CMOS 영상센서에 대한 영상 신호 전처리기의 구현

정 영식, 장 영조
한국기술교육대학교 정보기술공학부

Jeung Young Sic and Jang Young Jo
Korea University of Technology and Education
sikiga@hanmail.net and yjjang@kut.ac.kr

ABSTRACT

Recently, CMOS image sensor is rapidly used as an image capture device such as mobile phone or notebook PC. Because of poor quality of image by CMOS image sensor, ISP is essential step to improve image quality. In this paper, we implemented and simulated ISP algorithm for real time moving picture of CMOS image sensor. Especially, we concentrated on color interpolation, which extracts three color component from uncompleted color information. Several algorithms for color interpolation are implemented and analyzed to acquire a good quality of picture. Finally, we proposed an improved algorithm and confirmed the effectiveness by experimental simulation results.

Ⅰ. 서 론

단순한 문자나 음성의 통신에서 영상을 동반하는 통신이 일반화되고 있다. 개인 컴퓨터나 휴대폰 등에서 영상의 입력 장치로 비용, 소비 전력 등의 측면에서 CCD 영상 센서보다 유리한 CMOS 영상 센서를 사용한 카메라가 급속히 확산되고 있다. 그러나 CMOS 센서는 CCD 센서에 비해 화질이 좋지 않으므로 이를 개선해주는 전처리 단계인 ISP (Image Signal Processor)가 중요한 역할을 한다.

ISP는 크게 입력영상의 불충분한 영상 데이터를 기반으로 온전한 영상 데이터를 생성하는 Color Interpolation, 입력된 영상의 3가지색(빨강:R, 파랑:B, 초록:G)을 자연의 색에 맞게 변화시키는 White Balance, 각 픽셀들의 값이 R, G, B로 구성되어 있는 것을 Y, Cb, Cr로 바꾸는 Space Conversion, 입력된 영상과 디스플레이의 차이를 보정하기 위한 감마 보정 (gamma correction), 입력된 영상의 잡음을 제거하고 영상을 선명하게 하기위한 필터처리 등이 있다. 여기에서 Color Interpolation은 CMOS 영상 센서에서 출력되는 영상 데이터를 <그림1>과 같은 Bayer pattern 형태의 불충분한 값을 토대로 하여 원 영상과 비슷한 영상으로 나타내기위해 수행하는 과정이다. 즉, CMOS 영상 센서의 값을 가지고 초기의 영상을 획득하는 과정으로 다른 과정들에게 미치는 영향이 매우 크다. 본 논문에서는 CMOS 영상 센서 ISP를 위한 여러 가지 처리과정 중에서 Color Interpolation을 수행하기 위한 몇 가지 방법을 제시하고 비교하여 각각의 장·단점을 파악하고 가장 적합한 알고리즘을 제시하고 그 유효성을 실험으로 확인하고자 한다.

R	G	R	G
G	B	G	B
R	G	R	G
G	B	G	B

그림 1. CMOS 영상 센서의 출력

II. 본 론

2.1 2*2 매트릭스 알고리즘

이 알고리즘은 가장 기본적인 것으로 <그림2>와 같이 4개의 이웃하는 입력 데이터를 받아 그에 상응하는 픽셀들을 완전한 영상 데이터(R,G,B)로 계산한다. 블록에는 R과 B가 하나씩, G값은 2개가 포함된다. (하나의 매트릭스단위를 블록이라 하겠다.) 그래서 식 (1)과 식(2)에 의해서 한 블록의 모든 R 또는 B의 값은 동일하며, 식(3)으로 인해 G는 최대 3개의 각기 다른 값을 지니게 된다.

R	G	p1	p2	R	G
G	B	p3	p4	G	B
R	G	R	G	R	G

그림 2. 2*2 매트릭스 알고리즘

R11=R12=R21=R22=p1 **(1)**
B11=B12=B21=B22=p4 **(2)**
G11=G22=(p2+p3)/2 **(3)**

2.2 3*3 매트릭스 알고리즘

2*2 매트릭스 알고리즘에서 R과 B의 색을 보완하기 위한 방법으로 <그림3>과 같이 9개의 이웃하는 입력 데이터를 이용하여 그에 상응하는 영상 데이터를 계산한다. 그래서 블록에 포함되는 R과 B의 수가 증가하지만 홀수 번째의 블록과 짝수 번째의 블록들 사이에는 유효값의 수가 다르게 포함되어 계산식도 각각 다르게 표현된다. 픽셀 데이터를 구하는 수식 중 R의 경우를 들면,

R	G	R	p1	p2	p3
G	B	G	p4	p5	p6
R	G	R	p7	p8	p9

(홀수 번째 블록)

G	R	G	p1	p2	p3
B	G	B	p4	p5	p6
G	R	G	p7	p8	p9

(짝수 번째 블록)

그림 3. 3*3 매트릭스 알고리즘

- 홀수 번째 블록의 계산식 -

R11=p1 (4)
R12=(p1+p3)/2 (5)
R22=(p1+p3+p7+p9)/4 (6)

2.3 3*6 매트릭스 알고리즘

<그림4>와 같이 18개의 이웃하는 입력 데이터를 이용하여 그에 해당되는 픽셀 데이터를 계산하는 알고리즘으로 3*3 매트릭스를 이용한 것보다 색상을 원 영상에 가깝게 표현하게 된다. 또한 모든 색에 대하여 모든 블록의 유효값들의 수가 일정하다. 하지만 픽셀들의 색을 계산하는 수식이 복잡하다. 아래의 수식은 R의 경우를 예로 들었다.

