를 통하여 통신되거나, 예컨대 광학 디스크 판독 전용 메모리(ROM) 또는 이와 유사한 것과 같은 데이터 캐리어 등의 데이터 저장 매체(50) 상에 저장된다. 디코더(20)는, 예컨대 통신 네트워크(40)를 통하여 스트림되거나 데이터 저장 매체(50) 상에 제공되는, 인코딩된 데이터(D2 또는 D3)를 수신하도록 동작될 수 있고, 예컨대 오리지널 데이터(D1)와 실질적으로 유사한 대응하는 디코딩된 데이터(D5)를 생성하기 위하여, 역 방법 예컨대 역 ODelta 방법을 적용하도록 동작될 수 있다. 인코더(10) 및 디코더(20)는, 예컨대, 하나 이상의 소프트웨어 제품을 실행하도록 동작될 수 있는 컴퓨팅 하드웨어, 예컨대 이러한 설명부에서의 예시적인 실시형태로서 제공되는 코덱과 같은 디지털 하드웨어를 이용하여 유익하게 구현된다. 대안적으로, 인코더(10) 및/또는 디코더(20)는 전용 디지털 하드웨어를 이용하여 구현된다.

[0127] 인코더(10)에서 실행되는 바와 같이, ODelta 방법은 도 2에 도시된 바와 같은 단계들을 이용한다. 옵션적인 제1 단계(100)에서, 입력 데이터(D1)는 데이터 요소들의 값의 범위를 찾기 위하여 프로세싱된다. 옵션적인 제2 단계(110)에서, 값들의 범위로부터, 오프셋, 즉 프리-오프셋은 대응하는 세트의 변형된 요소들을 생성하기 위하여 데이터 요소들을 포지티브 레짐(regime)으로 변형하기 위해 계산된다. 제3 단계(120)에 있어서, 제2 단계(110)에서 옵션적으로 변형된 요소들은, 그 후 대응하는 ODelta 인코딩된 값들을 생성하기 위하여 다이렉트 ODelta 인코딩된다. 제4 단계(130)에 있어서, ODelta 인코딩된 값들 및 옵션적인 오프셋 값, 최소 값(lowValue), 및/또는 최대 값(highValue)은, 데이터(D2)로부터 데이터(D3)를 생성하기 위하여 예컨대 런-렝쓰 인코딩(RLE), 레인지 코딩, 또는 호프만 코딩을 이용하여 그 후 개별적으로 인코딩된다. 오프셋 값, 최소 값(lowValue), 및/또는 최대 값(HighValue)은 항상 압축될 수 없기 때문에, 인코더(10)로부터 디코더(20)까지 적절한 비트양을 이용하여 전달되는 값을 요구한다. 또한, 오프셋 값, 최소 값(lowValue), 및/또는 최대 값(highValue)은 다이렉트 ODelta 연산자에 대하여 옵션적인 특징이며; 예컨대, 오프셋 값은, 어떤 상황에서는, "0" 값을 가지며, lowValue는 값(MIN)을 가지며, highValue는 값(MAX)을 가지며, 즉 변형은 적용되지 않고, 풀 레인지가 사용되지 않는다. 특히, 다이렉트 ODelta 연산자가, 1-비트 데이터에 대하여 즉 비트-바이-비트 인코딩을 위하여 구현될 때, 이러한 연산자는 전혀 오프셋 값을 필요로 하지 않고, 그 후 단계들(100 및 110)은 항상 무시된다. 오프셋 값이 또한 단계 110에서 사용될 때, 최대 및 최소 값을 나타내는 레인지 값은 단계 110 내에서 업데이트되어야 한다. 차이 값의 수 즉, wrapValue는, 디코더(20)에 의해 또한 공지되어야 하거나 또는, 그렇지 않으면, 인코더(10)가 압축된 데이터 내에서 값을 디코더(20)에 전달해야 한다. 옵션적으로, 디폴트 wrapValue(= highValue - lowValue + 1)는 인코더에서 그리고 디코더에서 사용된다. 옵션적으로, 인코더(10) 및 디코더(20) 중 적어도 하

나는, 인코딩된 데이터(D2)를 생성하기 위하여 데이터(D1)를 최적으로 압축하기 위해, 예컨대 입력 데이터(D1)를 인코딩을 위한 섹션으로 세분하는 최적의 방식을 찾기 위하여 재귀적 방식으로 동작된다.

[0128] 디코더(20)에서 실행되는 바와 같은 역 ODelta 방법은 도 3에 도시된 바와 같은 단계를 이용한다. 제1 단계 200에서, 데이터(D2/D3 또는 D4)는, 디코딩된 ODelta 데이터를 생성하기 위하여 전술한 단계 130에서 사용되는 데이터에 대하여 역 인코딩되며, 여기서 디코딩된 ODelta 데이터는, ODelta-인코딩된 값을 가지며, 옵션적으로 개별적인 오프셋 값을 가진다. 제2 단계 210에서, ODelta-인코딩된 값들은 데이터 요소들의 시퀀스를 생성하기 위하여 디코딩된다. 제3 단계 220에 있어서, 데이터 요소들의 시퀀스는, 디코딩된 데이터(D5)를 생성하기 위하여 옵션적인 프리-오프셋 값을 이용하여 변형되며; 어떤 상황에 있어서, 이러한 변형은 "0" 값으로 설정되는데, 즉 변형이 효율적으로 적용되지 않는다. 또한, 예컨대 1-비트 인코딩 즉 비트-바이-비트 인코딩을 수행할 때, 오프셋 값을 사용할 필요 없이 방법을 실행하는 것이 가능하며, 이에 의해 단계 220을 무시할 수 있다. 또한, 디코더(20)는 적절한 방식으로 수신된 데이터 요소들을 디코딩할 수 있도록 wrapValue을 알아야 한다.

