碩士學位論文

확대 영상의 개선 알고리즘 제안

濟州大學校 産業大學院

電子電氣工學科

康　吉　奉

2001年　6月

확대 영상의 개선 알고리즘 제안

指導敎授　金 壯 亨

이 論文을 工學 碩士學位 論文으로 提出함

2001年　6月　　日

濟州大學校 産業大學院

電子電氣工學科
(컴퓨터工學)

康 吉 奉

康吉奉의 工學 碩士學位 論文을 認准함

2001年　6月　　日

審査委員長　邊 翔 庸

委　　員　金 壯 亨

委　　員　安 基 中

SUMMARY

In these days of information technology and multimedia, the needs of digital information such as a visual and audio are on the high increase. The field of transmission of multimedia data, however, is facing the problematic storage size. These traffic problem could be solved sending a small-sized image data and on the other side, receiving and approaching a magnified data.

This paper presents newer and hybrid interpolation rather than exiting one to get high-resolution still images.

The nearest neighbor interpolation and bilinear interpolation are generally used to process visual information. But the displaying interpolation is a compound of nearest neighbor interpolation excluding the mosaic and bilinear interpolation excluding the blurring. In such way, more advanced high-resolution still image could be obtained maintaining advantages and making up weak points of interpolations which are mentioned above.

목 차

Ⅰ. 서 론

초기에 컴퓨터는 계산을 빠르게 하는 기계였다. 하지만 현재의 컴퓨터는 일상 생활에 활용되고 있다. 현재의 컴퓨팅 환경은 기업체의 MIS나 사무자동화와 일반인의 인터넷과 멀티미디어 정보활용 분야에 주로 사용되고 있으며, 인터넷에서의 사용되는 정보는 멀티미디어 정보가 주를 이루고 있다.

멀티미디어 정보는 텍스트, 음성, 영상, 동영상 등이 포함되어 있는 정보를 말한다. 멀티미디어 정보에서 영상이나 동영상 정보는 다른 정보에 비하여 데이터의 양이 엄청나게 크다. 데이터의 양이 크다는 것은 인터넷이나 네트웍 환경에서의 전송 트래픽이 크다는 것을 의미한다. 인터넷에서 사용하는 영상은 특별한 경우를 제외하고는 특별히 화질이 좋을 필요가 없다. 즉, 사람이 인식할 수 있을 정도의 영상이면 되는 것이다. 본 논문은 이러한 영상의 전송에 있어서 작은 영상을 보내고 확대된 영상을 본다면 전송 트래픽을 줄일 수 있다는 것을 위한 영상 확대에 있어서의 개선된 보간법을 제안하고 있다.

영상의 확대(image magnification)는 크게 영상 확장(image expansion)과 영상 보간(image interpolation)의 단계로 나누어 설명할 수 있다. 먼저 영상의 확장이란 원 영상의 화소를 확대하고자 하는 간격만큼 띄어놓는 것을 의미한다. 다시 말해서 원 영상의 화소를 확대 영상의 격자에 일정 간격으로 재배치하는 것을 의미한다. 영상의 보간이란 확대 영상의 격자에 존재하지 않는 격자점들의 화소값을 적절한 값으로 채우는 과정이다. 이와 같이 영상 확장과 보간의 과정을 거친 영상은 원 영상에 비해 증가된 수의 화소들로 표현되며, 이는 바로 영상이 확대되었음을 의미한다[1]. 이를 해상도 측면에서 보면 영상 확대는 주어진 영상으로부터 보다 해상도가 향상된 영상을 얻는 과정이라고 정의 할 수 있다[13]. 기존의 연구들에서는 저해상도의 영상으로부터 고해상도의 영상을 만들어내는 방법은 주로 신호의 보간 방식에 의존하고 있다. 그러나 하나의 정지 영상 내에서 공간적으로

인접한 화소들의 정보만을 사용하는 보간 방식에 의한 확대는 정보량의 보존이라는 측면에서 볼 때, 영상의 크기는 증가시킬 수 있으나 원 영상으로부터 영상의 확대에 이용할 수 있는 정보량이 제한되어 있어 실질적인 해상도 향상에는 기여하지 못하는 것으로 알려져 있다[14].

본 논문에서는 영상의 확대에서 기존에 주로 사용하는 보간법들에 대한 분석과 새로운 보간법을 제안하였다. 제안된 보간 알고리즘은 객관적 평가 결과 향상된 화질의 영상을 얻을 수 있었다.

본 논문에서 제안하는 알고리즘은 운영체제로서 Microsoft사의 한글 Windows Me를 사용했으며, 컴파일러는 Visual C++ 6.0을 사용했다.

PC 환경으로는 Pentium Ⅲ 866Mz에다 메모리가 128Mb, Harddisk로서는 20GB의 용량에서 구현했다.

Ⅱ. 이론적 배경

높은 해상도의 디지털 영상을 얻기 위해서는 표본 격자 간격을 보다 조밀하게 구성해야 하는데, 이를 위해서는 각 화소에 해당하는 센서(photo-detector)의 크기가 표본 격자 간격에 비례해서 작아져야 한다. 그러나 센서의 크기가 어느 임계치 이하로 작아지면 입사되는 광량이 적어져서 잡음(short-noise)에 의한 영상의 열화(blur)를 피할 수 없게 된다[2,3]. 이와 같은 잡음이 나타나지 않는 센서의 최소 크기는 약 $50\mu m^2$로 알려져 있는데, 현재의 charge coupled device(CCD) 기술은 이미 이 경계에 도달해 있다[4].

따라서 이러한 한계를 뛰어 넘는 높은 해상도의 영상을 얻기 위해서는 디지털 영상 처리 기술의 적용이 요구된다.[6,7,14]

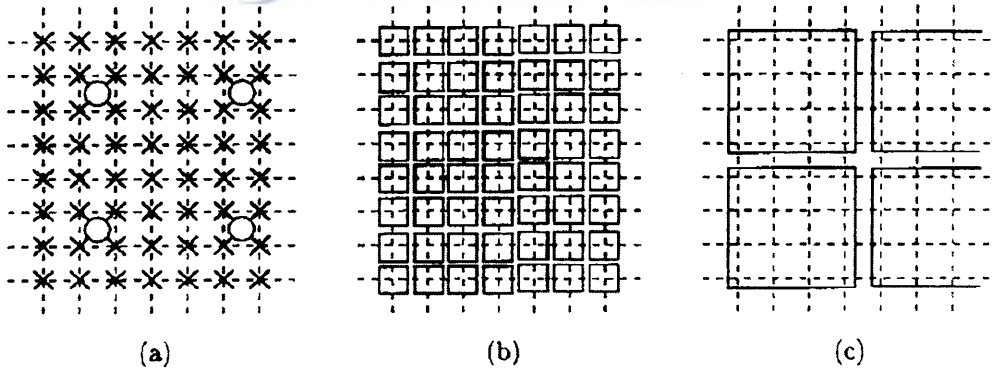


Fig. 2-1 직교격자배열을 사용한 영상 표본화의 예
(a) 표본주기 T_u, T_h 인 경우(×)와 표본주기 $4T_u, 4T_h$ 인 경우(◯)의 표본격자
(b) $x(m,n)$ 을 얻기 위한 센서의 배열구성
(c) $x_{\frac{1}{4}}(m,n)$ 을 얻기 위한 센서의 배열 구성

Fig. 2-1에서 연속 신호의 형태로 표시된 2차원 영상을 $x_c(p,q)$라 하면 수평, 수직 방향으로 각각 주기 T_v, T_h로 표본화된 이산 영상 신호는

$$x(m,n) = x_c(mT_v, nT_h), \quad \text{for } m,n = 0,1,\cdots,N-1 \qquad \text{식 (1-1)}$$

와 같이 표현할 수 있고, 이 식에서 $x(m,n)$으로 부터 수평, 수직 방향으로 각각 $\frac{1}{4}$씩 해상도가 저하된 영상은

$$x_{\frac{1}{4}}(m,n) = \frac{1}{16}\sum_{i=0}^{3}\sum_{j=0}^{3} x(4m+i,4n+j), \quad \text{for } m,n = 0,1,\cdots,\frac{N}{4}-1 \qquad \text{식 (1-2)}$$

이 된다.