R	G	R	G	R	G	p1	p2	p3	p4	p5	p6
G	B	G	B	G	B	p7	p8	p9	p10	p11	p12
R	G	R	G	R	G	p13	p14	p15	p16	p17	p18

그림 4. 3*6 매트릭스 알고리즘

R12=(p1+p3)/2 (7)
R22=(p1+p3+p13+p15)/4 (8)

2.4 개선된 3*3 매트릭스 알고리즘

기존의 3*3매트릭스를 이용한 알고리즘과 비슷하지만 <그림5>와 같이 3번째 열의 픽셀들은 다음 매트릭스의 1번째 열의 픽셀이 되어 계산이 된다. 또한, 세로로 이동할 때도 비슷하다. 그래서 4개의 유효한 픽셀 데이터를 구하게 된다. 수식 중 R의 경우를 들면

G		G		G
	G		G	
G		G		G
	G		G	
G		G		G

(G의 경우)

R		R		R
R		R		R
R		R		R

(R의 경우)

그림 4. 3*6 매트릭스 알고리즘

R12=(p1+p3)/2 (9)
R21=(p1+p7)/2 (10)
R22=(p1+p3+p7+p9)/4 (11)

2.5 여러 Color Interpolation 알고리즘의 분석

2*2 매트릭스를 이용한 알고리즘의 최대 장점

은 구현 알고리즘이 간단하고, 연산실행에 있어서 블록당 계산량이 적으며 하드웨어로 구현시 메모리를 적게 사용한다. 하지만 색상을 구현함에 있어서 4개의 이웃하는 입력 데이터에 R과 B의 유효 값이 한 픽셀뿐이라서 이들의 색상을 표현하는 것에는 한계를 지닌다. 그래서 보다 개선된 영상을 얻기 위해서는 R, B를 다시 한번 조정해주는 과정이 포함되어야 한다. 3*3 매트릭스를 이용한 알고리즘은 위의 단점이었던 R과 B의 색상표현을 개선하였다. 하지만 홀수의 블록들이 4개의 유효 값을 포함한데에 반해 짝수의 블록들은 2개의 유효 값만을 포함 하여 색상을 표현하는 데에 한계를 여전히 간직하고 있다. 3*6 매트릭스를 이용한 알고리즘은 위의 두 매트릭스의 단점을 보완하였다. 즉, 18개의 이웃하는 데이터를 이용하여 R과 B의 색상 표현에 결함을 최대한으로 줄였으며 연산 실행 속도를 개선하였다. 그러나 작업을 실행할 때 계산량이 많고, 하드웨어로 구현 시 메모리를 많이 차지하는 단점을 지니고 있다. 위의 단점들을 개선하여 제시하는 알고리즘이 개선된 3*3 매트릭스를 이용하는 것이다. 앞서 <그림5>에서 보았듯이 R과 B의 색상을 최대한으로 표현하기 위해서 각각의 매트릭스들의 시작 포인트를 다르게 하였고, 픽셀 계산부의 단순화를 위해 2픽셀 간격으로 매트릭스가 이동한다. 이로 인해서 연산의 실행속도는 2*2 매트릭스와 같게 되지만 블록에 포함되는 유효 픽셀값들의 수가 항상 일정하고 3*6 매트릭스를 이용한 것과 거의 유사한 결과를 얻을 수 있다. 또한 단순한 계산부로 인해 하드웨어로 구현하기 쉽고 메모리 또한, 3*6 매트릭스를 이용한 것보다 적게 차지한다.

