04장 화소점 처리

- 화소 점 처리의 개념
- 디지털 영상의 산술연산과 논리연산
- 디지털 영상의 다양한 화소 점 처리 기법

4장. 화소 점 처리

학습목표

- ✓ 화소 점 처리의 개념을 알아본다.
- ✓ 디지털 영상의 화소에서 산술연산과 그 효과를 알아본다.
- ✓ 디지털 영상에서 논리연산과 그 효과를 알아본다.
- ✓ 다양한 영상의 화소 점 처리 기법을 익힌다.

Section 01 화소 점 처리의 개념

- 화소 점 처리
  - 원 화소의 값이나 위치를 바탕으로 단일 화소 값을 변경하는 기술
  - 다른 화소의 영향을 받지 않고 단순히 화소 점의 값만 변경하므로 포인트 처리(Point Processing)라고도 함.
  - 산술연산, 논리연산, 반전, 광도 보정, 히스토그램 평활화, 명암 대비 스트레칭 등의 기법이 있음.
  - 디지털 영상의 산술연산은 디지털 영상의 각 화소 값에서 임의의 상수 값으로 덧셈, 뺄셈, 곱셈, 나눗셈을 수행하는 것
  - 그레이 레벨 영상에서 화소 값이 작으면 영상이 어둡고, 화소의 값이 크면 밝음.

산술연산과 논리연산

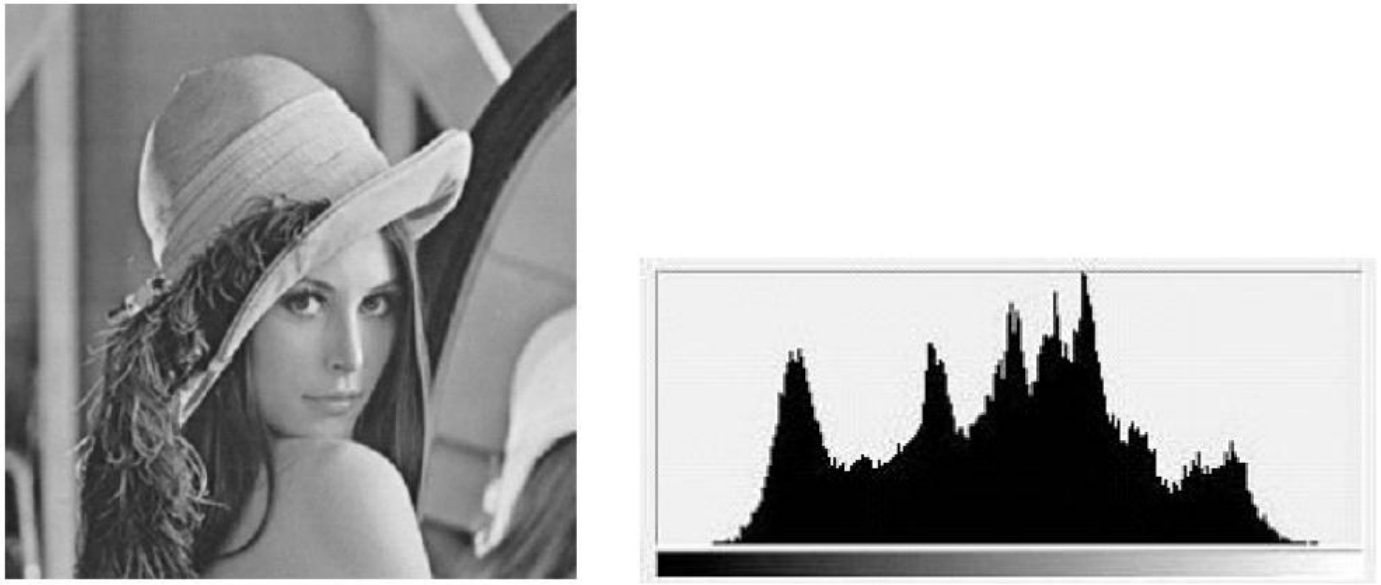
- 산술연산
  - 밝기 조정과 관련된 작업 수행

[그림 4-2] 디지털 영상의 밝기 변화

- 논리연산
  - 참과 거짓을 판별하는 연산
  - 화소의 상수 값에서 **AND, OR, XOR, NOT** 등의 연산을 수행하여 디지털 영상에서 차폐, 특징 추출, 형태 분석을 함.

히스토그램(Histogram)

- 히스토그램(Histogram)
  - 기둥그래프나 기둥 모양 그림이라고도 하며, 관측한 데이터가 분포된 특징을 한눈에 볼 수 있도록 기둥 모양으로 나타낸 것.
  - 가로축에는 레벨(Level)을, 세로축에는 각 레벨의 빈도수를 표시함.
  - 즉, 가로축은 영상의 밝기(Intensity) 값, 세로축은 가로축의 밝기 값에 대응하는 디지털 영상 내의 화소 수



[그림 4-3] 디지털 영상의 밝기 히스토그램

히스토그램 평활화와 명세화

- 히스토그램 평활화 기술
  - 편중된 디지털 영상의 히스토그램을 골고루 분산시켜 영상 전체의 명암 대비를 높여줌.
- 히스토그램 명세화 기술
  - 디지털 영상이 원하는 히스토그램을 갖게 해주는 기술.
  - 특정 부분의 명암 대비를 높일 수 있음.

[그림 4-4] 디지털 영상의 명암 대비 변화

- 화소의 밝기 값
  - 밝기의 단계 수는 화소를 표현하는 양자화 비트 수가 결정
  - 그레이 레벨 영상에서는 색은 없고 밝기만 있음.
  - 보통, 화소는 밝기를 나타내는데, 주로 양자화 비트 수를 8비트로 표현

- 명암 대비
  - 대비(Contrast): 영상 내에 있는 가장 밝은 값과 가장 어두운 값의 차이로, 영상의 품질을 결정하는 중요한 요소임.
  - 높은 대비를 보이는 디지털 영상: 어두운 명도와 밝은 명도의 차이가 너무 커서 시각적으로 좀더 명확하게 보임.
  - 낮은 대비를 보이는 디지털 영상: 밝기의 차이가 크지 않아 시각적으로 명확하지 못함.