2.1 영상의 스케일 변환

영상의 해상도 저하가 Fig. 2-2 에서 보는 바와 같이 고해상도의 영상 *x(i, j)*으로부터 저해상도 영상 *y(i, j)*으로의 천이(transition)되는 과정이라고 한다면, 영상의 확대는 *y(i, j)*가 주어진 상황에서 원래의 고해상도 영상 *x(i, j)*를 추정해 내는 과정이 된다.

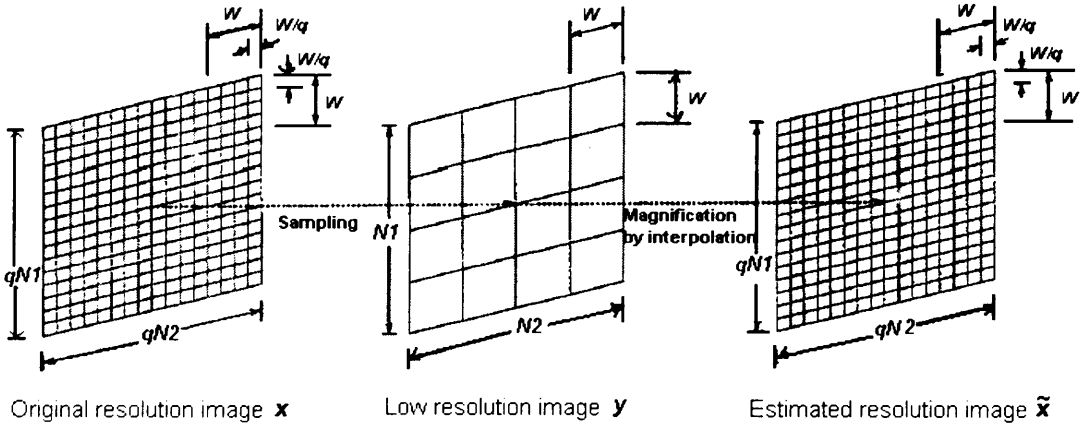


Fig. 2-2 해상도가 다른 두 영상 상호간의 관계

Fig. 2-2 에서 *y(i, j), (i = 0, · · ·, N_1 -1 ; j=0, · · ·, N_2-1)*의 화소가 *x(i, j)*의

$q \times q$개의 화소로부터 변화된 것이라면, $x(i, j)$ 영상의 화소수는 $qN_1 \times qN_2$가 될 것이다.

영상을 2배 확대하면 저해상도 영상의 한 점이 고해상도 영상의 4개의 점으로 사상되는 것이다. 즉, 저해상도 영상의 한 점으로부터 고해상도 영상의 4개의 점의 영상 화소를 추정하는 것이다. 그러나 한 점에서 4개의 점의 화소값을 추정하기는 어렵고, 또한 부정확한 결과의 영상을 얻을 수밖에 없다. 이러한 문제를 해결하는 방법으로 보간법이 사용된다 [14].

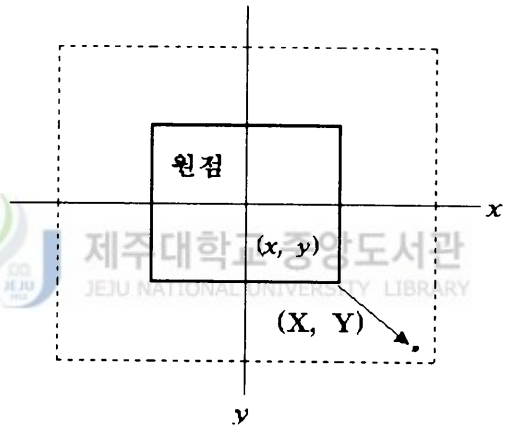


Fig. 2-3 영상 데이터의 크기 변환

영상의 크기를 변환하는 방법을 살펴보면, 어떤 점 (x,y)가 확대, 축소되어 (X,Y)로 위치가 변하면, 두 좌표 사이에는 다음과 같은 관계가 성립한다.

$$X = ax$$
$$Y = by \qquad \text{(식 2-1)}$$

위 식 (2-1)에서 a, b는 각각 x 방향, y 방향의 확대율이다. 이 수식을 이용하여 영

상을 처리한다면, a, b 가 1보다 큰 값을 가지는 경우, 영상 데이터는 확대가 되고, 1보다 작을 때에는 축소가 된 출력 데이터를 얻을 수 있다. 이 수식에 의해 우리는 모든 화소점 (x,y) 에 대해서 이 연산을 행하고, 출력 영상의 점 (X,Y) 의 농도값에 입력 영상의 점 (x,y) 의 농도값을 쓰면, 영상의 확대 또는 축소를 할 수 있다.

2.2 보간법(Interpolation)

원 영상이 1/4 인수의 해상도로 축소되는 경우 원 영상의 화소 격자에서 하나씩 건너뛰면서 새로운 영상의 격자에 써 가면 영상이 원 영상의 1/4 인수의 해상도로 줄어든다. 그러나 원 영상을 4의 인수에 의해 해상도를 확대하는 것은 화소의 값이 없는 격자가 생겨서 잘되지 않는다. 4의 인수에 의해 해상도가 확대되면 실제 화소값들 사이에 빈 공간이 생기게 되는 것이다.

보간법이란 주어진 데이터 점들의 정보로부터 그 점들 사이의 정보를 유추하는 수치적 방법이다. 보간법은 주변의 화소들을 분석함으로써 새로운 화소를 생성한다. 어떤 작업에 대하여 적절한 보간 함수를 선택하는 것은 어떤 응용 문제인가에 달여있으며, 보간 함수에 따른 질과 처리 시간 사이에는 트레이드 오프(trade-off)가 존재한다. 즉, 매우 복잡한 알고리즘은 영상의 질을 향상시키지만 보간 함수가 복잡하면 할수록 더 많은 처리 시간을 요구한다.

영상 처리에서 주로 사용되는 보간법에는 최근접 이웃화소 보간법과 양선형 보간법, , B-스플라인 보간법, Cubic 보간법[11] 등 여러 가지 방법이 있으나, 영상 처리에는 최근접 이웃화소 보간법과 양선형 보간법이 주로 사용되고 있다.

2.2.1 최근접 이웃화소 보간법

원 영상을 2배 확대하면 영상이 조금 변한다. 이것을 이해하기 위해, 다음 Fig. 2-4를 살펴보자.

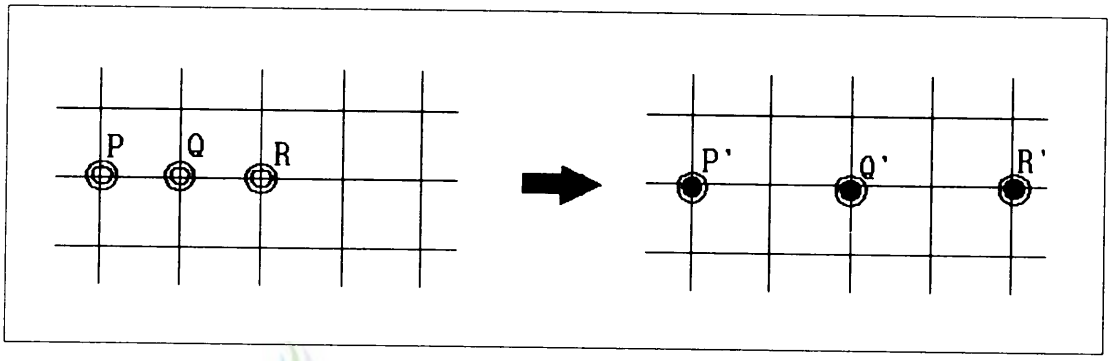


Fig. 2-4 영상의 2배 확대

Fig. 2-4 에서 보면, 입력 영상의 화소 P가 출력 영상의 화소 P′에 대응(mapping)된다. 그리고, 입력 영상 P의 인근에 있는 점 Q, R이 출력 영상에 각각 Q′, R′로 대응되면, Q′, R′는 확대율에 의하여, P′의 주변에 위치하고 있다. 2배 확대할 경우에는 출력 영상의 한 화소 P′점의 인접 화소에는 해당되는 데이터가 없기 때문에, 출력 영상 데이터에도 화소가 띄엄띄엄 있게 된다. 따라서, 이를 피하기 위해 '입력 영상을 기준으로 한 출력 영상의 매핑(mapping) 방법'을 사용하지 않고, 출력 영상을 기준으로 해서, 출력 영상의 화소가 입력 영상의 어떤 화소에 대응관계를 이용하는 것이 더 나은 결과를 얻을 수 있다는 것을 알 수 있다. 이 과정을 수행하기 위해, 식 (2-1)의 역변환을 생각해 보자. 식 (2-1)의 역변환은 다음과 같다.

$$x = X/a$$
$$y = Y/b \qquad \text{(식 2-2)}$$

출력 영상의 모든 화소 (X,Y) 에 대하여 식 (2-2)를 계산하고, 대응하는 입력영상을 구해 이 화소의 농도값을 쓰면, 위에서 언급한 현상은 일어나지 않을 것이다.

