Image data processing 소프트웨어를 이용한 CMOS image sensor device 테스트 시스템 구현

김성진
(주)테스티안
e-mail:sjkim@testian.com

A CMOS Image Sensor Device Test System with Image Data Processing Software

Kim Seongjin
(Ltd)TESTIAN

요 약

CMOS 이미지 센서는 모바일 디바이스, 특히 스마트 폰에 내장된 카메라에 가장 광범위하게 사용된다. 이러한 이미지 센서의 정상 동작을 검사하기 위해서는 불량화소 검출과 같은 테스트가 수행되어야 하며, 테스트를 위해서는 센서에 의해서 캡처된 이미지를 대상으로 이미지 처리를 할 수 있는 함수 제공이 필수적이다. 이 논문에서는 CMOS 이미지 센서의 동작을 효율적이고 엄격하게 판단할 수 있는 자동 검사 시스템을 구축하고 이미지 센서로부터 캡처되는 이미지 데이터에 대해서 목적에 맞는 테스트를 수행 할 수 있도록 이미지 처리 함수를 구현하고 실험하였다.

1. 서론

이미지 센서는 투입되는 빛을 전기적인 신호로 변환하는 반도체로서 카메라, 방송장비, CCTV와 같은 분야에서 많이 적용된다. 특히 스마트 폰, 태블릿과 같은 모바일 디바이스에 내장되는 카메라 수요가 급증함에 따라 저전력 및 고집적화가 가능하고 가격이 저렴한 CMOS 이미지 센서[1]의 사용량이 증가하였다. 이러한 CMOS 이미지 센서를 엄격하게 검사하기 위해서는 5단계의 테스트가 반드시 필요하다[2].

첫 단계로 핀 연속성 테스트(Pin continuity test)는 검사 상에 있는 디바이스 핀들에 대해서 Open/Short 및 Leakage 검사를 진행한다. 두 번째 단계에서는 디바이스에 테스트를 위한 입력 신호를 인가해야 한다. 여기서 말하는 입력 신호는 디바이스를 켜고 동작시키기 위해서 프로그램화 한 연속되는 신호 순서를 디바이스 핀으로 인가한다는 것을 뜻하며, 입력 신호와 더불어 CMOS 이미지 센서 디바이스가 빛의 투입량을 가질 수 있도록 광원이 요구되기도 한다. 세 번째 단계는 CMOS 이미지 센서 디바이스에서 생성한 이미지 데이터를 병렬 핀 인터페이스(Parallel pin interface) 혹은 MIPI(Mobile Industry Processor Interface)[3] 표준과 같은 고속 직렬 핀 인터페이스(High-speed serial pin interface)를 통해서 검사 장비에서 캡처하고 메모리와 같은 저장매체에 저장한다. 네 번째 단계에서는 검사 장비에 저장된 이미지 데이터를 고성능 워크스테이션과 같은 이미지 처리 장비에 병행 전송(Concurrent transfer)한다. 병행 전송은 CMOS 이미지 센서 디바이스에 입력 신호가 인가되거나 검사 장비에 이미지 데이터가 캡처되는 동안에도 데이터 전송이 가능한 것을 의미하며, 디바이스를 테스트하는 시간을 효율적으로 관리하고 검사 장비의 작업량을 높이기 위해서 반드시 필요한 사항이다. 마지막 단계에서는 CMOS 이미지 센서의 정상유무를 결정하기 위한 불량화소 검출과 같은 테스트를 수행하기 위해서 이미지 처리 장비의 소프트웨어를 이용하여 전송된 이미지 데이터를 처리한다[4-5].

