

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2010-0020359 (43) 공개일자 2010년02월22일
(51) Int. Cl. <i>H04N 9/67</i> (2006.01) (21) 출원번호 10-2008-0079109 (22) 출원일자 2008년08월12일 심사청구일자 2008년08월12일	(71) 출원인 이화여자대학교 산학협력단 서울 서대문구 대현동 11-1 이화여자대학교내 (72) 발명자 이병욱 서울 마포구 상암동 월드컵 파크 아파트 404동 1202호 양세정 서울특별시 강남구 도곡동 타워팰리스 E동 1017호 (74) 대리인 특허법인태동	

전체 청구항 수 : 총 4 항

(54) 정준부호숫자를 이용한 가역 색상 변환 방법

(57) 요약

본 발명은 정준부호숫자(Canonical Signed Digit; CSD)를 이용한 가역 색상 변환 방법에 관한 것으로서, (1) 각각의 색좌표 계수를 CSD 수로 변환하는 단계; (2) 상기 단계 (1)에서 CSD 수로 변환된 계수에서 공통패턴을 추출하여 미리 계산하는 단계; 및 (3) 상기 단계 (2)에서 공통패턴에 대하여 미리 계산된 결과를 이용하여, 가역 색상 변환을 수행하는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

본 발명에서 제안하고 있는 가역 색상 변환 방법에 따르면, 정수화에 따른 반올림 오차가 없는 역변환 행렬을 정확하게 구현할 수 있는 역변환 가능 정수 색상 변환 방법에 대하여 정준부호숫자(Canonical Signed Digit; CSD) 시스템과 공통패턴 기법을 적용함으로써 역변환 가능 정수 색상 변환의 수행속도를 대폭 향상시킬 수 있다.

대표도 - 도1



특허청구의 범위

청구항 1

정준부호숫자(Canonical Signed Digit; CSD)를 이용한 가역 색상 변환 방법으로서,

- (1) 각각의 색좌표 계수를 CSD 수로 변환하는 단계;
- (2) 상기 단계 (1)에서 CSD 수로 변환된 계수에서 공통패턴을 추출하여 미리 계산하는 단계; 및
- (3) 상기 단계 (2)에서 공통패턴에 대하여 미리 계산된 결과를 이용하여, 가역 색상 변환을 수행하는 단계를 포함하는 것을 특징으로 하는, CSD를 이용한 가역 색상 변환 방법.

청구항 2

제1항에 있어서,

상기 단계 (1)에서, 상기 CSD 수는, 각각의 수가 -1, 0, +1인 숫자의 집합인 삼진수이며, 연속되는 어떤 CSD 자릿수도 두 수 모두 0이 아닌 경우는 없는 것을 특징으로 하는, CSD를 이용한 가역 색상 변환 방법.

청구항 3

제1항에 있어서,

상기 단계 (2)에서, CSD 수로 변환된 계수 전체에 대하여 공통패턴을 추출하는 것을 특징으로 하는, CSD를 이용한 가역 색상 변환 방법.

청구항 4

제1항에 있어서,

상기 단계 (3)에서, 상기 가역 색상 변환은, 역변환 가능 정수 색상 변환인 것을 특징으로 하는, CSD를 이용한 가역 색상 변환 방법.

명세서

발명의 상세한 설명

기술 분야

[0001] 본 발명은 가역 색상 변환 방법에 관한 것으로서, 정수화에 따른 반올림 오차가 없는 역변환 행렬을 정확하게 구현할 수 있는 역변환 가능 정수 색상 변환 방법에 대하여 정준부호숫자(Canonical Signed Digit; CSD) 시스템과 공통패턴 기법을 적용함으로써 역변환 가능 정수 색상 변환의 수행속도를 대폭 향상시킨 가역 색상 변환 방법에 관한 것이다.

배경 기술

[0002] 색상을 표준적으로 나타내기 위한 색좌표 시스템은 여러 가지 다양한 목적으로 칼라 이미지 프로세싱에서 사용된다. 색좌표 시스템으로는, RGB 시스템, CMY 시스템, HSI 시스템, YUV 시스템, YCbCr 시스템 등이 있는데, 색좌표 시스템 사이의 변환을 위해 사용되는 색상 변환(Color Transform)은 보통 3×3 행렬로 구성된다. 흔히 사용되는 고정소수점 프로세서를 사용하여 색상 변환을 구현할 경우, 각각의 색좌표 계수 값들의 근사치를 사용하게 되고, 이로 인해 라운딩 에러가 생기며 역변환 행렬을 정확하게 구현할 수 없게 된다는 문제점이 있다.