[0129] 단지 포지티브 값을 달성하기 위하여 오프셋을 사용함으로서, 데이터(D2 또는 D3)에서의 더욱 효율적인 데이터 압축이 달성될 수 있다. 모든 데이터 값들이 이미 포지티브 값이면, 임의의 오프셋 값을 추가할 필요가 없다. 물론, 네가티브 오프셋 값들은 옵션적으로 다음 예에 도시된 바와 같이, 이용가능한 범위를 감소시키기 위하여 사용되지만, 이는 의무사항은 아니다.

[0130] 도 2 및 도 3의 방법들은, ODelta 코딩되는 이용가능한 값만을 이용함으로써, 옵션적으로 또한 최적화될 수 있다. 이러한 최적화는 사용된 값들이 공지되어 있어야 함을 요구한다. 예컨대, 전술한 내용의 예에 있어서, 1(=오리지널 최소)로부터 126(=오리지널 최대)까지의 값들만이 오리지널 데이터 세트(D1)에 제공된다. 그 후, 오프셋 값은 1(-> lowValue = 오리지널 최소 - 오프셋 = 1 - 1 = 0 그리고 highValue = 오리지널 최대 - 오프셋 = 126 - 1 = 125)이 된다. 프리-오프셋 값이 오리지널 데이터(D1)로부터 감소되었을 때, 이에 의해, 이하의 값들은 식 15(Eq. 15)에서 생성된다.

[0131] **64, 79, 125, 0, 61, 44, 88, 53, 65** 식 15

[0132] 식 15(Eq. 15)로부터, 최대 값 125는, "숫자" (= 최대 델타 값 = highValue - lowValue)가 이제 125일 수 있거나, wrapValue가 최소값 126(= 숫자 + 1 = highValue - lowValue + 1)일 수 있도록 결정된다(highValue = 오리지널 max - 오프셋 = 126 - 1 = 125). 이제, 이러한 값들을 저장 및/또는 전달할 필요가 있고, 그 후 이전의 예는 아래와 같이 프로세스 값들을 변경함으로써 변형될 수 있다.

wrapValue = 126 ("0" to "125" => 126 different values)
prediction Value = (highValue + lowValue + 1) div 2 = (wrapValue + 1) div 2 + lowValue =
[0133] **63**

[0134] 대응하는 다이렉트 ODelta 연산자 값들이 식 16(Eq. 16)에 제공된다.

[0135] **1, 15, 46, 1, 61, 109, 44, 91, 12** 식 16

[0136] 모든 "네가티브 델타 값" 은 이제 2(즉 = 레인지 변경 = 128 - 126)의 인자에 의해 감소된다는 점이 인식될 것이다. 유사하게도, 디코더(20)에 있어서, 프로세스 값들은 아래와 같이 변경되어야 한다.

wrapValue = 126
[0137] **predictionValue = (wrapValue + 1) div 2 + lowValue = 63**

[0138] 대응하는 역 ODelta 값들은 아래와 같이 식 17(Eq. 17)에 제공된다.

[0139] **64, 79, 125, 0, 61, 44, 88, 53, 65** 식 17

[0140] 프리-오프셋 값이 식 17(Eq. 17)에 추가될 때, 식 18(Eq. 18)에서의 이하의 결과가, 아래와 같이 식 15(Eq. 15)에서의 오리지널 데이터에 대응하여 획득된다.

[0141] **65, 80, 126, 1, 62, 45, 89, 54, 66** 식 18

[0142] 이 예에 있어서, 값들의 범위가 거의 채워지므로, 다이렉트 ODelta 연산자를 오프셋 및 최대 값(highValue)에 적용함으로써 유도되는 비교적 보통의 이익이 존재한다. 그러나, 엔트로피(E)에서의 감소는 계속해서 달성될 수

있는 데, 즉 값들이 적절하게 전달될 때, 주파수 테이블에서 또는 코드 테이블에서의 값들을 더 적게 만든다. 가장 큰 이익은, 범위가 덜 사용될 때 달성될 수 있다.

[0143] 실제 1-비트 다이렉트 및 역 ODelta 방법의 예시적인 실시형태 즉, 데이터를 인코딩 및 디코딩하는 방법 1 또는 방법 3이, 이제 실행 가능한 컴퓨터 소프트웨어 코드에 의해 제공되며; 이 방법은 전술한 다이렉트 및 역 ODelta 연산자 즉, 방법 1 또는 방법 3을 이용한다. 소프트웨어 코드는, 컴퓨팅 하드웨어 상에 실행될 때, 하나의 바이트 버퍼로부터 다른 바이트 버퍼로 비트를 프로세싱하기 위하여 동작될 수 있다. 소프트웨어 코드에 있어서, GetBit, SetBit 및 ClearBit 함수는 항상 HeaderBits 값을 업데이트한다. HeaderIndex 값은 또한 다음 비트가 다음 바이트 내에 있을 때 업데이트된다. 옵션적으로, 소프트웨어 코드는, 값들이 주어진 비트가 목적지 버퍼에 기록될 때에만 업데이트되도록, HeaderIndex 및 HeaderBits 값들의 하나의 세트만을 소스 및 목적지에 대하여 사용하기 위하여, 최적화될 수 있다.

```
procedure EncodeODelta1u(APtrSrc : PByte; ASrcDstBitLen : PCardinal; APtrDst : PByte)
var
  iSrcHeaderIndex, iSrcHeaderBits, iIndex,
  iDstHeaderIndex, iDstHeaderBits : Cardinal;
  bBit, bLastBit : Boolean;
begin
  // Reset offsets
  iSrcHeaderIndex := 0;
  iSrcHeaderBits := 0;
  iDstHeaderIndex := 0;
  iDstHeaderBits := 0;

  // Initialise delta value
  bLastBit := False;

  // Go through all bits
  for iIndex := 0 to ASrcDstBitLen^-1 do
  begin
    // Read bit
    bBit := GetBit(APtrSrc, @iSrcHeaderIndex, @iSrcHeaderBits);

    // Set destination bit if current source bit is different than previous source bit
    if (bBit <> bLastBit) then
    begin
      SetBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
      bLastBit := bBit;
    end
    else ClearBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
  end;
end;

function DecodeODelta1u(APtrSrc : PByte; ASrcDstBitLen : PCardinal; APtrDst : PByte) :
Boolean;
var
```

