한국산학기술학회논문지
Vol. 10, No. 11, pp. 3094-3097, 2009

임의의 비율을 지원하는 영상 축소 알고리즘과 하드웨어 구조

박현상[1*]
[1]공주대학교 전기전자제어공학부

Image Resolution Reduction Algorithm of Arbitrary Rate and Its Hardware Architecture

Hyun-Sang Park[1*]
[1]**Division of Electrical Electronic and Control Engineering Kongju National University**

요 약 임의의 입력 해상도와 출력 해상도의 비율로 주어지는 영상 축소 스케일러를 구현하려면 축소된 영상에 대한 화소의 좌표를 계산하기 위해서 범용 제산기의 사용이 요구된다. 이 범용 제산기는 매 화소마다 동작해야하기 때문에 처리속도를 높이기 위하여 LUT로 구현되나, LUT의 정밀도에 따라서 하드웨어의 규모가 비대해지는 문제가 야기된다. 본 논문에서는 제산기나 LUT 기반의 제산 연산을 수반하지 않는 영상 축소 알고리즘을 제안한다. 제안한 알고리즘은 비교기와 가산기 만으로 구성되어 있으며, 임의의 유리수로 표현되는 축소 비율을 허용함에도 불구하고, 기존 방식에 비해서 1/10 이하로 하드웨어 규모를 줄이는 것이 가능하다.

Abstract The use of general-purpose divider is inevitable to implement a image down-scaler when an arbitrary scaling ratio is given. To get an output at every clock from the divider, the divider should be implemented by LUT, however, its hardware size will be bigger and bigger as the precision level is increased. In this paper, a new image scaling algorithm is presented for a arbitrary scaling ratio, which do not requires a general-purpose or LUT-based divider. The proposed algorithm utilizes only comparators and adders such that the hardware size can be reduced by 1/10 compared to the conventional approaches.

Key Words : ISP, Scaler, Resizer

1. 서론

ISP(Image Signal Processor)[1]는 이미지 센서로부터 입력되는 Bayer 패턴[2]의 영상으로부터 R, G, B 색성분을 복원하여 이를 Y, Cb, Cr 영역으로 색변환하고, chroma 4:2:2 형식으로 부표본화하여 출력하는 등의 일련의 작업을 처리하는 영상 처리 프로세서를 지칭한다. ISP는 이러한 기본 기능 외에도 렌즈 왜곡 보정, 영상 잡음 제거, Gamma 보정, 자동 노출 조정, 자동 색온도 조정 등의 기능 들이 부가되어 이미지 센서에서 제공되는 영상의 화질이 광량, 광원 등과 같은 동작 환경에 영향을 받지 않고 일정한 수준을 유지할 수 있도록 조정하고 있다.

최근 완료된 SMIA[3]나 MIPI[4]와 같은 사실상의 국제표준에서는 ISP에서 다양한 형식의 영상 데이터가 출력될 것을 규정하고 있다. 여기에는 Bayer 패턴이나 YCbCr 4:2:2 부표본화 형식과 같은 보편적인 형식 외에도 YCbCr 4:2:0, RGB565, RGB444, JPEG 등까지도 추가되어 압축된 영상과 비압축된 영상, 그리고 YCbCr과 RGB라는 2개의 색체계(color space)까지도 지원하고 있다.

YCbCr 색체계는 JPEG, MPEG, H.264[5]와 같은 영상 압축 규격에서 사용되지만, RGB 색체계는 평판 디스플레이를 위해서 사용된다. 동영상이나 정지영상으로 촬상된 영상의 해상도는 표준 해상도를 따르지만, RGB 형식으로 출력하는 경우에는 임의의 해상도를 가지는 디스플레이 장치를 지원하기 때문에, 비표준 해상도를 따르는 경우가 많다.

*교신저자 : 박현상(vandammm@kongju.ac.kr)
접수일 09년 04월 29일 수정일 (1차 09년 09월 01일, 2차 09년 11월 06일) 게재확정일 09년 11월 12일