위의 결과는 앞서서 설명한 방식보다 더 좋은 결과임을 확인할 수 있다. 이 방법을 사사오입이라 부르며 그 개요는 다음과 같다.

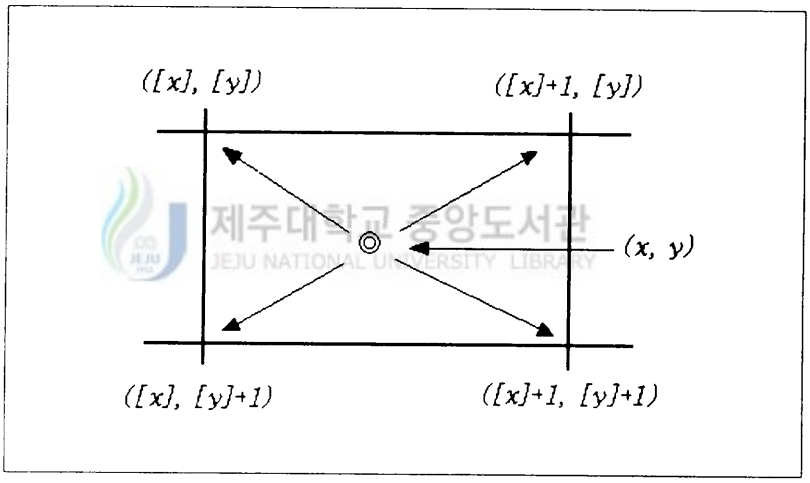


Fig. 2-5 화소값 구하는 방법

위에서 언급한 식 (2-2)의 계산은, 일반적으로 실수연산을 행할 필요가 있기 때문에 좌표 x, y는 소수부를 포함한다. 그러나 입력 영상의 화소의 위치(address)는 반드시 정수이어야 하므로 위치 계산에 있어서는, 어떤 형태로서 정수화할 필요가 있다. 이것을 영상에서 생각하면, Fig. 2-5 와 같이 좌표 (x,y)에 가장 근접한 격자점을 선택하는 것이다. 그러므로, 사사오입법을 최근접법이라고 부르기도 한다. 이 방법을 적용한 영상은 모자이크 현상이 나타나게 된다. 이 현상은 확대율을 크게 하면 두드러지게 나타난다.[8]

일반적으로 하나의 입력 화소의 값을 갖는 출력 화소들의 수가 크면 클수록 출력은 더 나쁘게 보인다. 큰 수에 의한 스케일링은 이것의 한 예이다. 또한 톱니 모양으로 알려진 시각적인 뭉툭함(blockiness)이 출력에 나타날 수 있다. 더 좋은 결과를 요구한다면 다른 보간 함수를 사용하는 것이 좋다.

Fig. 2-6 최근접 이웃화소 보간법에 의한 영상확대

2.2.2 양선형 보간법

최근접 이웃화소 보간법 보다 화질을 좋게 하기 위해 양선형 보간법(bilinear interpolation)이라 부르는 방법을 이용하는데, 영상 처리분야에서 일반적으로 사용하는 보간법이다. 양성형 보간법에서 새롭게 생성된 화소는 4개의 가장 가까운 화소들에 가중치를 곱한 값들의 합이다. 가중치들은 선형적으로 결정되어진다. 각각의 가중치는 각각의 존재하는 화소로부터의 거리에 정비례한다[11].

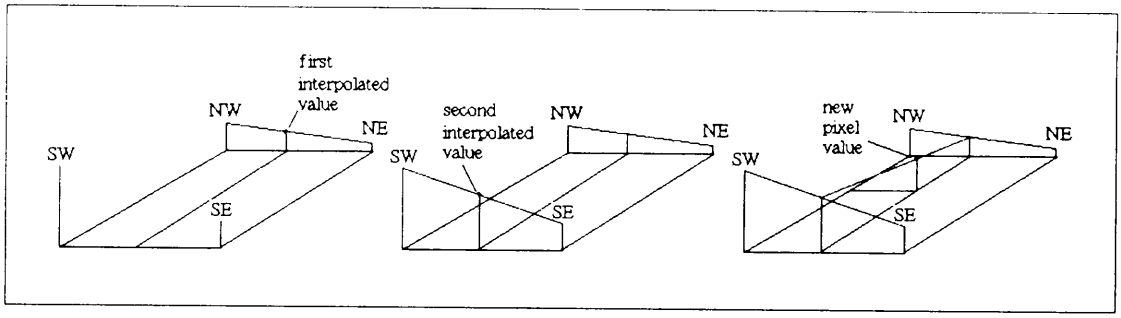


Fig. 2-7 양선형 보간법의 수행 단계

양선형 보간법은 Fig. 2-7 에서처럼, 세 번의 일차 보간들을 요구한다. 양선형 보간법은 최근접 이웃화소 보간법 보다 더 부드러운 영상을 산출한다. 화소당 세 번의 일차 보간을 요구하기 때문에 양선형 보간법은 최근접 이웃화소 보간법 보다 상당히 많은 계산을 요구한다. 양선형 보간법에 의한 영상확대 결과는 Fig. 2-8과 같다.

Fig. 2-8 양선형 보간법에 의한 영상확대

Fig. 2-8의 결과 영상을 보면 최근접 이웃화소 보간법에 의한 영상보다 확대 영상의 화질은 부드러우나 흐려지며, 윤곽선 부분에 화질의 향상이 더욱 필요하다는 것을 알 수 있다.

2.2.3 B-스플라인 보간법

최근접 이웃화소 보간법은 입력으로 한 개의 화소를 요구하고, 양선형 보간법은 새로운 화소를 생성하기 위해 4개의 가장 가까운 화소들을 사용한다. B-스플라인 보간법은 고등 차수 보간법의 하나로서 새로운 화소를 생성하기 위해 16개의 가장 가까운 화소들을 사용한다.

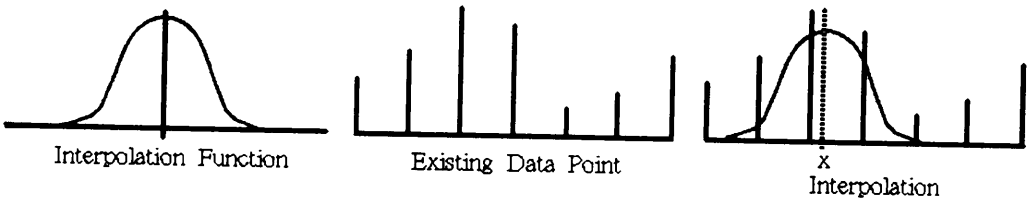


Fig. 2-9 1차원 보간법

모든 보간 함수들은 Fig. 2-10과 같은 원리에 의해서 보간을 실시한다. 보간 함수는 원하는 한 점의 중앙에 놓여지고 샘플 점들에서의 값들은 그러한 샘플들에 의해서 곱해지게 된다. 모든 결과들의 합은 새롭게 생성된 화소이다.

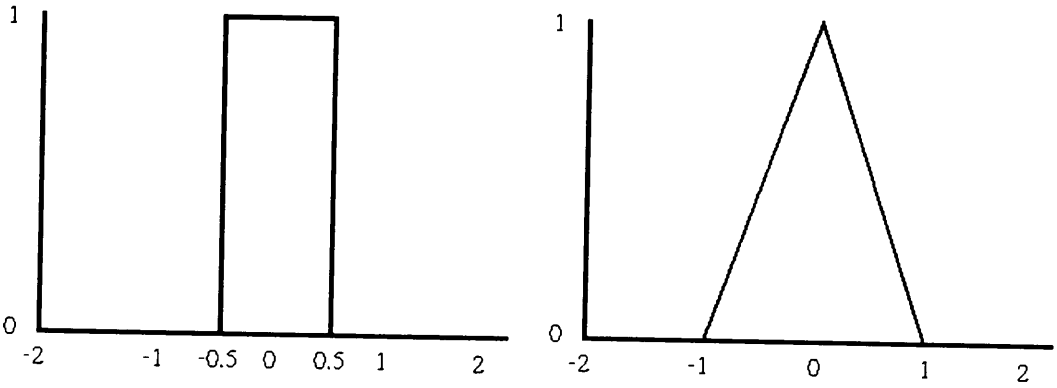


Fig. 2-10 1차원 보간 함수

이상적인 보간 함수는 저주파 통과 필터이다. B-스플라인 함수는 상당히 좋은 저주파 통과 필터를 만든다. 그러므로 B-스플라인 보간법은 보간 함수들 중 가장 부드러운 결과 영상을 산출한다(Fig. 2-12).

