08장 기하학적 변화

- 기하학적 변환의 개요
- 기하학적 변환의 사상 방법
- 보간법
- 영상의 스케일링 기하학 변환

한빛미디어(주)

학습목표

- ✓ 기하학 변환의 종류를 소개한다.
- ✓ 전방향 사상과 역방향 사상을 이해한다.
- ✓ 기하학 변환에서 역방향 사상의 효율성을 공부한다.
- ✓ 여러 가지 보간 기법을 공부한다.
- ✓ 영상의 확대와 축소 변환을 공부한다.

Section 01 기하학적 변환의 개요

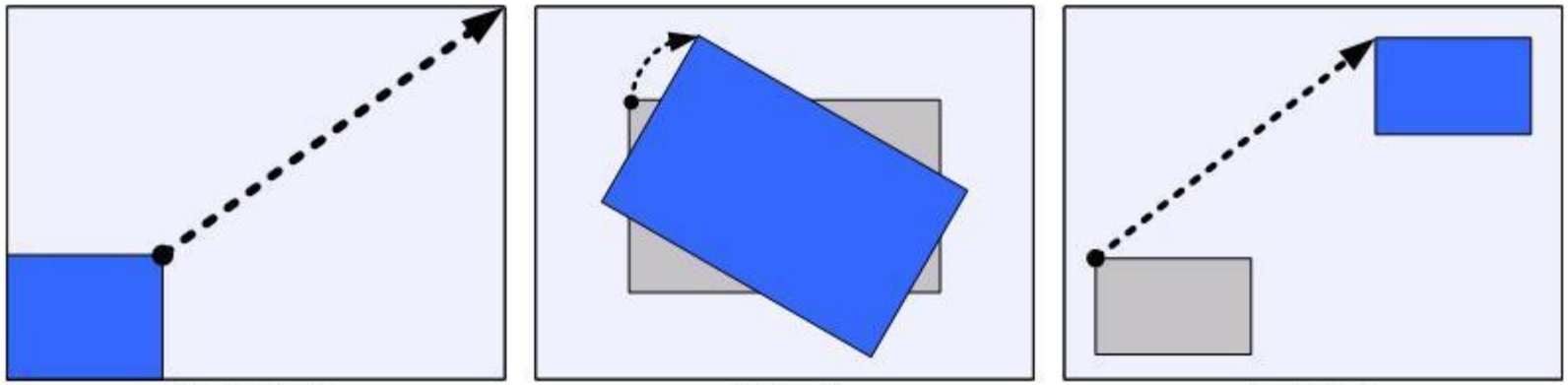
기하학적 변환

- 영상을 구성하는 화소의 공간적 위치를 재배치하는 과정

- 재배치가 되는 영상의 화소가 어떤 것이냐에 따라
  - 전방향 사상 : 하나는 입력 영상을 출력 영상으로 화소의 위치를 변환하는 과정
  - 역방향 사상 : 다른 하나는 출력 영상을 입력 영상으로 화소의 위치를 변환하는 과정

- 기본 형태에 따라
  - 선형 기하 변환 : 직선 처리처럼 선형적으로 처리하는 방법으로 평행이동(Translation), 회전(Rotation), 스케일링(Scaling) 등 화소의 재배치 수행
  - 비선형 기하 변환 : 영상을 찌그러뜨리고 구부려서 곡선으로 처리하는 방법으로, 워핑(Warping)과 모핑(Morphing)이 대표적

기하학적 변환의 개요(계속)



[그림 8-1] 선형 기하학적 변환

- 보간법
  - 영상을 확대하고 축소하는 스케일링 과정은 화소가 값을 할당 받지 못할 때 발생할 때 빈 화소에 값을 할당하는 과정

기하학적 변환의 개요(계속)

(a) x = y = 1/2

(b) x = y = 1

(c) x = y = 2

[그림 8-2] 스케일링 영상

(a) 입력 영상

(b) 출력 영상

[그림 8-3] 회전 영상

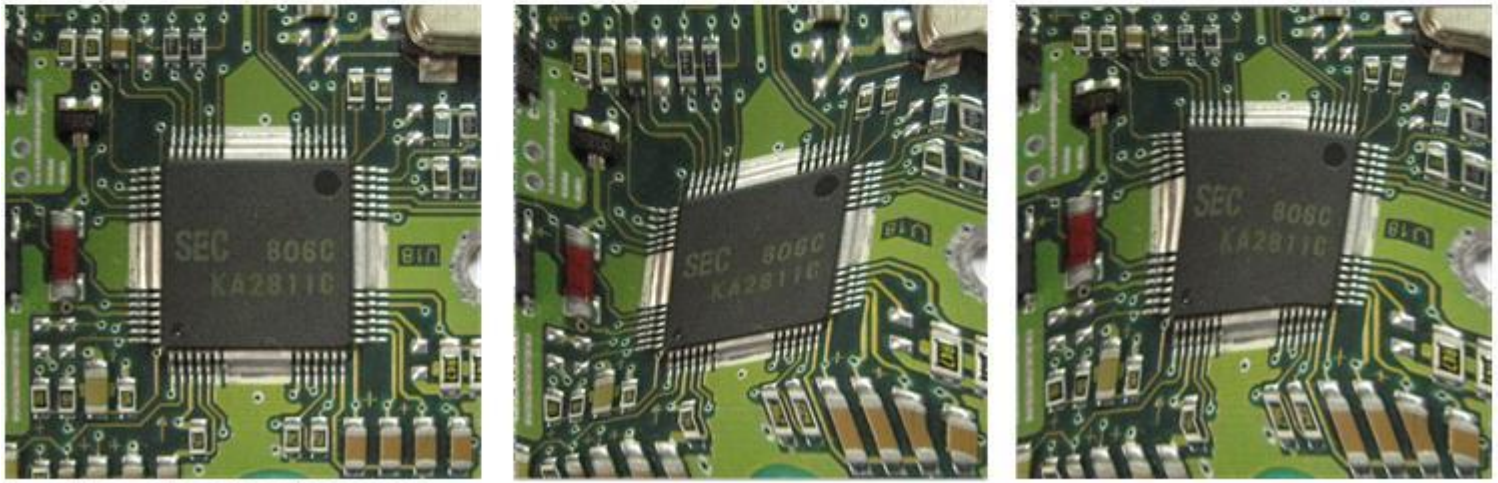
기하학적 변환의 개요(계속)

[그림 8-4] 이동 영상

기하학적 변환의 개요(계속)

(a) 입력 영상　(b) 좌우 대칭 영상　(c) 상하 대칭 영상

[그림 8-5] 대칭 영상



[그림 8-6] 워핑 처리 영상

기하학적 변환의 개요(계속)

(a) 입력 영상

(b) 중간 영상 단계 1

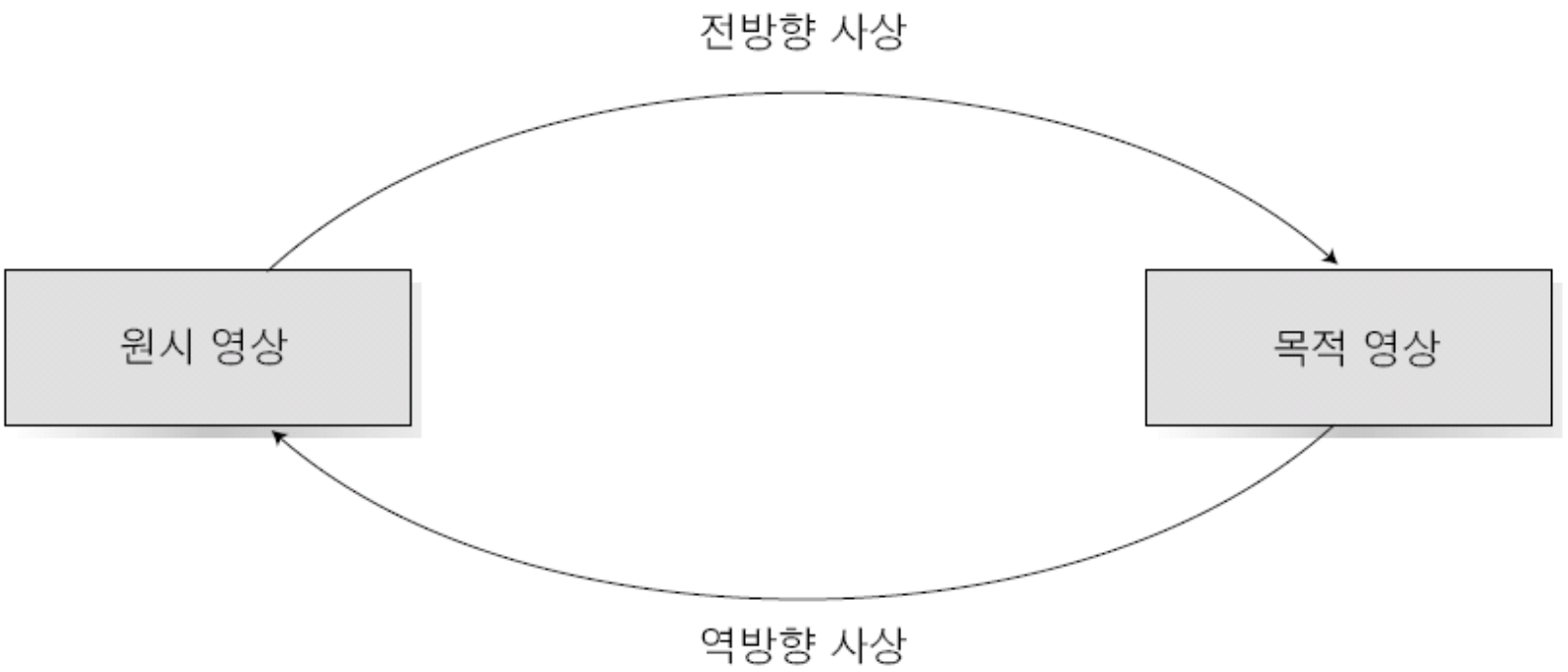
(c) 중간 영상 단계 2

(d) 중간 영상 단계 3

(e) 결과 영상

[그림 8-7] 모핑의 단계별 영상

- 사상(Mapping)
  - 주어진 조건에서 현재의 데이터를 원하는 목표로 만드는 것
    - 원시 영상의 화소가 목적 영상의 화소 위치로 이동(Shift)하면, 원시 영상 화소가 목적 영상의 화소로 대응되는 것
    - 변환(Transformation), 정합(Matching)이라는 뜻도 있음.