디지털 영상의 산술연산

- 화소 값의 덧셈연산
  - 화소의 밝기 값에 특정한 상수 값을 더해 화소의 밝기 값을 증가시켜 영상을 밝게 하는 처리 기술

> 화소 + α : 영상의 밝기 증가 = 밝아짐

  - 화소의 값에 임의의 상수를 더할 때 화소의 최대값을 넘기도 함.
  - 최대값인 255를 넘는 값은 모두 255로 처리

> (화소 값 + α) 〉 255이면, (화소 값 + α) = 255

디지털 영상의 산술연산(계속)

(a) 원본 영상

(b) 상수 값 10을 더한 영상

(c) 상수 값 50을 더한 영상

(d) 상수 값 100을 더한 영상

[그림 4-5] 덧셈 상수의 변화에 따른 디지털 영상의 밝기 증가

디지털 영상의 산술연산(계속)

- 화소 값의 뺄셈연산
  - 화소의 밝기 값에 특정한 상수 값을 빼 화소의 밝기 값을 감소시켜 영상의 밝기를 어둡게 하는 처리 기술

> 화소 $-\alpha$: 영상의 밝기 감소 = 어두워짐

  - 화소의 값에 임의의 상수를 뺄 때 화소의 최소값 0보다도 작은 음수가 발생할 수 있음.
  - 화소의 최소값인 0보다 작은 음수 값은 모두 0으로 처리

> (화소 값 $-\alpha$) $<$ 0이면, (화소 값 $+\alpha$) = 0

디지털 영상의 산술연산(계속)

- 화소 값의 곱셈연산
  - 화소의 밝기 값에 특정 상수 값을 곱해 전체적으로 화소의 밝기 값이 증가해 더 밝아짐.

> 화소 $* \alpha$: 영상의 밝기 차이 증가 = 뚜렷해짐

  - 밝은 부분은 더욱 밝아지고, 어두운 부분은 약간 밝아져 영상 내의 밝기에 커다란 차이가 생기는 것
  - 밝기의 차이가 커지므로 영상의 선명도 증가함.

- 화소 값의 나눗셈연산
  - 화소 값을 임의의 상수 값으로 나누면 전체적으로 화소의 밝기 값은 감소하고, 최대 밝기와 최소 밝기의 차이는 작아짐.
  - 밝은 부분은 많이 어두워지고, 어두운 부분은 약간 어두워짐.

> 화소 $/ \alpha$: 영상의 밝기 차이 감소 = 희미해짐

디지털 영상의 산술연산(계속)

(a) 원본 영상

(b) 상수 1.3을 곱한 영상

(c) 상수 1.5를 곱한 영상

(d) 상수 1.7을 곱한 영상

[그림 4-7] 곱셈 상수 변화에 따른 디지털 영상의 명도 대비 향상

디지털 영상의 산술연산(계속)

(a) 원본 영상

(b) 상수 1.3으로 나눈 영상

(c) 상수 1.5로 나눈 영상

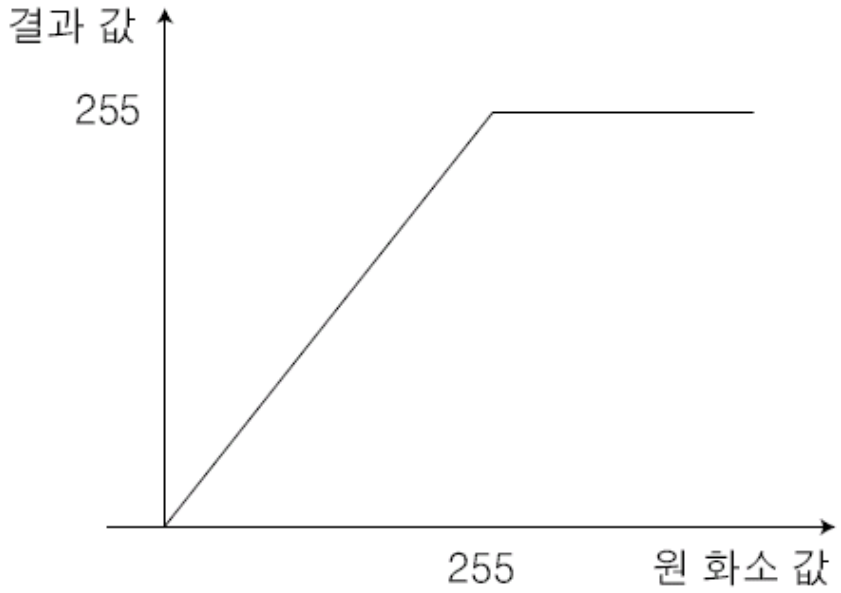
(d) 상수 1.7로 나눈 영상

[그림 4-8] 나눗셈 상수 변화에 따른 디지털 영상의 명도 대비 향상

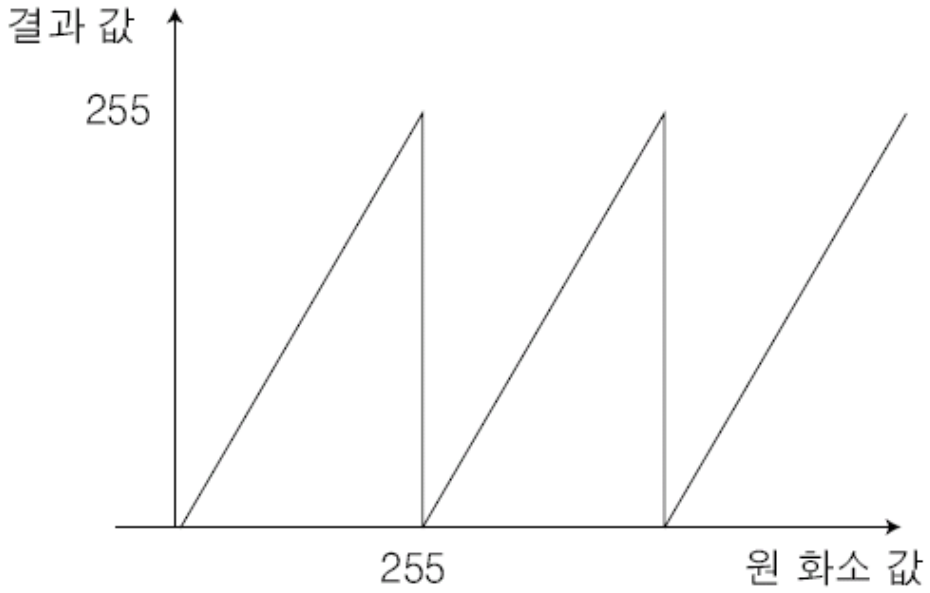
산술연산의 문제점과 해결 방법

- 문제점
  - 결과 값이 화소의 최대값과 최소값을 넘을 수 있음.
- 해결 방법
  - 클래핑(Clamping) 기법
    - 연산의 결과 값이 최소값보다 작으면 그 결과 값을 최소값으로, 최대값보다 크면 결과 값을 최대값으로 하는 기법
    - 8비트 그레이 영상의 최소값은 0, 최대값은 255
    - 음수는 0으로 설정하고, 255보다 큰 값은 255로 설정함.
  - 랩핑(Wraping) 기법
    - 연산의 결과 값이 최소값보다 작으면 그 결과 값을 최소값으로, 최대값보다 크면 최소값부터 최대값까지를 한 주기로 해서 이를 반복하는 기법
    - 최대값+1은 최소값이 되고, 연산의 결과 값이 최대값+상수 값일 때는 계속 상수 값-1로 설정함.
    - 8비트 그레이 영상의 최소값은 당연히 0이고, 최대값은 255
    - 음수는 0으로, 255보다 큰 결과 값 256은 0으로, 257은 1로 설정한 후 이런 방식으로 주기를 계속 반복

