(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년02월16일
(11) 등록번호 10-1707463
(24) 등록일자 2017년02월10일

(51) 국제특허분류(Int. Cl.)
G06T 15/40 (2011.01) *G06T 9/00* (2006.01)
H04N 19/597 (2014.01)
(52) CPC특허분류
G06T 15/405 (2013.01)
G06T 9/00 (2013.01)
(21) 출원번호 10-2016-7006561(분할)
(22) 출원일자(국제) 2011년12월29일
심사청구일자 2016년04월19일
(85) 번역문제출일자 2016년03월11일
(65) 공개번호 10-2016-0032277
(43) 공개일자 2016년03월23일
(62) 원출원 특허 10-2014-7017842
원출원일자(국제) 2011년12월29일
심사청구일자 2014년06월27일
(86) 국제출원번호 PCT/US2011/067902
(87) 국제공개번호 WO 2013/101095
국제공개일자 2013년07월04일
(56) 선행기술조사문헌
KR1020000049031 A*
KR1020100083980 A*
KR1020100102493 A
JP2011205529 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
아케닌-몰러 토마스 지
스웨덴 에스-227 38 엠 스웨덴 런드 로젠바겐 13
닐슨 짐 케이
스웨덴 에스-226 39 엠 스웨덴 런드 스벤스카 바겐 4
(뒷면에 계속)
(74) 대리인
제일특허법인

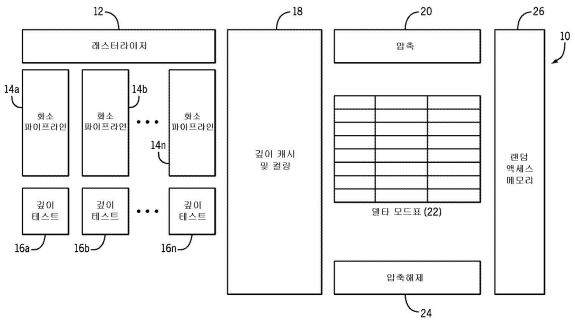
전체 청구항 수 : 총 30 항 심사관 : 조우연

(54) 발명의 명칭 **가변 깊이 압축**

(57) 요 약

일부 실시예에 따라, 깊이 압축에 할당된 비트수는 고려사항의 수에 기초하여 여러가지로 변경될 수 있다. 그 결과, 깊이 데이터는 더 효율적인 방식으로 압축될 수 있다.

대 표 도



(52) CPC특허분류
H04N 19/597 (2015.01)
G09G 2320/0261 (2013.01)
G09G 2320/10 (2013.01)
G09G 2340/02 (2013.01)
(72) 발명자
앤더슨 매그너스
스웨덴 스웨덴 에스-25375 엠 헬싱보르그 카프리폴 가탄 37
하셀그렌 존 엔
스웨덴 스웨덴 에스-21832 엠 벌켈플로스트랜드 스파르바겐 4

명 세 서

청구범위

청구항 1

깊이 데이터(depth data)를 압축하는 방법으로서,

깊이 데이터의 특성을 결정하는 단계와,

그래픽 프로세서를 이용하여 상기 깊이 데이터의 특성에 따라, 잔차값(residual values)을 인코딩하는 데 사용되는 비트수를 변동시킴(varying)으로써 상기 깊이 데이터를 압축하는 단계 - 상기 잔차값은 예측된 깊이 값과 실제 깊이 값 사이의 차이임 - 를 포함하는

방법.

청구항 2

제 1 항에 있어서,

상기 잔차값을 인코딩하는 데 사용되는 비트수를 변동시키는 것은 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 단계를 포함하는

방법.

청구항 3

제 1 항에 있어서,

상기 잔차값을 인코딩하는 것은 고정 지점 인코딩(anchor point encoding)을 이용하는 단계를 포함하는

방법.

청구항 4

제 2 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 단계는,

X 벡터 및 Y 벡터에 대한 비트수를 선택하는 단계와,

깊이당 잔차 비트수를 할당하는 단계를 포함하는

방법.

청구항 5

제 4 항에 있어서,

상기 선택 단계와 상기 할당 단계는 X 벡터, Y 벡터 및 깊이당 잔차 비트에 대한 상기 비트수 각각의 선택을 가능하게 하는 단계를 포함하는

방법.

청구항 6

제 2 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 단계는 복수의 선택 가능한 모드를 제공하는 단계를 포함하고,

각각의 모드는 X 벡터, Y 벡터 및 깊이당 잔차 비트의 특정 비트수를 특정하는

방법.

청구항 7

제 6 항에 있어서,

상기 특정 비트수를 특정하는 것은 고정 지점에 대해, 델타 벡터를 나타내는 데 필요한 최소 비트수를 계산하는 단계를 포함하는

방법.

청구항 8

제 7 항에 있어서,

상기 최소 비트수를 계산하는 단계는 상기 델타 벡터에 대해 필요한 비트수를 제공하는 최대 잔차 비트수를 갖는 모드를 선택하는 단계를 포함하는

방법.

청구항 9

제 8 항에 있어서,

상기 최대 잔차 비트수를 갖는 모드를 선택하는 단계는 주어진 고정 지점에 얼마나 많은 잔차 비트가 필요한지를 나타내는 표시자를 저장하는 단계를 포함하는

방법.

청구항 10

제 2 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 단계는,

상기 잔차값에 대한 상기 비트수가 대응하여 증가할 수 있는 경우에만 X값 또는 Y값에 대한 비트수를 감소시키는 단계를 포함하는

방법.