B-스플라인 보간법은 하나의 새로운 화소를 만들기 위해서 4번 샘플링 되어지며, 함수 값은 모두 양의 값을 가진다. B-스플라인 함수는 다음과 같이 정의된다.

$$f(x)=\begin{cases}\frac{1}{2}|x|^3-|x|^2+\frac{2}{3} & 0\le|x|<1\\ -\frac{1}{6}|x|^3-|x|^2+2|x|+\frac{4}{3} & 1\le|x|<2\\ 0 & 2\le|x|\end{cases}$$

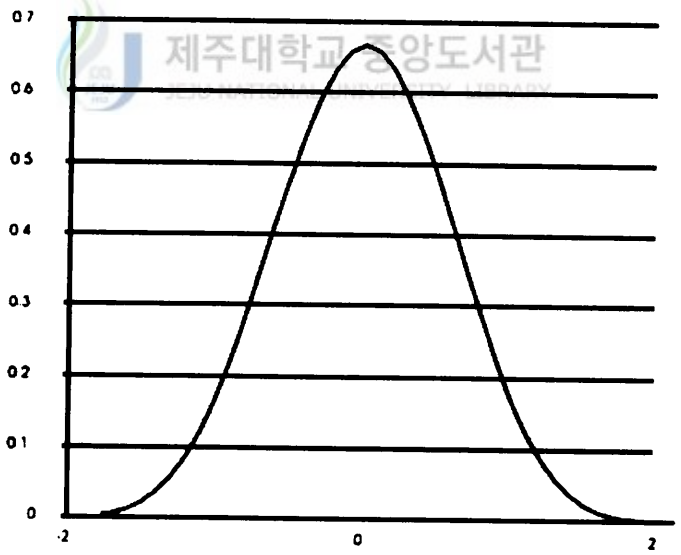


Fig. 2-11 B-스플라인 보간 함수

Fig. 2-12 B-스플라인 보간법에 의한 영상확대

2.3 객관적 평가

영상의 평가의 궁극적인 척도는 인간의 시각을 통한 평가이지만 이는 주관적인 성향이 강하므로 수학적인 계산을 통해 원 영상과의 차이를 객관적으로 측정하는 MSE(Mean Square Error)와 PSNR(Peak Signal to Noise Ratio)를 사용한다. 식(2-5)에서 X_{ij}는 원 영상을 의미하며 $\overline{X_{ij}}$는 확대된 영상을 의미한다.

$$MSE = \frac{1}{N^2} \sum_{j=1}^{N} \sum_{i=1}^{N} (X_{ij} - \overline{X_{ij}})^2 \qquad \text{(식 2-5)}$$

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \qquad \text{(식 2-6)}$$

확대 영상의 평가에서 PSNR값이 크다고 해서 반드시 주관적인 해상도 향상이 있었다고는 볼 수 없음에 유의해야 한다. 왜냐하면 이 지표는 원 영상과 확대 영상간의 전 영역에서 같은 위치에 해당하는 화소의 평균적 차분치를 의미하므로 PSNR이 높다고 해서 반드시 인간의 시각적인 특성을 만족시킨다고 볼 수는 없다.

Ⅲ. 구현 및 평가

일반적으로 확대된 영상에서 새로운 화소값을 구하는 보간법으로는 최근접 이웃화소 보간법과 양선형 보간법이 주로 사용되고 있다. 최근접 이웃화소 방법은 알고리즘이 간단하고 구현이 쉬운 반면, 모자이크 현상에 의해 영상의 화질이 떨어진다. 또한 양선형 보간법은 하나의 화소의 값을 구하기 위해 세 번의 선형 보간을 해야 하는 알고리즘의 복잡성에 비하여, 확대된 영상은 블러링 현상으로 인해 영상의 에지 부분이 부정확해진다는 것이다.

3.1 영상 확대

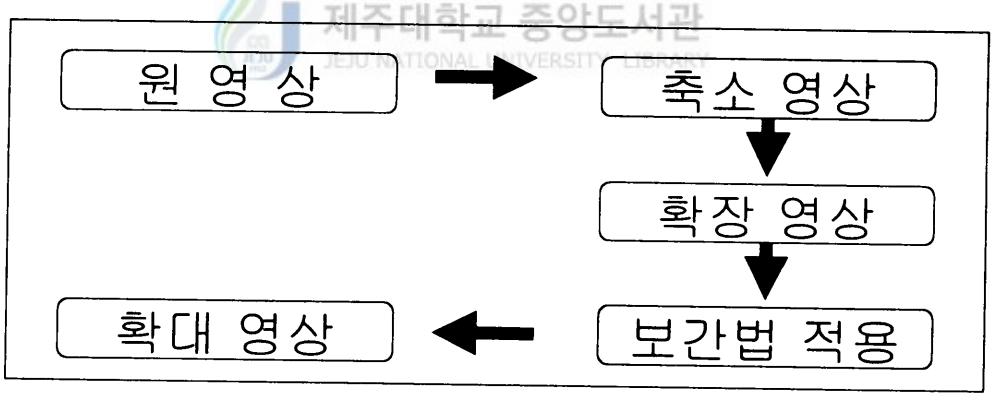


Fig. 3-1 일반적인 영상확대 구조도

본 논문에서는 최근접 이웃화소 보간법과 양선형 보간법의 장점을 취합하는 새로운 보간 방법을 제안하였다. 최근접 이웃 화소 방식에 의해 보간된 영상에 대하여 선형 보간을 함으로써 최근접 이웃화소 보간 방식과 양선형 보간 방식의 단점인 모자이크 현상과 블러링 현상을 제거하여 기존의 보간법 보다 화질이 좋은 확대영상을 얻는 것을 목표로 한다.

일반적인 영상 확대는 Fig. 3-1에 의한다. 확대된 영상의 화질은 Fig. 3-2에서 보는 바

와 같이 사용되는 보간법에 따라 다른 결과가 나타난다.

(a)축소 영상

(b) 최근접 이웃 화소 보간법

(c) 양선형 보간법

Fig. 3-2 보간법에 따른 영상확대

본 논문에서 제안하는 알고리즘은 기존의 영상확대와는 달리 확장된 영상의 보간 단계에서 최근접 이웃화소 보간법과 양선형 보간법을 조합하여 두 보간법들의 장점을 취합하고 단점을 상쇄시키는 것이다.

영상의 확대는 축소 영상을 확장된 영상의 격자에 사상시키는 영상 확장과 확장된 영상

의 존재하지 않는 화소의 값을 구하는 영상 보간의 단계로 볼 수 있다.