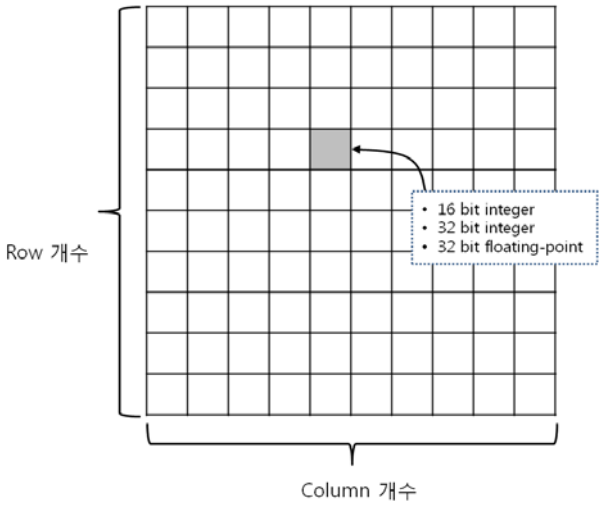
이 논문에서는 CMOS 이미지 센서 디바이스를 테스트하기 위한 자동 검사 시스템을 구성하고, 디바이스의 정상유무를 판단하기 위해서 캡처된 이미지를 테스트하기 위한 이미지 처리 함수를 구현하고 실험하였다. 이 논문의 구성은 제 2장에서 이미지 데이터 처리를 위해서 필요한 기본적인 개념을 정의한다. 3장에서는 CMOS image sensor가 캡처한 이미지 데이터를 목적에 맞게 테스트 할 수 있는 이미지 처리 함수에 대한 설명과 구현된 함수의 동작 여부와 처리 시간을 측정하였다. 마지막으로 4장에서는 이 연구 결과의 의미와 향후 연구 방안에 대해서 기술한다.

2. 이미지 데이터 처리

일반적으로 검사 상에 있는 CMOS 이미지 센서 디바이스가 캡처한 이미지 데이터가 메모리에 저장된 형태를 Image Plane이라고 하며, 본 논문에서 구현하고 실험한 모든 이미지 처리 함수들은 Image Plane의 데이터를 접근하고 제어할 수 있다.

2.1 Image Plane

Image Plane은 [그림 1]과 같이 2차원 배열로 표현되며 Pixel data의 bit width와 Plane의 크기와 같은 정보를 저장하고 있다. Image Plane의 한 개 Pixel은 보통 16bits 혹은 32bits 정수(integer) 값을 가지고 있으며, 부동 소수점(floating point) 값을 가지는 Image Plane을 정의 할 수도 있다. 이는 기본적으로 이미지 데이터는 정수형이지만 정확성을 높이기 위해 부동 소수점 계산을 해야 할 경우, 부동 소수점 데이터로 구성된 Image Plane을 사용할 수 있도록 한다.



(그림 1) Image Plane

2.2 Image Data Color

Image Plane은 Image Data Color의 집합체로 구성되어 있으며, 이미지를 처리할 때 마다 어떤 Color의 Pixel을 처리할 지 정의함으로써 Pixel 계산을 제한할 수 있다. [표 1]은 계산할 수 있는 14가지 타입의 Color를 보여준다.

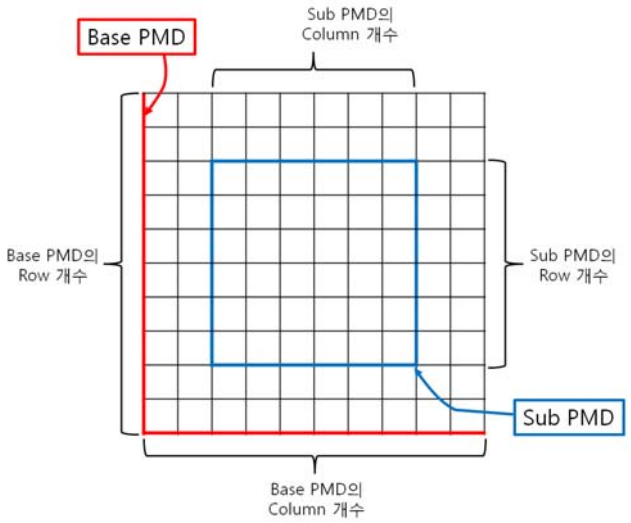
<표 1> Color Specification Code

표기	Internal Code	Color 처리 방법
ColorFlat	-2	흑백처리
ColorAll	-1	모드 Color 처리
ColorRed	0	Red만 처리
ColorGreen	1	Green만 처리
ColorBlue	2	Blue만 처리
ColorYellow	3	Yellow만 처리
ColorMagenta	4	Magenta만 처리
ColorCyan	5	Cyan만 처리
ColorRed2	6	Red2만 처리
ColorGreen2	7	Green2만 처리
ColorBlue2	8	Blue2만 처리
ColorYellow2	9	Yellow2만 처리
ColorMagenta2	10	Magenta2만 처리
ColorCyan2	11	Cyan2만 처리

Internal Code 0~5와 6~11은 각각 같은 Color를 의미하지만 같은 Color를 다른 Color처럼 구분하여 사용하고 싶을 경우를 대비하여 2가지 방법을 제공한다.