청구항 11

깊이 데이터를 압축하기 위해 명령어를 저장하는 컴퓨터 판독 가능한 기록 매체로서,

상기 명령어는 프로세서에 의해 실행되어,

깊이 데이터의 특성을 결정하고,

상기 깊이 데이터의 특성에 따라 잔차값을 인코딩하는 데 사용되는 비트수를 변동시킴으로써 상기 깊이 데이터를 압축 - 상기 잔차값은 예측된 깊이 값과 실제 깊이 값 사이의 차이임 - 하는

컴퓨터 판독 가능한 기록 매체.

청구항 12

제 11 항에 있어서,

상기 잔차값을 인코딩하는 데 사용되는 비트수를 변동시키는 것은 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 13

제 11 항에 있어서,

상기 잔차값을 인코딩하는 것은 고정 지점 인코딩을 이용하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 14

제 12 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은, X 벡터 및 Y 벡터에 대한 비트수를 선택하고, 깊이당 잔차 비트수를 할당하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 15

제 14 항에 있어서,

상기 선택하고 할당하는 것은 X 벡터, Y 벡터 및 깊이당 잔차 비트에 대한 상기 비트수 각각의 선택을 가능하게 하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 16

제 12 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은 복수의 선택 가능한 모드를 제공하는 것을 포함하고,

각각의 모드는 X 벡터, Y 벡터 및 깊이당 잔차 비트의 특정 비트수를 특정하는

컴퓨터 판독 가능한 기록 매체.

청구항 17

제 16 항에 있어서,

상기 특정 비트수를 특정하는 것은 고정 지점에 대해, 델타 벡터를 나타내는데 필요한 최소 비트수를 계산하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 18

제 17 항에 있어서,

상기 최소 비트수를 계산하는 것은 상기 델타 벡터에 대해 필요한 비트수를 제공하는 최대 잔차 비트수를 갖는 모드를 선택하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 19

제 18 항에 있어서,

상기 최대 잔차 비트수를 갖는 모드를 선택하는 것은 주어진 고정 지점에 얼마나 많은 잔차 비트가 필요한지를 나타내는 표시자를 저장하는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 20

제 12 항에 있어서,

상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은,

상기 잔차값에 대한 상기 비트수가 대응하여 증가할 수 있는 경우에만 X값 또는 Y값에 대한 비트수를 감소시키는 것을 포함하는

컴퓨터 판독 가능한 기록 매체.

청구항 21

깊이 데이터를 압축하는 장치로서,

깊이 데이터의 특성을 결정하고, 상기 깊이 데이터의 특성에 따라 잔차값을 인코딩하는 데 사용되는 비트수를 변동시킴으로써 상기 깊이 데이터를 압축하는 프로세서 - 상기 잔차값은 예측된 깊이 값과 실제 깊이 값 사이의 차이임 - 와,

상기 프로세서에 연결된 스토리지를 포함하는

장치.

청구항 22

제 21 항에 있어서,

상기 프로세서가 상기 잔차값을 인코딩하는 데 사용되는 비트수를 변동시키는 것은 델타값을 인에이블하는 데 사용된 비트수를 변동시키는

장치.

청구항 23

제 21 항에 있어서,

상기 프로세서가 상기 잔차값을 인코딩하는 것은 고정 지점 인코딩을 이용하는

장치.

청구항 24

제 22 항에 있어서,

상기 프로세서가 상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은, X 벡터 및 Y 벡터에 대한 비트수를 선택하고, 깊이당 잔차 비트수를 할당하는

장치.

청구항 25

제 24 항에 있어서,

상기 프로세서가 선택하고 할당하는 것은 X 벡터, Y 벡터 및 깊이당 잔차 비트에 대한 상기 비트수 각각의 선택을 가능하게 하는

장치.

청구항 26

제 22 항에 있어서,

상기 프로세서가 상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은 복수의 선택 가능한 모드를 제공하고,

각각의 모드는 X 벡터, Y 벡터 및 깊이당 잔차 비트의 특정 비트수를 특정하는

장치.

청구항 27

제 26 항에 있어서,

상기 프로세서가 상기 특정 비트수를 특정하는 것은 고정 지점에 대해, 델타 벡터를 나타내는 데 필요한 최소 비트수를 계산하는

장치.

청구항 28

제 27 항에 있어서,

상기 프로세서가 상기 최소 비트수를 계산하는 것은 상기 델타 벡터에 대해 필요한 비트수를 제공하는 최대 잔차 비트수를 갖는 모드를 선택하는

장치.

청구항 29

제 28 항에 있어서,

상기 프로세서가 상기 최대 잔차 비트수를 갖는 모드를 선택하는 것은 주어진 고정 지점에 얼마나 많은 잔차 비트가 필요한지를 나타내는 표시자를 저장하는

장치.

청구항 30

제 22 항에 있어서,

상기 프로세서가 상기 델타값을 인에이블하는 데 사용된 비트수를 변동시키는 것은 상기 잔차값에 대한 상기 비트수가 대응하여 증가할 수 있는 경우에만 X값 또는 Y값에 대한 비트수를 감소시키는

장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 전반적으로 그래픽 처리에 관한 것으로, 특히, 통계론적 모션 블러 래스터화(motion blur rasterization)를 위한 깊이 버퍼의 압축에 관한 것이다.

배 경 기 술

[0002] 모션 블러 래스터화는 움직이는 대상을 더 정확하게 표현하는 것, 특히 대상이 매우 빠르게 이동할 때 관찰되는 블러를 표현하는 것을 시도한다. 모션 블러를 무시하면, 장면 내의 프리미티브의 깊이에 대한 정보는 쉽게 압축될 수 있다. 특히, 각 프리미티브의 깊이는 이미징 장치로부터 프리미티브까지의 거리로서 결정될 수 있다. 모션 블러가 관련되지 않을 때 이 깊이를 감소시키는 알고리즘이 존재한다.