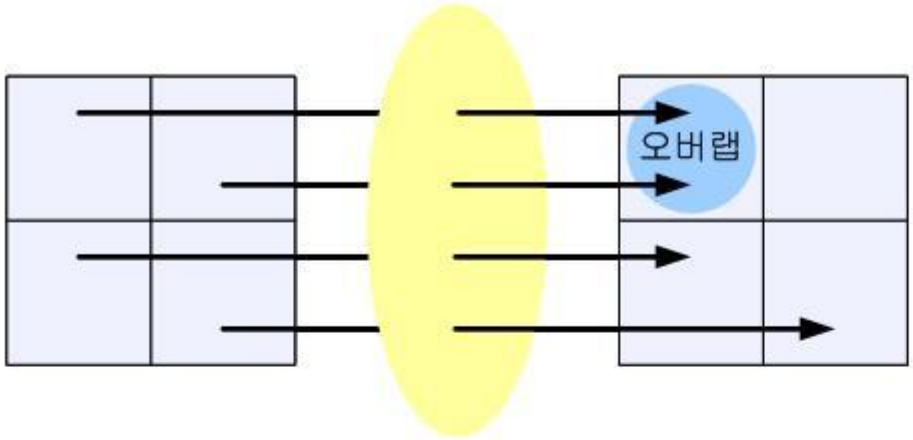
[그림 8-8] 전방향과 역방향 사상의 개념

전방향 사상

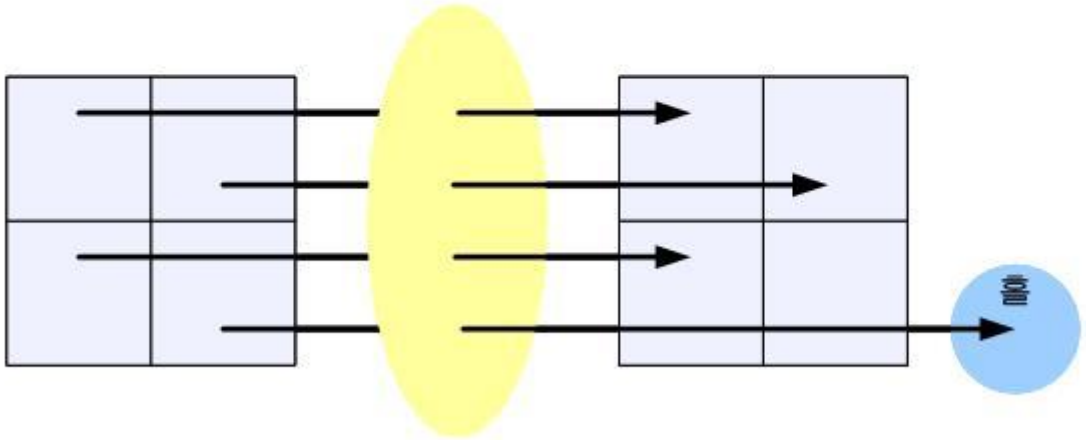
- 개념
  - 입력 영상의 모든 화소에서 출력 영상의 새로운 화소 위치를 계산하고, 입력 화소의 밝기 값을 출력 영상의 새로운 위치에 복사하는 방법

- 문제점
  - 오버랩(Overlap) 문제: 서로 다른 입력 화소 두 개가 같은 출력 화소에 사상되는 것
  - 홀(Hole) 문제: 입력 영상에서 임의의 화소가 목적 영상의 화소에 사상되지 않을 때

전방향 사상(계속)



(a) 오버랩(overlap) 문제

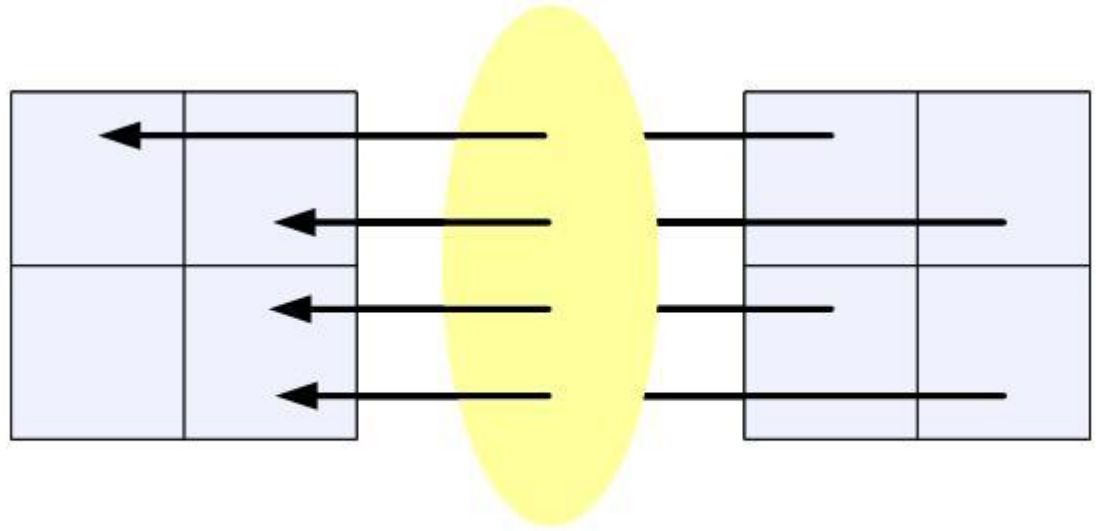


(b) 홀(hole) 문제

[그림 8-9] 전방향 사상에서의 오버랩과 홀 문제

역방향 사상

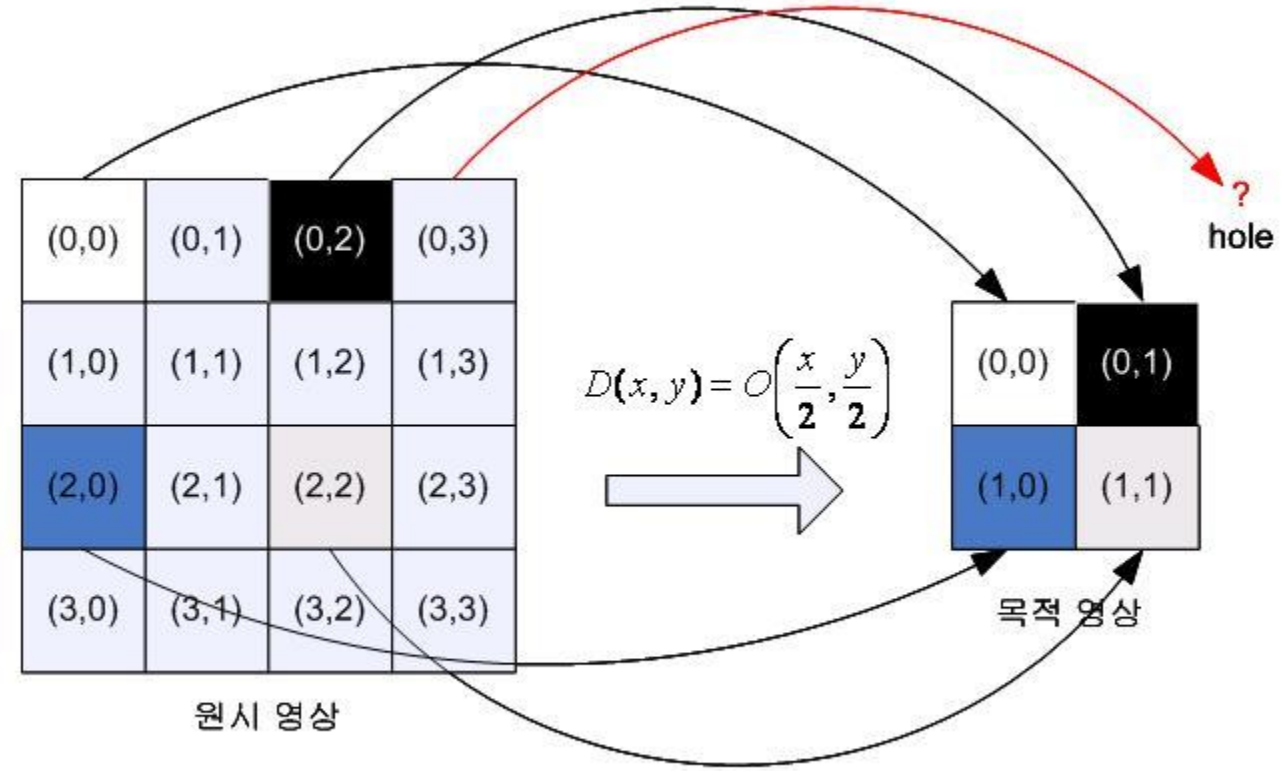
- 전방향 사상과는 반대되는 개념으로 목적 영상에서 원시 영상의 화소 값을 찾는 것
  - 목적 영상의 화소를 조사하여 몇 가지 역변환으로 원시 영상의 화소를 구한 뒤 목적 영상의 화소 값을 생성하려고 사용



[그림 8-10] 역방향 사상의 동작

기하학 처리를 위한 전방향 사상과 역방향 사상 비교

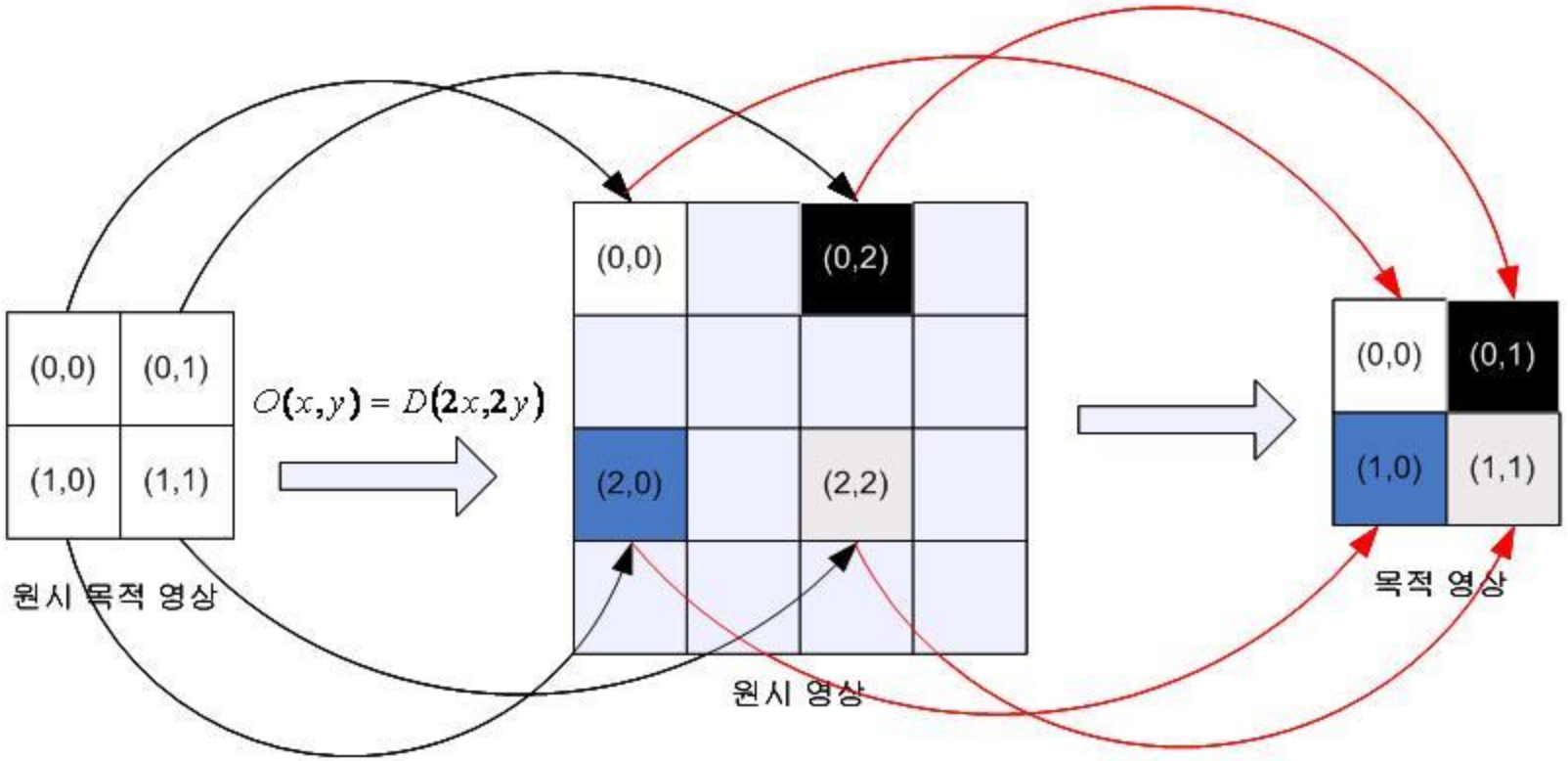
- 전방향 사상에서는 원시 영상의 좌표가 홀수면 목적 영상의 좌표 값에 소수점이 들어 있어 해당 좌표가 없게 되므로 홀 문제가 발생