[0144]

```
  iSrcHeaderIndex, iSrcHeaderBits, iIndex,
  iDstHeaderIndex, iDstHeaderBits : Cardinal;
  bBit, bLastBit : Boolean;
begin
  // Reset offsets
  iSrcHeaderIndex := 0;
  iSrcHeaderBits := 0;
  iDstHeaderIndex := 0;
  iDstHeaderBits := 0;

  // Initialise delta value
  bLastBit := False;

  // Go through all bits
  for iIndex := 0 to ASrcDstBitLen^-1 do
  begin
    // Read bit
    bBit := GetBit(APtrSrc, @iSrcHeaderIndex, @iSrcHeaderBits);

    // Change bit value if source bit is true
    if (bBit = True) then
    begin
      if (bLastBit = True) then
        bLastBit := False
      else bLastBit := True;
    end;

    // Set destination bit based on bit value (True or False)
    if (bLastBit) then
      SetBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits)
    else ClearBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
  end;
end;
```

[0145]

[0146] 전술한 다이렉트 및 역 ODelta 연산자 즉, 방법 1 또는 방법 3은, 디지털 포맷의 예컨대 비디오 데이터, 영상 데이터, 오디오 데이터, 그래픽 데이터, 지진 데이터, 의료 데이터, 측정 값, 참조 수치 및 마스크인 임의의 타입의 데이터를 압축하도록 유익하게 사용된다. 또한, 하나 이상의 아날로그 신호는 또한 예컨대 압축 이전에 ADC를 이용함으로써, 처음으로 대응하는 디지털 데이터로 변환될 때 다이렉트 ODelta 연산자를 이용하여 압축될 수도 있다. 데이터가 하나 이상의 아날로그 신호로 다시 변환되기를 원하면, 역 ODelta 연산자가 사용될 때, DAC는 연산 이후에 사용될 수 있다. 그러나, 다이렉트 ODelta 연산자 단독으로는 통상적으로 데이터를 압축하는데 유효하지 않지만, 예컨대 가변-길이 코딩(VLC), 산술 코딩, 레인지 코딩, 런-렝쓰 인코딩, SRLE, 엔트로피 모디파이어 등의 다른 인코딩 방법과 결합하여 사용될 때 효과적인 데이터 압축을 제공할 수 있다. 이러한 인코딩 방법은 다이렉트 ODelta 연산자가 인코더(10)에 사용된 이후에 데이터(D2)에 대하여 사용된다. 인코딩된 데이터(D2)는, 결과적으로 발생된 데이터가 디코더(20) 내에 구현된 역 ODelta 연산자에 전달되기 이전에 대응하여 다시 디코딩되어야 한다. ODelta 연산자는 또한 다른 타입의 엔트로피 모디파이어에 대하여 사용될 수 있다. 어떤 상황에 있어서, 다이렉트 ODelta 연산자는, 엔트로피(E)를 증가시키며, 데이터 압축 알고리즘은, 이것이 유익한 데이터 압축 성능을 제공할 때에만 예컨대, 이것이 압축되는 데이터의 본성에 기초하여 선택적으로 사용될 때, 예컨대, 전술한 바와 같이 입력 데이터(D1)의 선택된 부분들에 선택적으로 적용될 때, 데이터를 인코딩하는데 사용하기 위해 다이렉트 ODelta 연산자를 선택적으로 사용하도록 유익하게 동작될 수 있다.

[0147] 다이렉트 ODelta 연산자가, 예컨대 참조로 본원에 통합되는 미국 특허 출원 제13/584,005호에 개시된 바와 같이 블록 인코더와 결합하여 사용되도록 고안되었고, 역 ODelta 연산자는, 참조로 본원에 통합되는 미국 특허 출원 제13/584,047호에 개시된 바와 같이 블록 디코더와 결합하여 사용되도록 고안되었다. 옵션적으로, 다이렉트 ODelta 연산자 및 역 ODelta 연산자는 참조로 본원에 통합되는 미국 특허 출원 제13/657,382에 개시된 바와 같이 멀티레벨 코딩 방법과 결합하여 유익하게 사용된다. 유익하게, 바이너리 상태를 포함하는, 예컨대 데이터(D1)에 존재하는 모든 타입의 1-비트 데이터는, 대응하는 변환된 데이터를 생성하기 위하여 1-비트 버전의 다이렉트 ODelta 연산자 처리되며, 이 변환된 데이터는 그 후 인코딩된 데이터(D2 또는 D3)를 생성하기 위하여 실제 엔트로피 인코딩된다. 옵션적으로, 전술한 바와 같이, 다이렉트 ODelta 연산자는, 오리지널 데이터(D1)의 본성에 따라서 선택적으로 사용된다.

[0148] 옵션적으로, 다이렉트 ODelta 연산자의 이전 또는 이후에 데이터의 엔트로피를 변경하는 다른 방법들을 사용하도록 실현될 수 있다. 예컨대, 다이렉트 ODelta 연산자는, 일반화된 버전의 다이렉트 ODelta 연산자 내의 복수-비트 데이터에 대하여 직접적으로 또한 사용될 수 있다. 또한, 전술한 1-비트 버전의 다이렉트 ODelta 연산자는, 모든 사용된 비트가 먼저 비트의 직렬 시퀀스에 제공된 이후에, 유익하게도 복수-비트 데이터에 대하여 사용된다.