[0003] 고정소수점 프로세서의 단점을 극복하기 위한 연구가 많이 이루어져 왔고, 최근에 개발된 효과적인 방법으로 Pei 등에 의해 개발된 역변환 가능 정수 색상 변환이 있다. 역변환 가능 정수 색상 변환은, 정수화에 따른 반올림 오차가 없는 역변환 행렬을 정확하게 구현할 수 있다는 장점이 있지만, 그 수행속도를 보다 향상시킬 필요성이 있다.

발명의 내용

해결 하고자하는 과제

[0004] 본 발명은 기존에 제안된 방법들의 상기와 같은 문제점들을 해결하기 위해 제안된 것으로서, 정수화에 따른 반올림 오차가 없는 역변환 행렬을 정확하게 구현할 수 있는 역변환 가능 정수 색상 변환 방법에 대하여 정준부호숫자(Canonical Signed Digit; CSD) 시스템과 공통패턴 기법을 적용함으로써 역변환 가능 정수 색상 변환의 수행속도를 대폭 향상시킨 가역 색상 변환 방법을 제공하는 것을 그 목적으로 한다.

과제 해결수단

[0005] 상기한 목적을 달성하기 위한 본 발명의 특징에 따른, 정준부호숫자(Canonical Signed Digit; CSD)를 이용한 가역 색상 변환 방법은,

[0006] (1) 각각의 색좌표 계수를 CSD 수로 변환하는 단계;

[0007] (2) 상기 단계 (1)에서 CSD 수로 변환된 계수에서 공통패턴을 추출하여 미리 계산하는 단계; 및

[0008] (3) 상기 단계 (2)에서 공통패턴에 대하여 미리 계산된 결과를 이용하여, 가역 색상 변환을 수행하는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

[0009] 바람직하게는, 상기 단계 (1)에서, 상기 CSD 수는, 각각의 수가 -1, 0, +1인 숫자의 집합인 삼진수이며, 연속되는 어떤 CSD 자릿수도 두 수 모두 0이 아닌 경우는 없다.

[0010] 바람직하게는, 상기 단계 (2)에서, CSD 수로 변환된 계수 전체에 대하여 공통패턴을 추출한다.

[0011] 바람직하게는, 상기 단계 (3)에서, 상기 가역 색상 변환은, 역변환 가능 정수 색상 변환이다.

효 과

[0012] 본 발명에서 제안하고 있는 가역 색상 변환 방법에 따르면, 정수화에 따른 반올림 오차가 없는 역변환 행렬을 정확하게 구현할 수 있는 역변환 가능 정수 색상 변환 방법에 대하여 정준부호숫자(Canonical Signed Digit; CSD) 시스템과 공통패턴 기법을 적용함으로써 역변환 가능 정수 색상 변환의 수행속도를 대폭 향상시킬 수 있다.

발명의 실시를 위한 구체적인 내용

[0013] 이하에서는 첨부된 도면들을 참조하여, 본 발명에 따른 실시예에 대하여 상세하게 설명하기로 한다.

[0014] 도 1은 본 발명의 일 실시예에 따른 가역 색상 변환 방법의 구성을 나타내는 도면이다. 도 1에 도시된 바와 같이, 본 발명의 일 실시예에 따른 가역 색상 변환 방법은, CSD 변환 단계(S10), 공통패턴 기법 적용 단계(S20), 및 가역 색상 변환 수행 단계(S30)를 포함한다.

[0015] CSD 변환 단계(S10)에서는, 각각의 색좌표 계수를 CSD 수로 변환한다. 정준부호숫자(CSD) 시스템을 채택할 경우, 0이 아닌 수를 최소화시켜 하드웨어 곱셈기에서 수행되는 덧셈의 수를 줄여줄 수 있다. CSD 시스템의 인코딩 체계는 삼진수이고, 각 수가 -1, 0 또는 +1인 숫자 집합이다. 또한, 연속되는 어떤 CSD 자릿수도 두 수 모두 0이 아닌 경우는 없다. 즉, $c_i \times c_{i-1} = 0$ (c_i 는 i 번째 CSD 자릿수)으로 표현할 수 있다. 그러므로 어떤 임의의 N비트의 이진수를 CSD 수로 변환하면, $(N+1)/2$ 이하의 0이 아닌 수를 가지게 되며, 곱셈을 실행하는 데 있어서 그 수 이상의 덧셈이 필요하지 않게 된다. N비트의 CSD 수에서 0이 아닌 수에 대한 기대값은 점근적으로 $N/3 + 1/9$ 로 수렴한다. 평균적으로 CSD 수는 이진수에 비해 33% 적게 0이 아닌 수를 가진다. 표 1은 $N=4$ 인 이진수를 CSD 수로 변환하는 예이다. 여기서, *은 -1을 의미한다. 표 1을 통해, 이진수를 CSD 수로 변환할 경우, 음수들에서 0이 아닌 수의 숫자가 이진수보다 CSD 수에서 더 적다는 사실과 1의 개수가 총 32개에서 23개로 줄어든다는 사실을 분명하게 확인할 수 있다.