[0003] 그러나, 모션 블러가 관련되는 경우, 그 동작은 깊이 정보를 압축하는 것을 시도하는 것은 훨씬 더 복잡하다.

발명의 내용

해결하려는 과제

[0004] 그래픽 프로세서는 메모리 대역폭 사용이 증가함에 따라 증가하는 전력 소모에 매우 민감한 경향이 있다. 메모리 대역폭은 또한 현대의 그래픽 프로세서에서 그 자체가 부족한 리소스이다. 현재, 모션 블러 및 필드의 깊이의 통계론적 래스터화를 취득하는 것, 그리고 상호작용 또는 심지어 실시간 렌더링 기반으로 이 정보를 취득하는 것이 더 강조된다. 이것은 상당한 메모리 대역폭 사용을 수반하고, 증가된 전력 소모를 시사한다.

도면의 간단한 설명

[0005] 일부 실시예는 다음 도면에 대해 설명된다.

도 1은 본 발명의 일 실시예의 개략도이다.

도 2는 일 실시예에 따라 고정 지점용 모드 선택을 위한 흐름도이다.

도 3은 일 실시예에 따라 동적 잔차값 비트 할당을 위한 흐름도이다.

도 4는 일 실시예에 대한 시스템도이다.

발명을 실시하기 위한 구체적인 내용

[0006] 일부 실시예에 따라, 깊이 압축(depth compression)에 할당된 비트수는 고려사항의 수에 기초하여 다양하게 변경될 수 있다. 그 결과 깊이 데이터는 더 효율적인 방식으로 압축될 수 있다.

[0007] 깊이 버퍼 압축은 메모리 대역폭 사용을 감소시키는 한 기술이다. 일반적으로, 하나의 타일 내의 임의의 프리

미티브의 깊이 z는 일반적으로 삼각형인 프리미티브에 걸쳐 선형이다. 이것은 비용이 비싸지 않은 압축 및 압축해제 알고리즘을 구성하도록 활용될 수 있다.

[0008] 평면 인코딩(plane encoding)은 이 선형 관계의 이점을 취하도록 시도한다. 래스터라이저(rasterizer)는 압축기에 정확한 평면 방정식을 제공하고, 따라서, 잔차 항(residual)이 필요없다. 그러나, 모션 블러가 관련되는 경우, 각각의 정점이 선형 함수에 따라 이동할 수 있다. 그러면, 샘플에서의 깊이 함수는 유리 3차 함수이다. 이 함수는 단순한 예측변수 함수를 이용하여 전체 타일에 걸친 깊이를 예측하는 것을 상당히 더 어렵게 한다. 그 때문에, 표준 깊이 버퍼 압축 기술은, 특히 정확한 평면 방정식을 이용하는 것은, 그러한 노이즈 버퍼를 압축하는 데 실패할 수 있다.

[0009] 타일이라 불리는 화소의 블럭은 독립적으로 처리될 수 있다. 현재의 깊이 압축 방식은 모션 블러 및 필드의 깊이를 명료하게 처리하지 않고, 따라서, 그들은 시간 요소 또는 렌즈 파라미터를 갖지 않는다.

[0010] 일반적으로, 깊이 압축 알고리즘은 클러스터링, 예측변수 함수 생성 및 잔차 인코딩으로 불릴 수 있는 3개의 공통 단계를 이용한다. 클러스터링은, 타일의 샘플의 세트가 다른 층, 예컨대, 배경층 및 전경층에 속하는 경우에 사용된다. 이 경우, 동일한 예측변수 함수를 이용하여 타일의 모든 깊이를 압축하기는 어렵다.

[0011] 클러스터링 단계는 타일의 샘플을 2개 이상의 층으로 분리하도록 시도하고, 각 층의 샘플은 일반적으로 공통 평면에 놓여 있는 것과 같은 일부 특성을 공유한다. 2개 이상의 층으로 샘플을 분할하는 것의 목표는, 단일 층에서 모든 샘플을 압축하는 것에 비해 이론상 각각의 층이 압축하기가 더 간단해질 수 있는 것이다.

[0012] 예측변수 함수 생성에서, 각각의 층은 그 자신의 예측변수 함수를 생성한다. 하나의 목표는 저렴한 함수(inexpensive function)를 이용하여 각 샘플의 깊이를 예측하기 위해 예측변수 함수(predictor function)를 생성하도록 깊이 샘플 및 가능하다면 그들의 고정 좌표를 이용하는 것이다. 예컨대, 작은 화소당 변위를 갖는 작은 사각형이 타일에 렌더링되었다고 가정한다. 예측변수 함수로서, 대개 변위된 깊이가 놓인 곳에 대해 좋은 추측이 있기 때문에, 사각형의 평면을 이용할 수 있다. 추측이 정확하지 않은 경우라도, 결함은 다음 단계에서 처리될 수 있다.

[0013] 그래픽 프로세서의 통상 요건은 깊이 버퍼가 손실이 없는 것이기 때문에, 잔차 인코딩은 타일의 압축해제 중에 더 정확한 깊이가 재구성되게 할 수 있다. 잔차(residual)는 예측변수 함수와 실제 샘플 깊이 사이의 차이이다. 양호한 예측변수 함수를 고려하면, 샘플의 깊이와 예측변수 함수 사이의 잔차는 작을 수 있다. 그 결과, 그들 잔차는 비교적 적은 비트를 이용하여 인코딩될 수 있다. 그 후, (적은 수의 층이 있으면) 양호한 압축률이 달성될 수 있고, 예측변수 함수에 요구되는 스토리지는 작다.

