06장 화소 영역 처리

- 화소 영역 처리의 개념
- 회선 처리의 원리
- 블러링
- 샤프닝

6장. 화소 영역 처리

학습목표

- ✓ 영역 처리의 개념을 이해한다.
- ✓ 회선 처리를 이용한 영역 처리를 구현하는 방법을 학습한다.
- ✓ 블러링 효과를 이해하고 프로그램을 실습한다.
- ✓ 샤프닝 효과를 이해하고 프로그램을 실습한다.

화소 영역 처리

- 화소의 원값이나 위치를 바탕으로 화소 값을 변경하는 화소의 점 처리과 달리 해당 입력 화소뿐만 아니라 그 주위의 화소 값도 함께 고려하는 공간 영역 연산
- 회선 기법(또는 컨벌루션 기법, **Convolution Technique**)으로 수행하므로, 화소의 영역 처리를 회선 처리(**Convolution Processing**) 또는 컨벌루션 처리라고 함.
- 원시 화소와 이웃한 각 화소에 가중치를 곱한 합을 출력 화소로 생성

$$Output\_pixel[x,y] = \sum_{m=(x-k)}^{x+k} \sum_{n=(y-k)}^{y+k} \left(I[m,n] \times M[m,n] \right)$$

- **Output_pixel[x, y]:** 회선 처리로 출력한 화소
- **I[m, n]:** 입력 영상의 화소
- **M[m, n]:** 입력 영상의 화소에 대응하는 가중치

화소 영역 처리의 개념(계속)

I_1	I_2	I_3
I_4	I_5	I_6
I_7	I_8	I_9

(a) 입력 영상

M_1	M_2	M_3
M_4	M_5	M_6
M_7	M_8	M_9

(b) 회선 마스크

출력 픽셀 값 :

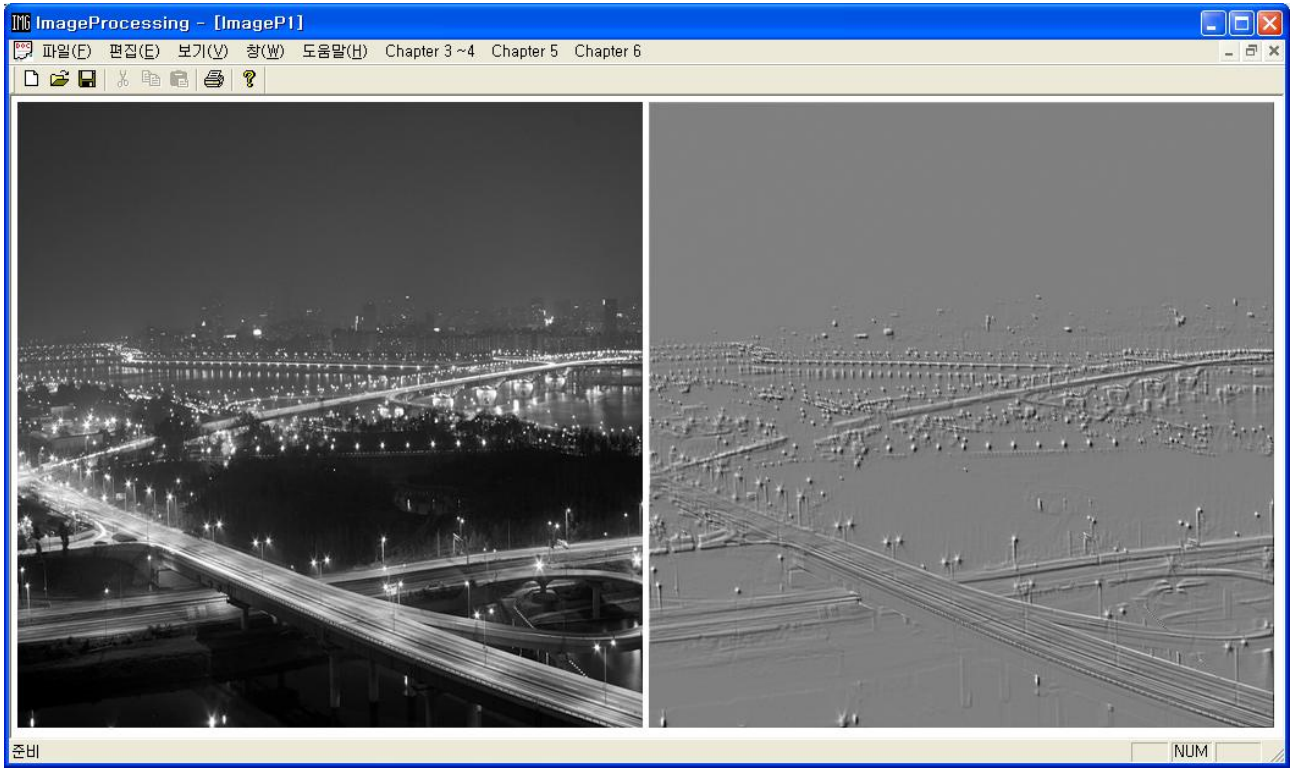
$$I_1 \times M_1 + I_2 \times M_2 + I_3 \times M_3 + I_4 \times M_4 + I_5 \times M_5 + I_6 \times M_6 + I_7 \times M_7 + I_8 \times M_8 + I_9 \times M_9$$

[그림 6-2] 회선 기법으로 출력 화소 생성

- 화소의 영역 기반 처리
  - **엠보싱(Embossing) 효과, 블러링(Blurring), 샤프닝(Sharpening), 경계선 검출(Edge Detection), 잡음 제거 등의 기술이 있음.**