[그림 8-11] 전방향 사상으로 생성된 목적 영상

기하학 처리를 위한 전방향 사상과 역방향 사상 비교(계속)

- 역함수로 원시 영상의 좌표 값을 계산하고, 그 좌표에 해당하는 화소 값을 찾아서 목적 영상에 할당
- 전방향 사상에서 발생했던 홀 문제가 일어나지 않음.



[그림 8-12] 역방향 사상으로 생성된 목적 영상

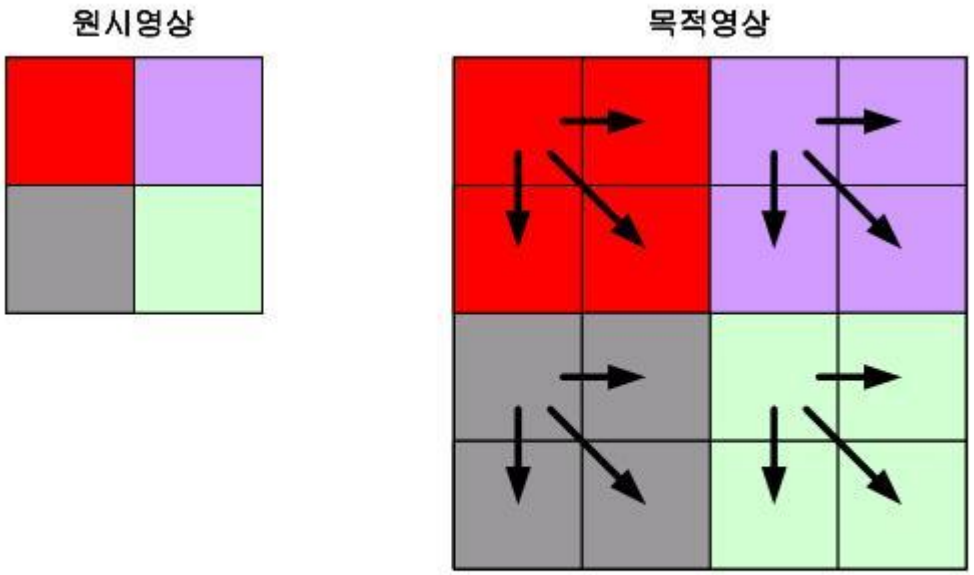
Section 03 보간법

- 보간법(Interpolation)의 개념
  - 화소 값을 할당받지 못한 목적 영상의 품질은 아주 좋지 못한데, 빈 화소에 값을 할당하여 좋은 품질의 영상을 만드는 방법
  - 화소의 값을 할당받지 못한 채 목적 영상을 만드는 대표적인 기하학 처리가 바로 영상의 확대

- 대표적인 보간법
  - 가장 인접한 이웃 화소 보간법(Nearest Neighbor Interpolation)
  - 양선형 보간법(Bilinear Interpolation)
  - 3차 회선 보간법(Cubic Convolution Interpolation)
  - B-스플라인 보간법(B-Spline Interpolation)

가장 인접한 이웃 화소 보간법(Nearest Neighbor Interpolation)

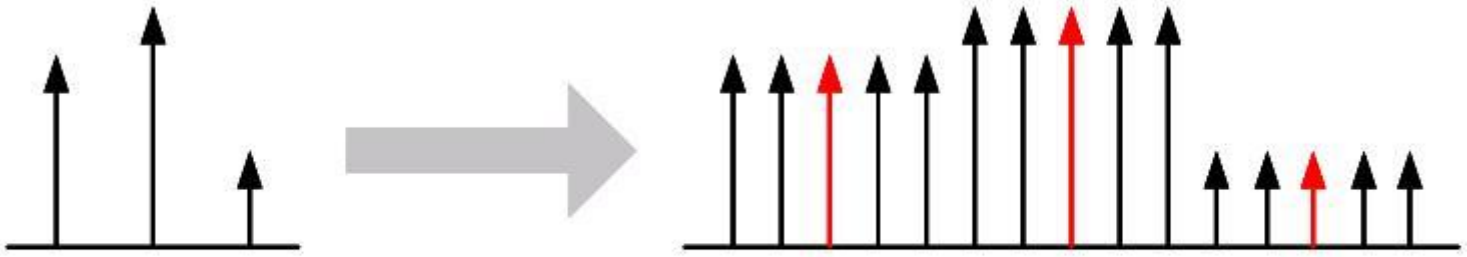
- 값을 할당받지 못한 목적 영상의 화소에서 가장 가깝게 이웃한 원시 화소의 값을 할당받은 목적 영상의 화소 값을 복사해서 사용하는 것



[그림 8-14] 가장 인접한 이웃 화소 보간법의 동작

가장 인접한 이웃 화소 보간법(계속)

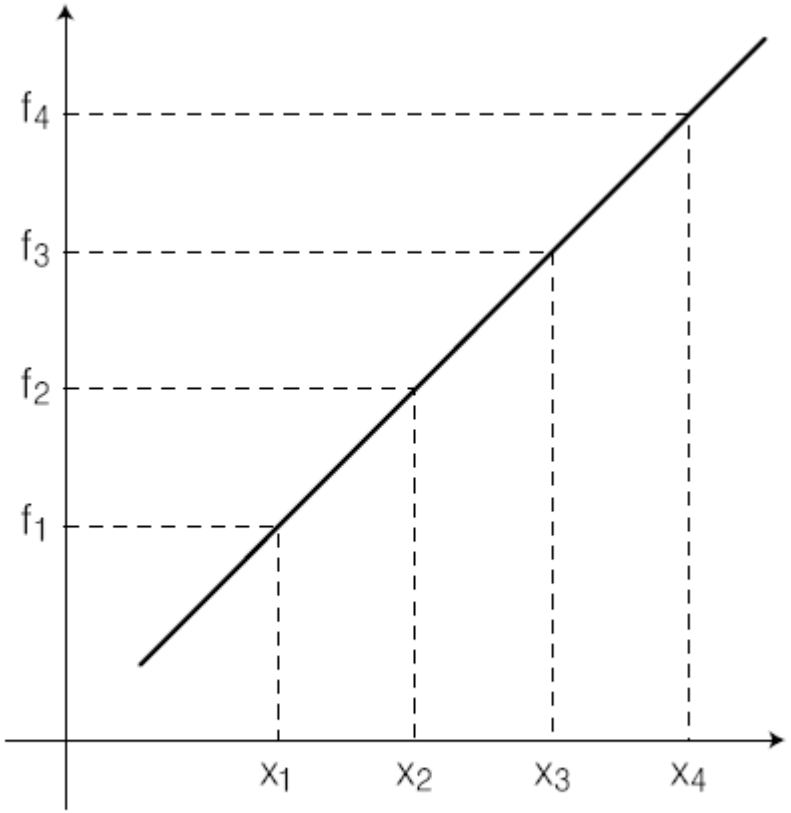
- 단순히 이웃 화소를 복사하여 사용하므로 처리 속도가 빠름
- 새로운 화소 값을 계산하지 않고 입력 화소 내에서만 찾기 때문에 원래의 영상과 전혀 다른 영상을 출력하는 오류가 발생
  - 하나의 입력 화소에 대응하는 출력 화소 수가 많을수록 영상의 질은 떨어지며, 영상 내에 톱니 모양이라고 하는 시각적인 뭉툭함(Blockiness)이 발생



[그림 8-16] 4배 보간된 영상에서 영상 품질 저하

선형 보간법 (Linear Interpolation)

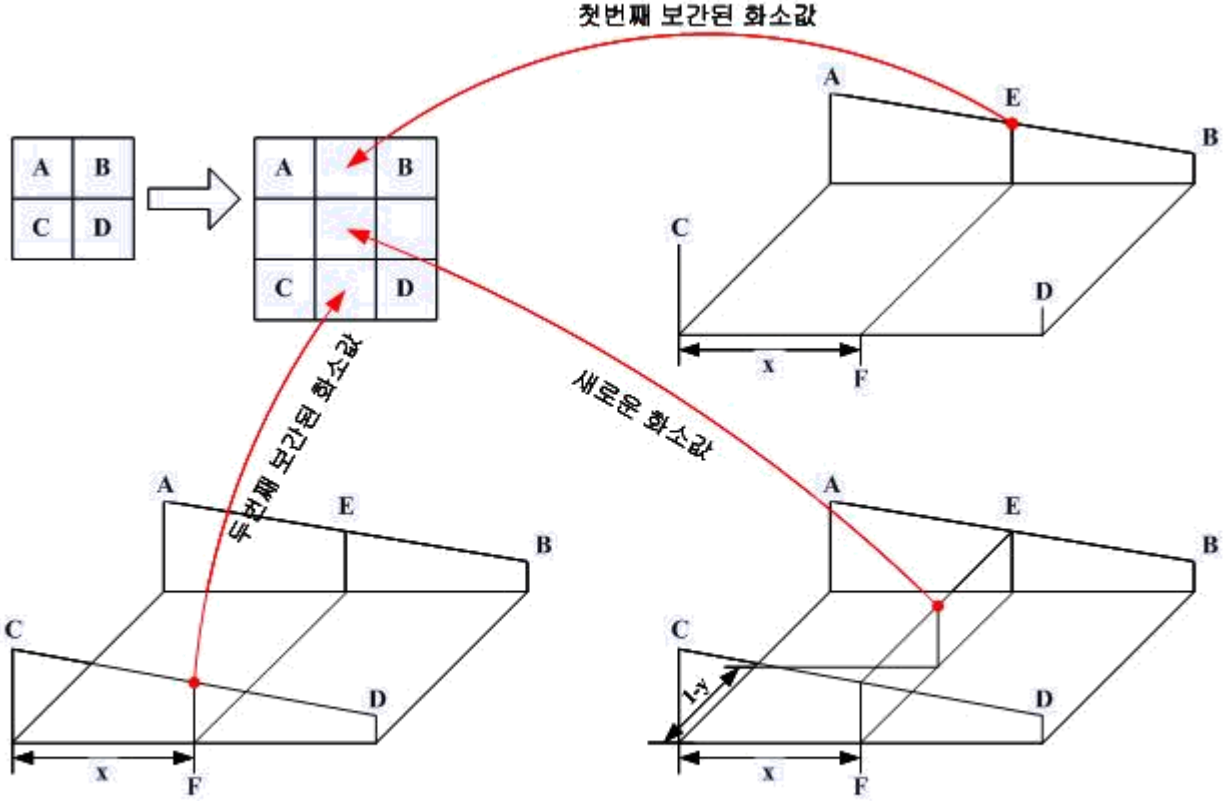
- 원시 영상의 화소 값 두 개를 이용하여 원하는 좌표에서 새로운 화소 값을 계산하는 간단한 방법