[0014] 일부 실시예에서, 깊이 버퍼 압축은 고정 인코딩(anchor encoding)을 이용할 수 있지만 평면 인코딩(plane encoding)을 포함하여 다른 기술도 마찬가지로 이용될 수 있다. 평면 방정식 및, 평면 데이터 도함수 또는 델타 벡터에 기초하는 평면 인코딩의 구성은, 진정한 도함수가 이들 도함수를 저장하기 위해 할당된 비트 예산에 비해 매우 크다는 점에서 과도한 평면을 처리하는 데 실패한다. 결과적으로, 실제 평면 표현 정확도는 저하될 수 있다. 이 정확도 저하는 델타 벡터가 추정되는 지점에서 타일 내의 거리를 증가시킨다. 한편, 단순히 델타 벡터에 너무 많은 비트를 할당하는 것은 깊이값 잔차를 저장하는데 이용 가능한 비트수를 감소시킨다.

[0015] 따라서, 일부 실시예에서, 델타 벡터에 이용 가능한 비트를 정적으로 할당하는 것의 영향은 깊이 데이터의 성질에 따라 비트의 동적 할당을 가능하게 함으로써 감소될 수 있다.

[0016] 래스터화된 구조에 따라, 타일은 다수의 프리미티브의 타겟일 수 있고, 시간 경과에 따라 깊이 값의 복잡한 분산을 초래한다. 이들 깊이 값의 기초 예측으로서 하나 이상의 평면 방정식을 이용하면, 잔차값은 압축된 깊이 데이터 포맷으로 인코딩된다. 예측변수 평면의 성질의 결과로서, 예측된 깊이 값은 거의 정확할 수 있고, 잔차 정정 비트에 대한 다양한 요구를 초래한다.

[0017] 현재의 깊이 압축 메커니즘은 모든 타일 위치에 대해 동일한 수의 잔차 비트를 정적으로 할당하고, 이는 잠재적으로 실제 깊이 값에 대해 그다지 일치하지 않을 수 있다. 일부 실시예에서, 잔차 정정 비트에 이용 가능한 비트를 정적으로 할당하는 것의 영향은, 이 대신, 개별적 타일 위치에 대해 다수의 잔차 비트를 동적으로 할당하게 함으로써 감소된다.

[0018] 고정 인코딩 실시예에서, 깊이 데이터 압축 메커니즘은 후보 고정 지점으로서 타일의 하나 이상의 지점을 선택함으로써 래스터화된 프리미티브의 평면 표현을 검출한다. 고정 지점의 추정은 깊이 데이터 값에 기초하여 국소 및 평면 X 및 Y 도함수의 계산을 포함한다. 예측변수 평면 표현 (zp(x,y)=a+b*x+c*y)는 a, b, c의 3개의

값으로서 압축된 깊이 데이터로 인코딩되고, 여기서 a는 고정 지점에서의 깊이값, b는 dZ/dX이며, c는 dZ/dY이다. 값 b 및 c는 델타값 또는 "벡터"이다.

[0019] 타일에서 각각의 깊이 z(x,y)에 대해, 잔차 깊이값은 예측된 값과 진정한 깊이값 사이의 차로서 저장된다. 값 d는 잔차값이다.

[0020] 델타 및 잔차값에 대해 할당된 다수의 비트를 선택함으로써, 타일 깊이 데이터에 대한 총 비트 예산은 깊이값 정확도의 손실 없이 감소될 수 있다. 깊이 데이터의 순 압축은 일부 실시예에서 달성될 수 있다.

[0021] 우리는 X, Y, R의 특정 조합을 지정하는 델타 모드의 개념을 소개하고, 여기서, X는 델타-X 벡터에 대한 비트의 수이고, Y는 델타-Y 벡터에 대한 비트의 수이고, R은 깊이당 잔차 비트의 수이다. 압축시에 이들 모드는 타일의 각각의 고정 지점에 대한 선택에 이용 가능하다.

[0022] 하나의 변형에서, 각각의 고정 지점은 잠재적으로 이용 가능한 모드의 그 자신의 세트를 가질 수 있고, 그 모드는 압축중에 동적으로 생성되고 압축결과에 의해 안내된다.

[0023] 다른 변형에서, 인코딩을 단순화하기 위해, 동일한 수의 비트가 양 방향(X 및 Y)에 대해 사용될 수 있다. 이하, 우리는 압축이 시작되기 전에 정적으로 생성되고 저장될 모드표를 고려한다. B=512, N=4, M=3, A=32, T=32인 경우 델타 모드의 예시적인 세트에 대해 표 1을 참조한다.

표 1

모드	X	Y	R
1	3	3	14
2	6	6	13
3	9	9	11
4	8	10	11
5	10	8	11
6	13	13	10
7	11	17	9
8	17	11	9

[0024] 델타 모드 표의 예

[0025] 타일 크기 및 압축을 위한 비트 예산을 고려하면, 델타값 및 잔차값에 이용 가능한 비트수 사이에 트레이드오프가 있다. X, Y, R의 모든 조합이 관련되는 것은 아니다. 단지, R이 적어도 1비트만큼 증가할 수 있으면 X 및/또는 Y를 감소시키는 것에 의미가 있다. 이것은 이용 가능한 모드의 수를 제한한다.

[0026] 다음의 조건은 타일의 압축을 가능하게 하기 위해 충족되어야 한다.