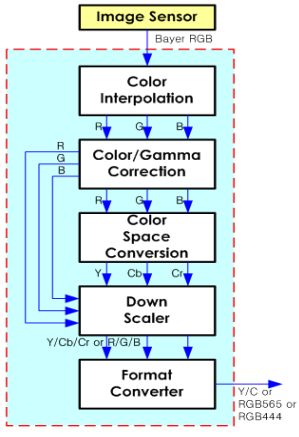
따라서 이미지 센서의 해상도와 이를 축소해서 얻어지는 영상의 해상도의 비율은 임의의 유리수로 표현되어, 무한소수로 표현되는 경우도 가능하기 때문에 완벽한 정밀도를 가지고 모든 종류의 축소 비율을 지원하려면 많은 하드웨어 자원을 요구하게 된다. 또한 유한소수로 표현되는 경우라 하더라도 임의의 값을 가지는 분모를 지원해야하기 때문에, 승산기와 제산기를 이용하여 좌표를 계산하는 일은 피할 수 없게 된다. 이러한 좌표 계산은 매 화소마다 수행될 수 있어야하기 때문에 제산기는 고속처리가 가능하도록 LUT (Look-Up Table)로 구현되며, 임의의 유리수 비율을 지원하기 위해서는 LUT의 정밀도를 높여야 하므로 결국 하드웨어의 규모가 크게 증가하게 된다.

본 논문에서는 고비용 하드웨어인 LUT 기반 고속 제산기를 사용하지 않고, 임의의 유리수로 표현되는 축소 비율을 처리하는 알고리즘과 이에 대한 하드웨어 구조를 제안한다.

2. 영상 축소 알고리즘

2.1 ISP의 구조

ISP는 Bayer 패턴의 영상 데이터를 JPEG이나 MPEG 등에서 필요로 하는 Y-Cb-Cr 데이터로 변환하여 출력하는 하드웨어를 통상적으로 일컫는 용어이다. ISP는 불완전한 색성분으로부터 부족한 다른 색성분을 복원하는 단계와 이를 Y-Cb-Cr 색체계로 변환하는 과정을 기본적으로 포함한다. 또한 인간이 인지하는 색분포로 변환해주는 색보정, 이미지 센서나 디스플레이 장치의 비선형성을 보상해주는 감마 보정이 포함된다.



[그림 1] ISP의 일반적인 구조

그림 1은 영상 축소 기능이 내장된 ISP의 기본적인 블록도를 나타낸다. 영상 축소는 Y/Cb/Cr이나 R/G/B 데이터의 색체계에 상관없이 24-bit 데이터에 대해서 동일한 방식으로 수행된다.

2.2 영상 축소 알고리즘

영상 축소는 2 단계로 구성된다[6]. 1 단계에서는 고주파 성분이 저주파 대역으로 내려오는 현상인 aliasing을 막기 위해서 anti-aliasing 필터를 적용하고, 2 단계에서는 축소 비율에 맞도록 부표본화(sub-sampling)를 수행한다. 1 차원적으로 N 개의 입력 데이터 {PI[i], $0 \le i \le N-1$}로부터 M 개의 데이터 {PO[j], $0 \le j \le M-1$}를 출력하고자 할 경우, 다음과 같이 출력 데이터를 결정한다.

```
for (j=0; j<M; j++){
  i = (int)(N/M*j+0.5);
  PO[j] = PI[i];
}
```
(a)

```
for (j=0; j<M; j++){
  i = (int)(N/M*j);
  w1 = (float)(N/M*j)-i;
  PO[j] = (1-w1)*PI[i] + w1*PI[i+1];
}
```
(b)

[그림 2] NN 알고리즘 기반 부표본화 방식

그림 2(a)는 공간상에서 가장 근접한 입력 데이터를 찾아서 취하는 방식으로 NN(Nearest Neighbour) 알고리즘이라고 한다[7]. 그림 2(b)는 출력 데이터를 추출하는 좌표와 인접한 입력 데이터 인덱스 간의 거리에 비례하는 가중치를 고려하여 부표본화하는 것으로써 선형적으로 인터폴레이션하는 방식이다[6]. NN 알고리즘보다 정밀한 방식으로 부표본화한다고 기대할 수 있으나, 매 화소마다 수행해야하는 계산량이 월등히 많은 것이 단점이다. 따라서 하드웨어의 크기나 이에 따른 전력소모량에 민감하거나 미묘한 화질 차이에 둔감한 응용분야에서는 NN 알고리즘이 주로 사용된다.

```
for (j=0; j<M; j++){
  i = (int)(N/M*j+0.5);
  PO[j] = PI[i];
}
```
(a)

```
for (j=0; j<M; j++){
  i = (int)(N/M*j);
  w1 = (float)(N/M*j)-i;
  PO[j] = (1-w1)*PI[i] +
w1*PI[i+1];
}
```
(b)