산술연산의 문제점과 해결 방법(계속)



(a) 클래핑 기법



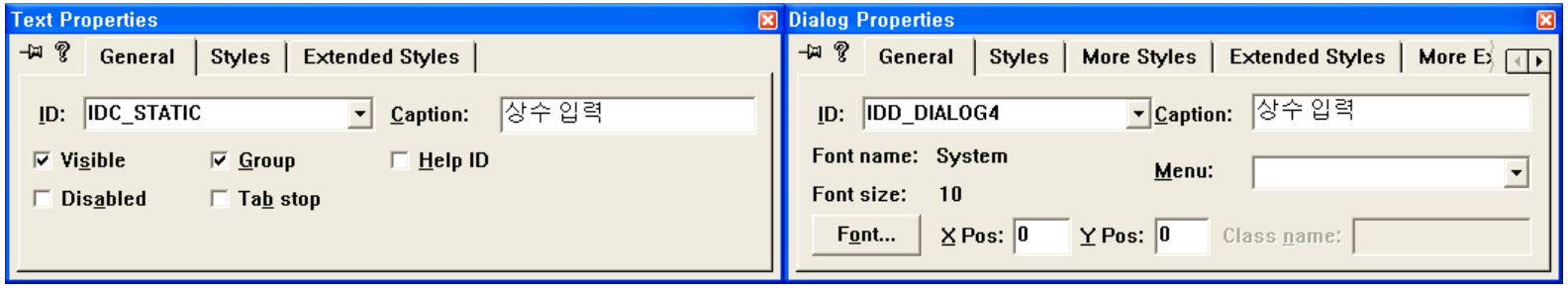
(b) 랩핑 기법

[그림 4-9] 산술연산의 문제점 해결 기법

[실습하기 4-1] 상수 값 입력 대화상자

① **Visual C++ 프로그램의 Workspace 창에서 [ResourceView] 탭을 클릭 → [ImageProcessing resources]-[Dialog] 폴더 클릭→ 바로가기 메뉴 [Insert Dialog]를 클릭해 대화상자 추가 → 대화상자를 편집, Text Properties와 Dialog Properties를 결정**

	ID	Caption
Text Properties	IDC_STATIC	상수 입력
Dialog Properties	IDD_DIALOG4	상수 입력
Edit Properties	IDC_EDIT1	



[실습하기 4-1] 상수 값 입력 대화상자

② **[View]-[ClassWizard] 메뉴 클릭→ [MFC ClassWizard] 대화상자를 이용해 생성된 대화상자에 새로운 클래스 등록**

Name	CConstantDlg

③ **생성된 edit box에 변수 할당: [MFC ClassWizard] 대화상자의 Object IDs 항목에서 CConstantDlg Class을 선택한 뒤 [Member Variables] 탭**

→ Control Ids 항목에서 ID_EDIT1을 선택하고 [Add Variable] 버튼을 클릭

→ [Add Member Variable] 대화상자에서 다음과 같이 실수형 변수를 선언

→ [OK] 버튼 클릭

Control IDs	Member variable name	Category	Variable type	Minimum Value	Maximum Value
IDC_EDIT1	m_Constant	Value	double	0	255

[실습하기 4-1] 상수 값 입력 대화상자

④ **Doc 클래스에서 사용하기 위해 ConstantDlg.h 선언**

```
#include "ImageProcessingDoc.h"
#include "DownSampleDlg.h"
#include "UpSampleDlg.h"
#include "QuantizationDlg.h"
#include "math.h"
#include "ConstantDlg.h" // 상수 입력 대화상자 사용을 위한 헤더 선언
```

[실습하기 4-2] 상수 값 상수 덧셈 프로그램

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

ID	ID_SUM_CONSTANT
Caption	화소 값 덧셈

② **[MFC ClassWizard]** 대화상자를 이용해 추가된 메뉴에서 화소 값의 덧셈을 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnSumConstant
Doc Class	void	OnSumConstant

③ **Doc** 클래스의 **OnSumConstant** 함수에 미리 작성한 **[상수 입력]** 대화상자를 호출하여 상수 값을 입력받아 화소 값의 덧셈연산을 수행하는 프로그램 작성

[실습하기 4-2] 상수 값 상수 덧셈 프로그램

```
void CImageProcessingDoc::OnSumConstant()
{

   CConstantDlg dlg; // 상수 값을 입력받는 대화상자

   int i;

   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){
     for(i=0 ; i<m_size ; i++){
       if(m_InputImage[i] + dlg.m_Constant >= 255)
          m_OutputImage[i] = 255;
       // 출력 값이 255보다 크면 255 출력
       else
       m_OutputImage[i]=(unsigned char)(m_InputImage[i] + dlg.m_Constant);
       // 상수 값과 화소 값과의 덧셈
     }
  }
}
```

[실습하기 4-2] 상수 값 상수 덧셈 프로그램

④ View 클래스의 OnSumConstant 함수에 Doc 클래스의 OnSumConstant 함수를 호출하여 화면에 출력하는 프로그램 작성

```
void CImageProcessingView::OnSumConstant()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    // 도큐먼트 클래스 참조
    ASSERT_VALID(pDoc); // 인스턴스 주소를 가져옴

    pDoc->OnSumConstant();

    Invalidate(TRUE);
}
```

[실습하기 4-2] 상수 값 상수 덧셈 프로그램

⑤ 프로그램 실행 결과 영상

상수 덧셈 프로그램 구현 결과[원본 영상(왼쪽), 원본 영상+30(오른쪽)]

[실습하기 4-3] 화소 값의 상수 빨셈 프로그램

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

ID	ID_SUB_CONSTANT
Caption	화소 값 빨셈

② **[MFC ClassWizard]** 대화상자를 이용해 추가된 메뉴에서 화소 값의 빨셈을 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnSubConstant
Doc Class	void	OnSubConstant