엠보싱 효과

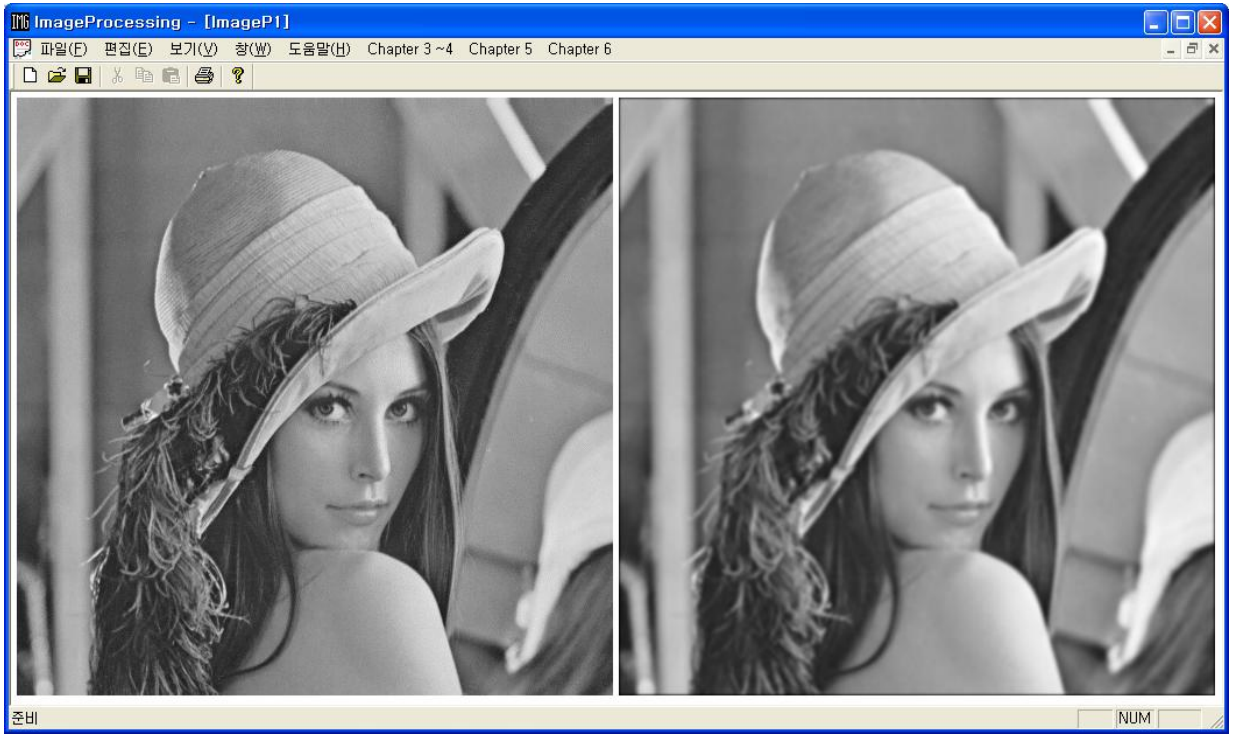
- 입력 영상을 양각 형태로 보이게 하는 기술



[그림 6-3] 입력 영상에 엠보싱 처리를 한 영상

블러링

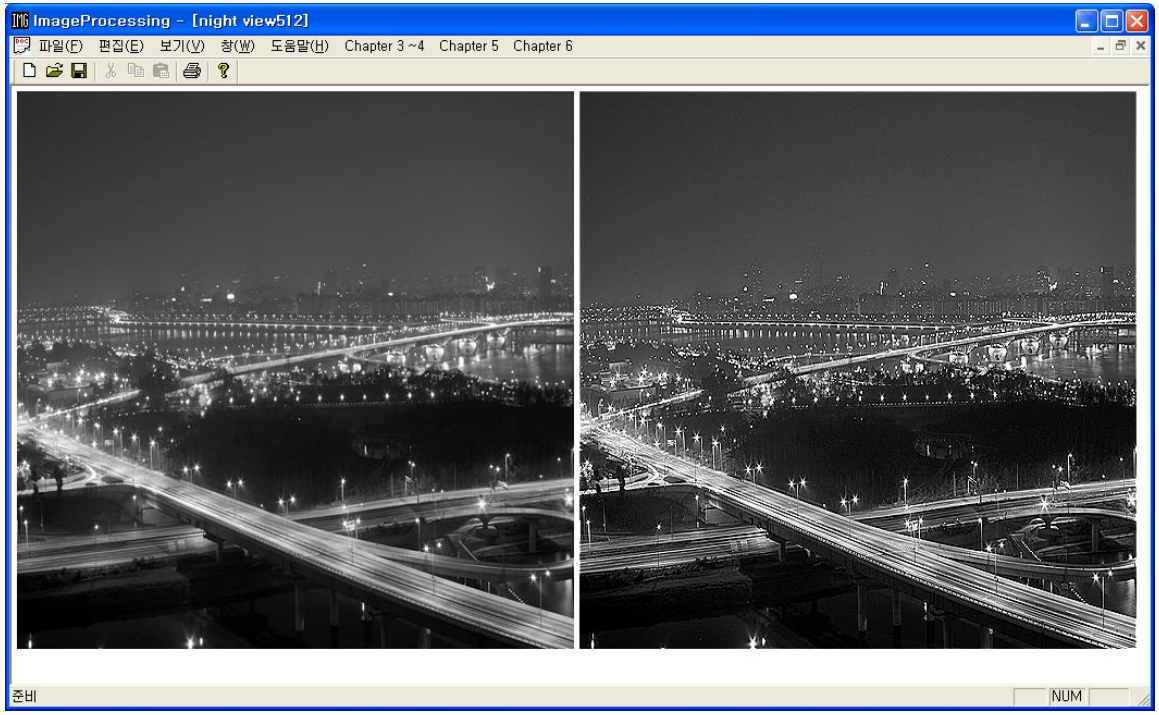
- 영상의 세밀한 부분을 제거하여 영상을 흐리게 하거나 부드럽게 하는 기술
- 영상의 세밀한 부분은 주파수 축에서 보면 고주파 성분인데, 블러링은 이 고주파 성분을 제거해 줌.
- 사용하는 가중치의 회선 마스크는 저역통과 필터(Low Pass Filter)가 됨



[그림 6-4] 입력 영상에 블러링 처리를 한 영상

샤프닝

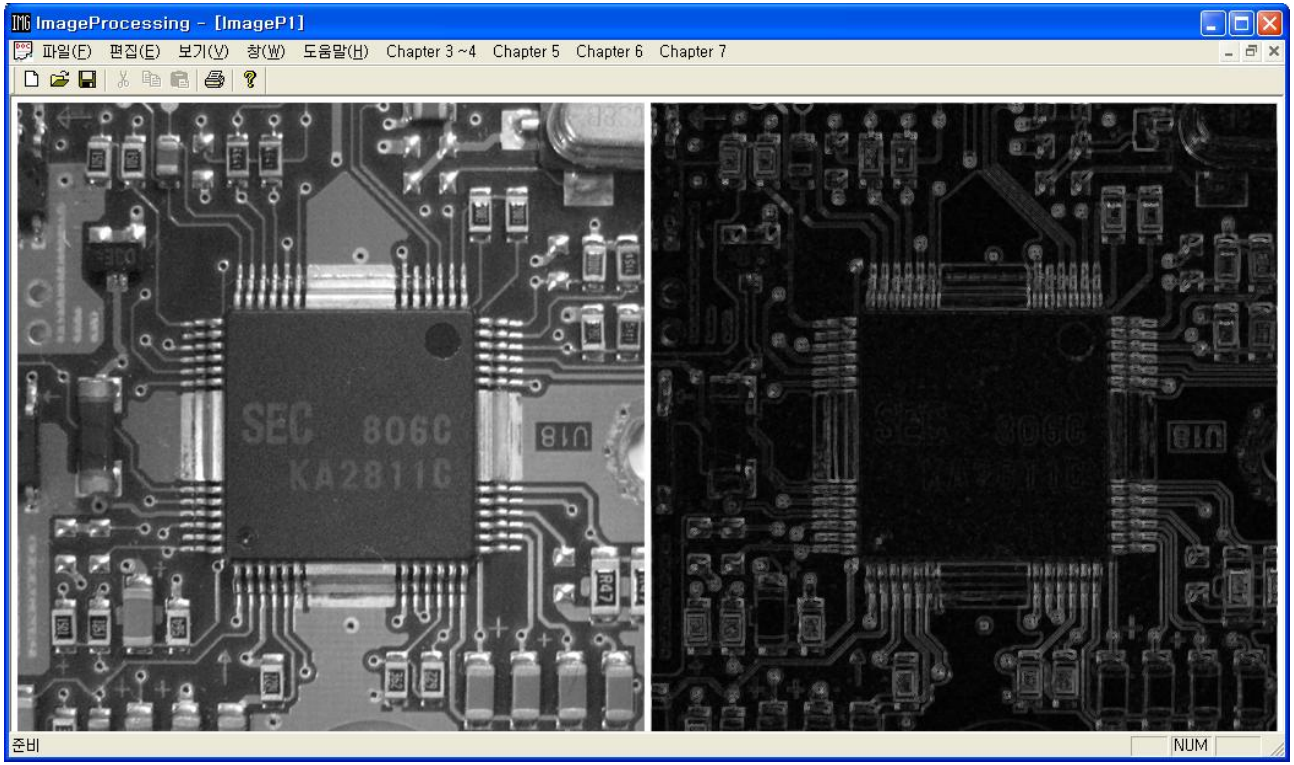
- 블러링과는 반대로 디지털 영상에서 상세한 부분을 더욱 강조하여 표현.
- 영상의 상세한 부분은 고주파 성분이므로 영상에서 저주파 성분만 제거하면 샤프닝 효과를 얻을 수 있음
- 사용되는 가중치의 회선 마스크는 고역통과 필터(**High Pass Filter**)가 됨.



[그림 6-5] 입력 영상에 샤프닝 처리를 한 영상

경계선 검출

- 디지털 영상의 경계선을 찾아내는 기술
- 경계선은 영상의 밝기가 낮은 값에서 높은 값으로 또는 높은 값에서 낮은 값으로 변하는 지점에 있으므로 입력한 영상의 정보가 많이 듦.