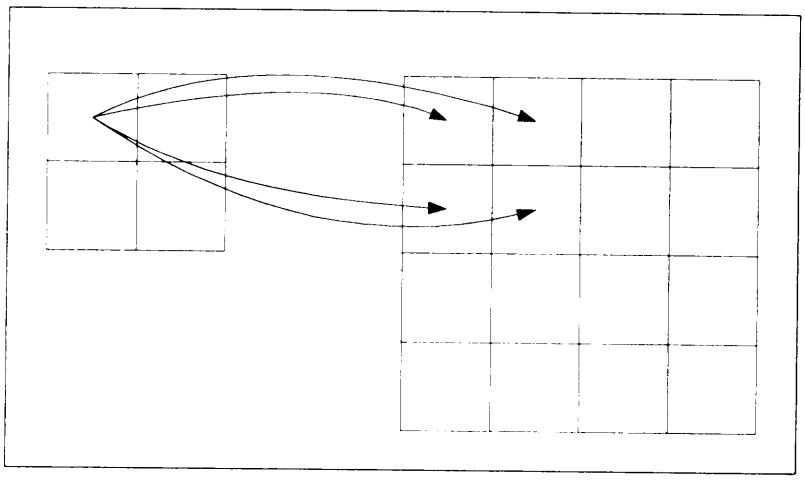
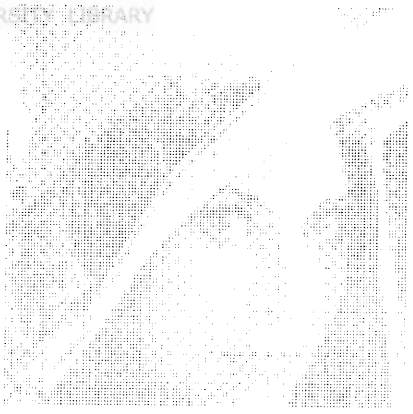


Fig. 3-3 축소 영상과 확장 영상의 1:4 사상 관계

(a) 축소 영상



(b) 2×2 확대 영상

Fig. 3-4 2배 확대 영상

영상의 2배 확대는 Fig. 3-3 에서와 같이 축소 영상의 1개의 화소가 확대 영상의 4개의 화소로 확장되는 것이다. Fig. 3-4는 축소 영상을 2배 확장한 영상이다. 축소 영상을 확장

하면 Fig. 3-4 (b)와 같이 확장된 영상에는 화소 값이 존재하지 않는 화소 3개가 생성된다. 이 존재하지 않는 화소에 원래의 영상이 가지고 있었을 화소의 값을 예측하는 것이 보간법이다. 영상 확대에서 보간법을 적용한다는 것은 그 만큼의 오류 요인을 포함하고 있다는 것이다. 즉 이 오류 요인을 최소화하는 것이 영상 확대에서 가장 중요한 것이다.

3.2 영상 보간

본 논문에서 제시하는 영상의 보간 기법은 확장된 영상에 대하여 최근접 이웃화소 보간 방식에 의하여 1차 보간을 하고 2차로 다시 양선형 보간을 실시하는 것이다.

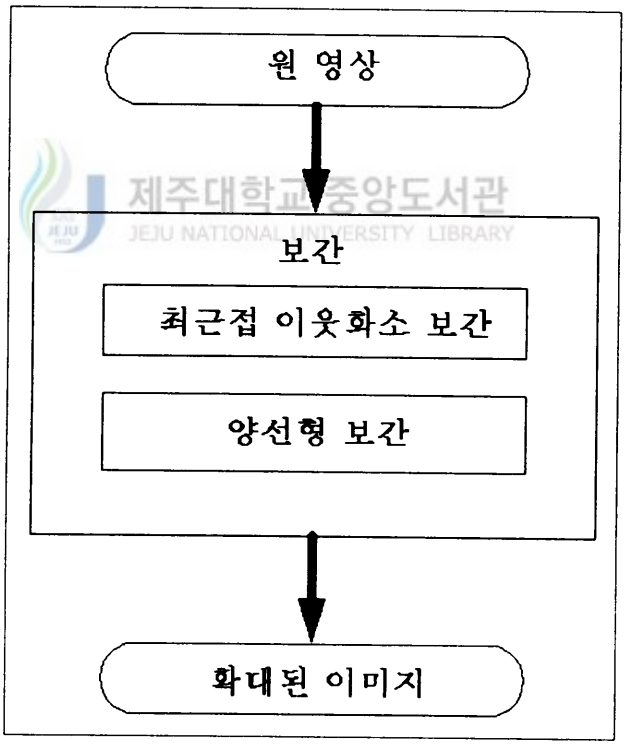


Fig. 3-5 제안 영상확대 구조도

본 논문에서 구현할 전체적인 영상 확대 구조도는 Fig. 3-5와 같다.

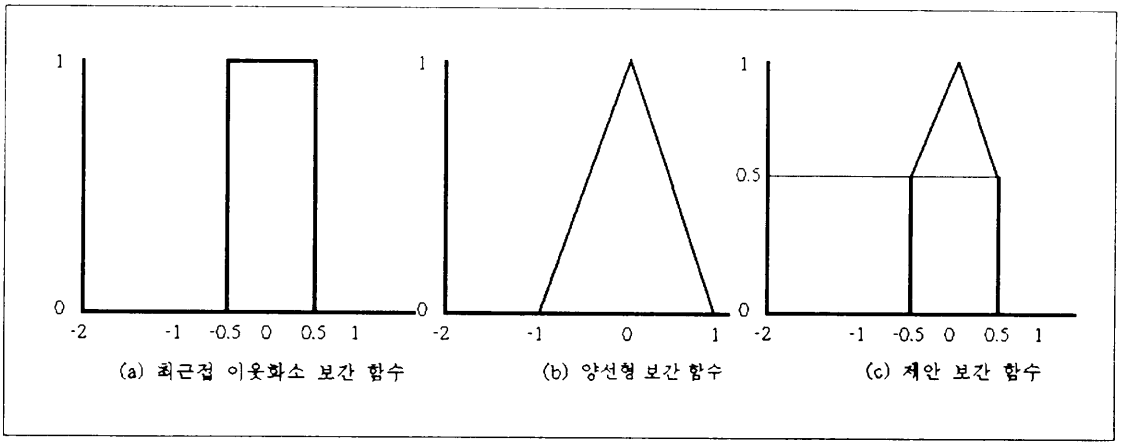


Fig. 3-6 보간 함수 비교

본 논문에서 제안하는 보간 함수는 Fig. 3-6과 같다. 제안하는 보간 함수는 최근접 이웃화소 보간 함수와 선형 보간 함수를 합친 것이다.

```
int x=0, y=0; int max=256;
Small();
Zoomin0();
    x =0; x < max; x+=2
        int y=0; y < max; y+=2
            m_Res[x+1][y] = m_Res[x][y];
            m_Res[x][y+1] = m_Res[x][y];
            m_Res[x+1][y+1] = m_Res[x][y];
        y = 1; y < max-1; y+=2
            m_Res[x][y] = (m_Res[x][y-1]+m_Res[x][y+1])/2;
            m_Res[x+1][y] = (m_Res[x+1][y-1]+m_Res[x+1][y+1])/2;
PSNR();
```


Fig. 3-7 제안 보간 함수에 대한 플로우 챠트

본 논문에서 제안하는 보간 함수에 대한 플로우 챠트는 Fig. 3-7에 나타내었다.

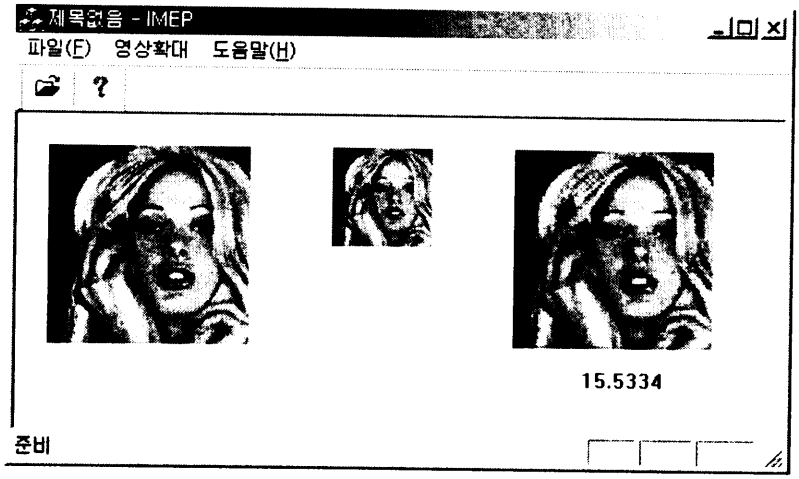


Fig. 3-8 구현된 영상확대 시스템

이 영상에 1차 선형보간을 실시하여 에지 부분의 모자이크 현상을 제거하고 양선형 보간법의 블러링 현상을 제거하는 효과를 얻었다. 본 논문에서 제시하는 방법은 알고리즘의 복잡도는 양선형 보간법 보다 간단하며, 확대된 영상의 주관적, 객관적 화질 또한 최근접 이웃화소 보간법과 양선형 보간법 보다 나은 영상을 얻었다.

3.3 결과 및 평가

다음 이미지들은 원 영상을 축소하고, 각기 다른 보간법을 적용하여 확대시킨 영상들을 비교하였다.