[0149] 복수의 방법들이, 인코더(10) 내의 다이렉트 ODelta 연산자와 관련하여 데이터 압축을 위하여 사용될 때, 대응하는 역 동작들이 예컨대, 디코더(20)에서 역순서로 수행된다.

[0150] 이하의 방법의 시퀀스는 인코더(10)에 사용된다.

```
[data D1]   => direct ODelta (method 2)
            => VLC-
            => EM
            => Arithmetic coding
            => [data D3]
```

[0151] 식 19

[0152] 이하의 방법의 역 시퀀스는 디코더(20)에 사용된다.

```
[data D3]   => inverse Arithmetic coding
            => Inverse EM
            => Inverse VLC
            => inverse ODelta (method2)
            => [data D5]
```

[0153] 식 20

[0154] 여기서 "VLC"는 가변 길이 코딩을 나타내며, "EM"은 엔트로피 변경을 나타낸다.

[0155] 앞에서 설명된 바와 같은 ODelta 연산자는 반전가능하며, 손실이 없다. 또한, ODelta 연산자는, 예컨대 비트-바이-비트 인코딩을 수행하지만, 또한 다른 데이터에 대해서도 수행할 때, 1-비트 데이터 스트림에 대하여 특별히 옵션적으로 구현될 수 있다. 유익하게도, 모든 타입의 데이터가, 다이렉트 ODelta 연산자의 일반화된 버전을 이용하여 프로세싱될 수 있다. 유익하게도, 다이렉트 ODelta 연산자는, 데이터가 압축될 때 사용되며, 대응하는 역 ODelta 연산자는 압축된 데이터가 압축 해제될 때 사용된다. 옵션적으로, ODelta 연산자가 사용될 때, 다이렉트 ODelta 연산자 및 그 대응하는 역 연산은, 역 순서로 사용되며, 다시 말해서, 역 ODelta 연산자는 오리지널 비트 스트림 상에 먼저 일시적으로 수행되고, 그 후, 오리지널 비트 스트림을 생성하기 위하여 다이렉트 ODelta 연산자가 후속된다. 하나의 ODelta 연산자는 엔트로피를 증가시키고, 다른 ODelta 연산자는 엔트로피를 감소시킨다. 다이렉트 ODelta 연산자가 전혀 엔트로피를 변경하지 않아야 하고, 그 후, 역 ODelta 연산자도 엔트로피를 변경하지 않는 것은 매우 드문 경우이다. 다이렉트 및 역 ODelta 연산자가 예컨대 방법 1에 대하여 사용될 때, 이러한 연산들의 역 순서는 방법 4의 정상 순서와 유사하다는 것에 주목해야 한다. 유사한 순서 변경이 방법 2 및 방법 3에 의해서 또한 가능하다.

[0156] 1-비트 버전에 있어서, 즉 비트-바이-비트 방식으로 데이터를 인코딩하기 위하여, 다이렉트 ODelta 연산자는 예측없이 즉, 초기 "0" 값의 예측을 디폴트로 가정하여 유익하게 개시된다. 일반화된 버전에서, ODelta 연산자는 이용가능한 데이터 범위의 절반을 나타내는 예측으로 개시되며; 예컨대, 5-비트가 데이터(D1)에서의 입력 데이터 값 즉, "0" 내지 "31"의 범위의 32개의 상이한 값에 대하여 사용되면, 예측 값은 32/2 = 16이다. 유익하게도, ODelta 연산자에는, 연산자를 이용하여 프로세싱될 데이터 요소에 대한 이용가능한 데이터 범위에 관한

정보가 제공될 필요가 있다.

[0157] 전술한 내용에 기재된 본 발명의 실시형태는, 비트 또는 임의의 디지털 값으로서 데이터(D1)에 제공되는 엔트로피(E)를 감소시킬 수 있다. 다이렉트 ODelta 연산자는 거의 항상 델타 코딩과 비교하여 개선된 엔트로피 감소를 제공한다. 델타 코딩이 바이트 랩어라운드와 함께 사용되고, 오리지널 예측(방법 1)에 대한 차이 ODelta 연산이 값 wrapValue=256, lowValue=MIN=0, 및 highValue=MAX=255을 사용하는 경우에만, 델타 디코딩 내에서 동일한 출력 결과를 생성한다. 다른 다이렉트 ODelta 방법이 사용되거나 또는 전체 데이터 범위가 입력 데이터에서 이용가능하지 않으면, ODelta 연산자는, 선택된 방법 또는 lowValue 및/또 highValue를 보냄으로써 더 나은 결과를 생성하며, 즉 이 연산자는 또한 wrapValue을 자동적으로 변경한다. 더 작은 엔트로피는, 데이터를 더 높은 데이터 압축비로 압축할 수 있게 한다. 더 높은 데이터 압축비는, 에너지 소비에서의 대응하는 감소에 대하여, 더 작은 용량 데이터 저장소를 사용할 수 있게 하고, 또한 압축된 데이터를 통신할 때 더 느린 데이터 대역폭을 사용할 수 있게 한다.

[0158] 전술한 내용에 있어서, 차 및 합 계산의 형태가 인코더(10)에서 실행되고, 대응하는 역 계산이 디코더(20)에서 수행된다는 것을 인식할 수 있다. 인코더(10) 내에 사용되는 다른 예측 방법을 이용하는 것이 또한 가능하며, 그 후, 대응하는 역 예측이 디코더(20)에서 수행된다. 이는 실제로 적어도 4개의 대응하는 역 ODelta 방법뿐만 아니라 적어도 4개의 상이한 다이렉트 ODelta 방법이 존재한다는 것을 의미한다. 이러한 방법의 상세하고 정확한 설명이 후속된다. 옵션적으로, 계산은, 인코딩된 데이터(D2)[또는 D3]에서 더 높은 등급의 데이터 압축을 획득하기 위하여 재귀적 방식으로 수행된다. 이러한 재귀적 계산을 실행할 때, 변경되는 숫자 범위는, 얼마나 많은 재귀적 계산이 사용되었는지에 대한 함수로서 사용된다. 예컨대, 인코더(10)에서, 이하의 계산 시퀀스는 인코딩된 데이터(D2)[또는 D3]를 생성하기 위하여 데이터(D1) 상에서 수행된다.