③ **Doc** 클래스의 **OnSubConstant** 함수에 미리 작성한 **[상수 입력]** 대화상자를 호출하여 상수 값을 입력받아 화소 값의 빨셈연산을 수행하는 프로그램 작성

[실습하기 4-3] 화소 값의 상수 뺄셈 프로그램

```
void CImageProcessingDoc::OnSubConstant()
{
   CConstantDlg dlg;

   int i;

   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){
      for(i=0 ; i<m_size ; i++){
         if(m_InputImage[i] - dlg.m_Constant < 0)
                  m_OutputImage[i] = 0; // 출력 값이 255보다 크면 255를 출력
         else
            m_OutputImage [i]
                  = (unsigned char)(m_InputImage[i] - dlg.m_Constant);
         // 상수 값과 화소 값과의 뺄셈
      }
   }
}
```

[실습하기 4-3] 화소 값의 상수 뺄셈 프로그램

④ View 클래스의 OnSubConstant 함수에 Doc 클래스의 OnSubnstant 함수를 호출하여 화면에 출력하는 프로그램

```
void CImageProcessingView::OnSubConstant()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();// 도큐먼트 클래스 참조
    ASSERT_VALID(pDoc); // 인스턴스 주소를 가져옴

    pDoc->OnSubConstant();

    Invalidate(TRUE);
}
```

[실습하기 4-3] 화소 값의 상수 뺄셈 프로그램

⑤ 프로그램을 실행 결과 영상

상수 뺄셈 프로그램 구현 결과[원본 영상(왼쪽), 원본 영상-50(오른쪽)]

[실습하기 4-4] 화소 값의 상수 곱셈 프로그램

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

ID	ID_MUL_CONSTANT
Caption	화소 값 곱셈

② **[MFC ClassWizard]** 대화상자에서 추가된 메뉴에서 화소 값의 곱셈을 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnMulConstant
Doc Class	void	OnMulConstant

③ **Doc** 클래스의 **OnMulConstant** 함수에 미리 작성한 **[상수 입력]** 대화상자를 호출하여 상수 값을 입력받아 화소 값의 곱셈연산을 수행하는 프로그램 작성

[실습하기 4-4] 화소 값의 상수 곱셈 프로그램

```
void CImageProcessingDoc::OnMulConstant()
{
    CConstantDlg dlg;

    int i;

    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){
        for(i=0 ; i<m_size ; i++){
            if(m_InputImage[i] * dlg.m_Constant > 255)
                    m_OutputImage[i] = 255;
            // 곱의 값이 255보다 크면 255를 출력
            else if(m_InputImage[i] * dlg.m_Constant < 0)
                    m_OutputImage[i] = 0;
            // 곱의 값이 0보다 작으면 0을 출력
            else
                m_OutputImage [i]
                      = (unsigned char)(m_InputImage[i] * dlg.m_ Constant);
            // 상수 값과 화소 값 곱셈
        }
    }
}
```

[실습하기 4-4] 화소 값의 상수 곱셈 프로그램

④ View 클래스의 OnMulConstant 함수에 Doc 클래스의 OnMulConstant 함수를 호출하여 화면에 출력하는 프로그램 작성

```
void CImageProcessingView::OnMulConstant()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument(); // 도큐먼트 클래스 참조
    ASSERT_VALID(pDoc); // 인스턴스 주소를 가져옴

    pDoc->OnMulConstant();

    Invalidate(TRUE);
}
```

[실습하기 4-4] 화소 값의 상수 곱셈 프로그램

⑤ 프로그램을 실행 결과 영상

상수 곱셈 프로그램 구현 결과[원본 영상(왼쪽), 원본 영상×2(오른쪽)

[실습하기 4-5] 화소 값의 상수 나눗셈 프로그램

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

ID	ID_DIV_CONSTANT
Caption	화소 값 나눗셈

② **[MFC ClassWizard]** 대화상자에서 추가된 메뉴에서 화소 값의 나눗셈을 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnDivConstant
Doc Class	void	OnDivConstant

③ **Doc** 클래스의 **OnDivConstant** 함수에 미리 작성한 **[상수 입력]** 대화상자를 호출하여 상수 값을 입력받아 화소 값의 나눗셈연산을 수행하는 프로그램

[실습하기 4-5] 화소 값의 상수 나눗셈 프로그램

```
void CImageProcessingDoc::OnDivConstant()
{
    CConstantDlg dlg;

    int i;

    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){
        for(i=0 ; i<m_size ; i++){
            if(m_InputImage[i] / dlg.m_Constant > 255)
                    m_OutputImage[i] = 255;
            // 나눗셈의 값이 255보다 크면 255를 출력
            else if(m_InputImage[i] / dlg.m_Constant < 0)
                    m_OutputImage[i] = 0;
            // 나눗셈의 값이 0보다 작으면 0을 출력
            else
                m_OutputImage [i]
                        = (unsigned char)(m_InputImage[i] / dlg.m_Constant);
            // 상수 값과 화소 값 나눗셈
        }
    }
}
```

[실습하기 4-5] 화소 값의 상수 나눗셈 프로그램

④ View 클래스의 OnDivConstant 함수에 Doc 클래스의 OnDivConstant 함수를 호출하여 화면에 출력하는 프로그램 작성

```
void CImageProcessingView::OnDivConstant()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument(); // 도큐먼트 클래스 참조
    ASSERT_VALID(pDoc); // 인스턴스 주소를 가져옴

    pDoc->OnDivConstant();

    Invalidate(TRUE);
}
```