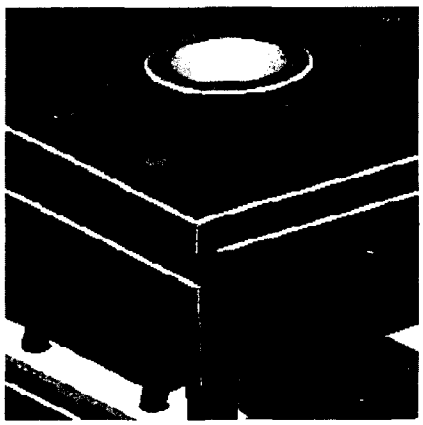
III. 결 론

본 논문에서는 CMOS 영상 센서의 전처리를 위한 ISP의 Color Interpolation을 수행하는 몇가지 알고리즘에 대해서 다루었다. 이 과정은 ISP 과정 중 픽셀의 값을 표현하는 각각의 색상을 결정하는 것으로 출력영상의 기틀을 마련한다. 따라서 색상을 표현함에 있어서 많은 데이터를 참고해야 되지만, 하드웨어의 설계를 기반으로 하기 때문에 구현의 용이성을 위해 계산식의 단순화, 사용되는 메모리의 크기, 연산수행 속도 등을 고려해보지 않을 수가 없다. <표 1>을 보면 블록 당 계산량은 2*2 매트릭스, 전체 블록수를 보면 3*6 매트릭스의 성능이 가장 우수하다. 그리고 <그림6>을 보면 모든 알고리즘이 불완전한 영상 데이터를 입력받아 원영상과 비슷하게 복원하는 것을 볼 수 있다. 그러나 경계부분에서는 색상의 깨짐 현상이 나타나서 영상의 차이를 확실하게 알 수 있다. 3*6 매트릭스와 개선된 3*3 매트릭스는 비슷하지만 2*2 매트릭스는 색상의 깨짐 현상이 현저하게 증가했음을 확인할 수 있다. 3*6 매트릭스는 블록의 계산식이 복잡하여 연산부가 복잡하고 연산과정에서 사용되는 메모리가 많이 필요하게 되어 하드웨어로 구현하기가 쉽지 않다. 그러나 개선된 3*3 매트릭스는 계산식이 간단하여 연산부가 간단하고 메모리 할당량도 적어서 하드웨어로 구현하기 쉽다. 여러 알고리즘 중에서 계산식의 단순화, 화질의 정도 등에서 개선된 3*3 매트릭스를 이용한 알고리즘의 성능이 뛰어난 것을 볼 수 있다. 그러므로 제어부를 설계할 때 다소 어려움이 있더라도 ISP의 하드웨어를 설계하는 데에는 개선된 3*3 매트릭스를 이용한 알고리즘을 적용시키는 것이 가장 올바른 판단이라고 제안한다.

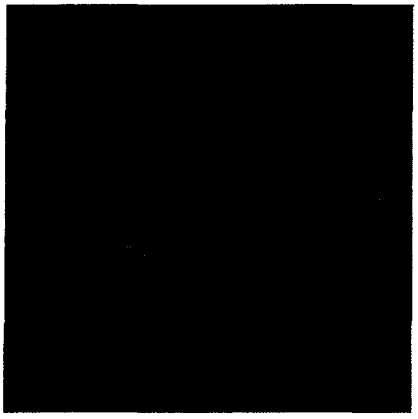
표.1 여러 알고리즘의 비교

	알고리즘			
	2*2	3*3	3*6	개선된 3*3
블록당 계산량	1	12	46	14
전체 블럭수	25344	11328	5664	25344
화질	중	중	상	상

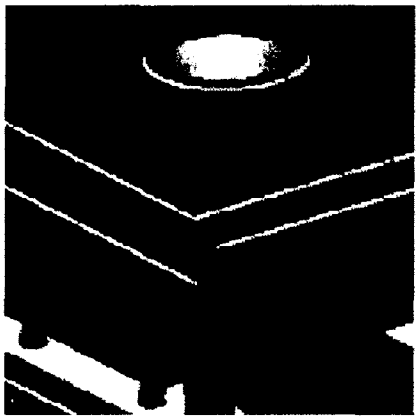
* 한 프레임은 352*288을 기준으로 함
* 덧셈연산을 기준으로 블록 당 계산량을 산출함



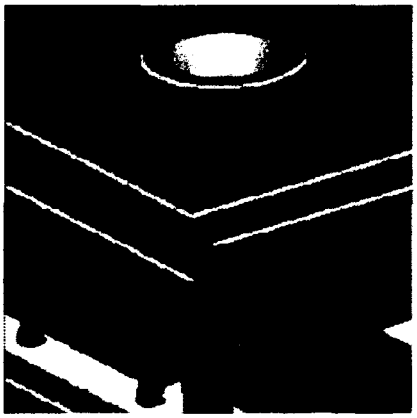
a. 원영상



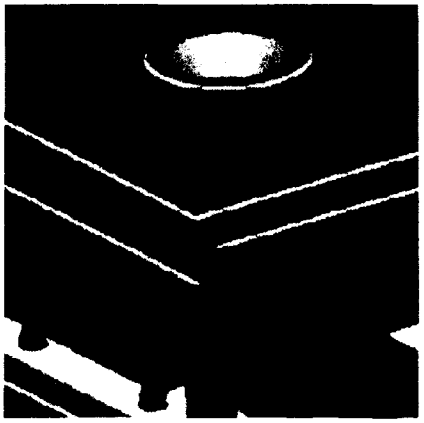
b. CMOS 영상 센서의 출력



c. 2*2 매트릭스 알고리즘



d. 3*6 매트릭스 알고리즘



e. 개선된 3*3 매트릭스 알고리즘

그림.6 Color Interpolation의 테스트 결과

Ⅳ. 참고문헌

[1] 영상처리 이론과 실제 - 홍릉과학출판사, 1999.8.20

[2] 영상처리 기초편 - 기한재출판사

[3] Visual C++를 이용한 디지털 영상처리의 구현 Disital Image Processing - PC 어드밴스