[0027] $T*\log2(N)+ N * (M + A + X + Y) + (T - N * 3) * R < B$

[0028] 여기서, B=압축을 가능하게 하는 총 비트 예산

[0029] N=고정 지점의 수

[0030] M=고정 지점에 대해 사용된 모드를 표현하는 log2(모드의 수) 비트

[0031] A=고정 지점 깊이값에 사용된 비트 수

[0032] X=X 델타에 사용된 비트수

[0033] Y=Y 델타에 사용된 비트수

[0034] T=타일 내의 깊이의 총 수

[0035] R=잔차 깊이값에 사용된 비트수

[0036] T*log2(N) 항은 깊이가 사용되는 평면 방정식을 표시하기 위해 타일의 깊이당 log2(N)를 예약한다. (T-N*3) 항은 제 1차 도함수(dZ/dX 및 dZ/dY) 뿐만 아니라 고정 지점 깊이가 정확하다는, 즉 그들이 잔차 비트의 스토리지를 필요로 하지 않는 것을 시사한다. 다른 실시예에서, 고정 지점 뿐만 아니라 제 1 차 도함수도 정확하지 않고, 이는 그들 깊이에 대한 잔차 비트도 필요로 할 것이고, 상기 불균형의 마지막 항을 (T-N*3) 대신 T로 만든다. 고정 지점은 정확하지만 도함수가 그렇지 않으면, 그 항은 대신 (T-N)이 된다. 상기 예에서, 도함수 뿐만

아니라 고정 값은 정확한 깊이 값에 정확하게 이르고, 이는 종종 타당한 가정이다.

[0037] 각각의 고정 지점에 대한 모드를 선택하는 알고리즘은 다음과 같이 일 실시예에 따른다.

[0038] 2M 열을 갖는 표는 각각의 방향 (X 및 Y)에 대해 다수의 할당된 델타 비트를 저장하도록 구성된다. 예컨대 표 1을 참조하라, 그러면, N 개의 고정 지점 위치가 주어지고, N=1, 2, 3, …, T(통상 1 과 4 사이)이다. 다음에, 각각의 고정 지점에 대해, 델타 벡터를 표현하기 위해 최소로 필요한 비트수가 계산된다. 이것은 고정 지점의 깊이부터 dZ/dX에 대해 오른쪽(또는 왼쪽) 인근까지, 및, dZ/dY에 대해 위쪽(또는 아래쪽) 인근까지의 델타 벡터를 간단히 계산함으로써, 그리고 얼마나 많은 비트가 필요한지 조사함으로써(첫번째 최상위 세트 비트를 찾음으로써) 행해질 수 있다. 잔차 비트의 최대의 수에 대응하는 모드는 이 고정 지점에 대한 모드로서 선택된다.

[0039] 이 간단한 방식의 이점은, 그것이 더 많은 타일이 압축되게 하고, 결국, 깊이에 대해 메모리 대역폭 사용을 감소시킨다(이는 그래픽 프로세서의 대역폭을 상당히 소비하는 것이었음). 예컨대, 동시에 깊이당 다수의 잔차 비트를 필요로 하는 큰 델타 벡터를 갖는 타일이 압축될 수 있다. 유사하게, 많은 잔차 비트를 필요로 하는 작은 델타 벡터를 갖는 타일이 마찬가지로 압축될 수 있다.

[0040] 불충분한 정확도 및/또는 정밀도로 인해, 결국, 과도한 평면 표현 또는 그저 복잡한 타일 데이터로 인해, 평면 예측변수는 정확한 깊이 값을 조정하기 위해 타일 위치당 잔차 비트를 필요로 한다. 예측의 부정확한 양에 따라, 다양한 수의 잔차 비트가 예측과 정확한 값 사이의 차를 인코딩하기 위해 실제로 필요하다.

[0041] 얼마나 많은 잔차 비트가 그 위치에 대해 필요한지 나타내는 표시자를 저장하기 위해 타일 위치당 하나의 엔트리와 함께 잔차 모드 마스크를 이용하는 것을 제안한다. 그 목표는, 각 위치에 대해 가능한 한 적은 비트를 이용하고, 그 결과 필요한 총 비트를 최소화하는 것이다.

[0042] 각 위치에 대해 사용된 잔차 비트의 가능한 수는 적절한 값으로 정적으로 할당될 수 있고, 또는 타일 데이터에 기초하여 동적으로 계산될 수 있다.

[0043] 동적으로 할당된 잔차 모드에 대해 알고리즘은 일 실시예에 따라 다음과 같이 동작한다.

[0044] 타일의 압축 동안, 예측변수 깊이는 실제 깊이와 비교되고 잔차값을 계산한다. 그 후 최단 모드(즉, 가장 적은 비트수로 필요한 차를 캡쳐하는 모드)가 그 잔차를 인코딩하기 위해 선택된다. 다음에, 대응하는 모드값이 잔차 모드 마스크에 채워진다. 마지막으로, 잔차 모드 마스크가 인코딩된 타일 데이터에 저장된다.

[0045] 그 기술은 일례로 예시될 수 있다.

표 2

모드	잔차 비트
00=0	2
01=1	8
10=2	12
11=3	16

[0046] 4개의 모드를 갖는 예시적 잔차 모드표

표 3

	X0	X1	X2	X3
Y0	00	00	01	10
Y1	00	01	11	00
Y2	00	10	01	00
Y3	01	00	00	00

[0047] 대응하는 인코딩을 갖는 4x4 화소의 예시적 타일

[0048] 예에서, 2비트를 사용하는 모드 00의 9가지 경우가 있기 때문에 잔차에 대한 총 비트수는 90(9x2+4x8+2x12+16)이다. 전체 값 잔차를 저장한 결과 256비트(16x16)이다.