[실습하기 4-5] 화소 값의 상수 나눗셈 프로그램

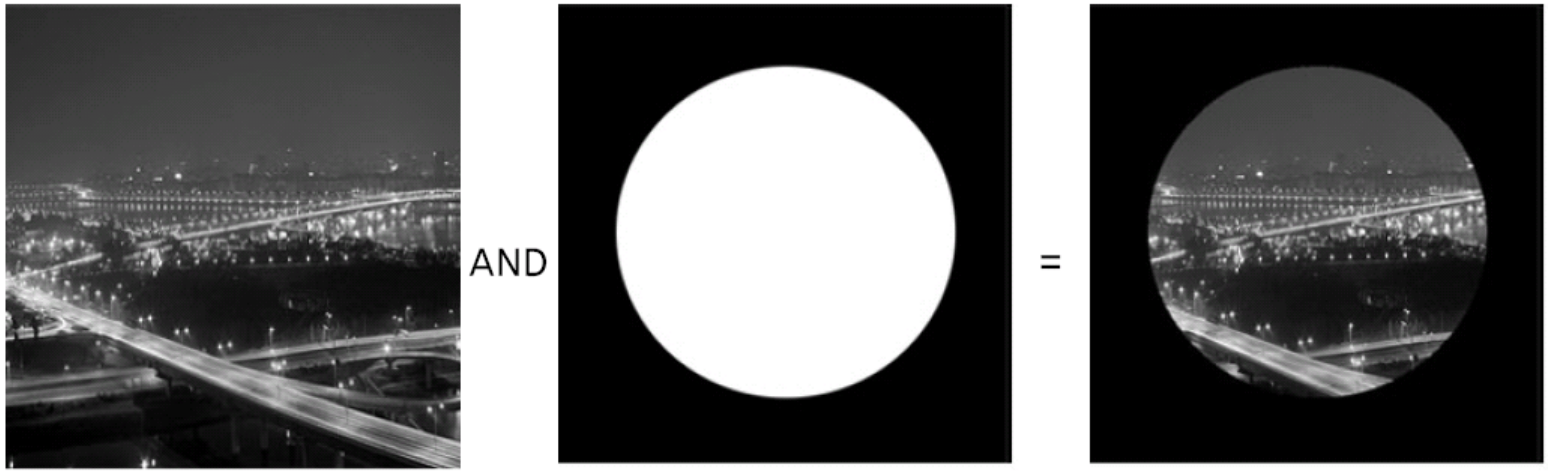
⑤ 프로그램 실행 결과 영상

상수 나눗셈 프로그램 구현 결과[원본 영상(왼쪽), 원본 영상/2(오른쪽)]

디지털 영상의 논리연산

- 화소 값의 AND 연산
  - 원하는 비트를 선택적으로 0으로 만드는 기능이 있어 마스크(mask) 연산이라고도 함
  - 영상의 특정 화소 비트에서 0으로 구성된 이진 데이터와 AND 연산을 수행

화소 비트 :	1 0 1 1	0 1 0 1	연산 전
이진 데이터 :	0 0 0 0	1 1 1 1	마스크(AND) 연산
마스크 결과 :	0 0 0 0	0 1 0 1	연산 후

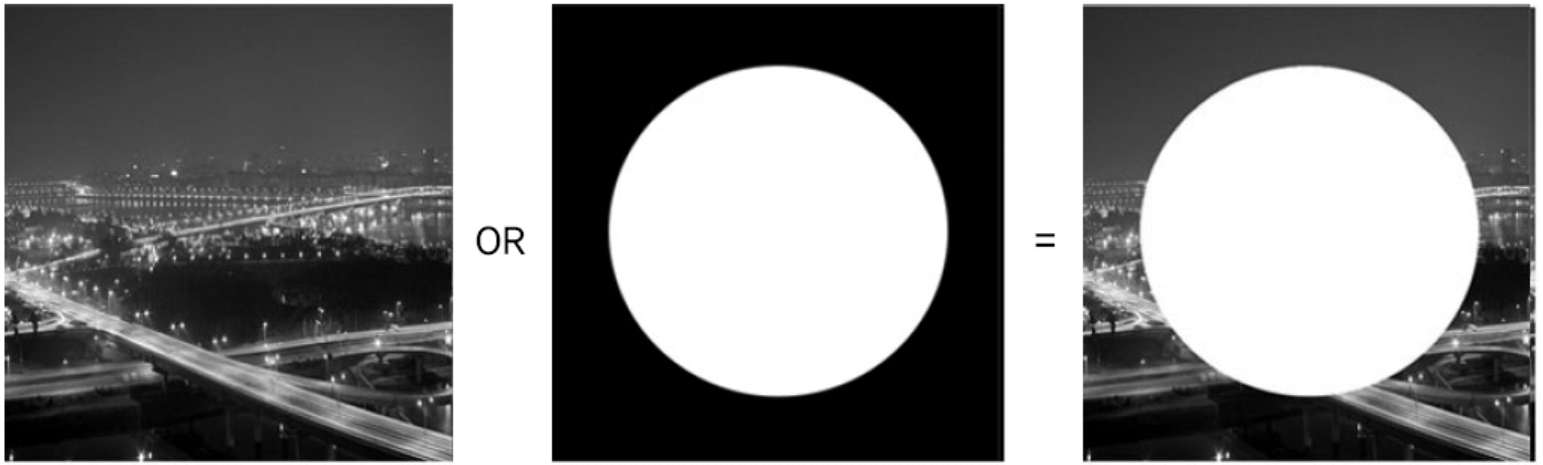


[그림 4-10] 영상에서 AND 연산 수행

디지털 영상의 논리연산(계속)

- 화소 값의 OR 연산
  - 특정 비트를 선택적으로 1로 구성할 수 있어 선택적-세트(selective-set)연산이라고도 함
  - 화소의 특정 비트를 1로 만들려고 원하는 비트 위치가 1로 구성된 이진 데이터와 OR 연산을 수행

화소 비트 :	1 0 0 1 0 0 1 0	연산 전
이진 데이터 :	0 0 0 0 1 1 1 1	선택적-세트(OR)연산
선택적 세트 결과 :	1 0 0 1 1 1 1 1	연산 후



[그림 4-11] 영상에서 OR 연산 수행

디지털 영상의 논리연산(계속)

- 화소 값의 XOR 연산
  - 입력이 서로 다를 때만 1을 출력하는 연산으로, 두 데이터를 비교하므로 비교(compare)연산이라고도 함
  - 같은 비트에서만 0을 출력함

화소 비트 :	1 1 1 0 0 0 0 1	연산 전
비교 데이터 :	0 1 0 1 1 0 0 1	비교(XOR)연산
비교 결과 :	1 0 1 1 1 0 0 0	연산 후

[그림 4-12] 그레이 영상을 상수 값 128로 XOR 연산 수행

디지털 영상의 논리연산(계속)

- 화소 값의 NOT 연산
  - 화소 비트를 반전시키는 일을 함.
  - 영상에서는 검정색이 흰색으로, 흰색이 검정색으로 반전됨

화소 비트 :	1 0 0 1	0 0 1 0	연산 전
반전 화소 비트 :	0 1 1 0	1 1 0 1	반전(NOT)연산

[그림 4-13] 영상에서 NOT 연산 수행

[실습하기 4-6] 화소 값의 AND 프로그램

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

ID	ID_AND_OPERATE
Caption	AND 연산

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnAndOperate
Doc Class	void	OnAndOperate