(a)원 영상 (b)축소 영상 (c)최근접 이웃화소 보간법

(d)양선형 보간법 (e)제안 보간법

Fig. 3-9 보간법에 따른 확대영상 비교 (Woman)

본 논문에서 제안하는 알고리즘에 의해 확대된 영상의 평가에 사용된 방법은 객관적 평가 방법으로써 MSE를 이용한 PSNR을 사용하여 영상의 품질 평가를 적용하였다.

3.3.1 객관적 평가

객관적 평가를 위해서 여러 가지 이미지들에 대해서 확대영상의 PSNR값을 구하여 비교하였다.

	최근접	양선형	제안	PSNR 이득
Lena	10.60	13.78	13.75	3.15
Airfield	12.21	14.73	14.65	2.44
Peppers	10.19	13.04	13.00	2.81
Couple	10.47	13.54	13.51	3.04
Girl	8.78	11.09	11.06	2.28
Man	10.16	13.13	13.09	2.93
Woman	11.52	15.57	15.53	4.01
Baboon	10.06	12.89	12.85	2.79
Crowd	9.02	11.41	11.38	2.36
Airplane	12.85	17.67	17.55	4.70
평균	10.59	13.69	13.64	3.05

Fig. 3-10 확대된 영상의 PSNR 비교

여기서 PSNR 이득은 제안한 보간법이 다른 보간법들 중에서 가장 작은 PSNR값과의 차이를 의미한다.

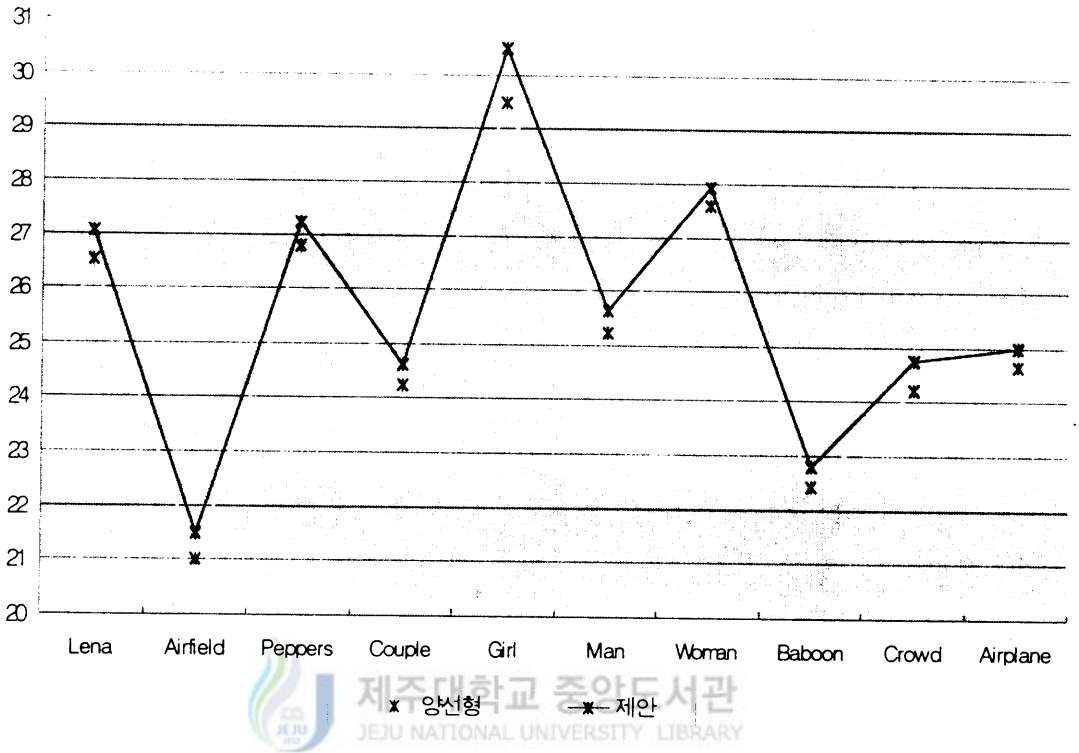


Fig. 3-11 양선형 보간법과 제안 보간법의 PSNR 비교

본 논문에서 제안하는 보간 알고리즘은 Fig. 3-10과 Fig. 3-11에서와 같이 확대된 영상의 PSNR 비교에서 여러 영상에 대하여 평균적으로 약 3db의 향상을 보였다.

Ⅳ. 결 론

본 논문에서는 정지 영상의 고해상도 확대에 있어서 확장 영상의 보간 단계에서 새로운 보간 기법을 제시하였다. 정지 영상의 확대는 원 영상을 축소하고, 축소시킨 영상을 다시 확대 영상의 결과를 얻기 위해서 축소 영상이 가지고 있는 정보량의 한계에 의하여 진정한 의미의 고해상도 영상을 얻을 수 없다. 이러한 제한된 정보량에 대하여 축소시킨 영상을 확장하고 확장된 영상에 제안하는 보간법을 적용하여 확대 영상에서 최근접 이웃 화소 보간법의 단점인 모자이크 현상을 제거하고, 양선형 보간법의 단점인 블러링 현상을 제거한 향상된 화질의 확대 영상을 얻을 수 있음을 확인하였다.

영상의 평가 방법으로는 PSNR(Peak Signal to Noise Ratio)에 의한 객관적인 평가 방법을 사용하여 제안하는 보간법이 최근접 이웃화소 보간법과 양선형 보간법보다 좋은 결과를 얻을 수 있음을 증명하였다.

본 연구의 적용 범위는 네트웍 환경에서 영상의 전송이나, 영상의 저장 등 많은 응용분야를 포함할 수 있다.

향후 연구 과제로는 칼라 영상에 대한 확대와 동영상 확대 기법에 대한 연구가 필요하며, 정지영상의 품질을 측정하는 주관적 평가의 기준에 대한 연구도 필요하다.

참 고 문 헌

[1] A. K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall, 1989.

[2] A. Papoulis, "Probability, Random Variables, and Stochastic Processes", 3rd., McGraw-Hill, 1991.

[3] K. Aizawa, T. Komastu, and T. Satio, "A Scheme for Acquiring Very High Resolution Images Using Multiple Cameras", Proc. 1992 Int. Conf. Acoust., Speech, Signal Processing, vol. 3, pp. 289-292, 1992.

[4] T. Ando, "Trend of High-Resolution and High-Performance Solid State Imageing Technology", Journal. ITE Japan, vol. 44, no. 2, pp. 105-109, February 1992.

[5] R. R. Schultz and R. L. Stevenson, "A Bayyesian Approach to Image Expansion for Inproved Definition", IEEE Trans. Image Processing, vol. 3, no 3, pp. 233-242, May 1994.

[6] B. C. Tom and A. K. Katsaggelos, "Reconstruction of a High Resolution Image From Multiple Degraded Mis-Registered Low Resolution Image", Proc. 1994 Viusal Comm., Image Processing, vol. 2308, no. 3, pp. 971-981, May 1994.

[7] B. C. Tom and A. K. Katsaggelos, "An Iterative Algorthm for Improving the Resolution of Video Sequences", Proc. 1996 Visual Comm., Image processing, vol. 2727, no. 3, pp. 1430-1438, March 1996.

[8] 이문호, 염재훈, "C언어를 이용한 영상신호처리", 대왕사, 1998.

[9] Gonzalez & Woods 著 ; 하영호 外 共譯, "디지털 영상처리", 그린, 1998.

[10] 이문호, "C언어 · MATLAB를 이용한 디지털 필터설계", 대왕사, 1997.

[11] Randy Crane 著; 최형일 外 共譯, 영상처리 이론과 실제, 홍릉과학출판사, 1999.

[12] 宋萬均, 제로트리 코딩을 이용한 이산 웨이브렛 필터의 성능 평가, 충북대학교 석사학위논문, 1998.

[13] 申定浩, 다채널 영상복원 및 확대기법에 관한 연구, 중앙대학교 석사학위논문, 1997.

[14] 서윤진, "확대된 정지영상의 개선을 위한 필터의 혼합적용에 관한 연구", 제주대학교석사학위논문, 1999.