[Data D1] edirect ODelta (method 3) =>
edirect ODelta (method 3) =>
eEM =>
edirect ODelta (method 1) =>
eVLC [Data D3]

[0159] 식 21

[0160] 그리고 대응하는 역 연산들이 디코더(20)에서 수행된다.

[Data D3] dVLC =>
dinverse ODelta (method 1) =>
dEM =>
dinverse ODelta (method 3) =>
dinverse ODelta (method 3) [Data D5]

[0161] 식 22

[0162] 각 시간 데이터는, 식 21(방법 1에 대응하는 Eq. 21), 식 22(방법 2에 대응하는 Eq. 22), 식 23(방법 3에 대응하는 Eq. 23), 및 식 24(방법 4에 대응하는 Eq. 24)에 의해 표시되는 바와 같이, 이러한 4 개의 방법에서 연산되며, 이러한 방법들 중 하나가 다른 방법들보다 많이 프로세싱되는 데이터의 엔트로피를 감소시킬 수 있기 때문에, 모든 방법을 사용하도록 시도하는 것이 옵션적으로 가능하다. 인코더(10) 및/또는 디코더(20) 내에서의 방법들의 이용을 최적화하는 경우, 요구된 데이터 내의 정보량과 비교될 때, 많은 횟수로 동일하거나 또는 상이한 방법들을 사용하고, 그리고 선택된 방법, 또는 방법들이기만 하면, 엔트로피를 감소시키는 것이 유리하다. 이에 따라 방법 1 내지 4는 수치 값들을 인코딩하기 위하여 이용될 수 있고, 여기서 "수치 값"은, 복수-비트 값들뿐만 아니라 비트-바이-비트 방식에 의한 비트의 인코딩된 스트림에서와 같이, 그 정의 내에서 논-바이너리 숫자뿐만 아니라 1-비트 데이터를 포함한다.

[0163] 차 연산은 연속적인 숫자 값의 나머지를 나타내며; 이에 따라서, 합 연산은 연속적인 숫자 값의 합을 나타낸다. 인코더(10)에서 실행되는 이러한 연산들은 디코더(20)에서의 이들 자신의 대응하는 역 연산들을 가진다. 차 또는 합은, 현재 입력 값 그리고 예측 값으로서 사용되는 이전의 입력 또는 결과 값에 기초하여 계산될 수 있다. 다른 예측 값들이 또한 사용될 수 있고, 이러한 값들은, 디코더 내에서 또한 이와 같이 행하도록 반전될 수 있는 한, 예컨대 예측을 생성하기 위하여 인코더 내에서 초기에 입력 및 출력 값을 사용할 수 있다.

[0164] 이러한 방법들 중 어느 것도 인코더(10) 및 디코더(20) 내에서 현저하게 데이터를 압축하지 않지만, 모든 방법들은, 엔트로피를 감소시키기 위하여 유익하게 사용되므로, 일부 다른 압축 방법은 그 후 엔트로피-감소된 데이터를 더욱 유효하게 압축할 수 있다. 이러한 다른 압축 방법은, 옵션적으로 호프만 코딩, 산술 코딩, 레인지 코

딩, RLE 코딩, SRLE 코딩, 엔트로피 모디파이어 코딩 중 적어도 하나이다. 그러나, 모든 방법들에 대하여, 예컨대, 데이터의 손실없는 압축 및 후속의 손실없는 압축 해제가 달성되면, 연산 및 그 역 연산이 항상 정확하게 실행될 수 있는 적은 숫자 값들을 통신할 필요가 있다. 물론, 인코더(10) 및 디코더(20)는 어떤 종류의 숫자 값들이 입력 데이터(D1) 내에 포함되는 지에 관한 정보를 가진다. 유익하게도, 즉 MIN 및 MAX에 의해 규정된 숫자 범위가 공지된다고 가정된다. 원칙적으로, 방법들은 항상 기존 데이터 범위에 기초하여 직접적으로 기능할 수 있다. 연산들이 필요한 숫자 값들은, 최저 발생 숫자 값(lowValue) 및 최대 발생 숫자 값(highValue)이며; lowValue은 MIN 이상이며, highValue은 MAX 이하이다.

[0165] 이러한 값들에 기초하여, 다른 필요한 숫자 값들이 유도될 수 있다. 유익하게도, 이러한 값들은 다양한 형태로 통신되며, 여기서 소실된 값들이 유익하게 계산된다. 예컨대, 세트[“lowValue”, “highValue”, “숫자(number)”]로부터의 2개의 값이 알려지면, “숫자”는 [highValue - lowValue]이 되며, 그 후, 제3 값이 이로부터 계산될 수 있다. 데이터(D2) 내의 임의의 값들을 생략하고, 그 후 디코더(20)에서 이러한 값들을 유도하는 것은 데이터(D2)에서의 더 큰 데이터 압축을 제공할 수 있다.