[실습하기 4-6] 화소 값의 AND 프로그램

③ **Doc 클래스의 OnAndOperate 함수에 다음 프로그램 추가**

```
void CImageProcessingDoc::OnAndOperate()
{
    CConstantDlg dlg;
    int i;

    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){
        for(i=0 ; i<m_size ; i++){
            // 비트 단위 AND 연산
            if((m_InputImage[i] & (unsigned char)dlg.m_Constant)>= 255)
                                {m_OutputImage[i] = 255;
            }
            else if((m_InputImage[i] & (unsigned char)dlg.m_Constant)< 0)
                                {m_OutputImage[i] = 0;
            }
            else{
                 m_OutputImage [i] = (m_InputImage[i]
                                  & (unsigned char)dlg.m_Constant);
            }
        }
    }
}
```

[실습하기 4-6] 화소 값의 AND 프로그램

④ View 클래스의 OnAndOperate 함수에 다음 프로그램 추가

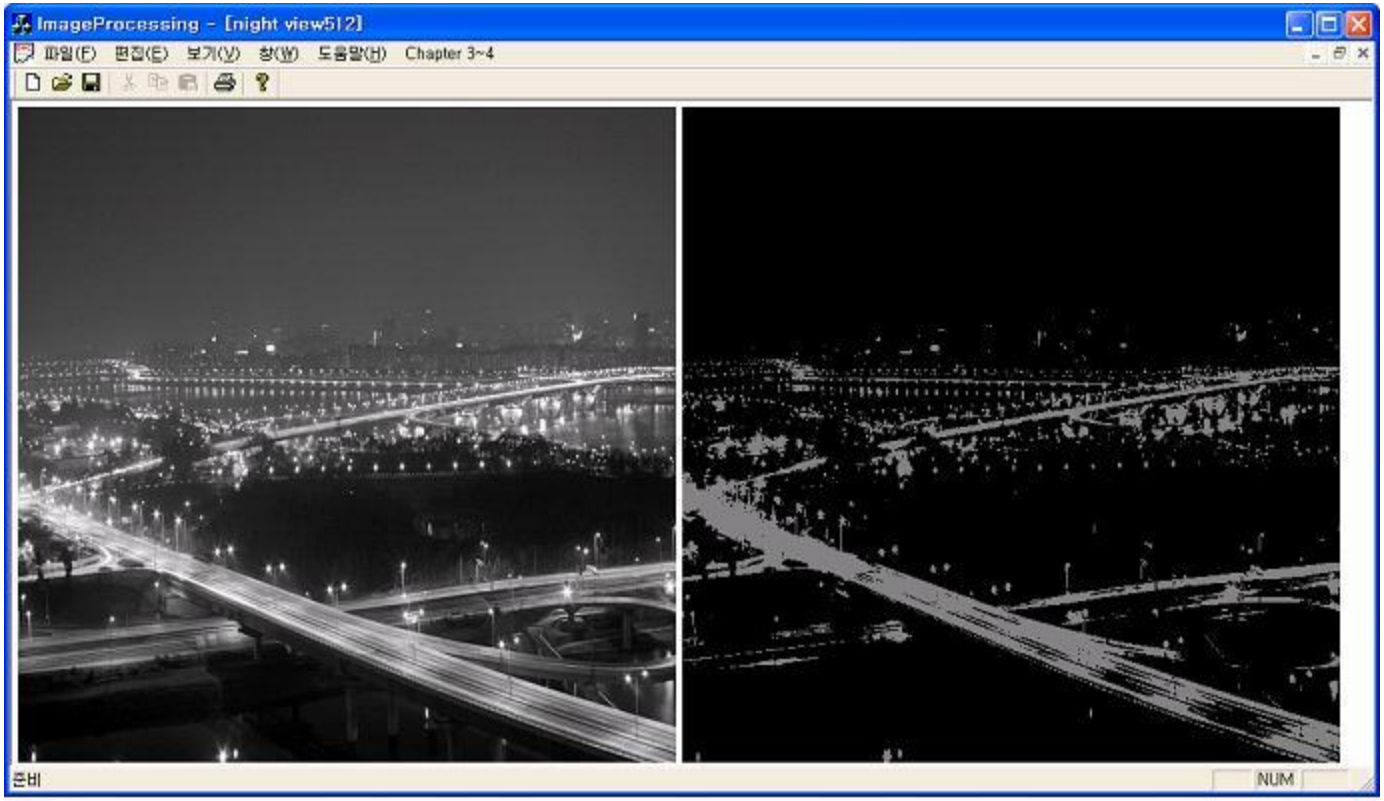
```
void CImageProcessingView::OnAndOperate()
{
// TODO: Add your command handler code here
CImageProcessingDoc* pDoc = GetDocument();
ASSERT_VALID(pDoc);

   pDoc->OnAndOperate();

   Invalidate(TRUE);
}
```

[실습하기 4-6] 화소 값의 AND 프로그램

⑤ 프로그램 실행 결과 영상



영상의 화소 값에 상수 128을 AND 연산한 결과 영상

[실습하기 4-7] 화소 값의 OR 프로그램

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

ID	ID_OR_OPERATE
Caption	OR 연산

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnOrOperate
Doc Class	void	OnOrOperate

③ **Doc 클래스의 OnOrOperate 함수에 다음 프로그램 추가**

[실습하기 4-7] 화소 값의 OR 프로그램

```
void CImageProcessingDoc::OnOrOperate()
{
   CConstantDlg dlg;
   int i;

   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){
      for(i=0 ; i<m_size ; i++){
        // 비트 단위 OR 연산
        if((m_InputImage[i] | (unsigned char)dlg.m_ Constant) >= 255){
                        m_OutputImage[i] = 255;
        }
        else if((m_InputImage[i] |(unsigned char)dlg.m_ Constant) < 0){
                        m_OutputImage[i] = 0;
        }
       else{
            m_OutputImage [i] = (m_InputImage[i] |
                        unsigned char)dlg.m_Constant);
       }
     }
  }
}
```