부록 A. 보간법에 따른 확대 영상들의 비교

(a)원 영상

(b)축소 영상

(c)최근접 이웃화소 보간법

(d)양선형 보간법

(e)제안 보간법

Fig. 4-1 보간법에 따른 확대영상 비교 (Baboon)

(a)원 영상

(b)축소 영상

(c)최근접 이웃화소 보간법

(d)양선형 보간법

(e)제안 보간법

Fig. 4-2 보간법에 따른 확대영상 비교 (Lena)

(a)원 영상

(b)축소 영상

(c)최근접 이웃화소 보간법

(d)양선형 보간법

(e)제안 보간법

Fig. 4-3 보간법에 따른 확대영상 비교 (Couple)

(a)원 영상

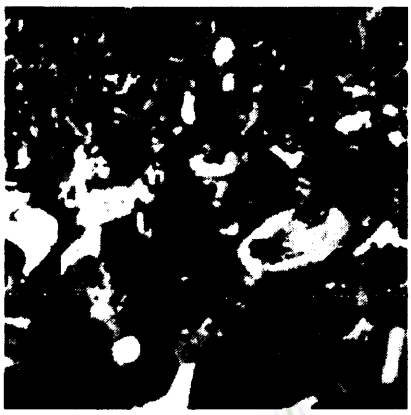
(b)축소 영상

(c)최근접 이웃화소 보간법

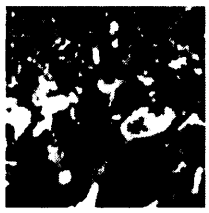
(d)양선형 보간법

(e)제안 보간법

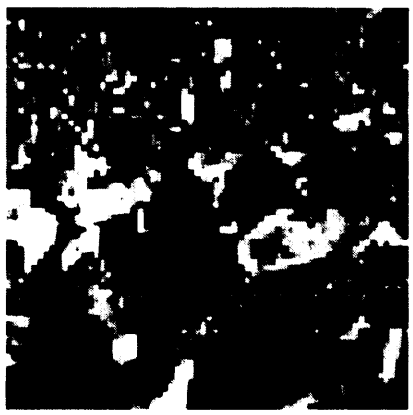
Fig. 4-4 보간법에 따른 확대영상 비교 (Girl)



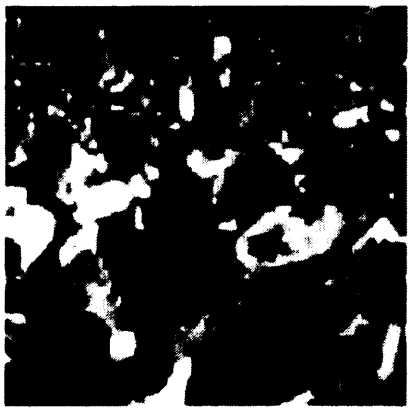
(a)원 영상



(b)축소 영상



(c)최근접 이웃화소 보간법



(d)양선형 보간법

(e)제안 보간법

Fig. 4-6 보간법에 따른 확대영상 비교 (Crowd)

(a)원 영상

(b)축소 영상

(c)최근접 이웃화소 보간법

(d)양선형 보간법

(e)제안 보간법

Fig. 4-5 보간법에 따른 확대영상 비교 (Man)

부록 B. 확대 영상의 제안 알고리즘에 대한 소스 코드

이 절에서는 확대 영상을 구현하는데 있어서 핵심적인 기술인 최근접 이웃 화소 보간법 및 양선형 보간법과 제안하는 혼합형 보간법의 알고리즘을 일부 표현한다.

```
// IMEP.cpp : Defines the class behaviors for the application.

#include "stdafx.h"
#include "IMEP.h"

#include "MainFrm.h"
#include "IMEPDoc.h"
#include "IMEPView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CIMEPApp

BEGIN_MESSAGE_MAP(CIMEPApp, CWinApp)
        //{{AFX_MSG_MAP(CIMEPApp)
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Standard print setup command
        ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIMEPApp construction

CIMEPApp::CIMEPApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CIMEPApp object

CIMEPApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CIMEPApp initialization

BOOL CIMEPApp::InitInstance()
```

```
{
	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a
shared DLL
#else
	Enable3dControlsStatic(); // Call this when linking to MFC statically
#endif

	// Change the registry key under which our settings are stored.
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization.
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));

	LoadStdProfileSettings();  // Load standard INI file options (including MRU)

	// Register the application's document templates.  Document templates
	//  serve as the connection between documents, frame windows and views.

	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CIMEPDoc),
		RUNTIME_CLASS(CMainFrame),	// main SDI frame window
		RUNTIME_CLASS(CIMEPView));
	AddDocTemplate(pDocTemplate);

	// Parse command line for standard shell commands, DDE, file open
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);

	// Dispatch commands specified on the command line
	if (!ProcessShellCommand(cmdInfo))
		return FALSE;

	// The one and only window has been initialized, so show and update it.
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();

	return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
```

```
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog

/////////////////////////////////////////////////////////////////////////////
// CIMEPApp message handlers
```

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//       are changed infrequently
//

#if
!defined(AFX_STDAFX_H__2C9DE8FB_E865_49AC_9420_317B29ADB97E__INCLUDED_)
#define AFX_STDAFX_H__2C9DE8FB_E865_49AC_9420_317B29ADB97E__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN                    // Exclude rarely-used stuff from Windows
headers

#include <afxwin.h>          // MFC core and standard components
#include <afxext.h>          // MFC extensions
#include <afxdtctl.h>              // MFC support for Internet Explorer 4 Common
Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows Common
Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_STDAFX_H__2C9DE8FB_E865_49AC_9420_317B29ADB97E__INCLUDED_)
```

```
// IMEP.h : main header file for the IMEP application
//

#if !defined(AFX_IMEP_H__6B2B1F2C_ED23_4DBD_8B2D_23A693324A07__INCLUDED_)
#define AFX_IMEP_H__6B2B1F2C_ED23_4DBD_8B2D_23A693324A07__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// main symbols

/////////////////////////////////////////////////////////////////////////////
// CIMEPApp:
// See IMEP.cpp for the implementation of this class
//

class CIMEPApp : public CWinApp
{
public:
	CIMEPApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CIMEPApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

// Implementation
	//{{AFX_MSG(CIMEPApp)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_IMEP_H__6B2B1F2C_ED23_4DBD_8B2D_23A693324A07__INCLUDED_)
```

```
// IMEPDoc.cpp : implementation of the CIMEPDoc class
//

#include "stdafx.h"
#include "IMEP.h"

#include "IMEPDoc.h"
#include "Math.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CIMEPDoc

IMPLEMENT_DYNCREATE(CIMEPDoc, CDocument)

BEGIN_MESSAGE_MAP(CIMEPDoc, CDocument)
        //{{AFX_MSG_MAP(CIMEPDoc)
                // NOTE - the ClassWizard will add and remove mapping macros
here.
                //    DO NOT EDIT what you see in these blocks of generated
code!
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIMEPDoc construction/destruction

CIMEPDoc::CIMEPDoc()
{
        // TODO: add one-time construction code here
        scale = 2;
        max = 256;
        psnr=0.0;
}

CIMEPDoc::~CIMEPDoc()
{
}

BOOL CIMEPDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)

        return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
```

```
// CIMEPDoc serialization

void CIMEPDoc::Serialize(CArchive& ar)
{
	/*
	if (ar.IsStoring())
	{
		// TODO: add storing code herear
		CFile fp;
		fp.Open("Result.raw",CFile::modeCreate | CFile::modeWrite);
		fp.WriteHuge (m_Res, 128 * 128);
		fp.Close();
	}
	else
	{
		// TODO: add loading code here
		CFile* fp = ar.GetFile ();
		if(fp->GetLength() != 128 * 128)
		{
			AfxMessageBox("128*128 이미지가 아닙니다 ! ");
			return;
		}
		else
		{
			ar.Read(m_Org, fp->GetLength());
		}
		Flag = true;
	}
*/
}

/////////////////////////////////////////////////////////////////////////////
// CIMEPDoc diagnostics

#ifdef _DEBUG
void CIMEPDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CIMEPDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CIMEPDoc commands

// 2배 확장
void CIMEPDoc::ZoomIn0()
{
	InitImage();


	for(int y = 0; y < 128; y++)
	{
```

```
                for (int x = 0; x < 128; x++)
                {
                        m_Res[y*scale][x*scale] = m_small[y][x];
                }
        }
}

// 최근접 이웃화소 보간법
void CIMEPDoc::ZoomIn1()
{
        Small();
        for(int x =0; x < 64; x++)
        {
                for(int y=0; y < 64; y++)
                {
                        m_Res[x*scale][y*scale] = m_small[x][y];
                        m_Res[x*scale][y*scale+1] = m_small[x][y];
                        m_Res[x*scale+1][y*scale] = m_small[x][y];
                        m_Res[x*scale+1][y*scale+1] = m_small[x][y];
                }
        }