[그림 6-6] 입력 영상에 경계선 검출을 한 영상

Section 02 회선 처리의 원리

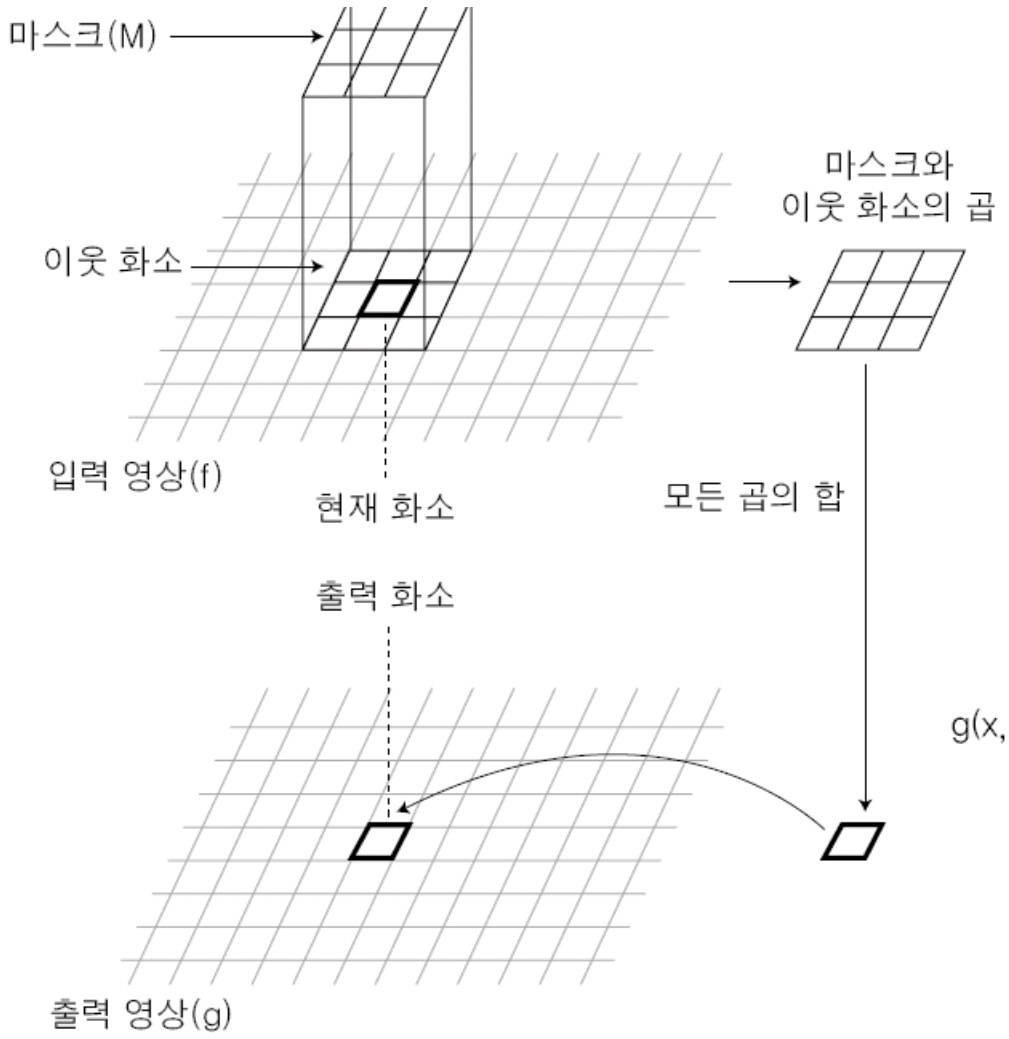
화소의 영역 처리

- 디지털 영상처리 시스템은 선형 시불변 시스템

  → 디지털 영상처리의 결과는 컨벌루션 또는 회선 기법으로 얻을 수 있음

- 회선 기법으로 생성되는 새로운 화소 값
  - 이웃한 화소 값과 이에 대응하는 회선 마스크의 가중치를 곱한 뒤 곱한 값을 더해서 얻음.
  - 가중치는 작은 행렬인 회선 마스크 또는 회선 커널로 구성됨.

- 디지털 영상에서 회선 기법
  - 가중치를 포함한 회선 마스크가 이동하면서 수행

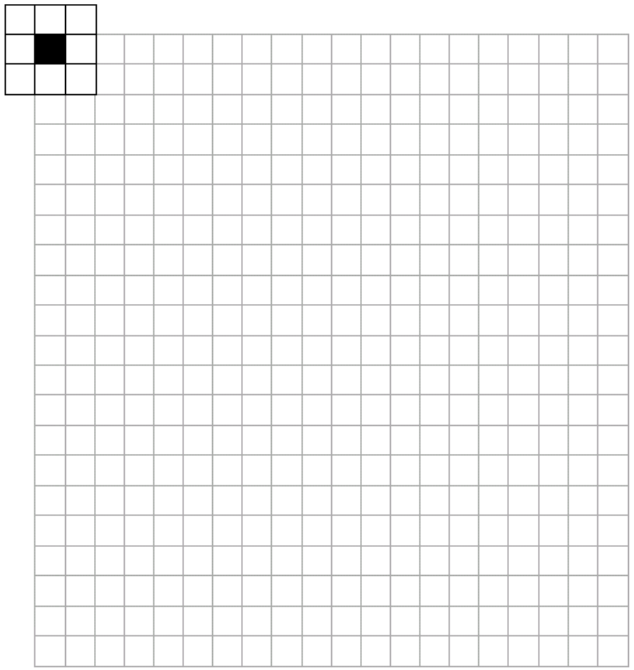


$$g(x, y)=M(0, 0)\times f(x-1, y-1)+M(0, 1)\times f(x-1, y)$$
$$+M(0, 2)\times f(x-1, y+1)+M(1, 0)\times f(x, y-1)$$
$$+M(1, 1)\times f(x, y)+M(1, 2)\times f(x, y+1)$$
$$+M(2, 0)\times f(x+1, y-1)+M(2, 1)\times f(x+1, y)$$
$$+M(2, 2)\times f(x+1, y+1)$$

[그림 6-7] 디지털 영상에서 회선을 처리하는 과정

회선 수행 방법

- 가중치를 포함한 회선 마스크가 이동하면서 수행
- 회선 마스크가 영상의 왼쪽 위 화소에서 오른쪽으로 한 화소씩 차례로 이동하면서 수행하여 새로운 화소를 만들어 냄.
- 한 줄에서의 회선 수행이 끝나면, 다음 줄로 이동하여 다시 한 화소씩 오른쪽으로 이동하면서 차례로 수행됨.



[그림 6-8] 회선 수행이 시작되는 위치

회선의 경계 부분 처리

- 회선의 경계 부분 처리
  - 화소의 영역을 처리하려면 이웃 화소가 있어야 하지만 시작이나 끝부분에는 주변 화소가 없는 것처럼 회선 마스크에 대응할 요소가 없는 영상의 화소를 처리하는 방법을 경계 부분 처리라고 함.

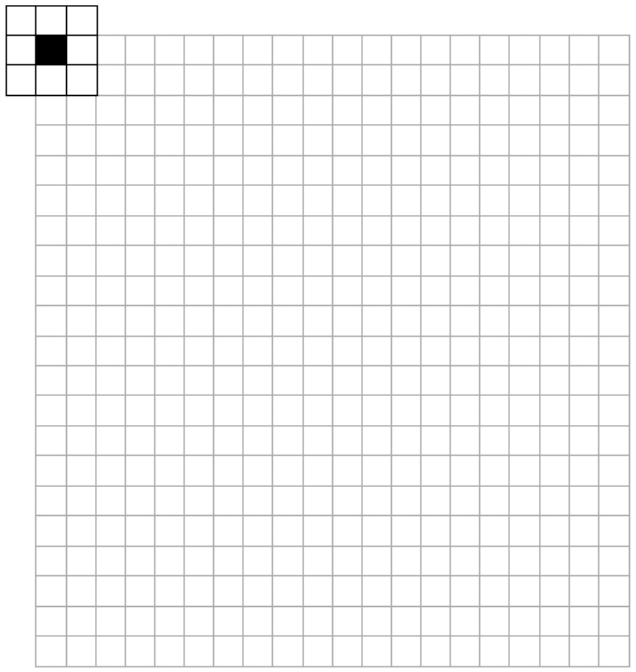
- 0 삽입
  - 회선 마스크에 대응되는 빈 영상의 화소 값을 모두 0으로 가정해서 회선을 수행하는 방법
  - 0으로 설정해서 영상의 경계 부분에서 정확한 회선 처리가 불가능하므로 손실이 발생함.
  - 프로그램을 단순하게 구현할 수 있다는 장점이 있음.