[그림 8-17] 선형 보간법의 원리

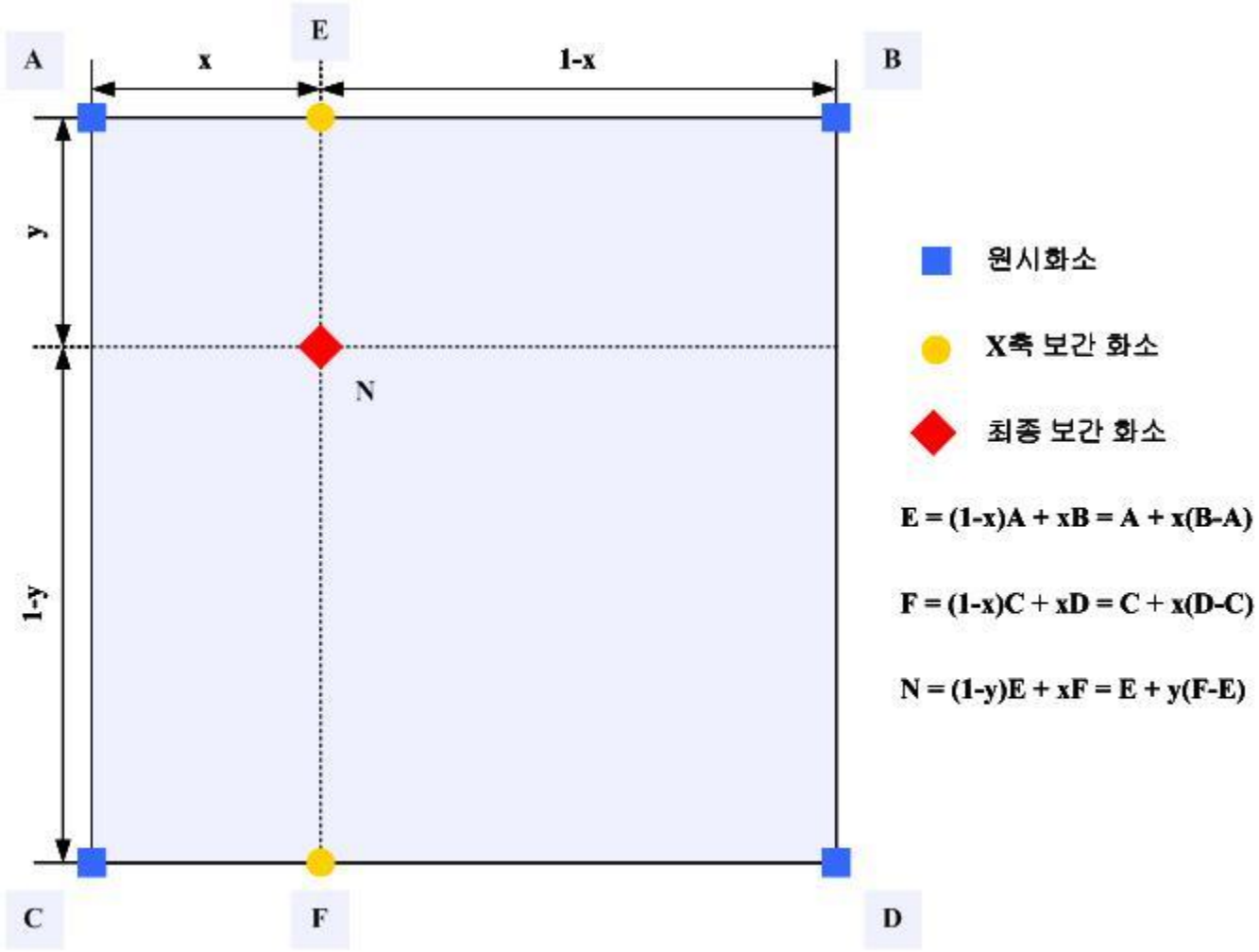
선형 보간법(계속)

- 양선형 보간법(Bilinear Interpolation)
  - 선형 보간을 바탕으로 수행
  - 화소당 선형 보간을 세 번 수행하며, 새롭게 생성된 화소는 가장 가까운 화소 네 개에 가중치를 곱한 값을 합해서 얻음.
    - 각 가중치는 각 화소에서의 거리에 정비례하도록 선형적으로 선택



[그림 8-18] 보간을 세 번 수행하는 양선형 보간법

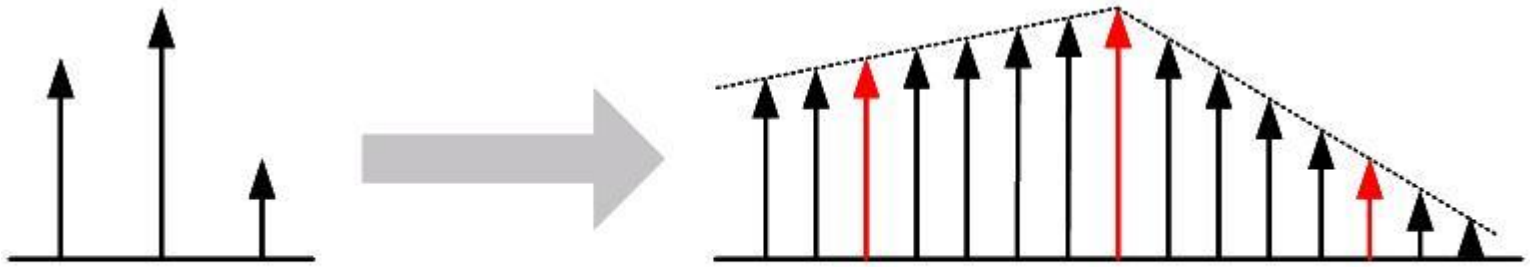
선형 보간법(계속)



[그림 8-19] 양선형 보간법으로 새로운 화소 값 설정

선형 보간법(계속)

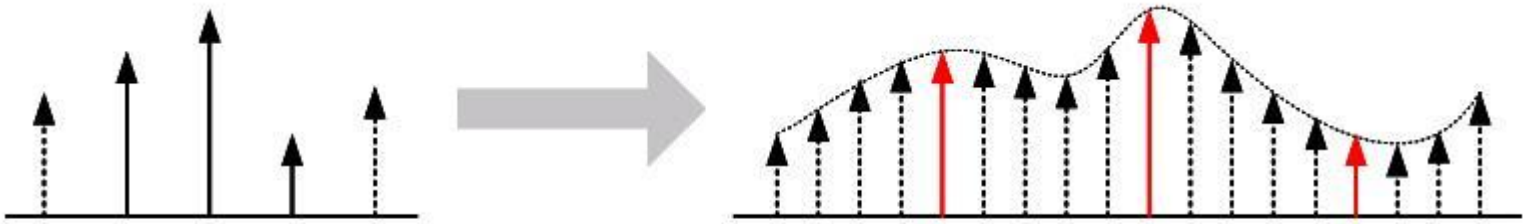
- 양선형 보간법의 장단점
  - 장점: 가장 인접한 화소 보간법에 비해 더 스무딩한 영상을 출력함.
  - 단점: 화소당 선형 보간을 세 번씩 수행해야 하므로 상당히 많은 계산량이 소모됨.



[그림 8-20] 4배 확대된 양선형 보간

고차 보간법

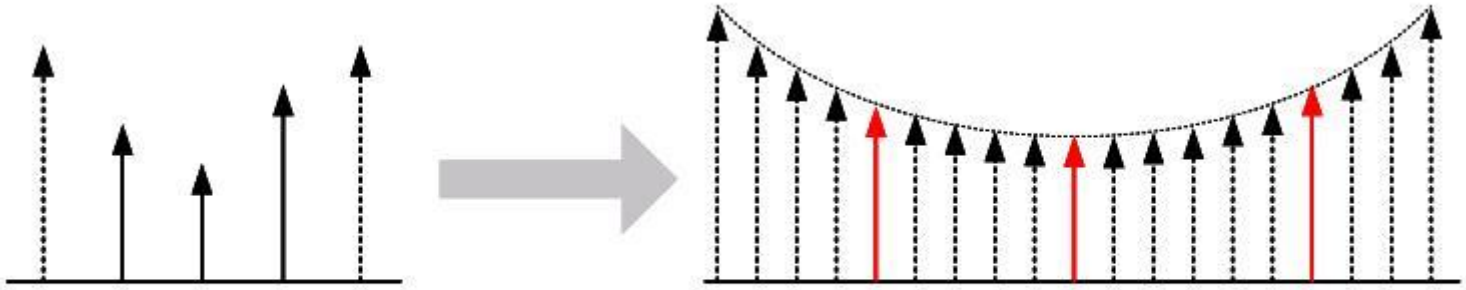
- 더 많은 이웃 화소를 참조하므로 값을 할당받지 못한 화소 값을 쉽게 추정할 수 있음.
- 3차 회선과 B-스플라인이 대표적
- 3차원 회선 보간법
  - 4×4의 이웃 화소를 참조하여 보간을 수행.
  - 양선형 보간법보다 더 많은 화소를 참조하므로 보간된 영상의 품질도 더 좋음.
  - 이웃 화소를 16개 참조하므로 계산 시간이 더 소요됨.