[0166] 이러한 값들에 이외에, 제1 값의 계산시에 이전의 값으로서 사용될 수 있는 숫자 P 즉 “예측”이 요구된다. “0”과 “숫자” 값 사이의 값은 항상 숫자 P 즉, “예측”을 위하여 선택될 수 있다. 또한, 전술한 연산들은, 디코더(20)에서 데이터(D2/D3 또는 D4)를 디코딩할 때 회복가능하게 기능하기 위하여 즉, 연산들을 가능한한 작게 되도록 생성하는 값 범위를 감소시키기 위하여, “wrapValue”값이 제공될 필요가 있다. 그러나, 이러한 “wrapValue”은 “숫자”보다 더 크게 되어야 하며, 유익하게도, 이는 “숫자”값 +1을 가져야 한다. 옵션적으로, 데이터(D1)의 본성에 따라서, 제1 “예측”값은, 예컨대 데이터(D1)가 더 큰 값보다 더 작은 값을 포함하도록 가정되면, 전술한 바와 같이 “0”이 되도록 선택될 수 있다. 대안적으로, 제1 “예측”값은, 데이터(D1)가 더 작은 값들보다 보다 더 큰 값들을 포함하도록 가정되면, “숫자”와 동일하게 되도록 선택될 수 있다. 값들의 크기에 대하여 가정이 행해지지 않은 경우에, 그 후, “예측”값에 대하여 “(wrapValue + 1)/2 + lowValue”값을 이용하는 것이 바람직하다.

[0167] 이제, 본 개시의 실시형태들을 구현할 때 컴퓨팅 하드웨어에서 수행되는 동작들의 예를 설명한다.

[0168] 인코더(10)에 있어서, 제1 다이렉트 차 연산 즉, 방법 1은, 모든 데이터 값들에 대하여, 소프트웨어 루프에서 계산되는, 출력 값 즉, “결과”, 입력 값에 대응하는 것 즉, “오리지널”값에 대하여 유익하게 구현된다.

[0169]
```
result = original – prediction
if result < lowValue then result = result + wrapValue
```

[0170] 최종적으로, 다음 입력에 대한 예측 값은 현재 입력과 동일하게 되도록 설정된다.

[0171]
```
prediction = original
```

[0172] 디코더(20)에 있어서, 제1 역 차 연산 즉, 방법 1은, 모든 데이터 값들에 대하여, 소프트웨어 루프에서 계산되는, 출력 값 즉, “결과”, 입력 값에 대응하는 것 즉, “오리지널”값에 대하여 유익하게 구현된다.

[0173]
```
result = original + prediction
if result > highValue then result = result - wrapValue
```

[0174] 최종적으로, 다음 입력에 대한 예측 값은 현재 결과와 동일하게 되도록 설정된다.

[0175]
```
prediction = result
```

[0176] 인코더(10)에 있어서, 제2 다이렉트 차 연산 즉, 방법 2는, 모든 데이터 값들에 대하여, 소프트웨어 루프에서 계산되는, 출력 값 즉, “결과”, 입력 값에 대응하는 것 즉, “오리지널”값에 대하여 유익하게 구현된다.

[0177]
```
result = original – prediction
if result < lowValue then result = result + wrapValue
```

[0178] 최종적으로, 다음 입력에 대한 예측 값은 현재 결과와 동일하게 되도록 설정된다.

[0179]
```
prediction = result
```

[0180] 디코더(20)에 있어서, 제2 역 차 연산 즉, 방법 2는, 모든 데이터 값들에 대하여, 아래와 같이 소프트웨어 루프에서 계산되는, 출력 값 즉, “결과”, 입력 값에 대응하는 것 즉, “오리지널”값에 대하여 유익하게

구현된다.

[0181]
```
result = original + prediction
if result > highValue then result = result - wrapValue
```

[0182] 최종적으로, 다음 입력에 대한 예측 값은 아래와 같이 현재 입력과 동일하게 되도록 설정된다.

[0183]
```
prediction = original
```

[0184] 인코더(10)에 있어서, 제1 다이렉트 합 연산 즉, 방법 3은, 모든 데이터 값들에 대하여, 아래와 같이 소프트웨어 루프에서 계산되는, 입력 값 즉, "결과", 입력 값에 대응하는 것 즉, "오리지널" 값에 대하여 유익하게 구현된다.

[0185]
```
result = original + prediction
if result > highValue then result = result – wrapValue
```

[0186] 최종적으로, 다음 입력에 대한 예측 값은 아래와 같이 현재 입력과 동일하게 되도록 설정된다.

[0187]
```
prediction = original
```

[0188] 디코더(20)에 있어서, 제1 역 합 연산 즉, 방법 3은, 모든 데이터 값들에 대하여, 아래와 같이 소프트웨어 루프에서 계산되는, 입력 값 즉, "결과", 입력 값에 대응하는 것 즉, "오리지널" 값에 대하여 유익하게 구현된다.

[0189]
```
result = original – prediction
if result < lowValue then result = result + wrapValue
```

[0190] 최종적으로, 다음 입력에 대한 예측 값은 아래와 같이 현재 결과와 동일하게 되도록 설정된다.

[0191]
```
prediction = result
```

[0192] 인코더(10)에 있어서, 제2 다이렉트 합 연산 즉, 방법 4는, 모든 데이터 값들에 대하여, 아래와 같이 소프트웨어 루프에서 계산되는, 입력 값 즉, "결과", 입력 값에 대응하는 것 즉, "오리지널" 값에 대하여 유익하게 구현된다.

[0193]
```
result = original + prediction
if result > highValue then result = result - wrapValue
```

[0194] 최종적으로, 다음 입력에 대한 예측 값은 아래와 같이 현재 결과와 동일하게 되도록 설정된다.

[0195]
```
prediction = result
```

[0196] 디코더(20)에 있어서, 제2 역 합 연산 즉, 방법 4는, 모드 데이터 값들에 대하여, 아래와 같이 소프트웨어 루프에서 계산되는, 입력 값 즉, "결과", 입력 값에 대응하는 것 즉, "오리지널" 값에 대하여 유익하게 구현된다.

[0197]
```
result = original - prediction
if result < lowValue then result = result + wrapValue
```

[0198] 최종적으로, 다음 입력에 대한 예측 값은 아래와 같이 현재 입력과 동일하게 되도록 설정된다.