회선의 경계 부분 처리(계속)

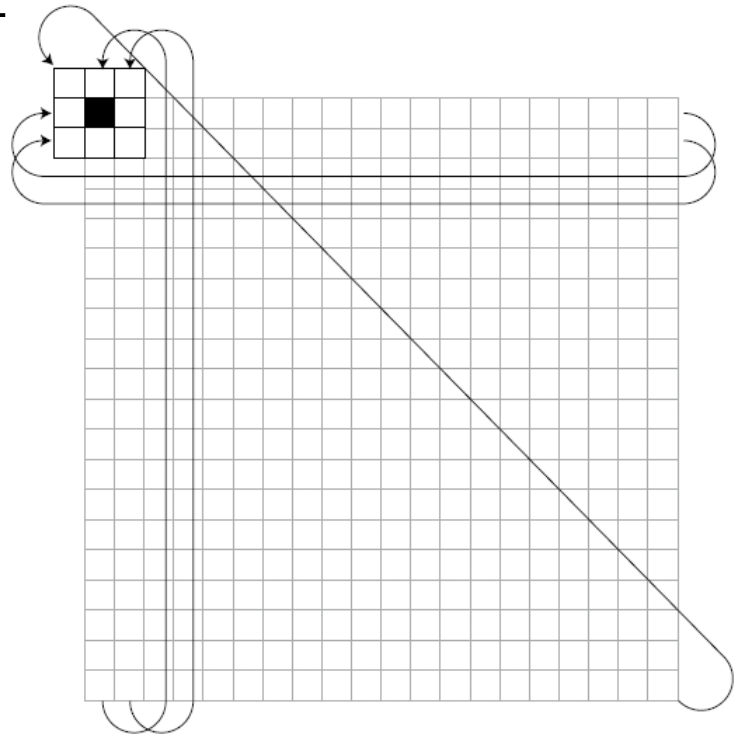
- 중첩 부분에서만 회선 수행
  - 회선 마스크와 영상이 완전하게 중첩되는 위치에서 회선을 시작하도록 하는 방법
  - 회선 마스크의 크기가 3×3이면 모든 회선 마스크의 요소와 영상의 화소가 중첩되는 영상 위치 (1, 1)에서 회선을 시작함.
  - 중첩 부분에서 회선 수행이 끝나면 경계 부분은 입력 영상과 같은 화소 값을 복사해서 사용
  - 경계 부분은 회선 처리가 되지 않아 모든 영역이 회선 처리된 영상의 새로운 화소 값을 얻을 수는 없음.

회선의 경계 부분 처리(계속)

- 영상의 크기를 조정하여 회선 수행
  - 영상의 시작과 끝부분이 연결된 것으로 처리하는 방법
  - 영상이 폐곡선을 형성해 영상의 경계 부분이 빈 영상의 화소 값을 대신
  - 입력 영상의 첫 번째와 두 번째 행의 마지막 열에 있는 화소, 마지막 행의 첫 번째와 두 번째 열에 있는 화소, 마지막으로 대각선 방향으로 마지막 행의 마지막 열의 화소를 복사해 사용



[그림 6-8] 회선 수행이 시작되는 위치



[그림 6-9] 영상의 크기를 조정하여 회선 수행

회선 마스크

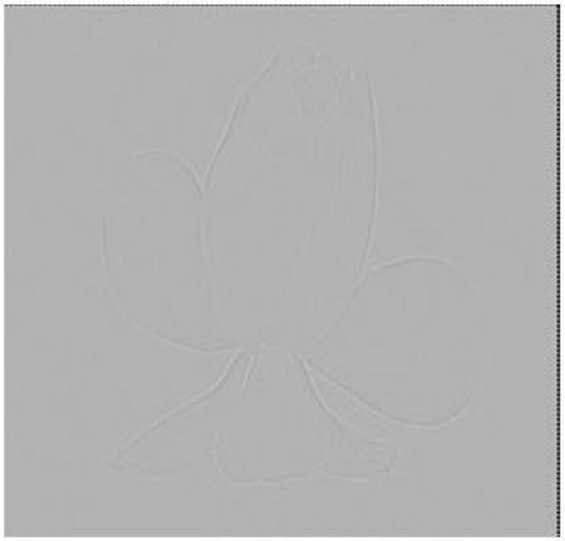
- 회선 마스크의 특징
  - 주변 화소의 값을 각 방향에서 대칭적으로 고려해야 함. 이것은 각 방향에 있는 같은 수의 이웃 화소에 기반을 두고 새로운 화소 값을 생성하기 때문
  - 회선 마스크의 크기는 행과 열 모두 홀수의 크기를 사용하여 3×3, 5×5, 7×7 등
  - 회선 처리 기법으로 생성된 출력 영상은 밝기 에너지를 보존해야 하므로 영상의 평균 밝기를 원 영상과 똑같이 유지해야 함.
  - 회선된 영상의 평균 밝기 값이 원본 영상과 같도록 많은 회선 마스크의 계수 합이 1이 되도록 함.
  - 경계선 검출 등에서 사용되는 일부 회선 마스크에서는 음수의 계수를 포함하며, 계수 합이 0이 되도록 설계
  - 음의 계수에서는 음의 화소 값이 생성될 수도 있으나 밝기는 항상 양의 값만 있으므로 생성된 화소 값에 일정한 상수(최대 밝기의 1/2)를 더해서 양의 화소 값이 나오도록 함.

회선 마스크(계속)

원본 영상

0	0	0
0	1	0
0	0	0

Sum=1



1	1	1
1	−8	1
1	1	1

Sum=0

[그림 6-10] 회선 마스크에 따른 영상의 회선 처리

회선 마스크(계속)

- 회선 마스크에 따른 회선 처리 예
  - 회선 마스크에 따른 결과를 간단히 확인할 수 있도록 1차원 데이터에서 1차원 회선 마스크를 사용하여 회선 처리

[그림 6-11] 회선 마스크에 따른 회선 처리의 간단한 계산 예

엠보싱

- 회선 처리를 이용한 가장 기본적인 영상처리 방법
- 경계선 검출 기법에서 사용되는 회선 마스크와 같은 회선 마스크 사용
- 적절하게 구분된 경계선으로 영상이 볼록한 느낌을 갖게 됨(구리 판에 양각한 것 같은 효과)
- 가운데에 있는 계수가 다른 계수를 상쇄시키도록 구성해서 경계선을 검출. 이 경계선에서 양각한 효과를 얻을 수 있음.
- 마스크에는 음의 계수 값 -1이 있으므로 회선 처리로 생성된 영상의 화소 값은 음수

0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

(a) 원본 영상

0	128	255	255	128	0
0	128	255	255	128	0
0	128	255	255	128	0
0	128	255	255	128	0
0	128	255	255	128	0
0	128	255	255	128	0

(b) 엠보싱 처리 영상

[그림 6-12] 간단한 엠보싱 처리

[실습하기 6-1] 엠보싱 프로그램

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

ID	ID_EMBOSSING
Caption	엠보싱 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 엠보싱 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnEmbossing
Doc Class	void	OnEmbossing
Doc Class	double**	OnMaskProcess
Doc Class	double**	OnScale
Doc Class	double**	Image2DMem

- OnMaskProcess : 화소 영역을 처리하는 함수
- OnScale : 영상의 화소 값을 정규화시키는 함수
- Image2DMem : 화소 영역 처리 프로그램을 간단히 하려고 1차원 배열을 2차원 배열로 할당하는 함수

[실습하기 6-1] 엠보싱 프로그램

③ **Doc Class OnEmbossing, OnMaskProcess, OnScale, Image2DMem 함수 추가**
→ [Add Member Variable] 대화상자에서 double m_tempImage 변수 선언**

❶ OnEmbossing 함수 추가하기

```
void CImageProcessingDoc::OnEmbossing()
{
   int i, j;
   double EmboMask[3][3] = {{-1., 0., 0.}, {0., 0., 0.}, {0., 0., 1.}};
   // 마스크 선택
   // double EmboMask[3][3] = {{0., 0., 0.}, {0., 1., 0.}, {0., 0., 0.}};
   // double EmboMask[3][3] = {{1., 1., 1.}, {1., -8.,1.}, {1., 1., 1.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;
   m_OutputImage = new unsigned char [m_Re_size];
   m_tempImage = OnMaskProcess(m_InputImage, EmboMask);
   // OnMaskProcess 함수를 호출하여 회선 처리를 한다.

   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   } // 회선 처리 결과가 0~255 사이 값이 되도록 한다.
```

[실습하기 6-1] 엠보싱 프로그램

❶ OnEmbossing 함수 추가하기 (계속)

```
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);
   // 정규화 함수를 사용할 때

   // 회선 처리 결과나 정규화 처리 결과는 2차원 배열 값이 되므로
   // 2차원 배열을 1차원 배열로 바꾸어 출력하도록 한다.
   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-1] 엠보싱 프로그램

❷ OnEmbossing 함수 추가하기