[그림 8-21] 3차원 회선 보간법으로 보간

고차 보간법(계속)

- **B-스플라인 보간법**
  - 이상적인 보간 함수는 저주파 통과 필터인데, B-스플라인 함수는 그중에서도 상당히 좋은 저주파 통과 필터
    → B-스플라인 함수는 보간 함수 중에서 가장 스무딩한 영상 출력



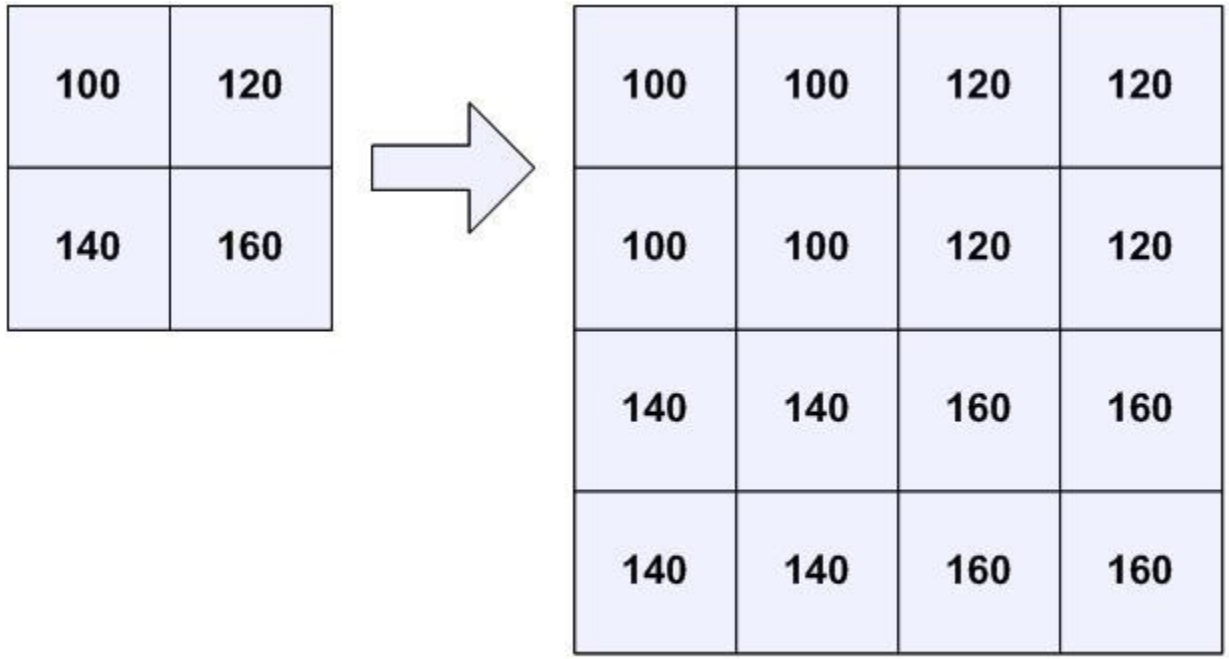
[그림 8-22] B-스플라인 보간법으로 보간

Section 04 영상의 스케일링 기하학적 변환

- 스케일링(Scaling)
  - 디지털 영상의 모양은 변화시키지 않은 채 크기만을 확대하거나 축소하는 변환
    - 영상을 확대하는 것을 확대(Magnification), 스케일링 업(Scaling Up), 줌(Zooming), 업 샘플링(Up Sampling)이라고 하며,
    - 영상을 축소하는 것을 축소(Minification), 스케일링 다운(Scaling Dawn), 데시메이션(Decimation), 다운 샘플링(Dawn Sampling)이라 함.
  - 스케일링 변환은 원 영상의 해상도를 떨어뜨리는 특징이 있어 결과 영상의 품질은 당연히 떨어짐.

영상의 확대 스케일링 변환

- 가장 인접한 이웃 화소 보간법을 이용한 영상 확대
  - 확대 배율만큼 화소를 반복적으로 복사해서 사용하므로 쉽고 빠르게 확대와 보간이 수행됨.



[그림 8-24] 영상을 2배 확대하여 가장 인접한 이웃 화소로 보간

[실습하기 8-1] 가장 인접한 이웃 화소 보간법으로 영상 확대 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_NEAREST
Caption	인접한 이웃 화소 보간법

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 인접한 이웃 화소 보간법을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnNearest
Doc Class	void	OnNearest

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 8-1] 가장 인접한 이웃 화소 보간법으로 영상 확대 프로그램

```
void CImageProcessingDoc::OnNearest()
{
   int i,j;
   int ZoomRate = 2; // 영상 확대 배율
   double **tempArray;

   m_Re_height = int(ZoomRate*m_height); // 확대된 영상의 높이
   m_Re_width = int(ZoomRate*m_width); // 확대된 영상의 너비
   m_Re_size = m_Re_height * m_Re_width;

   m_tempImage = Image2DMem(m_height, m_width);
   tempArray = Image2DMem(m_Re_height, m_Re_width);

   m_OutputImage = new unsigned char [m_Re_size];

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }

   for(i=0 ; i< m_Re_height ; i++){
      for(j=0 ; j< m_Re_width ; j++){
         tempArray[i][j] = m_tempImage[i/ZoomRate][j/ZoomRate];
         // 이웃한 화소를 이용한 보간
      }
   }

   for(i=0 ; i< m_Re_height ; i++){
      for(j=0 ; j< m_Re_width ; j++){
         m_OutputImage[i* m_Re_width + j] = (unsigned char)tempArray[i][j];
      }
   }
}
```

[실습하기 8-1] 가장 인접한 이웃 화소 보간법으로 영상 확대 프로그램

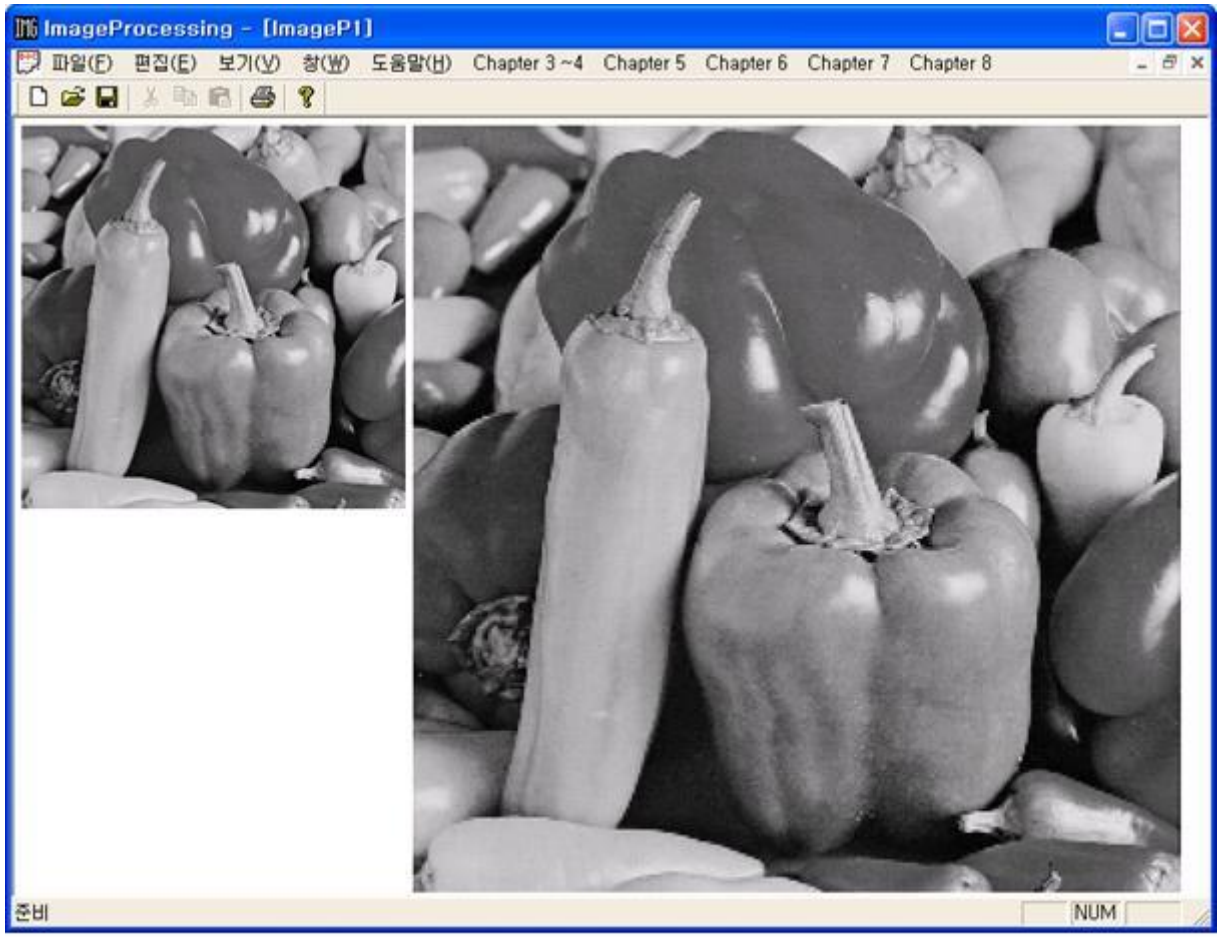
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnNearest()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnNearest();

    Invalidate(TRUE);
}
```

[실습하기 8-1] 가장 인접한 이웃 화소 보간법으로 영상 확대 프로그램

⑤ 프로그램 실행 결과 영상

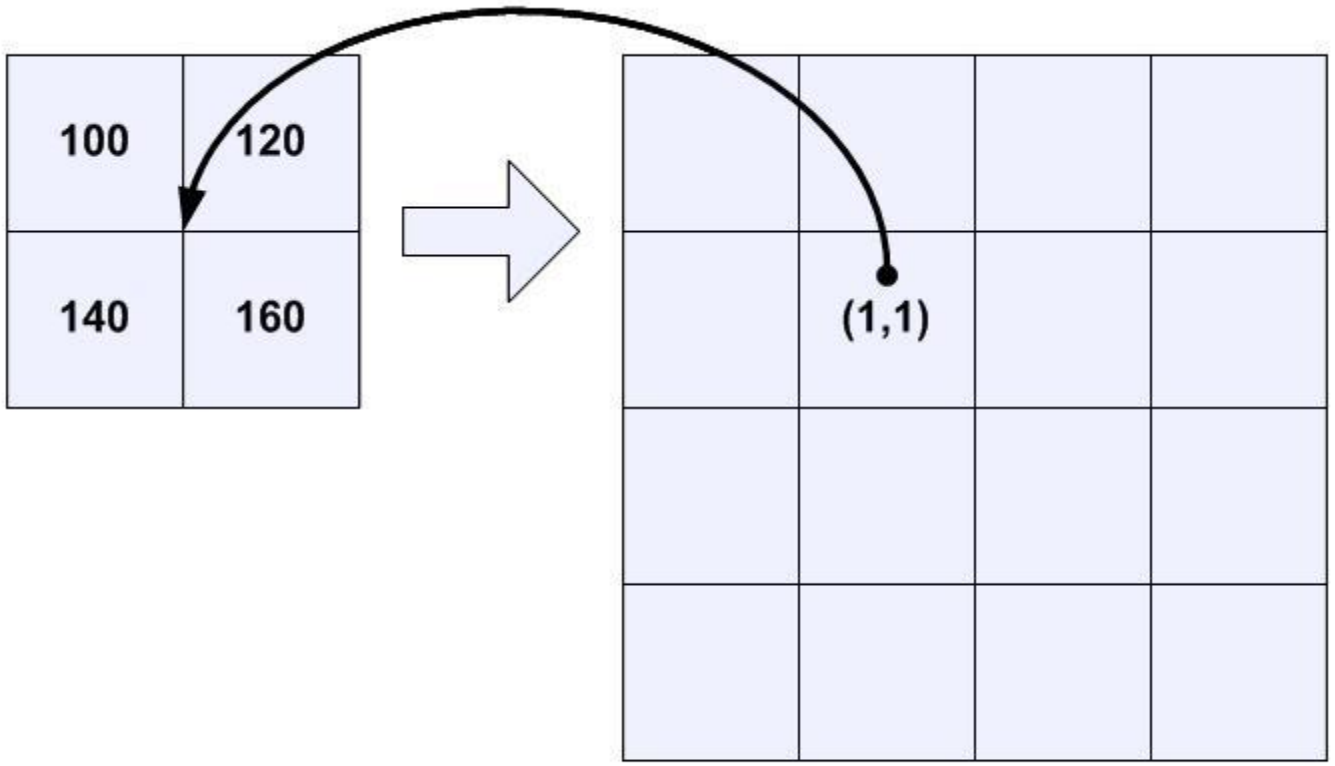
- 원시 영상을 2배로 확대한 뒤 빈 화소에 보간을 수행한 영상이라서 부드럽지 못하고 톱니 모양의 뭉툭함이 나타남.



가장 인접한 이웃 화소 보간법을 사용한 보간 결과 영상

영상의 확대 스케일링 변환(계속)

- 양선형 화소 보간법을 이용한 영상 확대



[그림 8-25] 양선형 보간법을 실제로 적용한 예

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_BILINEAR
Caption	양선형 보간법

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 양선형 보간법을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnBilinear
Doc Class	void	OnBilinear

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