[0049] 본 예에서, 우리는 각 깊이 값에 대해 2비트 값을 갖는다. 그러나, 누군가는 (사용된 2비트 값의 수를 감소시키기 위해) 각 2x2 깊이에 대해 2비트 값을 사용하도록 선택할 수도 있다. 일반적으로, 각각의 WxH 깊이는 그

모드를 나타내기 위해 Q비트를 사용할 수 있다.

[0050] 정적 할당에 대해, 우리는 선택하기 위해 타일 당 여러가지 모드를 공급할 수 있다. 예컨대, 하나의 모드는 잔차 비트가 완전히 균일한 방식으로 깊이에 걸쳐 분산되는 것일 수 있다. 다른 모드는 고정 지점에 가까울수록 더 적은 비트를, 깊이가 타일에 위치한 고정 지점으로부터 더 멀수록 더 많은 비트를 사용할 수 있다.

[0051] 고정 인코딩 기반 압축 기술을 이용하는 실시예가 기술되었지만, 동적 잔차 비트 할당은 잔차를 인코딩하는 임의의 압축기에도 사용될 수 있다. 동적 델타값 비트 할당은 다른 예측변수와 함께 사용하는 것 또한 가능하다. 예컨대, 우리가 예측변수로서 바이리니어 패치(a bilinear patch)(4개의 깊이 값에 그들 x, y 위치를 더한 것으로부터 생성됨)를 이용하면, 그 패치를 하나의 고정 지점 및 2개의 델타 벡터로서 인코딩할 수 있고, 그 후 처음 3개의 지점과 네번째 지점으로부터의 평면 방정식 사이의 차이인 4개의 지점에 대한 잔차값을 인코딩할 수 있다. 이들 2개의 델타 벡터에서 소비된 비트수 및 4개의 지점에 대한 잔차 비트는 고정 인코딩에 대해 상기에 기술된 바와 마찬가지 방식으로 동적으로 할당될 수 있다. 동일한 것이 다른 예측변수 함수에 마찬가지로 쉽게 적용될 수 있다.

[0052] 도 1을 참조하면, 장치(10)는 래스터라이저 및 일련의 화소 파이프라인(14a-14n)을 포함할 수 있다. 깊이 테스트 유닛(16a-16n)은 각 화소 파이프라인에 제공될 수 있다. 깊이 캐시 및 컬링(culling) 유닛(18)은 깊이 값 컬링에 사용될 수 있다. 압축은 압축기(20)에서 행해질 수 있고, 압축해제는 압축해제기(24)에서 행해질 수 있다. (표 1과 같은) 델타 모드표(22)가 마찬가지로 제공될 수 있다. 마지막으로, 랜덤 액세스 메모리(26)가 압축 및 압축해제 엔진에 제공될 수 있다.

[0053] 도 2를 참조하면, 고정 지점에 대한 모드 선택 시퀀스(30)가 하드웨어, 소프트웨어, 및/또는 펌웨어로 구현될 수 있다. 소프트웨어 및 펌웨어 실시예에서, 그것은 자기, 광 또는 반도체 메모리 등의 비 일시적 컴퓨터 판독 가능한 매체에 저장된 컴퓨터 실행 명령어에 의해 구현될 수 있다.

[0054] 고정 지점에 대한 모드 선택은, 블럭 32에 표시된 바와 같이, 델타 비트에 대한 표를 구성함으로써 시작된다. 그 후 고정 지점 위치가 블럭 34에서 주어진다. 마지막으로, 블럭 36에서 각 고정 지점에 대한 각 델타 벡터를 나타내기 위한 비트가 계산된다.

[0055] 동적 잔차값 비트 할당을 위한 시퀀스(38)가 소프트웨어, 펌웨어 및/또는 하드웨어로 구현될 수 있다. 소프트웨어 및 펌웨어 실시예에서, 그것은 광, 자기 또는 반도체 메모리 등의 비 일시적 컴퓨터 판독 가능한 매체에 저장된 컴퓨터 실행 명령어에 의해 구현될 수 있다.

[0056] 블럭 40에 표시되어 있는 바와 같이, 시퀀스는 예측 깊이를 실제 깊이와 비교함으로써 시작된다. 그 후 블럭 42에 도시된 바와 같이 잔차 값이 계산된다.

[0057] 그 후 잔차값 비트를 인코딩하는 최단 모드가 선택된다(블럭 44). 모드값은 잔차 모드 마스크에 채워진다(블럭 46). 마지막으로, 블럭 48에 표시된 바와 같이, 잔차 모드 마스크가 인코딩된 타일 데이터에 저장된다.

[0058] 도 4에 도시된 컴퓨터 시스템(130)은 버스(104)에 의해 칩셋 코어 로직(110)에 연결된, 하드 드라이브(134) 및 탈착 가능한 매체(136)를 포함할 수 있다. 컴퓨터 시스템은 스마트폰, 태블릿 또는 모바일 인터넷 장치 등의 스마트 모바일 장치를 포함하여 임의의 컴퓨터 시스템일 수 있다. 키보드 및 마우스(120) 또는 다른 종래의 구성요소는 버스(108)를 통해 칩셋 코어 로직에 연결될 수 있다. 코어 로직은 버스(105)를 통해 그래픽 프로세서(112)에 연결될 수 있고, 일 실시예에서는 중앙 프로세서(100)에도 연결될 수 있다. 그래픽 프로세서(112)는 또한 버스(106)에 의해 프레임 버퍼(114)에 연결될 수 있다. 프레임 버퍼(114)는 버스(107)에 의해 디스플레이 스크린(118)에 연결될 수 있다. 하나의 실시예에서, 그래픽 프로세서(112)는 SIMD(single instruction multiple data) 구조를 이용하는 멀티스레드, 멀티 코어 병렬 프로세서일 수 있다.