```
double** CImageProcessingDoc::OnMaskProcess(unsigned char *Target, double Mask[3][3])
{ // 회선 처리가 일어나는 함수
   int i, j, n, m;
   double **tempInputImage, **tempOutputImage, S = 0.0;

   tempInputImage = Image2DMem(m_height + 2, m_width + 2);
   // 입력 값을 위한 메모리 할당
   tempOutputImage = Image2DMem(m_height, m_width);
   // 출력 값을 위한 메모리 할당

   // 1차원 입력 영상의 값을 2차원 배열에 할당한다.
   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         tempInputImage[i+1][j+1]
            = (double)Target[i * m_width + j];
      }
   }

   // 회선연산
   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         for(n=0 ; n<3 ; n++){
            for(m=0 ; m<3 ; m++){
               S += Mask[n][m] * tempInputImage[i+n][j+m];
            }
         } // 회선 마스크의 크기 만큼 이동하면서 값을 누적
         tempOutputImage[i][j] = S; // 누적된 값을 출력 메모리에 저장
         S = 0.0; // 다음 블록으로 이동하면 누적 값을 초기화
         }
      }
   return tempOutputImage; // 결과 값 반환
}
```

[실습하기 6-1] 엠보싱 프로그램

❸ OnEmbossing 함수 추가하기

```
double** CImageProcessingDoc::OnScale(double **Target, int height, int width)
{ // 정규화를 위한 함수
   int i, j;
   double min, max;

   min = max = Target[0][0];

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         if(Target[i][j] <= min)
            min = Target[i][j];
      }
   }

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         if(Target[i][j] >= max)
            max = Target[i][j];
      }
   }

   max = max - min;

   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         Target[i][j] = (Target[i][j] - min) * (255. / max);
      }
   }

   return Target;
}
```

[실습하기 6-1] 엠보싱 프로그램

❹ Image2DMem 함수 추가하기

```
double** CImageProcessingDoc::Image2DMem(int height, int width)
{ // 2차원 메모리 할당을 위한 함수
   double** temp;
   int i, j;
   temp = new double *[height];
   for(i=0 ; i<height ; i++){
      temp[i] = new double [width];
   }
   for(i=0 ; i<height ; i++){
      for(j=0 ; j<width ; j++){
         temp[i][j] = 0.0;
      }
   } // 할당된 2차원 메모리를 초기화
   return temp;
}
```

[실습하기 6-1] 엠보싱 프로그램

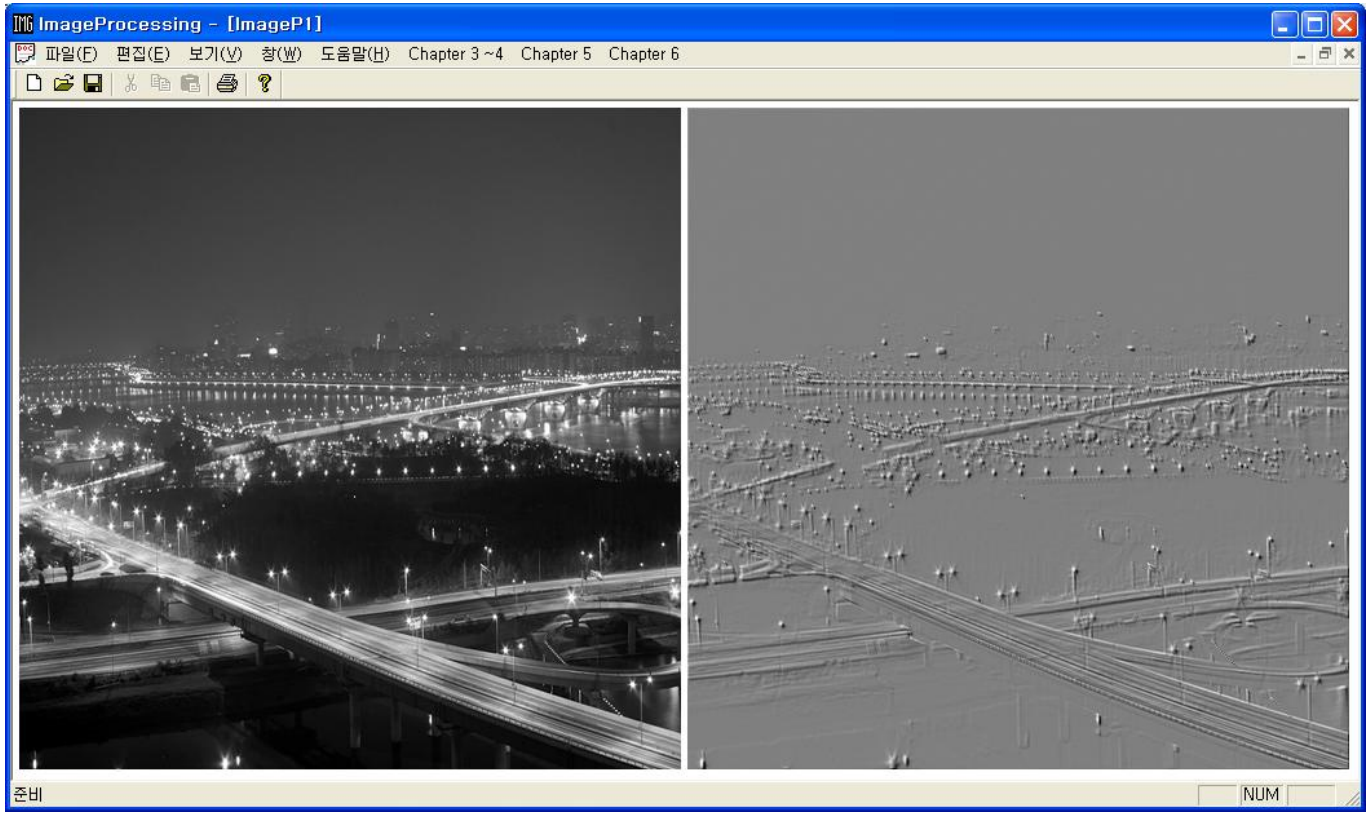
④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnEmbossing();

    Invalidate(TRUE);
}
```

[실습하기 6-1] 엠보싱 프로그램

⑤ 프로그램 실행 결과 영상

✓ 엠보싱 처리 기법을 실제 8비트 그레이 레벨 영상에 적용한 예로 음각과 양각된 것처럼 느껴짐.



그레이 영상을 엠보싱 처리한 결과 영상

컬러 영상의 회선 처리

- 컬러 영상의 회선 처리
  - 컬러 영상은 **R, G, B** 채널 세 개를 이용해 다양한 색을 표현하므로 이 채널을 응용하여 회선 처리 수행
  - 독립 채널별 회선 수행과 **HSI** 컬러 모델로 변경한 뒤 회선을 처리하는 두 가지 방법이 있음

- 독립 채널별 회선 수행
  - **RGB** 컬러 영상을 **R, G, B** 채널로 분리하여 채널별로 각각 회선을 수행한 뒤 회선 처리된 각 채널을 다시 조합해서 회선된 컬러 영상 생성
  - **RGB** 컬러 영상은 채널 세 개를 조합해서 색을 표현하므로 회선 과정에서 아주 작은 오류만 발생해도 조합된 회선에서 정확한 결과를 만들지 못함.