2.3 Base PMD와 Sub PMD

PMD(Pixel Map Definition)는 Image Plane에 저장되어 있는 이미지 데이터 가운데 얼마만큼을 접근할 것인지 정의하며, [그림 2]와 같이 크게 두 가지 타입으로 나눠진다.



(그림 2) BasePMD와 SubPMD

그림에서 BasePMD는 Image Plane 전체에 대한 Pixel Map을 의미하며, SubPMD는 Image Plane 일부에 대한 Pixel Map을 의미한다.

2.4 Color Map

Color Map은 Image Plane의 Color filter가 어떻게 나열되어 있는지를 정의한다. [그림 3]은 Color filter arrangement의 예를 보여준다.

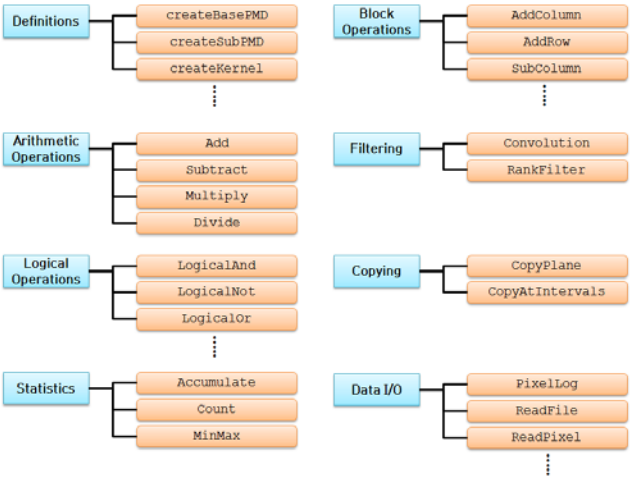
(그림 3) Color Filter Arrangement

[그림 3]에서는 최상단 좌측부터 (G, B)와 (R, G)가 2 X 2 형태로 반복 되는 것을 알 수 있으며, 이렇게 반복되는 형태를 Color Map Array라고 한다. 또한 [표 1]의 Color Specification Code를 적용하면 G는 1, B는 2, R은 0이며 (1, 2)와 (0, 1)의 형태로 표현될 수 있다.

3. 이미지 처리 함수 구현 및 실험

3.1 이미지 처리 함수

CMOS 이미지 센서로부터 캡처된 이미지를 목적에 맞게 여러 형태로 테스트하기 위해서는 Image Plane에 접근하여 이미지를 처리할 수 있는 함수가 필요하다. [그림 4]는 이미지 처리를 위해서 구현된 함수의 종류를 보여준다.



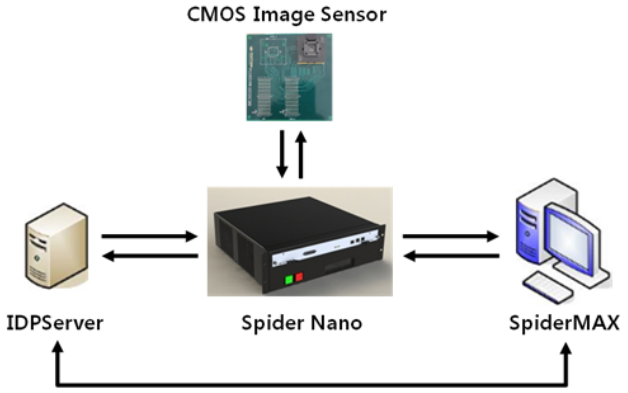
(그림 4) 이미지 처리 함수

이미지 처리를 위한 함수는 PMD 정의, 사칙연산, 논리연산, 통계, 블록연산, 필터링, 복사, DATA 입출력으로 나눠지며, 각 항목의 목적에 맞는 총 50개의 함수를 구현하였다. 그림에서 필터링의 경우 이미지 처리를 하는데 있어서 가장 많이 사용되는 항목으로써 처리하고자 하는 Pexel과 이웃한 Pixel들을 대상으로 가중치를 부여하고 처리하려는 Pixel이 새로운 값을 가질 때까지 값을 계속 더해가는 과정인 Convolution 함수와 비선형 필터링을 할 수 있는 RankFilter 함수를 포함한다.