표 1

수	2의 보수	CSD
0.875	0111	100*
0.750	0110	10*0
0.625	0101	0101

0.500	0100	0100
0.375	0011	010*
0.250	0010	0010
0.125	0001	0001
0	0000	0000
-0.125	1111	000*
-0.250	1110	00*0
-0.375	1101	0*01
-0.500	1100	0*00
-0.625	1011	0*0*
-0.750	1010	*010
-0.875	1001	*001
-1	1000	*000

[0017] 공통패턴 기법 적용 단계(S20)에서는, 단계 S10에서 CSD 수로 변환된 계수에서 공통패턴을 추출하여 미리 계산한다. 본 발명에서는, CSD 수의 덧셈기와 뺄셈기를 줄이기 위해 공통패턴 기법을 채택한다. 공통패턴 기법을 적용함으로써, 즉 모든 CSD 계수에서 공통패턴을 추출해서 미리 계산해 둬으로써 덧셈과 뺄셈의 수를 줄일 수 있다. 여러 개의 곱셈기가 있는 하드웨어에서 공통패턴을 사용함으로써 총 계산량의 50%를 줄일 수 있다. 도 2는 RGB 색좌표에서 UVW 색좌표로의 변환 행렬에서 총 8개의 계수들 중 마지막 4개를 CSD 형으로 표현한 것을 나타내는 도면이다. 색상 변환에서 각각의 곱셈은 $g_i \cdot x$ 로 쓸 수 있는데, 여기서 g_i 는 한 열의 매트릭스 계수이며($i = 1, \dots, 8$), x 는 이진 입력으로서 1 또는 0의 값을 갖는다. 각 계수들이 CSD 형으로 $g_i = g_{i0}g_{i1}g_{iM-1}$ (M: 전체 비트 길이)과 같이 표현된다고 하면, 그 곱셈식은 다음 수학식 1과 같다.

수학식 1

$$\sum_{j=0}^{M-1} g_{ij} \cdot (x \gg j)$$

[0018]

[0019] 도 2에서, 각각의 계수에서 같은 원은 같은 공통패턴을 가진다. 도 2에 표시된 g_5 내지 g_8 에 대한 산술 연산은 각각 다음 수학식 2와 같다.

수학식 2

[0020]

$$g_5 \cdot x = -x + x \ll 2$$

[0021]

$$g_6 \cdot x = -x \ll 3 + x \ll 5$$

[0022]

$$g_7 \cdot x = x + x \ll 3 + x \ll 6$$

[0023]

$$g_8 \cdot x = x + x \ll 2 + x \ll 5$$

[0024] 여기서, $x \ll a$ 는 x 를 a 번 왼쪽으로 이동함(좌측 시프트)을 의미한다. 상기 4개의 계수를 위한 공통패턴은 10*(실선)과 100*(점선)으로서, 다음 수학식 3과 같이 정의할 수 있다.

수학식 3

[0025]

$$w_1 = -x + x \ll 2$$

[0026]

$$w_2 = -x + x \ll 3$$

[0027]

수학식 3을 이용하면, 수학식 2를 다음 수학식 4와 같이 표현할 수 있다.

수학식 4

[0028]

$$g_5 \cdot x = w_1$$

[0029] $g_6 \cdot x = w_1 < 3$

[0030] $g_7 \cdot x = x + w_2 < 3$

[0031] $g_8 \cdot x = x + w_2 < 2$

[0032] 상기 수학식 4를 상기 수학식 2와 비교해 보면, 공통패턴 w_i 에 의해 덧셈의 수가 9개에서 5개로 줄어드는 것을 확인할 수 있다.

[0033] 도 3은 8개의 변환 계수를 모두 사용하여 공통패턴을 표시한 것을 나타내는 도면이다. 도 3에서 확인할 수 있는 바와 같이, 전체 덧셈의 수가 이진수로 표현할 경우 33개인데 비하여, CSD형으로 변환하고 공통패턴을 공유함으로써 16개로 줄어든다. 도 3에서와 같이 8개의 계수를 모두 사용할 경우, 도 2에 도시된 것처럼 4개의 계수만을 사용했을 때보다 더 큰 감소율을 보인다. 이를 통해 공통패턴 방식은 더 많은 계수를 가질수록 그리고 더 많은 곱셈을 수행할수록 더 효과적임을 확인할 수 있다.

[0034] 도 4는 여러 가지 종류의 색상 변환 계수들에 대하여 공통패턴을 표현한 것을 나타내는 도면이다. 도 4에 도시된 바와 같이, RGB 색좌표를 KLA 색좌표로, RGB 색좌표를 IV_1V_2 색좌표로, RGB 색좌표를 $YC_B C_R$ 색좌표로, RGB 색좌표를 DCT 색좌표로, RGB 색좌표를 YIQ 색좌표로, RGB 색좌표를 XYZ 색좌표로, RGB 색좌표를 UVW 색좌표로 변환하는 경우의 변환 계수들에 대하여 공통패턴 기법을 적용할 수 있다.