컬러 영상의 회선 처리(계속)

- **HSI 컬러 모델로 변경 후 회선 처리**
  - **RGB 컬러 영상을 우선으로 해서 HSI 컬러 모델로 변환하여 색상(H), 명도(I), 채도(S) 성분을 얻음**
  - **밝기 성분인 명도(I) 부분만 회선 처리를 수행하고 나머지 두 성분은 수행하지 않음. 마지막으로 HSI는 다시 RGB 컬러 영상으로 변환됨.**
  - **색상(H)에서 회선 처리를 하지 않아 원 영상의 색상 부분이 그대로 보존되므로 독립 채널별로 회선을 처리하는 방법보다 더 정확**

컬러 영상의 회선 처리(계속)

[그림 6-13] 채널별 회선 수행 뒤 합성한 컬러 회선 영상

[그림 6-14] HSI 변환 뒤 밝기(I)만 회선 처리된 영상

- 블러링 회선 마스크
  - 블러링 회선 마스크는 모든 계수가 양수로 전체 합은 1
  - 디지털 영상에서 세세한 부분은 화소 값이 극단적인 값에 속함. 이 극단적 값을 제거하는 대표적인 방법이 바로 평준화로, 평균값으로 대체하는 것
  - 블러링 회선 마스크의 계수는 평균을 구하는 데 사용되므로 모두 값이 같음.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

[그림 6-15] 블러링 마스크의 회선 계수

블러링 회선 마스크(계속)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

90	90	90	90	90	90	90
90	90	90	90	90	90	90
90	90	255	255	255	90	90
90	90	255	255	255	90	90
90	90	255	255	255	90	90
90	90	90	90	90	90	90
90	90	90	90	90	90	90

(a) 원본 영상

90	90	90	90	90	90	90
90	105	120	135	120	105	90
90	120	150	180	150	120	90
90	135	180	255	180	135	90
90	120	150	180	150	120	90
90	105	120	135	120	105	90
90	90	90	90	90	90	90

(b) 블러링 영상

[그림 6-16] 간단한 블러링 처리 예

[실습하기 6-2] 블러링 프로그램

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

ID	ID_BLURR
Caption	블러링 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 블러링 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnBlurr
Doc Class	void	OnBlurr

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 6-2] 블러링 프로그램

```
void CImageProcessingDoc::OnBlurr()
{
   int i, j;
   double BlurrMask[3][3] = {{1./9., 1./9., 1./9.},
      {1./9., 1./9., 1./9.}, {1./9., 1./9., 1./9.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   m_tempImage = OnMaskProcess(m_InputImage, BlurrMask);
   // 블러링 처리
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

   // 정규화
   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   }
   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-2] 블러링 프로그램

④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnBlurr();

    Invalidate(TRUE);
}
```

⑤ 프로그램 실행 결과 영상

✓ 엠보싱 처리 기법을 실제 8비트 그레이 레벨 영상에 적용한 예로 음각과 양각된

(a) 입력 영상 (b) 3×3 마스크 적용 영상 (c) 5×5 마스크 적용 영상

블러링 회선 마스크 크기에 따른 회선 처리 영상

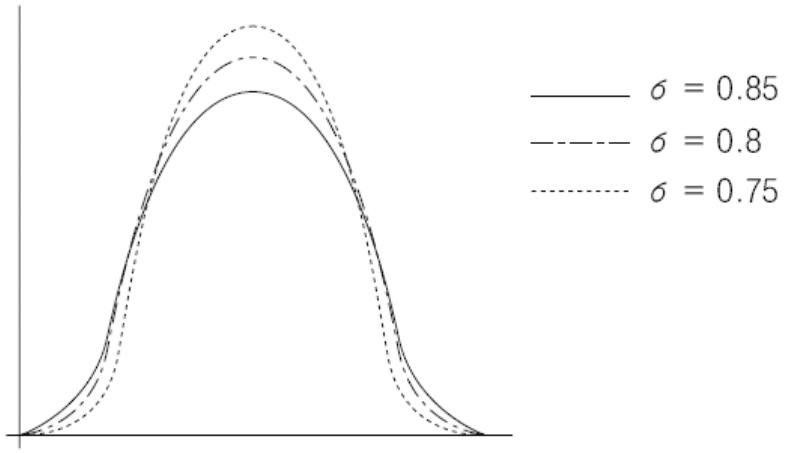
가우시안 스무딩 필터링 처리

- 영상의 세세한 부분을 제거하여 부드럽게 하므로 스무딩(Smoothing) 처리라고도 함
- 스무딩 처리에 사용되는 대표적인 저역통과 필터로 가우시안 필터 (Gaussian Filter)가 있음
- 이 필터는 수학적으로 잘 정의된 가우시안 함수에서 얻음.

$$G[x, y] = \frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{2\pi\sigma^2}$$

가우시안 스무딩 필터링 처리(계속)

- σ에 따른 가우시안 함수 그래프
  - σ 값이 클수록 높이는 낮지만 폭은 넓어지므로 많은 저주파 성분을 통과시킴.
  - σ 값이 작을 수록 적은 저주파 성분만 통과시킴.



[그림 6-17] σ 값에 따른 가우시안 그래프

- 가우시안 필터
  - 가우시안 함수를 표본화하여 마스크의 계수를 결정
  - 오른쪽은 3×3 가우시안 필터 계수. 모든 계수는 양의 값으로 그 합은 1

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

[실습하기 6-3] 가우시안 필터 처리 프로그램

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

ID	ID_GAUSSIAN_FILTER
Caption	가우시안 필터 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 가우시안 필터 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnGaussianFilter
Doc Class	void	OnGaussianFilter

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 6-3] 가우시안 필터 처리 프로그램

```
void CImageProcessingDoc::OnGaussianFilter()
{
   int i, j;
   double GaussianMask[3][3] = {{1./16., 1./8., 1./16.},
         {1./8., 1./4., 1./8.}, {1./16., 1./8., 1./16.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   m_tempImage = OnMaskProcess(m_InputImage, GaussianMask);
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   }

   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-3] 가우시안 필터 처리 프로그램

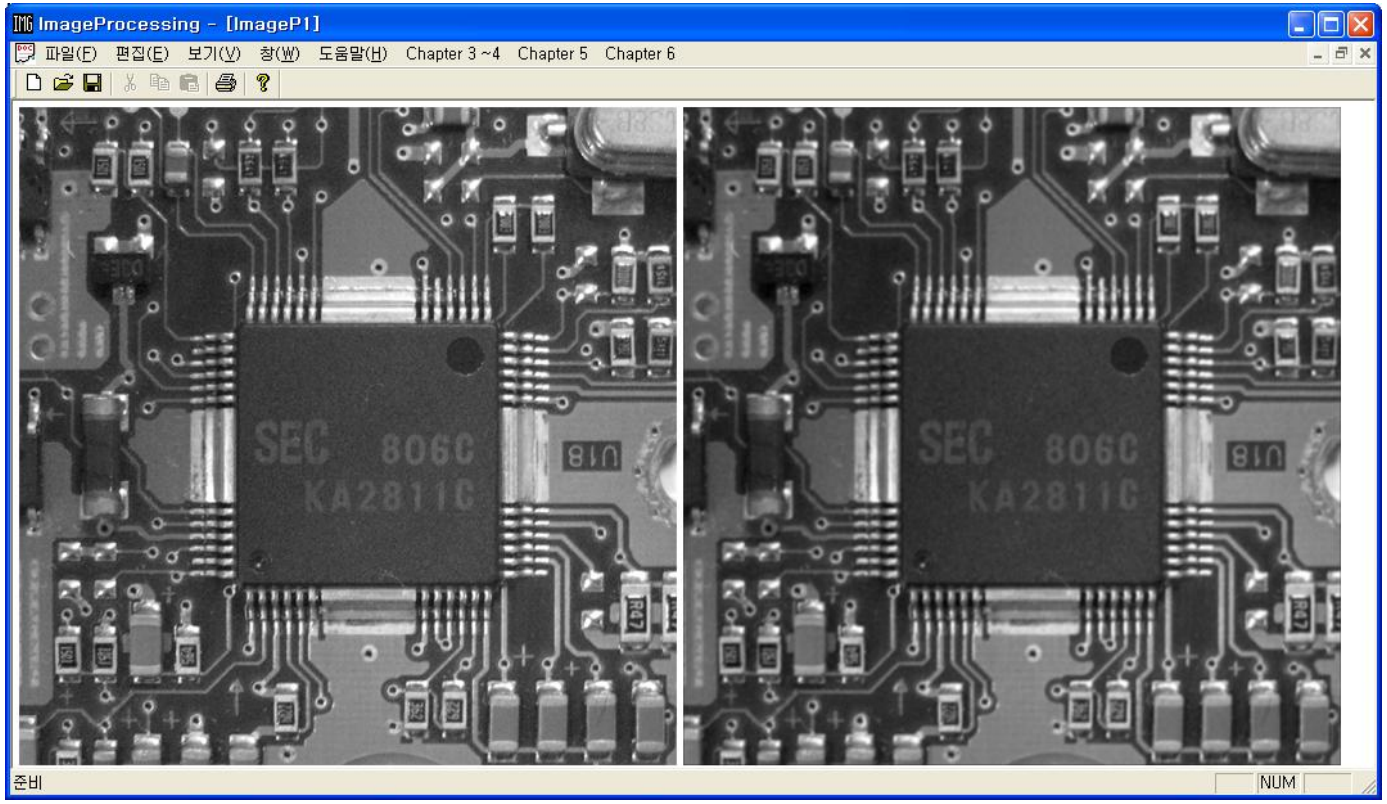
④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnGaussianFilter();

    Invalidate(TRUE);
}
```

[실습하기 6-3] 가우시안 필터 처리 프로그램

⑤ 프로그램 실행 결과 영상

✓ 3 x 3 가우시안 필터를 이용한 스무딩한 영상으로, 전체적으로 경계선이 악화되어 흐려짐.