[0199]
```
prediction = original
```

[0200] 이러한 합 및 차 연산, 모든 4개의 방법은, 즉, 인코더(10) 및 디코더(20)의 ODelta 버전을 구현할 때, 또한 1-비트 데이터에 즉 비트-바이-비트로 적용될 수 있다. 1-비트 데이터의 상황에서, 다음 값들은 양쪽 인코더(10) 및 디코더(20)에 의해 이미 알려져 있으며 즉, MIN = 0, MAX = 1이다. 또한, lowValue = MIN = 0, 및 highValue = MAX = 1이 유익하게 가정된다. 또한, 이러한 경우에, "숫자"는 그러므로 [highValue - lowValue = 1 - 0 = 1]이고, wrapValue은 "숫자" + 1 = 1 + 1 = 2가 되도록 유익하게 선택된다. 유익하게, 예측 값은, lowValue = MIN = 0으로부터 개시되는 포지티브 값만을 가질 수 있는, 고려중인 1-비트 데이터만이 존재하기 때문에, "0" 값이 되도록 선택된다. 1-비트 데이터에 대하여, 방법 1 및 방법 3은 서로 유사한 코딩 결과를 산출

한다. 이와 유사하게, 방법 2 및 방법 4는 서로 유사한 코딩 결과를 산출한다. 이러한 지식을 가지는 것은, 여러 가지 디폴트가 가정될 수 있기 때문에, 데이터(D2)에 보내질 필요가 있는 정보를 유익하게 단순화하며, 방법 1 또는 방법 2 중 어느 하나로 그리고 선택된 예측[입력 값(방법 1)) 또는 결과 값(방법 2)]에 의해, 즉 단지 차 연산의 실행 횟수에 대한 정보를 보낼 필요가 있으므로, 디코더(20)는 디코딩된 데이터(D5)를 생성하기 위하여 데이터(D2, D3 또는 D4)를 디코딩할 때, 정확한 역 차 연산을 필요한 횟수로 실행할 수 있다.

[0201] 유사한 출력을 생성하는 방법 1 또는 방법 3을 이용함으로써 생성된 제1 예는, 또한 유사한 출력을 생성하는 방법 2 또는 방법 4 중 어느 하나를 이용함으로써 프로세싱될 수 있다. 이하에 도시된 결과는, 데이터 Eq. 1에 적용될 때 이러한 방법들에 의해 달성될 수 있다.

[0202] **011001000111100111111111111101010101**

[0203] 이 때, 프로세싱된 데이터는 24개의 "1" 그리고 13 개의 "0" 가지는데, 즉 엔트로피는 제1 예에서와 동일하지만, "1" 및 "0" 의 카운트는 경우에 따라서 변한다. 이는 또한 이러한 상이한 방법들 사이에서의 종종 발생하는 엔트로피 변화와는 달리, 항상 발생하지 않는다. 예컨대, 데이터의 4개의 제1 요소 이후에, 방법 1 및/또는 방법 3은 3개의 "1" 과 하나의 "0" 을 생성하는 반면에, 오리지널 데이터 및 방법 2 및/또는 방법 4에 의해 프로세싱되는 데이터는 2개의 "1" 과 2개의 "0" 을 가진다. 따라서, 방법 1 및/또는 방법 3은 이러한 경우에 방법 2 및/또는 방법 4보다 더 작고, 또한 오리지널보다 더 작은 엔트로피를 생성할 수 있다.

[0204] 멀티-비트 구현에 있어서, 데이터(D1)가 -64 내지 +63의 범위의 값을 포함하면, MIN = -64이고 MAX = 63이다. lowValue = MIN이고 highValue = MAX라고 가정함으로써, "숫자" = 127이고 wrapValue은 128이 되도록 유익하게 선택된다. 그러나, 데이터(D1)가 랜덤으로 변할 때, "예측" 값은 값[(wrapValue + 1)/2 + lowValue = 64 + -64 = 0]으로 유익하게 설정된다.

[0205] 제1 값이 예컨대 -1이면, 다이렉트 ODelta 방법 및/또는 방법 2으로 코딩된 제1 값은, -1 - 0 = -1이 되고, 이에 따라서, 다이렉트 ODelta 방법 3 및/또는 방법 4로 코딩된 값은, -1 + 0 = -1이 된다는 점에 주목해야 한다. 다음 값은 그 후 어떻게 데이터가 진행되는지에 따라서 변할 수 있는데, 예컨대 제2 값이 5이면, 다이렉트 ODelta 방법 1은 5 - -1 = 6을 생성하고, 다이렉트 ODelta 방법 2는 5 - -1 = 6을 생성하고, 다이렉트 ODelta 방법 3은 5 + -1 = 4을 생성하고, 다이렉트 ODelta 방법 4는 5 + -1 = 4를 생성한다. 디코더(20)는, 이 경우에 있어서, 역 ODelta 방법 1 및/또는 방법 2를 이용할 때 제1 값으로서 -1 + 0 = -1을 생성할 수 있고, 역 ODelta 방법 3 및/또는 방법 4를 이용할 때 제1 값으로서 -1 - 0 = -1을 생성할 수 있다. 이에 따라서, 역 ODelta 방법 1을 이용한 제2 값은 6 + -1 = 5이 되고, 역 ODelta 방법 2에 의해, 제2 값은 6 + -1 = 5이 되고, 역 ODelta 방법 3에 의해, 제2 값은 4 - -1 = 5이 되고, 역 ODelta 방법 4에 의해, 제2 값은 4 - -1 = 5이 된다.