[실습하기 4-7] 화소 값의 OR 프로그램

④ View 클래스의 OnOrOperate 함수에 다음 프로그램 추가

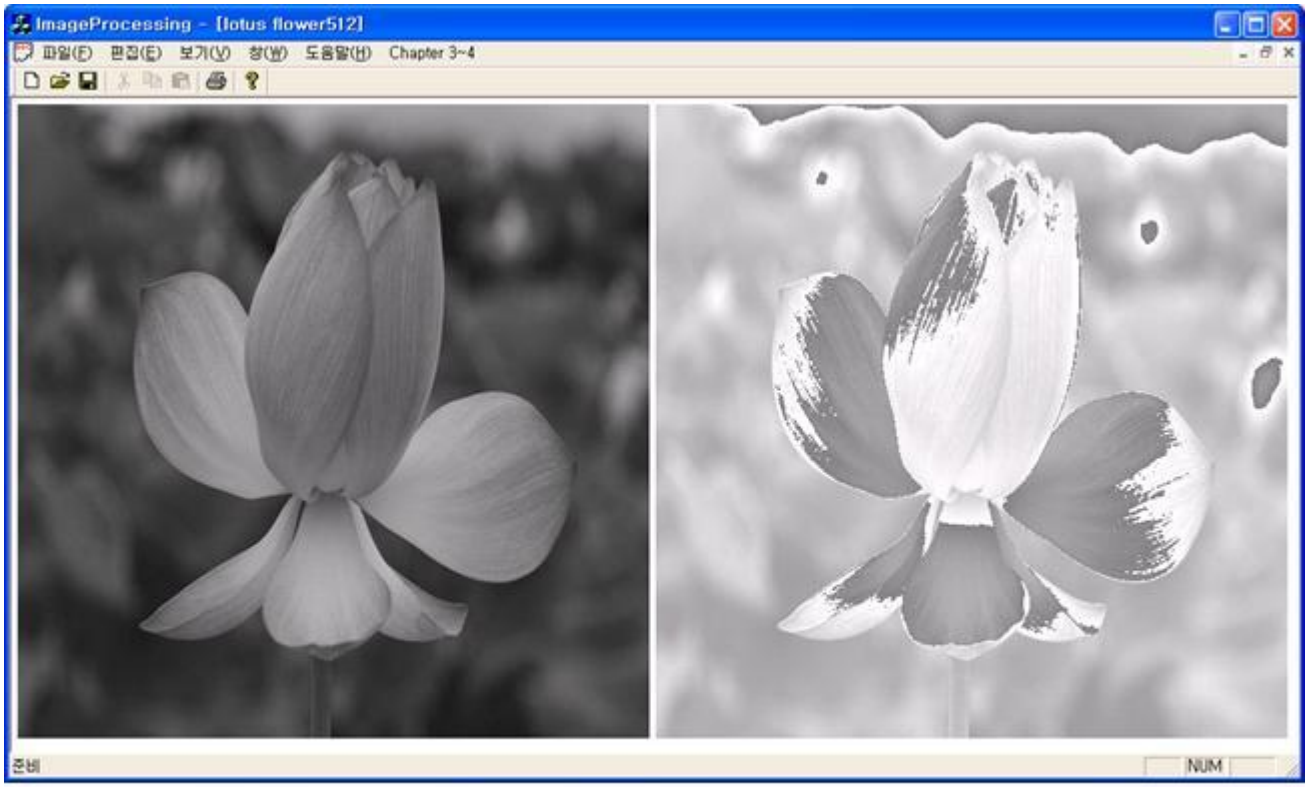
```
void CImageProcessingView::OnOrOperate()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnOrOperate();

    Invalidate(TRUE);
}
```

[실습하기 4-7] 화소 값의 OR 프로그램

⑤ 프로그램 실행 결과 영상



영상의 화소 값에 상수 128을 OR 연산한 결과 영상

[실습하기 4-8] 화소 값의 XOR 프로그램

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

ID	ID_XOR_OPERATE
Caption	XOR 연산

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnXorOperate
Doc Class	void	OnXorOperate

③ **Doc 클래스의 OnXorOperate 함수에 다음 프로그램 추가**

[실습하기 4-8] 화소 값의 XOR 프로그램

```
void CImageProcessingDoc::OnXorOperate()
{
    CConstantDlg dlg;
    int i;

    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){
        for(i=0 ; i<m_size ; i++){
            // 비트 단위 XOR 연산
            if((m_InputImage[i] ^ (unsigned char)dlg.m_Constant) >= 255){
                  m_OutputImage[i] = 255;
            }
            else if((m_InputImage[i] ^ (unsigned char)dlg.m_ Constant) < 0){
                  m_OutputImage[i] = 0;
            }
            else{
                m_OutputImage [i] = (m_InputImage[i]
                                     ^ (unsigned char)dlg.m_Constant);
            }
        }
    }
}
```