```
void CImageProcessingDoc::OnBilinear()
{
   int i, j, point, i_H, i_W;
   unsigned char newValue;
   double ZoomRate = 2.0, r_H, r_W, s_H, s_W;
   double C1, C2, C3, C4;

   m_Re_height = (int)(m_height * ZoomRate); // 확대된 영상의 높이
   m_Re_width = (int)(m_width * ZoomRate); // 확대된 영상의 너비
   m_Re_size = m_Re_height * m_Re_width;

   m_tempImage = Image2DMem(m_height, m_width);
   m_OutputImage = new unsigned char [m_Re_size];

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }
```

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

```
for(i=0 ; i< m_Re_height ; i++){
   for(j=0 ; j< m_Re_width ; j++){
      r_H = i / ZoomRate;
      r_W = j / ZoomRate;

      i_H = (int)floor(r_H);
      i_W = (int)floor(r_W);

      s_H = r_H - i_H;
      s_W = r_W - i_W;

      if(i_H < 0 || i_H >= (m_height-1) || i_W < 0
         || i_W >= (m_width-1))
      {
         point = i* m_Re_width + j;
         m_OutputImage[point] = 255;
      }
```

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

```
            else
            {
                C1 = (double)m_tempImage[i_H][i_W];
                C2 = (double)m_tempImage[i_H][i_W+1];
                C3 = (double)m_tempImage[i_H+1][i_W+1];
                C4 = (double)m_tempImage[i_H+1][i_W];

                newValue = (unsigned char)(C1*(1-s_H)*(1-s_W)
                    +C2*s_W*(1-s_H)+C3*s_W*s_H+C4*(1-s_W)*s_H);
                point = i* m_Re_width+j;
                m_OutputImage[point] = newValue;
            }
        }
    }
}
```

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

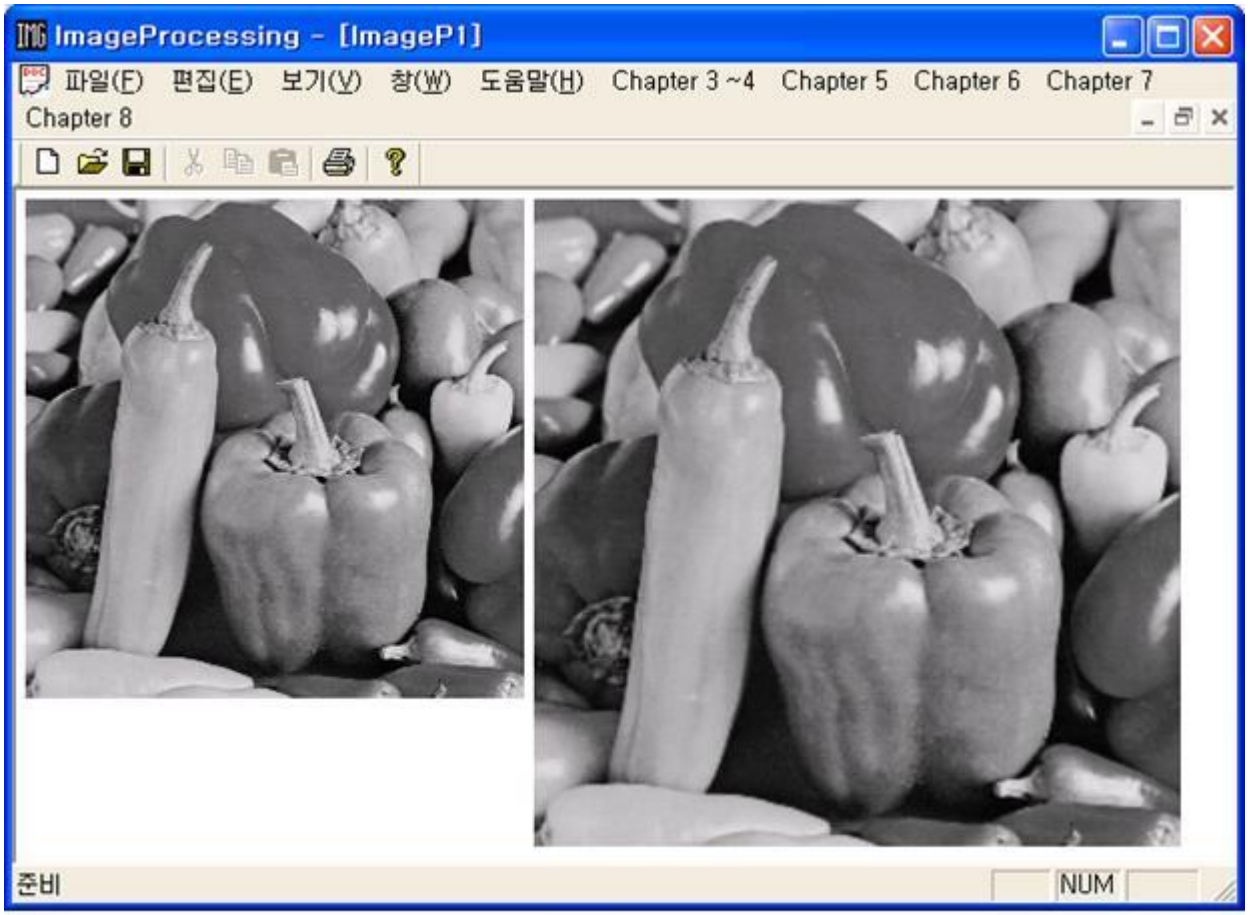
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnBilinear()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnBilinear();

    Invalidate(TRUE);
}
```

[실습하기 8-2] 양선형 보간법 이용한 영상 확대 프로그램

⑤ 프로그램 실행 결과 영상

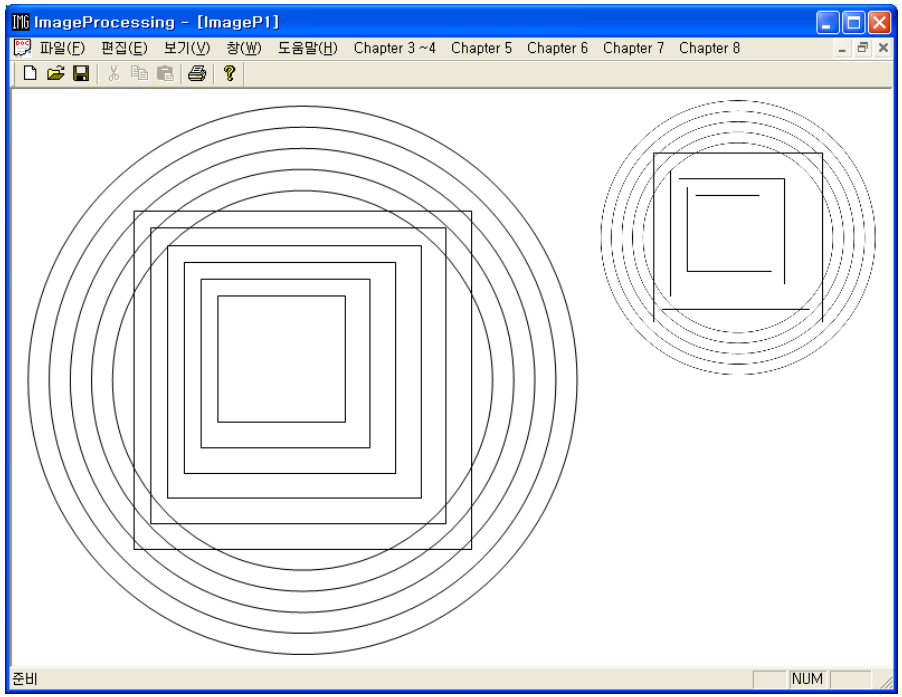
- 원시 영상을 2배로 확대한뒤 빈 화소에는 양선형 보간법을 적용한 것
- 결과 영상이 가장 인접한 이웃 화소 보간법의 결과보다 훨씬 부드러워짐.