[0059] 소프트웨어 구현예인 경우, 적절한 코드가 메인 메모리(132)(139로 표기됨) 또는 그래픽 프로세서 내의 임의의 이용 가능한 메모리를 포함하여, 임의의 적당한 반도체, 자기, 또는 광 메모리에 저장될 수 있다. 따라서, 일 실시예에서, 도 2 및 3의 시퀀스를 수행하기 위한 코드는, 메모리(132) 및/또는 그래픽 프로세서(112) 및/또는 중앙 프로세서(100) 등의 비 일시적 기계 또는 컴퓨터 판독 가능한 매체(130)에 저장될 수 있고, 일 실시예에서 프로세서(100) 및/또는 그래픽 프로세서(112)에 의해 실행될 수 있다.

[0060] 도 2 및 도 3은 흐름도이다. 일부 실시예에서, 이들 흐름도에 묘사된 시퀀스는 하드웨어, 소프트웨어 또는 펌웨어로 구현될 수 있다. 소프트웨어 실시예에서, 반도체 메모리, 자기 메모리 또는 광 메모리 등의 비 일시적 컴퓨터 판독 가능한 매체는 명령어를 저장하도록 사용될 수 있고, 도 2 및 도 3에 도시된 시퀀스를 구현하기 위

해 프로세서에 의해 실행될 수 있다.

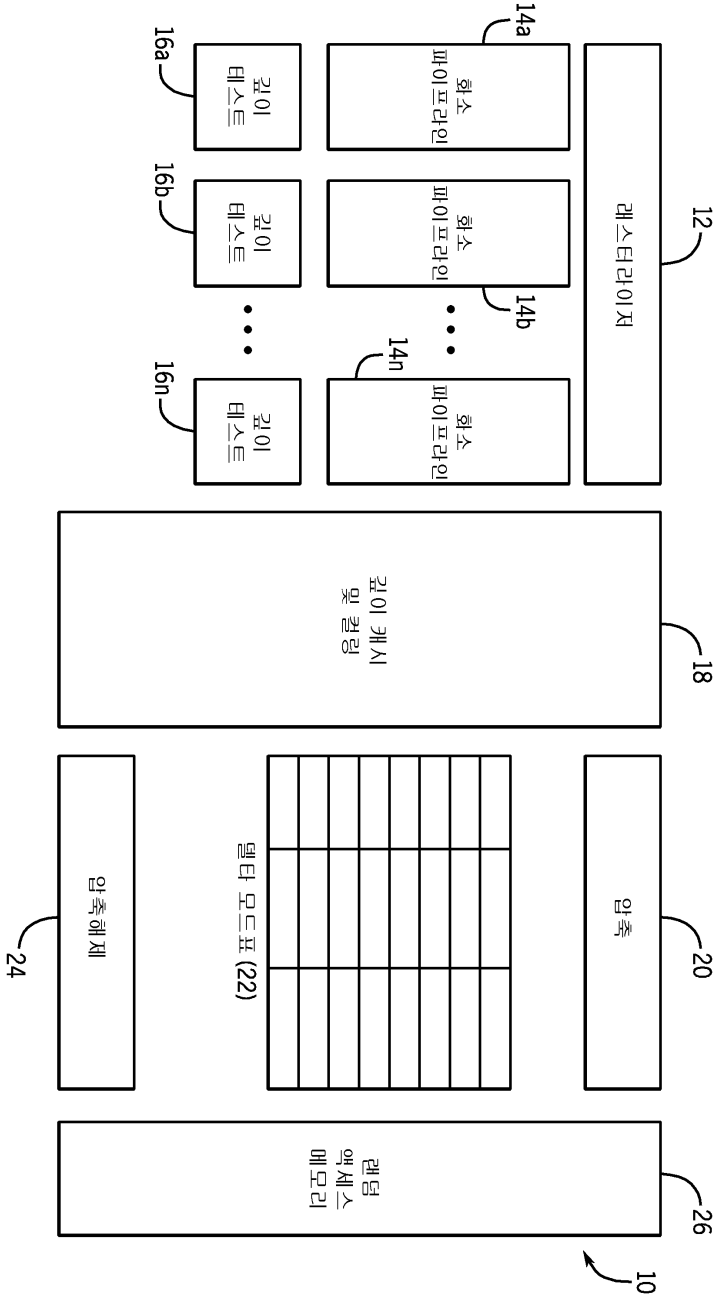
[0061] 여기에 기술된 그래픽 프로세싱 기술은 여러가지 하드웨어 구조로 구현될 수 있다. 예컨대, 그래픽 기능성은 칩셋 내에 통합될 수 있다. 이와 달리, 이산적 그래픽 프로세서가 사용될 수 있다. 또 다른 실시예로서, 그래픽 기능은 멀티코어 프로세서를 포함하여 범용 프로세서에 의해 구현될 수 있다.