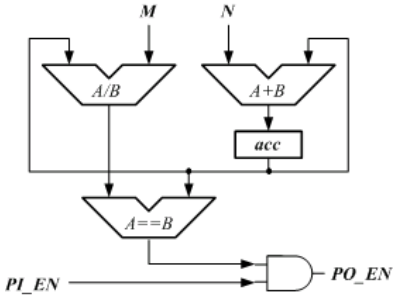
[그림 3] NN 알고리즘의 다른 표현 방식

그림 2(a)는 축소되어 출력되는 데이터 관점에서 표현되었으나, 하드웨어 설계를 지향한다면 그림 3(a)와 같이 입력되는 데이터 관점으로 수정해야한다. 즉, 순차적으로 입력되는 N개의 데이터로부터 출력되어야 하는 위치에 가장 근접한 데이터를만을 선택해서 출력해야 한다.

그림 3(a)는 승산과 제산을 포함하고 있으나, 승산은 그림 3(b)와 같이 누적가산(accumulation) 연산으로 대치할 수 있다. 그러나 이 경우에는 매 사이클마다 제산 연산을 수행해야하기 때문에 LUT 기반의 고속 제산기가 필요하다.

2.3 NN 알고리즘의 하드웨어 구조

그림 3(b)의 하드웨어 구조는 그림 4와 같다. *N*은 입력되는 데이터의 수, *M*은 출력되는 데이터 수를 나타내며, **PI_EN**은 유효한 입력 데이터가 발생하는 구간을, **PO_EN**은 유효한 출력 데이터가 발생하는 구간을 각각 나타낸다.



[그림 4] NN 알고리즘의 하드웨어 구조

누적가산기 ***acc***는 (M>>1)로 초기화되었다가 비교기의 출력과 ***PI_EN***의 논리곱 연산 신호인 ***PO_EN***에 의해서 *N*을 누적시킨다. 제산기에서는 ***acc***를 *M*으로 나눈 몫만을 출력한다.

3. 제안한 영상 축소 알고리즘

3.1 단순 연산 기반 NN 알고리즘

그림 3을 기반으로 구현된 NN 알고리즘은 제산기를 포함하고 있기 때문에, 하드웨어 구현 시 많은 비용이 소모된다. 그림 3(a)에 제산이 필요한 부분은 다음 조건식을 계산하는 경우이다.

$$i == (\text{int})\left(\frac{N\cdot j}{M}+0.5\right) \tag{1}$$

식(1)에서 (int)는 정수값을 취하는 연산이다. 상기 조건식은 식(2)의 조건식과 같이 전개하는 것이 가능하다.

$$i \le \frac{N\cdot j}{M}+0.5 < i+1 \tag{2}$$

식(2)에 정수 *M*을 곱하면 식(3)과 같은 조건식으로 정리된다.

$$M\cdot i \le N\cdot j+\left(\frac{M}{2}\right) < M\cdot i+M \tag{3}$$

식(3)에는 3개의 승산이 포함되어 있는데 이들은 덧셈으로 대치하는 것이 가능하다. 하드웨어로 구현할 때 *N*개의 입력 데이터는 순차적으로 입력되므로, *i*번째 입력에 대해서 *M*×*i*를 구하는 대신, *M*을 *i*번 누적시켜도 같은 값을 가진다. 마찬가지로 *j*번째 출력에 대해서 *N*×*j*를 구하는 대신 *N*을 *j*번 누적시켜도 같은 값을 갖게 된다. 데이터가 입력될 때마다 *M*을 누적시키는 변수를 ***acc0***라고 하고, 데이터가 출력될 때마다 *N*을 누적시키는 변수를 ***acc1***이라고 하면, 그림 3(a)의 알고리즘은 다음과 같이 표현할 수 있다. 단, 알고리즘간의 동치를 위해서 ***acc1***의 초기값은 (*M*>>1)이다.

```
acc0 = 0;
acc1 = (M>>1);
j = 0;
for (i=0; i<N; i++){
  acc0_nxt = acc0 + M;
  if (acc1>=acc0 && acc1<acc0_nxt){
    PO[j] = PI[i];
    j = j + 1;
    acc1 += N;
  }
  acc0 = acc0_nxt;
}
```