양선형 보간이 수행된 결과 영상

영상의 축소 스케일링 변환

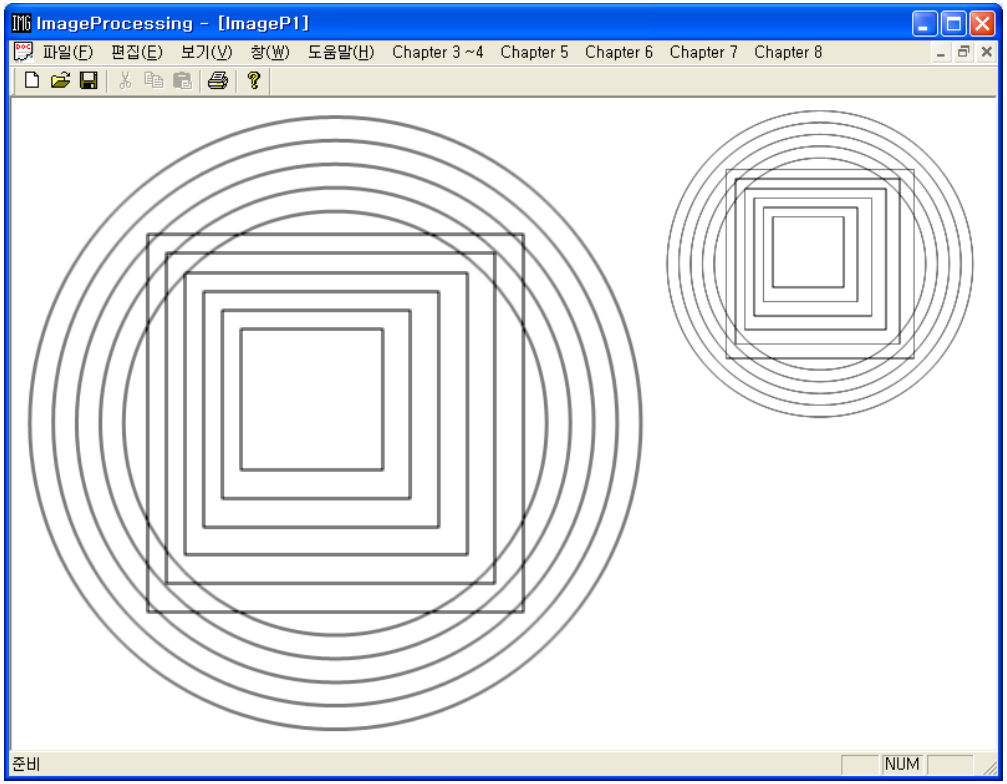
- 영상의 크기를 줄이는 영상의 축소 스케일링 변환식은 다음과 같이 확대와는 반대 개념
- 에일리어싱(Aliasing): 영상의 크기를 많이 축소하려고 너무 낮은 비율로 샘플링을 수행하면 화소 수를 너무 적게 취하게 되어 영상의 세부 내용을 상실하게 되는 현상



[그림 8-26] 서브 샘플링으로 세밀한 정보가 손실된 결과 영상

영상의 축소 스케일링 변환(계속)

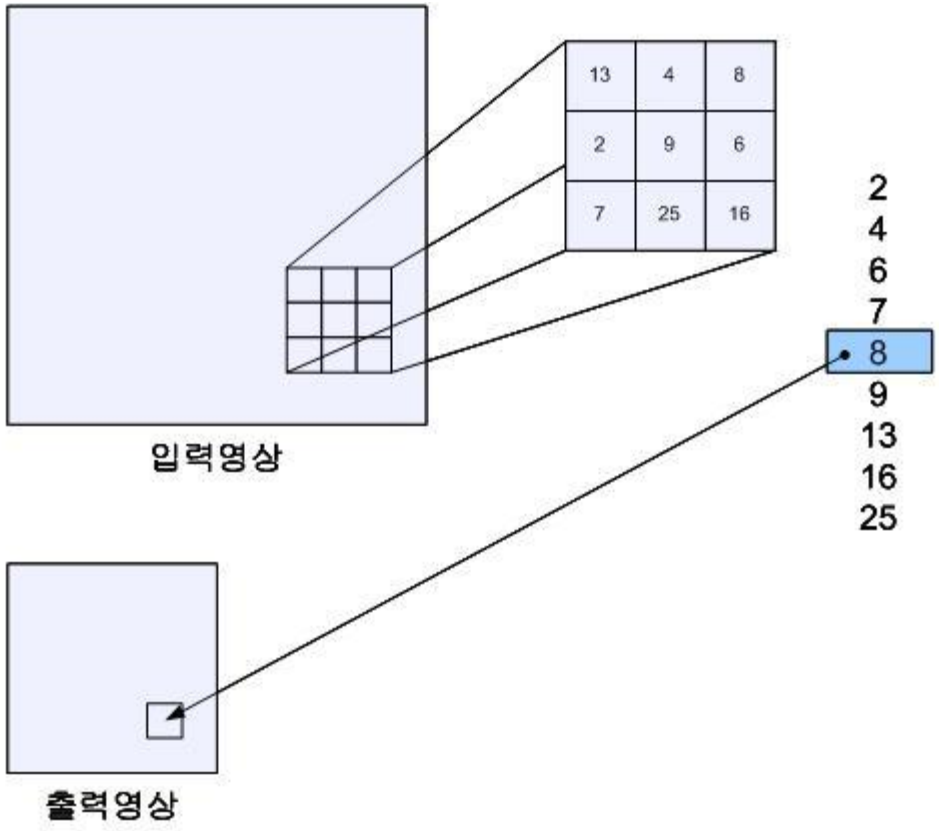
- 서브 샘플링 과정에서 세부 내용을 잃어버리는 문제를 해결하려면 서브 샘플링을 수행하기 전에 먼저 영상의 블러링(Blurring)을 수행하면 됨.
- 즉, 저주파 통과 필터링을 통과하여 블러링된 영상에서 서브 샘플링을 수행하면 세부 내용을 보존할 수 있음.



[그림 8-27] 블러링 처리한 뒤 서브 샘플링된 결과 영상

영상의 축소 스케일링 변환(계속)

- 미디언 표현
  - 미디언(Median) 표현은 화소 블록을 중간(Median) 값으로 대치한 뒤 이 값을 샘플링하여 축소 영상의 화소로 사용하는 방법



[그림 8-28] 미디언 표현을 이용한 서브 샘플링 동작 과정

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_MEDIAN_SUB
Caption	미디언 서브 샘플링

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 미디언 서브 샘플링을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnMedianSub
Doc Class	void	OnMedianSub
Doc Class	void	OnBubleSort(double *A, int MAX)
Doc Class	void	OnSwap(double *a, double *b)

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

❶ **OnMedianSub 함수 추가하기**

```
void CImageProcessingDoc::OnMedianSub()
{
   int i, j, n, m, M = 2, index = 0; // M = 서브 샘플링 비율
   double *Mask, Value;

   Mask = new    double [M*M]; // 마스크의 크기 결정

   m_Re_height = (m_height + 1) / M;
   m_Re_width = (m_width + 1) / M;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];
   m_tempImage = Image2DMem(m_height + 1, m_width + 1);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }
```

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

❶ **OnMedianSub 함수 추가하기(계속)**

```
    for(i=0 ; i<m_height-1 ; i=i+M){
        for(j=0 ; j<m_width-1 ; j=j+M){
            for(n=0 ; n<M ; n++){
                for(m=0 ; m<M ; m++){
                    Mask[n*M + m] = m_tempImage[i+n][j+m];
                    // 입력 영상을 블록으로 잘라 마스크 배열에 저장
                }
            }
            OnBubleSort(Mask, M*M); // 마스크에 저장된 값을 정렬
            Value = Mask[(int)(M*M/2)]; // 정렬된 값 중 가운데 값을 선택
            m_OutputImage[index] = (unsigned char)Value;
            // 가운데 값을 출력
            index++;
        }
    }
}
```

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

❷ **OnBubleSort 함수 추가하기**

```
void CImageProcessingDoc::OnBubleSort(   double * A, int MAX)
{ // 데이터의 정렬을 처리하는 함수
   int i, j;
   for(i=0 ; i<MAX ; i++){
      for(j=0 ; j<MAX-1 ; j++){
         if(A[j] > A[j+1]){
            OnSwap(&A[j], &A[j+1]);
         }
      }
   }
}
```

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

❸ **OnSwap 함수 추가하기**

```
void CImageProcessingDoc::OnSwap(double *a, double *b)
{ // 데이터 교환 함수
   double temp;
   temp = *a;
   *a = *b;
   *b = temp;
}
```

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

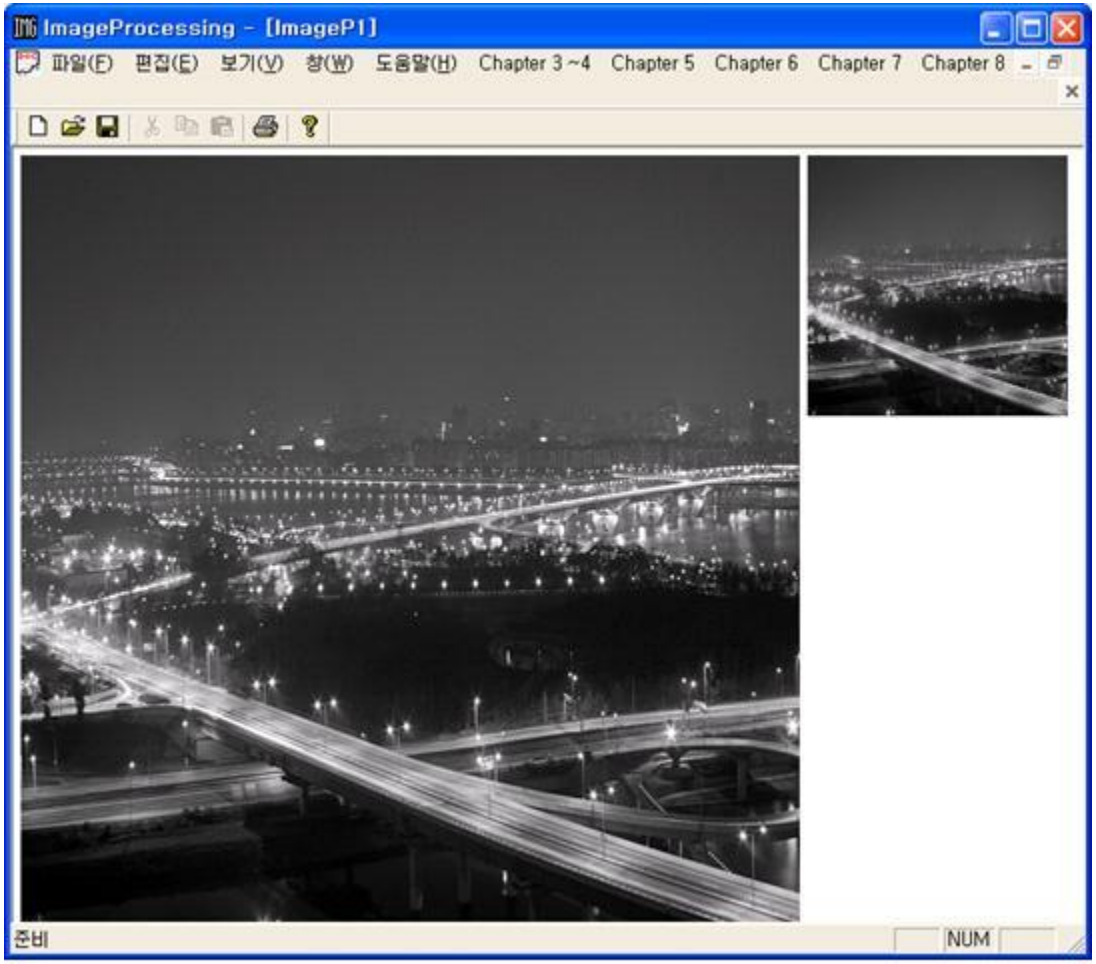
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnMedianSub()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnMedianSub();