[0206] 그 후, 이 솔루션은 숫자 범위가 실제로 -20 내지 +27 사이의 값만을 포함하면 최적화될 수 있다. 이러한 예시적인 경우에서, 예컨대 lowValue = -20 및 highValue = 27을 송신하는 것이 실현될 수 있다. 양쪽이 송신되면, 숫자 = 47이 되도록 계산하는 것이 실현될 수 있고, 그 후, wrapValue은 48이 되도록 유익하게 선택된다. 이제, 예측을 위하여 값[48/2 + -20 = 4]을 계산하는 것이 실현될 수 있다. 그 후, 이전의 예는 예컨대 ODelta 방법 1 또는 방법 2가 사용될 때 값 -1에 대하여 [-1 - 4 = -5]을 산출하며, ODelta 방법 또는 방법 4가 사용될 때 [-1 + 4 = 3]을 산출한다. 이와 유사하게, 제2 값은 ODelta 방법에 대하여 (5 - -1) = 6, (5 - -5) = 10, (5 + -1) = 4, 및 (5 + 3) = 8로서 이루어진다. 디코딩(20)은 다시 정확하게 기능하며, 방법 1 및/또는 방법 2에 대하여 -5 + 4 = -1로서 제1 값을 산출하고, 방법 3 및/또는 방법 4에 대하여 3 - 4 = -1로서 제1 값을 산출한다. 이에 따라서, 상이한 방법에 대한 제2 값은 (6 + -1) = 5, (10 + -5) = 5, (4 - -1) = 5, 및 (8 - 3) = 5로서 디코딩된다.

[0207] 상기 이러한 예들에서의 모든 값은 범위 즉, -64 내지 +63 또는 -20 내지 +27 내에 있고, 이에 따라서 이러한 예시적인 값 내에 보정항을 수행할 필요가 없지만, 만일 임의의 네가티브 또는 포지티브 변화가 충분히 크면, 범위 내에 결과 값을 유지하기 위하여 데이터 값에 대한 보정이 주어진 식 21 내지 24 (Eq. 21 내지 Eq. 24)에 의해 행해져야 한다. 여기서 보정항은 랩어라운드 값을 지칭한다는 점에 주목해야 한다.

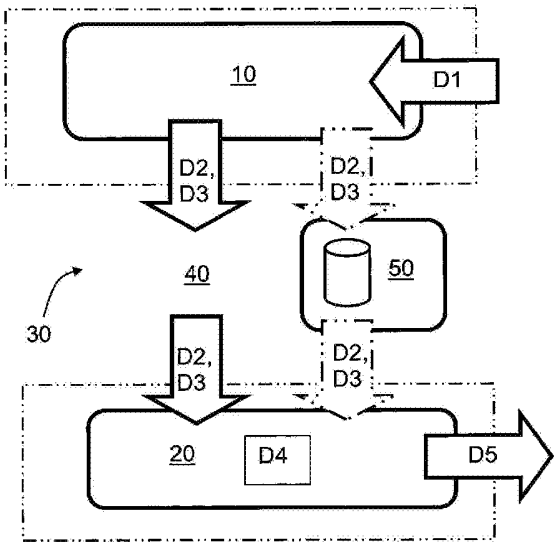
[0208] lowValue이 알려질 때, 코딩된 값들은, 인코더(10)로부터 디코더(20)까지 엔트로피 인코딩된 데이터(D3)가 보내져야 하는 코딩 테이블을 간략화하기 위하여, 0으로부터 개시하여 "숫자" 값으로 종료되도록 유익하게 배치된다. 이러한 연산은 포스트-오프셋으로 지칭되며, 이러한 포스트-오프셋은, 엔트로피 디코딩 이후에 데이터(D4)에 대한 역 ODelta 연산 이전에 코딩된 데이터 값으로부터 삭제되어야 한다.

[0209] 전술한 바와 같이, 프리-오프셋 기능을 가진 오프셋을 구현하는 것이 또한 가능하며, 여기서 오리지널 입력 데이터(D1)는 이미 ODelta 방법의 실제 실행 이전에, 0부터 "숫자" 까지의 값을 포함할 수 있는 포지티브 요소로 변환된다. 또한 이러한 상황에 있어서, 이러한 동작이 요구되는 정보 전송은, "프리-오프셋" 및 ODelta 방법이 동일한 정보를 반복적으로 송신하지 않거나 또는 일부 다른 방법 때문에 이미 알려진 것을 무시하는 이러한 방식으로 수행하는 것이 유익하다. 이러한 프리-오프셋 결과는, 적절한 D5 출력 데이터를 생성하기 위하여 역 ODelta 연산 이후에 디코딩된 데이터로부터 삭제되어야 한다.

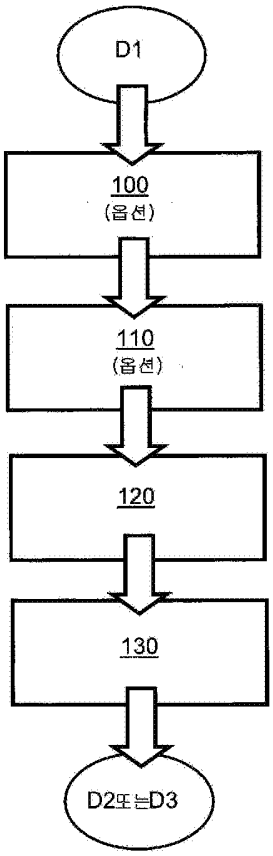
[0210] 전술한 본 발명의 실시형태들은 첨부된 특허 청구범위에서 규정하는 본 발명의 범위로부터 벗어나지 않고 수정될 수 있다. 본 발명을 설명하고 청구하기 위해 사용되는 "포함하는", "구비하는", "통합하는", "이루어진", "가진", "...인" 등의 표현은 비 배타적 방식, 즉, 명시적으로 설명하지 않은 항목, 구성요소 또는 요소들도 또한 존재할 수 있는 것으로 해석되어야 한다. 단수형으로의 인용은 복수형과 관련이 있는 것으로 또한 해석되어야 한다. 첨부된 특허 청구범위에서 괄호 안에 포함된 번호들은 특허 청구범위의 이해를 돕기 위한 것으로 의도되고, 어떻게든 특허 청구범위에 의해 청구되는 주제를 제한하는 것으로 해석되지 않아야 한다.

도면

도면1



도면2



도면3