3.2 전체 시스템

구현된 이미지 처리 함수를 검증하기 위해서 [그림 5]와 같이 자동 검사 시스템을 구성하였다. 그림에서 Spider Nano[6]는 Desktop 계측기형 ATE(Automated Test Equipment) 장비로서 CMOS 이미지 센서로부터 전송되는 이미지 데이터를 효율적으로 저장하고 이미지 처리를 위한 워크스테이션으로 데이터를 전송해 주는 역할을 한다. IDPServer는 이미지 데이터를 입력으로 받아 목적에 맞는 테스트를 할 수 있는 이미지 처리 워크스테이션 역할을 하며, 본 논문에서 구현한 이미지 처리 함수를 구동한다. SpiderMAX는 User PC에서 동작하는 소프트웨어로서 Spider Nano와 IDPServer를 운영하며, 테스트를 시작하고 이미지 처리 결과를 확인할 수 있다.

전체 시스템의 동작은 우선 CMOS 이미지 센서를 소켓 보드에 장착한 후 Spider Nano에 연결한다. SpiderMAX에서는 디바이스에 신호 인가를 위해서 패턴을 로딩시킨 후 IDPServer와의 연결을 시도한다. 연결 후에는 Spider Nano에서 이미지를 캡처하기 위한 테스트를 시작한다.

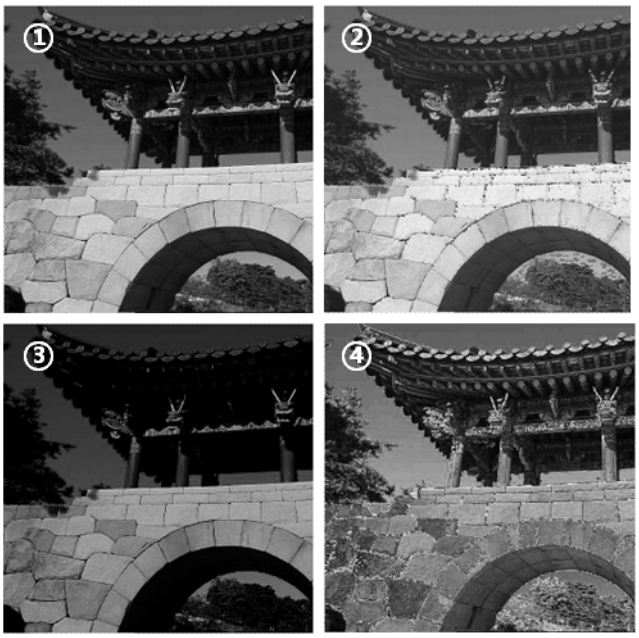


(그림 5) 전체 시스템 구성

캡처된 이미지는 Spider Nano를 통해서 IDPServer로 전송되고, IDPServer에서는 전송된 이미지 데이터를 테스트에 맞게 처리를 한다. 마지막으로 처리된 이미지 데이터를 SpiderMAX로 전송하여 이미지를 확인한다.