[0062] 본 명세서 전반에 걸쳐 "하나의 실시예" 또는 "일 실시예"와 같은 언급은 그 실시예와 연결되어 기술된 특정 특징, 구조 또는 특성이 본 발명 내에 포함된 적어도 하나의 구현예에 포함되는 것을 의미한다. 따라서, "하나의 실시예" 또는 "일 실시예"의 구절의 출현은 반드시 동일한 실시예를 언급하는 것이 아니다. 더욱이, 특정 특징, 구조 또는 특성은 도시된 특정 실시예가 아닌 다른 적합한 형태로 도입될 수 있고, 그러한 모든 형태는 본 특허 청구범위 내에 포함될 수 있다.

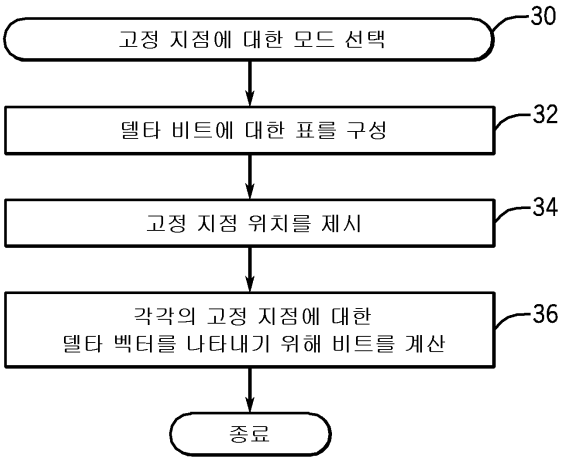
[0063] 본 발명은 제한된 수의 실시예에 대해 기술되었지만, 당업자는 그로부터 다수의 수정예 및 변형예를 인식할 것이다. 첨부되는 청구범위는 본 발명의 진정한 사상 및 범위 내에 포함되는 그러한 모든 수정예 및 변형예를 포함하는 것이 의도된다.

도면

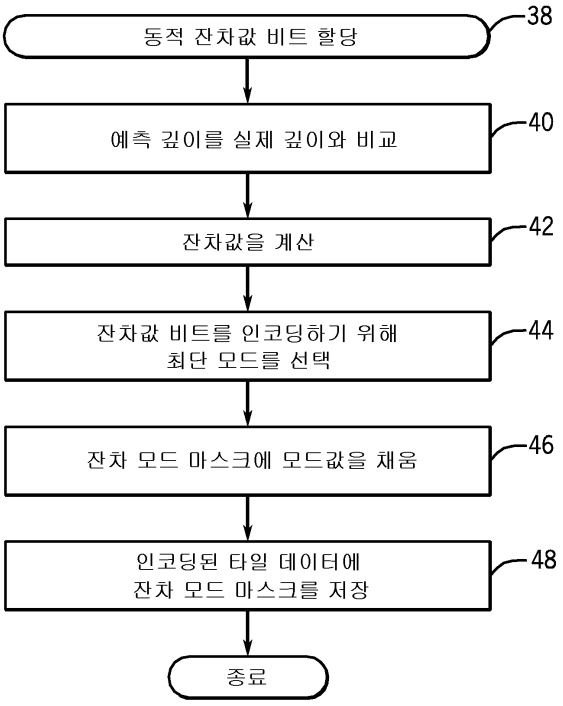
도면1


래스터라이저
12
14a
14b
14n
픽셀 파이프라인
16a
16b
16n
깊이 테스트
깊이 캐시 및 컬링
18
압축
20
멀티 모드 표 (22)
압축 해제
24
랜덤 액세스 메모리
26
10

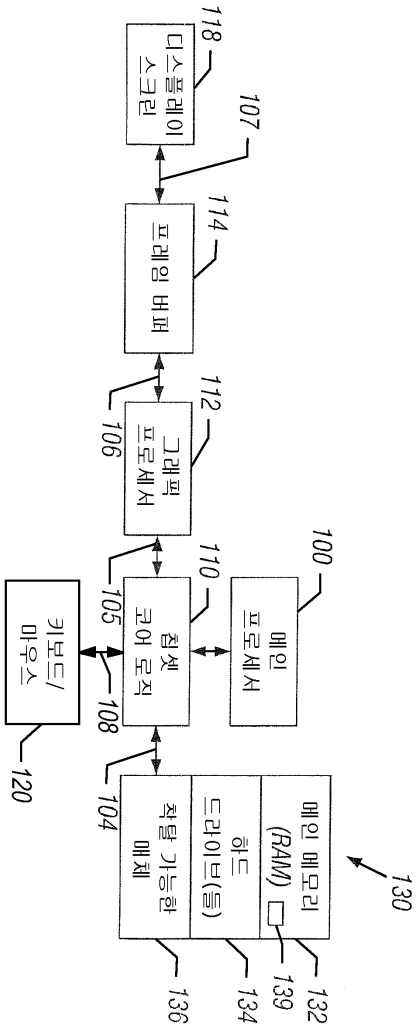
도면2


고정 지점에 대한 모드 선택
30
델타 비트에 대한 표를 구성
32
고정 지점 위치를 제시
34
각각의 고정 지점에 대한
델타 벡터를 나타내기 위해 비트를 계산
36
종료

도면3


동적 잔차값 비트 할당
38
예측 깊이를 실제 깊이와 비교
40
잔차값을 계산
42
잔차값 비트를 인코딩하기 위해
최단 모드를 선택
44
잔차 모드 마스크에 모드값을 채움
46
인코딩된 타일 데이터에
잔차 모드 마스크를 저장
48
종료

도면4


118
디스플레이 스크린
107
114
프레임 버퍼
112
106
그래픽 프로세서
110
105
칩셋 코어 로직
100
메인 프로세서
키보드/ 마우스
120
108
104
착탈 가능한 매체
하드 드라이브(들)
메인 메모리 (RAM)
130
136
134
139
132