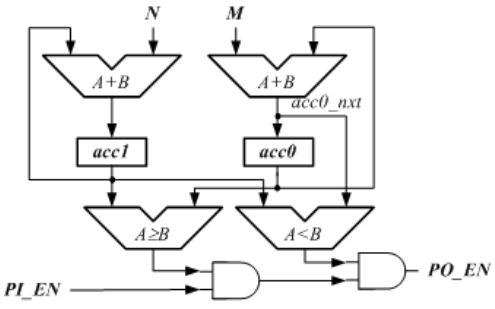
[그림 5] 승산/제산이 배제된 부표본화 알고리즘

그림 4에서 구현된 NN 알고리즘은 제산기를 포함하기 때문에, 하드웨어 구현 시 많은 비용이 소모된다. 그림 5는 제산 연산을 사용하지 않고, 2개의 가산기와 2개의 비교기로 구현할 수 있도록 NN 알고리즘의 처리 방식을 재구성한 것이다.

3.2 제안한 NN 알고리즘의 하드웨어 구조

그림 5에 대한 하드웨어 구조는 그림 6과 같다. 각종 신호나 파라미터는 그림 4와 동일하다. 그림 4의 구조와 비교할 때, 제산기는 누적연산기로 대치되었고, 같은 값

을 비교하는 비교기 대신에 크기를 비교하는 비교기로 대치되었다.



[그림 6] 제안한 NN 알고리즘의 하드웨어 구조

비교기가 구성하는 하드웨어 비용을 본다면, 등호에 대한 비교기는 XOR 게이트로 구현되지만, 크기에 대한 비교기는 감산기(subtracter)로 구현되기 때문에 2배 이상의 비용이 소요된다. 그러나 LUT로 구현되는 제산기로 인한 비용증가는 비교기로 인한 비용증가를 훨씬 상회하기 때문에, 전체적인 비용은 제안한 구조를 채택했을 때 대폭 줄어들게 된다.

4. 결론

본 논문에서 제안한 NN 알고리즘과 LUT 기반 NN 알고리즘에 대한 하드웨어를 TSMC 0.18 μm 공정 셀 라이브러리를 사용해서 *M*과 *N*이 각각 임의의 10 비트로 표시되는 경우에 대해서 합성한 결과를 표 1에 정리했으며 단위는 μm^2이다. LUT의 정밀도에 따라서 하드웨어 규모는 달라지는데, 10 비트의 해상도라면 LUT는 최소 25 비트 이상이어야 한다. 그러나 이 경우 하드웨어 규모는 10배 이상 차이가 난다.

[표 1] 1차원 NN 알고리즘의 하드웨어 규모 비교

LUT depth	LUT-based	Proposed
30-bit	107725.46	9245.50
25-bit	91908.43	
20-bit	67369.58	
15-bit	24578.77	

제안한 구조의 NN 알고리즘을 화소 단위와 라인 단위에 대해서 적용하면, 영상 축소 하드웨어를 바로 구현하는 것이 가능하다. 실제로 제안한 알고리즘에 기반을 둔 영상 축소 하드웨어의 규모는 35,443 μm^2에 불과하여 LUT 기반의 1차원 하드웨어에 비해서도 적은 규모이다.

결론적으로 제안한 NN 알고리즘을 적용한 영상 축소 하드웨어는 기존 방식에 비해서 1/10 이하의 규모를 가지므로, 하드웨어 비용에 민감한 이미지 센서에 적용하더라도 적은 비용 추가로 다양한 영상 축소를 지원하는 것이 가능하다.

참고문헌

[1] J. Nakamura, *Image Sensors and Signal Processing for Digital Still Cameras*, CRC Press, 2006.
[2] 박상식, *CCD/CMOS 이미지 센서*, (주)전자신문인터넷, 2007.
[3] SMIA 1.0 Part 2: CCP2 Specification (ttp://www.smia-forum.org)
[4] MIPI Alliance Standard for Camera Serial Interface 2 (http://www.mipi.org)
[5] Iain Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*, Wiley, 2003.
[6] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing, 1992.
[7] http://en.wikipedia.org/wiki/Nearest-neighbor_interpolation

박 현 상(Hyun-Sang Park) **[종신회원]**

- 1993년 8월 : 한국과학기술원 전기및전자공학과 (공학석사)
- 1999년 8월 : 한국과학기술원 전기및전자공학과 (공학박사)
- 1998년 12월 ~ 2005년 2월 : 삼성전자 LSI사업부 책임연구원
- 2005년 3월 ~ 현재 : 공주대학교 전기전자제어공학부 부교수

<관심분야>
영상처리, 카메라 신호처리, 멀티미디어 SoC