[0035] 가역 색상 변환 수행 단계(S30)에서는, 단계 S20에서 공통패턴에 대하여 미리 계산된 결과를 이용하여, 가역 색상 변환을 수행한다. 앞서 설명한 바와 같이, 본 발명에서 수행하고자 하는 가역 색상 변환은, 역변환 가능 정수 색상 변환이다.

[0036] 다음 표 2는 각 가역 색상 변환에 대한 계산량을 비교해 놓은 것이다. 표 2를 통해, 기존 가역 색상 변환 방법(2의 보수를 이용하는 방법)에 비하여, 본 발명에서 제안하고 있는 가역 색상 변환 방법(CSD 수 및 공통패턴을 이용하는 방법)이 약 40 ~ 65%까지 덧셈기의 수를 줄여주는 것을 분명하게 확인할 수 있다.

표 2

	RGB → KLA	RGB → IV_1V_2	RGB → $YC_B C_R$	RGB → DCT	RGB → YIQ	RGB → XYZ	RGB → UVW
2의 보수에서의 덧셈기의 수	50	42	31	34	34	33	33
CSD에서의 덧셈기의 수	18	19	17	20	18	20	16
절약율	64.0%	54.8%	45.2%	41.2%	47.1%	39.4%	51.5%

[0038] 이상 설명한 본 발명은 본 발명이 속한 기술분야에서 통상의 지식을 가진 자에 의하여 다양한 변형이나 응용이 가능하며, 본 발명에 따른 기술적 사상의 범위는 아래의 특허청구범위에 의하여 정해져야 할 것이다.

도면의 간단한 설명

[0039] 도 1은 본 발명의 일 실시예에 따른 가역 색상 변환 방법의 구성을 나타내는 도면.

[0040] 도 2는 RGB 색좌표에서 UVW 색좌표로의 변환 행렬에서 총 8개의 계수들 중 마지막 4개를 CSD 형으로 표현한 것을 나타내는 도면.

[0041] 도 3은 8개의 변환 계수를 모두 사용하여 공통패턴을 표시한 것을 나타내는 도면.

[0042] 도 4는 여러 가지 종류의 색상 변환 계수들에 대하여 공통패턴을 표현한 것을 나타내는 도면.

[0043] <도면 중 주요 부분에 대한 부호의 설명>

[0044] S10: CSD 변환 단계

[0045] S20: 공통패턴 기법 적용 단계

[0046] S30: 가역 색상 변환 수행 단계

도면

도면1



도면2

	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
g_5	0	0	0	0	0	1	0	*
g_6	0	0	1	0	*	0	0	0
g_7	0	1	0	0	*	0	0	1
g_8	0	0	1	0	0	*	0	1

도면3

2's complement	# of 1's	CSD	# of 1's	CSD	# of 1's
11011010	5	00*0*010	3	00*0*010	3
01111111	7	1000000*	2	1000000*	2
01000110	3	010010*0	3	010010*0	2
11100111	6	00*0100*	3	00*0100*	3
00000011	2	0000010*	2	0000010*	1
00110000	2	010*0000	2	010*0000	1
00111001	4	0100*001	3	0100*001	2
00011101	4	00100*01	3	00100*01	2
total	33	total	21	total	16

도면4

	RGB to KLA	RGB to IV_1V_2	RGB to $YC_B C_R$	RGB to DCT	RGB to YIQ	RGB to XYZ	RGB to UVW
g_1	00*0]0*0*	0100*0[0*	00[00]000	0[0*]01010	000[000]0	01010010	00*0*0]0
g_2	101001000	00[0*]000*	001010101	001010001	0100*0001	0[*0]0100*	1000000*
g_3	00*0]0*0*	00*0*0*0*	000[0*]000	00100*001	00*0*000*	[0*]0[0*]01	0100[0*]0
g_4	[00*]00*0]	0*0[0*]000	*0]010[0*	*0]010*0*	0*0010010	00100100*	00*0[00*]
g_5	*00[01010	100000000	0*000*00*	0*0[0*]000	*0]0*0[*0]	0[0*]0000*	00000[0*]
g_6	100000101	0[00]0*0]	000*00001	00*000[0*]	0100*0010	000*00101	0[0*]0000
g_7	000*00*0*	000100*0]	00[0*]0010	0[0*]01010	0[0*]0[0*]0	*0*0*0*0*	0[00*]001
g_8	00000000*	0*00*0101	000*0*001	001010001	000*00001	[00*]00100	00[00*]01