가우시안 스무딩 처리한 영상

Section 04 샤프닝

- 블러링과는 반대되는 효과를 보이는 기법을 샤프닝(sharpening) 또는 영상 강화라고 함.
- 고주파에 해당하는 상세한 부분을 더욱 강조하여 대비 효과를 증가시킴
- 흐린 영상을 개선하여 선명한 영상을 생성하는 데 주로 사용됨.

−1	−1	−1
−1	9	−1
−1	−1	−1

(a) 샤프닝 회선 마스크 1

0	−1	0
−1	5	−1
0	−1	0

(b) 샤프닝 회선 마스크 2

[그림 6-18] 크기가 3×3인 샤프닝 회선 마스크

샤프닝(계속)

0	−1	0
−1	5	−1
0	−1	0

10	10	10	10	10	10	10
10	10	10	10	10	10	10
10	10	50	50	50	10	10
10	10	50	50	50	10	10
10	10	50	50	50	10	10
10	10	10	10	10	10	10
10	10	10	10	10	10	10

(a) 원본 영상

10	10	10	10	10	10	10
10	10	0	0	0	10	10
10	0	130	90	130	0	10
10	0	90	50	90	0	10
10	0	130	90	130	0	10
10	10	0	0	0	10	10
10	10	10	10	10	10	10

(b) 샤프닝 영상

[그림 6-19] 간단한 샤프닝 처리 예

[실습하기 6-4] 샤프닝 처리 프로그램

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

ID	ID_SHARPENING
Caption	샤프닝 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 샤프닝 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnSharpening
Doc Class	void	OnSharpening

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 6-4] 샤프닝 처리 프로그램

```
void CImageProcessingDoc::OnSharpening()
{
   int i, j;
   //double SharpeningMask[3][3] = {{-1., -1., -1.},
      {-1., 9., -1.}, {-1., -1., -1.}};
   double SharpeningMask[3][3] = {{0., -1., 0.}, {-1., 5.,
      -1.}, {0., -1., 0.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   m_tempImage = OnMaskProcess(m_InputImage, SharpeningMask);
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   }

   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-4] 샤프닝 처리 프로그램

④ View 클래스에 다음 프로그램 추가

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

   pDoc->OnSharpening();

   Invalidate(TRUE);
}
```

⑤ 프로그램 실행 결과 영상

✓ 적용 마스크에 따라 경계 부분이 강조된 정도가 다르게 나타남. 마스크1을 적용한 (b) 보다 마스크 2를 적용한 (c)가 더욱 강조되었음.

(a) 입력 영상

(b) 마스크 1 적용

(c) 마스크 2 적용

마스크에 따른 샤프닝 처리 영상

고주파 통과 필터를 이용한 샤프닝 처리

- 고주파 필터는 영상 신호 성분 중 고주파 성분은 통과시키고 저주파 성분은 차단
- 필터 계수의 합은 0으로 샤프닝 회선 마스크하고는 다르나, 나머지 동작 특성은 같음.
- 가운데 큰 양수 값과 주변의 작은 음수 값으로 마스크되어 경계선을 더욱 두드러지게 함.
- 다음은 대표적인 고주파 필터의 계수

−1/9	−1/9	−1/9
−1/9	8/9	−1/9
−1/9	−1/9	−1/9

[실습하기 6-5] 고주파 필터 샤프닝 처리 프로그램

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

ID	ID_HPF_SHARP
Caption	고주파 필터 샤프닝 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 고주판 필터 샤프닝 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnHpfSharp
Doc Class	void	OnHpfSharp

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 6-5] 고주파 필터 샤프닝 처리 프로그램

```
void CImageProcessingDoc::OnHpfSharp()
{
   int i, j;
   double HpfSharpMask[3][3] = {{-1./9., -1./9., -1./9.},
      {-1./9., 8./9., -1./9.}, {-1./9., -1./9., -1./9.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   m_tempImage = OnMaskProcess(m_InputImage, HpfSharpMask);
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   }

   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-5] 고주파 필터 샤프닝 처리 프로그램

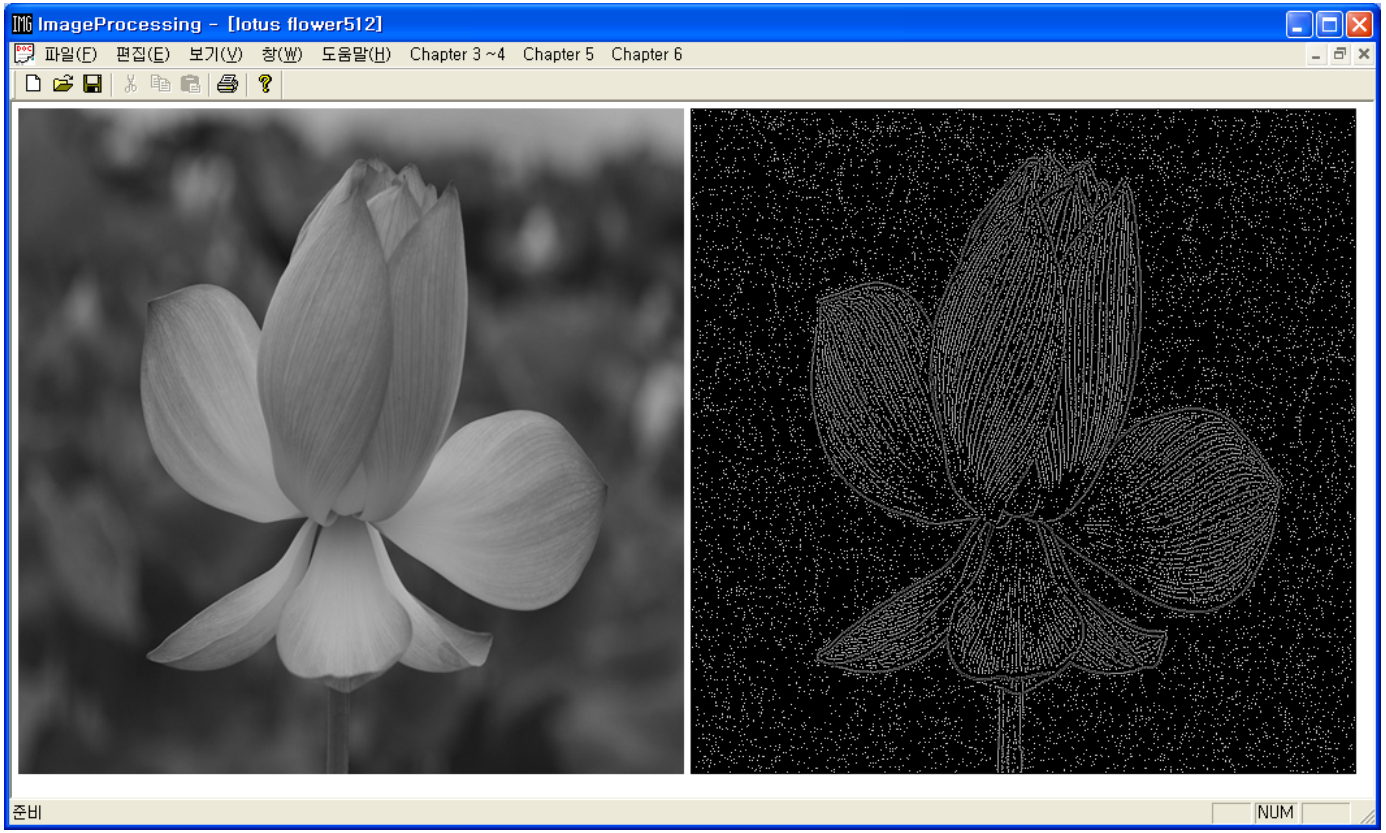
④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnHpfSharp();

    Invalidate(TRUE);
}
```

[실습하기 6-5] 고주파 필터 샤프닝 처리 프로그램

⑤ 프로그램 실행 결과 영상

✓ 샤프닝 회선 마스크로 샤프닝된 영상과 비교하면 더 많은 저주파 성분이 제거되어 단지 경계 부분만 확인할 수 있음.