        PSNR();
}

void CIMEPDoc::InitImage()
{
        for(int y = 0; y < 256; y++)
        {
                for(int x = 0; x < 256; x++)
                {
                        m_Res[y][x] = 255;
                }
        }
}

// 양선형 보간
void CIMEPDoc::ZoomIn2()
{
        int i=0,j=0;
        Small();
        // *********영상 확대**********
        ZoomIn0();

        // Step 1
        int y = 0;
        int x = 0;

        for(i =1; i < 256; i+=2)
        {
                for(j=1; j < 256; j+=2)
                {
                        m_Res[i][j] = (m_Res[y][x] + m_Res[y+2][x] +
                                                                m_Res[y][x+2] +
m_Res[y+2][x+2])/4;
                        x+=2;
                }
```

```
                x=0;
                y+=2;
        }

        // Step 2
        y=0;
        x=0;
        for(i =0; i < 256; i+=2)
        {
                for(j=1; j < 256; j+=2)
                {
                        m_Res[i][j] = (m_Res[y][x] + m_Res[y][x+2])/2;
                        x+=2;
                }
                x=0;
                y+=2;
        }

        // Step 3
        y=0;
        x=0;
        for(i =0; i < 256; i+=2)
        {
                for(j=1; j < 256 ; j+=2)
                {
                        m_Res[j][i] = (m_Res[y][x] + m_Res[y+2][x])/2;
                        y+=2;
                }
                y=0;
                x+=2;
        }
        PSNR();
}


void CIMEPDoc::PSNR()
{
        double err=0.0;
        psnr =0;
        mse=0;
        for(int i=0; i < 256; i++)
        {
                for(int j=0; j<256; j++)
                {
                        err = err + ((double)Org_Img[i][j] -
(double)m_Res[i][j])*((double)Org_Img[i][j] - (double)m_Res[i][j]);
                }
        }
        mse = err/(256.0*256.0);
        psnr = 10.0*log10((256.0*256.0)/mse);
}


void CIMEPDoc::DoubleLoad()
{
        CFileDialog dlg(true);
        CFileDialog dlg2(true);
```

```
        if(dlg.DoModal () == IDOK)
        {
                CFile file(dlg.GetFileName (), CFile::modeRead);
                file.Read (m_Org, sizeof(m_Org));
                file.Close ();
        }

                if(dlg2.DoModal () == IDOK)
        {
                CFile file(dlg2.GetFileName (), CFile::modeRead);
                file.Read (Org_Img, sizeof(Org_Img));
                file.Close ();
        }
}
void CIMEPDoc::Small()
{
        InitImage();

        int i = 0, j = 0;

        for(int y = 0; y < 128; y++)
        {
                for (int x = 0; x < 128; x++)
                {
                        i = int(y * scale_y2);
                        j = int(x * scale_x2);

                        m_small[i][j] = m_Org[y][x];
                }
        }
}
void CIMEPDoc::Large()
{
        InitImage();

        int i = 0, j = 0;

        for(int y = 0; y < 64; y++)
        {
                for (int x = 0; x < 64; x++)
                {
                        i = int(y * scale_y);
                        j = int(x * scale_x);

                        m_Res[i][j] = m_small[y][x];
                }
        }

}
//B스플라인
void CIMEPDoc::ZoomIn4()
{
        int x=0, y=0;
        int max=256;

        //영상확대
        Small();
```

```
        for(x=0; y <64; x++){
                for(y=0;y < 64; y++){
                        m_Res[x*2][y*2] = m_small[x][y];
                        m_Res[x*2+1][y*2] = m_small[x][y];
                        m_Res[x*2][y*2+1] = m_small[x][y];
                        m_Res[x*2+1][y*2+1] = m_small[x][y];
                        if(y > 1){
                                m_Res[x*2][y*2] = (m_Res[x*2][y*2-1] +
m_Res[x*2][y*2+1])/2;
                                m_Res[x*2+1][y*2] = (m_Res[x*2+1][y*2-1] +
m_Res[x*2+1][y*2+1])/2;
                        }
                }
        }
        PSNR();
}
//제안보간법
void CIMEPDoc::ZoomIn3()
{
        int x=0,y=0;
        int max=256;

        Small();
        // 영상 확대

        ZoomIn0();
        for(x =0; x < max; x+=2)
        {
                for(int y=0; y < max; y+=2)
                {
                        m_Res[x+1][y] = m_Res[x][y];
                        m_Res[x][y+1] = m_Res[x][y];
                        m_Res[x+1][y+1] = m_Res[x][y];
                }
                for(y = 1; y < max-1; y+=2)
                {
                        m_Res[x][y] =  (m_Res[x][y-1]+m_Res[x][y+1])/2;
                        m_Res[x+1][y] = (m_Res[x+1][y-1]+m_Res[x+1][y+1])/2;
                }
        }
        PSNR();
}
```

```
// IMEPView.h : interface of the CIMEPView class
//
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_IMEPVIEW_H__D94B4834_15E5_4FDA_8B4B_FAA37AAAFB71__INCLUD
ED_)
#define
AFX_IMEPVIEW_H__D94B4834_15E5_4FDA_8B4B_FAA37AAAFB71__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CIMEPView : public CView
{
protected: // create from serialization only
        CIMEPView();
        DECLARE_DYNCREATE(CIMEPView)

// Attributes
public:
        CIMEPDoc* GetDocument();
        int max;
// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CIMEPView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        protected:
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CIMEPView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CIMEPView)
        afx_msg void OnZoomin1();
        afx_msg void OnZoomin2();
        afx_msg void OnZoomin3();
        afx_msg void OnZoomin4();
```

```
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in IMEPView.cpp
inline CIMEPDoc* CIMEPView::GetDocument()
   { return (CIMEPDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_IMEPVIEW_H__D94B4834_15E5_4FDA_8B4B_FAA37AAAFB71__INCLUD
ED_)
```

감사의 글

무더운 여름이 시작된 것 같습니다.

호기심과 설레는 마음으로 대학원 생활을 시작한 지도 엊그제 같은데 벌써 2년 6개월이라는 세월이 흘러 무언가 이뤘다는 마음보다 여러 가지로 부족함을 많이 느껴 좀 더 열심히 했더라면 하는 아쉬움이 남습니다. 저의 부족함을 열성과 사랑으로 지도하여 주신 김장형 지도교수님께 먼저 진심으로 감사를 드리고, 논문 심사를 맡아 세심한 부분까지 지도를 해주신 변상용 교수님과 안기중 교수님께 깊은 감사를 드립니다. 그리고 재학기간 동안 저를 지도해 주신 이상준 교수님, 곽호영 교수님, 송왕철 교수님, 이동희 교수님께도 감사를 드립니다.

남들 보다 늦은 나이에 공부를 다시 하려니 어려움이 많았으나 항상 저에게 격려와 관심을 가져준 제주관광대학 관광정보처리계열 교수님들과 호텔경영과 교수님들께도 깊은 감사를 드리고, 지금은 몽고에 가있지만 가기 전에 논문을 작성하는데 많은 도움을 준 박사 과정인 김무영씨, 연구실에서 하얀 밤을 지새워 공부하며 프로그램의 구현에 많은 도움을 준 박사 과정인 강진석씨, 강영도씨, 박재필씨, 김정효씨, 박창희씨, 석사 과정인 변태보씨, 허동진씨, 양영수씨, 강명화씨, 홍유기씨, 박종섭씨, 이정아씨 에게도 고마운 마음을 전합니다. 또한 멀티미디어 연구실에 공부하는 모든 사람들과 저를 아는 모든 사람들에게도 감사를 드립니다.

사업과 강의, 대학원 공부를 병행하면서 많은 시간들을 가족과 함께 보낼 수 없었던 것이 못내 아쉽기는 하지만 그래도 항시 늦게 귀가하는 저를 언제나 따뜻하게 맞아주고 격려해 주었던 아내 차용숙씨와 아들 성민, 정민 에게도 고맙다는 마음을 전합니다. 이 자그마한 결실이 맺기까지 충고를 아끼지 않고 해주신 부산동명대학에 전자과 교수로 계신 강길범 형님께도 감사를 드립니다.

끝으로 무한한 희생과 사랑으로 이 아들을 걱정해 주시는 아버님과 돌아가신 어머님의 영전에 이 논문을 바칩니다.