    Invalidate(TRUE);
}
```

[실습하기 8-3] 미디언 표현을 이용한 서브 샘플링 프로그램

⑤ 프로그램 실행 결과 영상

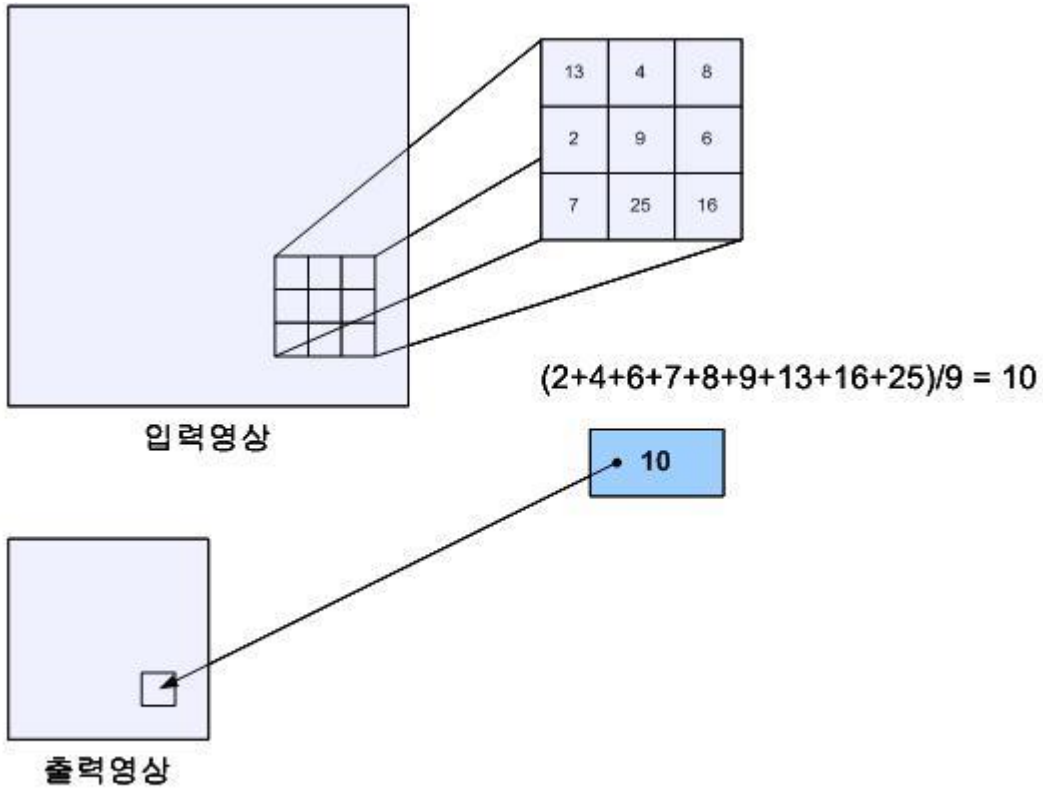
- 블러링 전처리로 서브 샘플링된 영상과는 다르게 원본 영상만큼 화질이 선명



미디언 표현을 이용해 서브 샘플링된 결과 영상

영상의 축소 스케일링 변환(계속)

- 평균 표현
  - 평균(Mean) 표현은 미디언 표현과 비슷하게 화소 블록을 블록 내 화소의 평균값으로 대치하는 방법
  - 이렇게 얻은 평균값이 해당 축소 영상의 화소 값으로 사용됨.



[그림 8-29] 평균 표현을 이용한 서브 샘플링의 동작

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_MEAN_SUB
Caption	평균 서브 샘플링

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 평균 서브 샘플링을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnMeanSub
Doc Class	void	OnMeanSub

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

```
void CImageProcessingDoc::OnMeanSub()
{
   int i, j, n, m, M = 3, index = 0, k; // M = 서브 샘플링 비율
   double *Mask, Value, Sum = 0.0;

   Mask = new    double [M*M];

   m_Re_height = (m_height + 1) / M;
   m_Re_width = (m_width + 1) / M;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];
   m_tempImage = Image2DMem(m_height + 1, m_width + 1);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }
```

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

```
    for(i=0 ; i<m_height-1 ; i=i+M){
        for(j=0 ; j<m_width-1 ; j=j+M){
            for(n=0 ; n<M ; n++){
                for(m=0 ; m<M ; m++){
                    Mask[n*M + m] = m_tempImage[i+n][j+m];
                }
            }
            for(k=0 ; k<M*M ; k++)
                Sum = Sum + Mask[k];
                // 마스크에 저장된 값을 누적
            Value = (Sum / (M*M)); // 평균을 계산
            m_OutputImage[index] = (unsigned char)Value;
            // 평균값을 출력
            index++;
            Sum = 0.0;
        }
    }
}
```

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

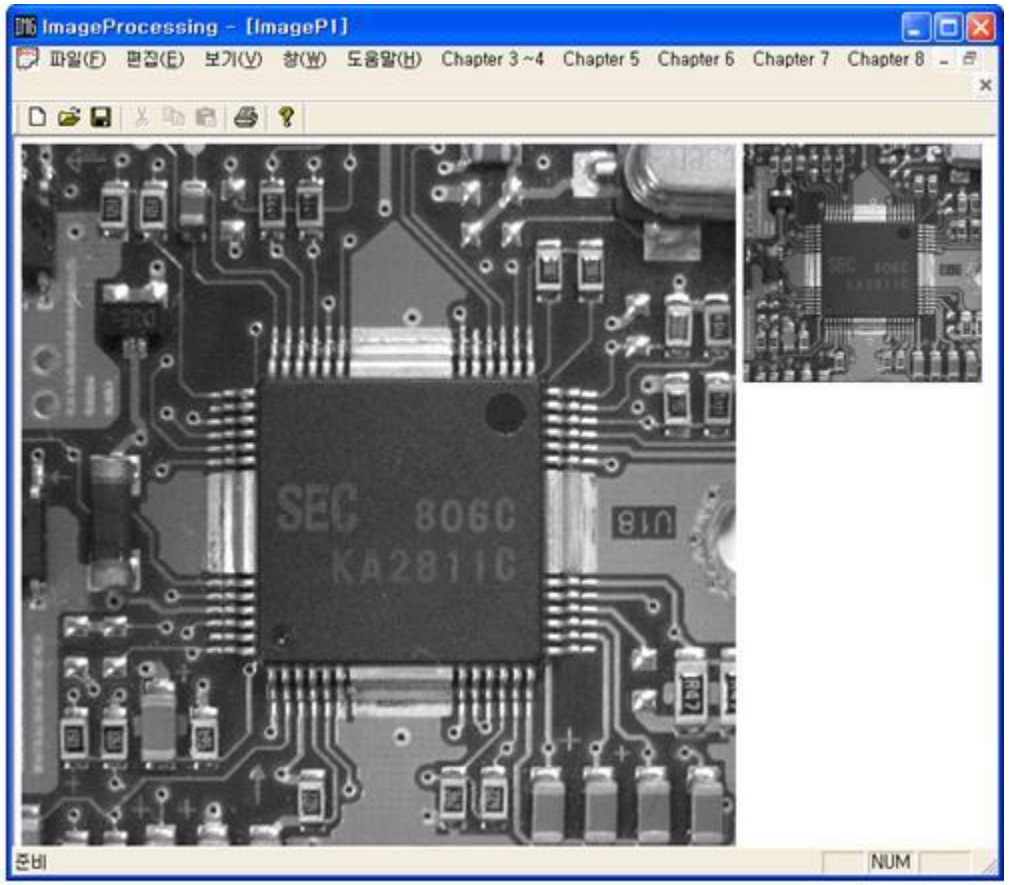
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnMeanSub()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnMeanSub();

    Invalidate(TRUE);
}
```

[실습하기 8-4] 평균 표현을 이용한 서브 샘플링 프로그램

⑤ 프로그램 실행 결과 영상

- 영상의 크기가 작아지고 스무딩해짐.



평균 표현을 이용해 서브 샘플링된 결과 영상

요약

- 기하학적 변환
  - 영상을 구성하는 화소의 공간적 위치를 임의의 기하학적 변환으로재배치하는 과정
- 선형 기하 연산
  - 곡선이 전혀 없는 영상을 대상으로 평행이동, 회전, 스케일링 등 화소의 재배치를 수행
- 비선형 기하 처리
  - 영상을 찌그러뜨리고 구부려서 곡선으로 처리하는 방법
  - 워핑 변환과 모핑 변환이 대표적
- 사상
  - 주어진 조건에서 현재의 데이터를 원하는 목표로 만드는 것

요약

- 전방향 사상
  - 입력 영상의 모든 화소에서 출력 영상의 새로운 화소 위치를 계산하고, 입력 화소의 밝기 값을 출력 영상의 새로운 위치에 복사하는 방법
  - 오버랩 문제와 홀 문제 발생
    - 오버랩 문제: 서로 다른 입력 화소 두 개를 똑같은 출력 화소에 사상하는 것. 새롭게 생성된 화소 값이 어떤 입력 화소에 근거하는지 정할 수 없으므로 처리된 결과 영상이 불명확
    - 홀 문제 : 전방향 사상에서 입력 영상의 화소가 목적 영상 내의 출력 화소에 없는 것. 입력 화소에 출력 화소 값이 배당되지 않으므로 정확한 영상처리를 할 수 없음.

- 역방향 사상
  - 목적 영상의 화소를 조사하여 몇 가지 역변환으로 원시 영상의 화소를 구한 뒤 목적 영상의 화소 값을 생성하려고 사용
  - 홀과 오버랩 문제가 발생하지 않아 기하학 처리에서 유용

요약

- 보간법
  - 화소 값을 할당받지 못해 품질이 좋지 못한 것을 방지하기 위해 빈 화소에 값을 할당하여 좋은 품질의 영상을 만드는 방법
- 가장 인접한 이웃 화소 보간법
  - 값을 할당받지 못한 목적 영상의 화소에서 가장 가깝게 이웃한 원시 화소의 값을 할당받은 목적 영상의 화소 값을 복사해서 사용하는 것
  - 원시 화소에서 계산된 좌표가 정수가 아니면 가장 가까이에 있는 유효한 화소 좌표를 선택하는 것
  - 처리 속도가 빠르나 하나의 입력 화소에 대응하는 출력 화소 수가 클수록 영상의 질은 떨어지며, 영상 내에 톱니 모양이라고 하는 시각적인 뭉툭함이 발생
- 양선형 보간법
  - 화소당 선형 보간을 세 번 수행, 새롭게 생성된 화소는 가장 가까운 화소 네 개에 가중치를 곱한 값을 합해서 얻음. 각 가중치는 각 화소에서의 거리에 정비례하도록 선형적으로 선택
  - 가장 인접한 화소 보간법보다 더 스무딩한 영상을 출력하나 화소당 선형 보간을 세 번씩 수행해야 하므로 상당히 많은 계산량이 소모됨.

요약

- 고차 보간
  - 더 많은 이웃 화소를 참조하므로 값을 할당받지 못한 화소 값을 쉽게 추정할 수 있음.
- 3차원 회선 보간법
  - 4×4의 이웃 화소를 참조하여 보간 수행
  - 양선형 보간법보다 더 많은 화소를 참조하므로 보간된 영상의 품질이 더 좋으나 이웃 화소를 16개 참조하므로 계산 시간이 더 소요됨.
- B-스플라인 보간 함수
  - 상당히 좋은 저주파 통과 필터로, 보간 함수 중에서 가장 스무딩한 영상을 출력
- 스케일링
  - 디지털 영상의 모양은 변화시키지 않은 채 크기만을 확대하거나 축소하는 변환

Thank you