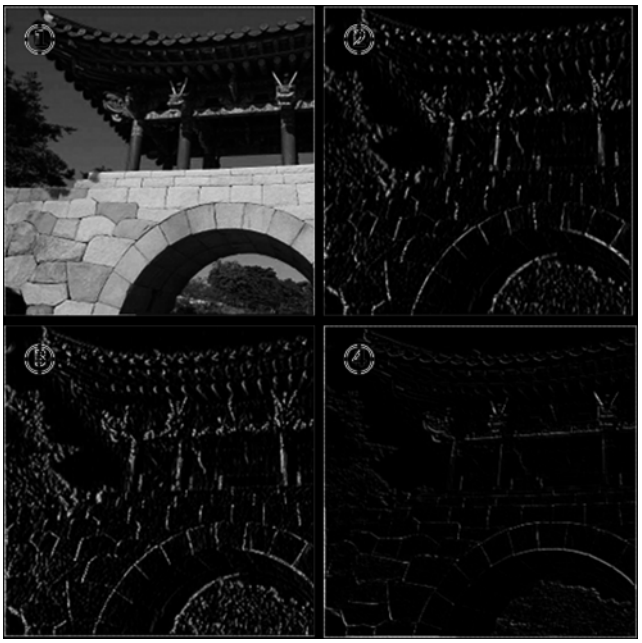
3.3 실험 결과

실험에서 이미지 처리 함수의 동작 여부를 검증하기 위해서 사칙연산을 이용한 단일영상 처리와 Convolution 함수를 이용한 에지검출(Edge detecting)[7]을 실행하였으며, 그 결과는 [그림 6]과 [그림 7]과 같다.



(그림 6) 산술연산 이미지 처리

[그림 6]에서 1번이 원본 이미지이며, 2~4번과 [그림 7]의 1번은 각각 Add, Subtract, Multiply, Divide 함수를 적용한 결과 이미지이다. [그림 7]의 2~4번은 Convolution 함수를 이용하여 각각 Prewitt, Sobel, Robert 에지 검출 기법이 적용된 이미지 결과이다.



(그림 7) 산술연산과 Convolution 이미지 처리

또한 이미지 처리 함수 중 Data 입출력을 제외한 나머지 모든 함수에 대해서 디바이스 수와 이미지 사이즈를 증가시키면서 수행하였을 때의 처리 시간을 측정하였으며, 그 결과는 [표 2]와 같다.

<표 2> 이미지 처리 시간

이미지 크기	1M(1024 X 1024)			
DUT 수	1DUT	2DUT	3DUT	4DUT
수행시간	1,100	1,128	1,275.5	1,400
이미지 크기	4M(2048 X 2048)			
DUT 수	1DUT	2DUT	3DUT	4DUT
수행시간	4,558.5	5,591.5	6,336.50	7,298.00
이미지 크기	8M(4096 X 2048)			
DUT 수	1DUT	2DUT	3DUT	4DUT
수행시간	9,075.5	10,993	12,516.5	14,316

단위(ms)

결과에서 테스트 시간은 캡처 된 이미지 데이터가 IDPServer에 전송되는 시간과 이미지 처리 함수가 수행되는 시간을 포함한 것이다.

4. 결론 및 향후 연구

본 논문에서는 CMOS 이미지 센서에서 캡처한 이미지를 목적에 맞게 여러 형태의 테스트를 진행할 수 있는 이미지 처리 함수를 구현하고 실험하였다. 이를 위해, CMOS 이미지 센서로부터 전송되는 이미지 데이터를 저장하고 처리하기 위한 자동 검사 시스템을 구성하였으며, 구현한 이미지 처리 함수를 이용하여 전송된 이미지 데이터를 처리하였다. 또한 함수들의 검증을 위하여 실제 이미지를 대상으로 Pixel 처리 기법을 적용한 결과와 디바이스 수 및 이미지 사이즈를 증가시키면서 이미지 처리 함수의 실행 시간을 측정하고 제시하였다.

향후 연구에서는 이미지 사이즈가 증가함에 따라 다소 느려지는 처리 시간을 보완하기 위해서 이미지 처리 함수별 최적화가 필요할 것이다.

참고문헌

[1] E. R. Fossum, "CMOS image sensors: Electronic camera-on-a-chip", IEEE Trans. Electron Devices, vol. 44, pp.1689 -1698, Dec 1997.

[2] M. Burns and G. W. Roberts "An Introduction to Mixed-Signal IC Test and Measurement", Oxford, 2001.

[3] http://www.mipi.org

[4] Xu Youqing, Yu Shengsheng, Zhou Jingli, Fang Zuyuan "Detection and Compensation of Bad Pixel for CMOS Image Sensor," International Conference on Sensor and Control Techniques(ICSC 2000) Proceedings of SPIE, Vol. 4077, pp.208-212, June 2000.

[5] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing Third Edition, Prentice- Hall, 2007.

[6] http://www.testian.com/_products_02_1.htm

[7] C. C. Kang and W. J. Wang, "A Novel Edge Detection Method Based on Maximization of the Objective Function," Pattern Recognition, Vol.40, No.2, pp.609-618, 2007.