고주파 통과 필터를 사용하여 샤프닝 처리한 영상

저주파 통과 필터를 이용한 샤프닝 처리

- 고주파 통과 필터를 통과한 결과 영상은 저주파 통과 필터를 활용하여 얻을 수도 있음.
- 원본 영상에서 저주파 통과 필터를 통과한 결과 영상을 뺄셈하여 얻는데, 이를 언샤프 마스킹(Unsharp Masking)이라고 함.

Unsharp Masking = (원 영상) – (저주파 통과 필터링 결과 영상)

- 고주파 통과 필터는 세부 정보를 강조하지만 영상에서 중요한 부분에 해당하는 낮은 공간 주파수 성분이 손실됨.
- 고주파 지원(High-Boost) 필터는 저주파 영역에서 손실한 양에 해당하는 일정량의 이득을 주어 저주파 성분의 손실을 어느 정도 보상받을 수 있음.
- 원본 영상의 밝기를 증가시킨 뒤 저주파 영상을 뺄셈하는 방법을 이용하여 처리

High-Boost = α(원 영상) – (저주파 통과 필터링 결과 영상)

[실습하기 6-6] 저주파 필터 샤프닝 처리 프로그램

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

ID	ID_LPF_SHARP
Caption	저주파 필터 샤프닝 처리

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 저주파 필터 샤프닝 처리를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnLpfSharp
Doc Class	void	OnLpfSharp

③ **Doc 클래스에 다음 프로그램 추가. 저주파 필터 샤프닝 처리에서는 이전에 추가한 ConstantDlg 대화상자를 이용하여 alpha값을 입력받음.**

[실습하기 6-6] 저주파 필터 샤프닝 처리 프로그램

```
void CImageProcessingDoc::OnLpfSharp()
{
   CConstantDlg dlg; // 상수를 입력받으려고 대화상자 선언

   int i, j, alpha;
   double LpfSharpMask[3][3] = {{1./9., 1./9., 1./9.},
      {1./9., 1./9., 1./9.}, {1./9., 1./9., 1./9.}};

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   if(dlg.DoModal() == IDOK){
      alpha = (int)dlg.m_Constant;
      // 대화상자를 이용하여 상수를 입력받는다.
   }

   m_tempImage = OnMaskProcess(m_InputImage, LpfSharpMask);

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

[실습하기 6-6] 저주파 필터 샤프닝 처리 프로그램

```
   // m_tempImage = OnScale(m_tempImage, m_Re_height, m_Re_width);

   for(i=0 ; i<m_Re_height ; i++){
      for(j=0 ; j<m_Re_width ; j++){
         if(m_tempImage[i][j] > 255.)
            m_tempImage[i][j] = 255.;
         if(m_tempImage[i][j] < 0.)
            m_tempImage[i][j] = 0.;
      }
   }

   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)m_tempImage[i][j];
      }
   }
}
```

[실습하기 6-6] 저주파 필터 샤프닝 처리 프로그램

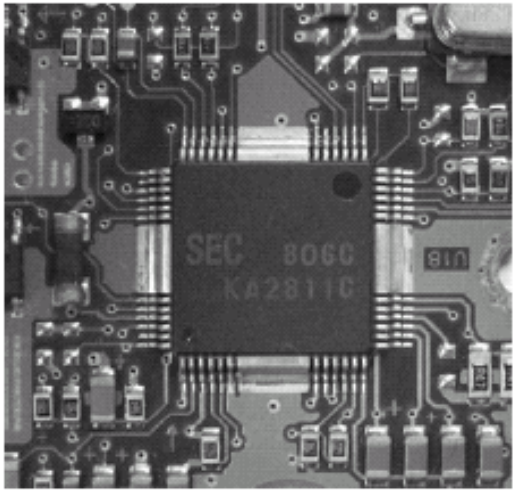
④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnLpfSharp();

    Invalidate(TRUE);
}
```

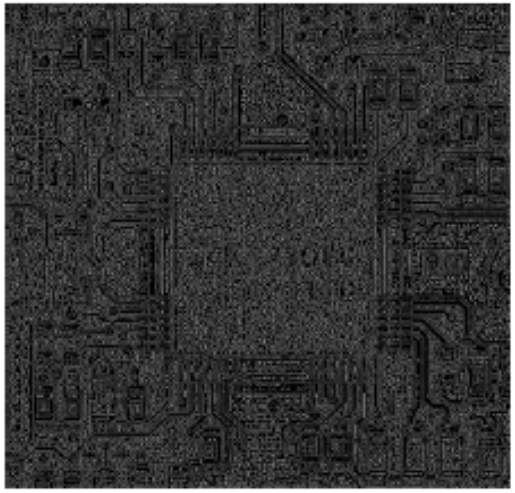
[실습하기 6-6] 저주파 필터 샤프닝 처리 프로그램

⑤ 프로그램 실행 결과 영상

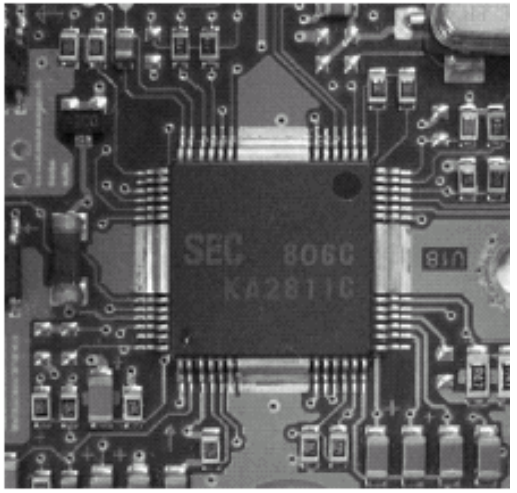
✓ **Unsharp Mask로 처리된 (b)는 고주파 통과 필터를 직접 사용한 것과 같은 효과. 고주파 지원으로 처리된 (c)는 샤프닝 회선 마스크를 이용한 것과 결과가 비슷**



(a) 원본 영상



(b) Unsharp Mask 처리



(c) 고주파 지원(α=17)

저주파 통과 필터를 사용한 샤프닝 처리 결과 영상

요약

- 화소의 영역 처리
  - 출력 영상의 새로운 화소 값을 결정하려고 해당 입력 화소뿐만 아니라 그 주변의 화소도 함께 고려하는 공간 영역 연산
  - 회선 기법(처리) 또는 컨벌루션 처리라고도 함.
- 회선 기법
  - 주변 화소의 조합을 결정하여 새로운 화소를 출력해 줌.
  - 원시 화소에 이웃한 각 화소에 가중치를 곱한 합을 출력 화소로 생성
- 화소의 영역 기반 처리
  - 엠보싱 효과, 블러링, 샤프닝, 경계선 검출, 잡음 제거 등이 있음
- 엠보싱 효과
  - 입력 영상을 양각 형태로 나타냄=영상의 특정 부분이 볼록해 보이도록 만듦.
- 블러링
  - 영상의 세밀한 부분을 제거하여 영상을 흐리게 하거나 부드럽게 나타내는 기술. 고주파 성분을 제거하는 기술(영상의 세밀한 부분은 고주파 성분).
- 경계선 검출
  - 디지털 영상에 있는 경계선(Edge)을 찾아내는 기법

요약

- 화소의 영역 처리를 수행하는 디지털 영상처리 시스템은 선형 시불변 시스템을 만족시킴
  - 디지털 영상처리의 결과는 컨벌루션 또는 회선 처리로 얻을 수 있음
- 회선 기법으로 생성되는 새로운 화소 값
  - 이웃 화소 값과 이에 대응하는 회선 마스크의 가중치를 곱한 뒤 곱한 값을 더해서 얻음.
  - 여기서 가중치는 작은 행렬인 회선 마스크 또는 회선 커널로 구성됨.
- 디지털 영상에서 화소의 영역 처리를 수행하는 회선 기법
  - 가중치를 포함한 회선 마스크가 이동하면서 수행
- 경계 부분 처리
  - 회선 마스크에 대응할 요소가 없는 영상의 화소를 처리하는 방법
- 영상에서의 주파수 개념은 화소 값의 변화율을 나타냄.
- 블러링 처리나 스무딩 처리를 하려고 영상처리에서 사용하는 대표적인 저역 통과 필터로 가우시안 필터가 있음.

Thank you