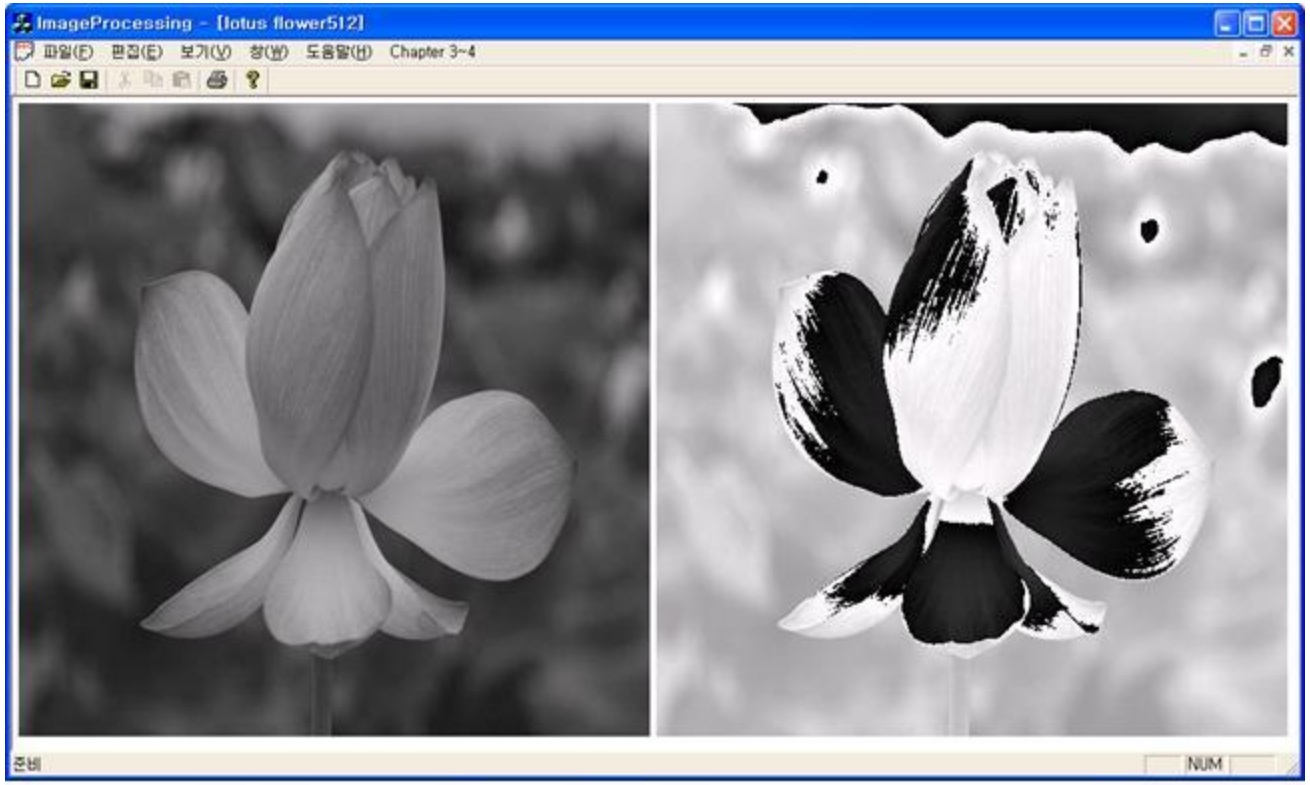
[실습하기 4-8] 화소 값의 XOR 프로그램

④ View 클래스의 OnXorOperate 함수에 다음 프로그램 추가

```
void CImageProcessingView::OnXorOperate()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    pDoc->OnXorOperate();
    Invalidate(TRUE);
}
```

[실습하기 4-8] 화소 값의 XOR 프로그램

⑤ 프로그램 실행 결과 영상



영상의 화소 값에 상수 128을 XOR 연산한 결과 영상

- 화소 점 처리 기법

$$Output(q) = T[Input(p)]$$

  - p는 입력 영상의 화소 값이고, T로 화소 값을 변환함
  - q는 값을 변환하여 얻는 출력 화소 값

- 명암 변환(Intensity Transform)
  - 밝기를 변경하는 것
  - 미리 지정된 변환 함수를 기반으로 입력 영상의 이전 화소를 새로운 화소로 변환하는 점 처리 기법

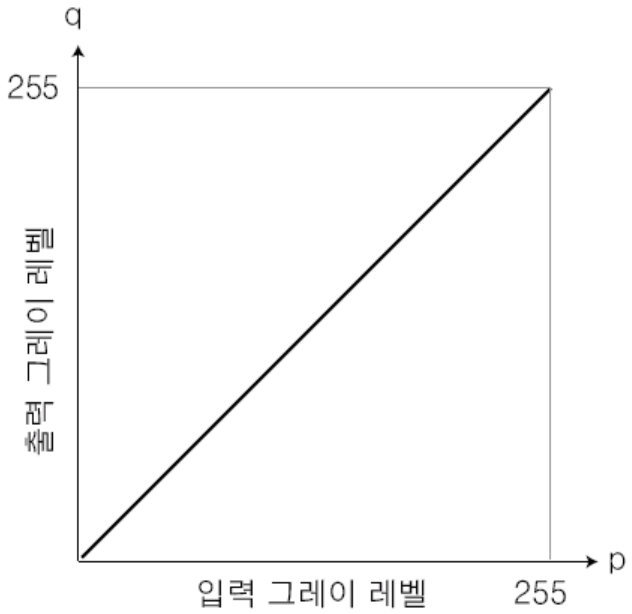
널 변환(Null Transform)

- 입력 영상을 출력 영상으로 변환해도 변화가 없는 것
- 단순히 입력 화소를 출력 화소로 바꾸는 변환
- 널 변환의 변환 함수

$$\text{Output}(q) = \text{Input}(p)$$

(a) 입력 영상

(b) 널 변환 함수 그래프



(c) 출력 영상

[그림 4-14] 8비트 그레이 레벨 영상에서의 널 변환 함수 그래프와 출력 영상

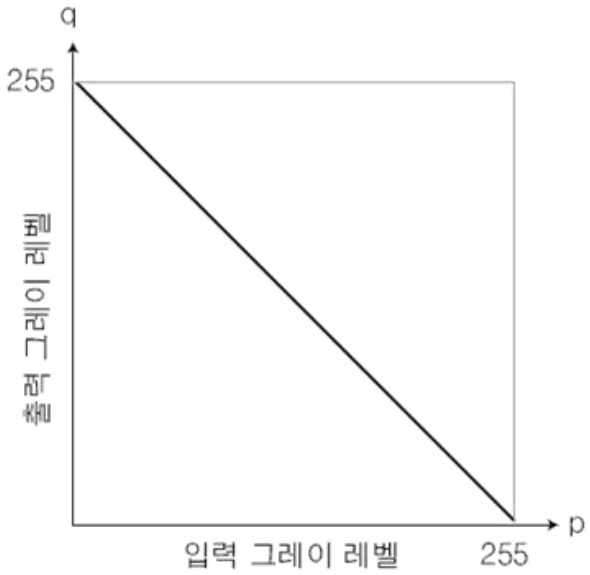
영상의 반전 변환(Negative Transform)

- 사진학적 역변환
- 각 화소의 값이 영상 내에 대칭이 되는 값으로 변환
- 8비트 그레이 레벨의 영상을 반전시키면 화소 값 0번은 255번으로, 화소 값 1번은 254번으로 변환됨
- 반전 변환의 변환 함수

$$Output(q) = 255 - Input(p)$$

(a) 입력 영상

(b) 반전 변환 함수 그래프



(c) 출력 영상

[그림 4-15] 8비트 그레이 레벨 영상에서의 반전 변환 함수 그래프와 출력 영상

감마 보정(Gamma Correction)

- 입력 값을 조정하여 출력을 제대로 만드는 과정
- 감마 보정 함수

$$\text{Output}(q) = [\text{Input}(p)](1/\gamma)$$

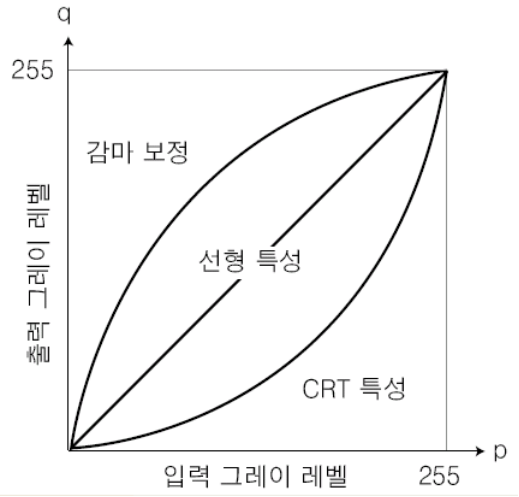
- 함수의 감마 값(γ)에 따라 영상을 밝게 하거나 흐리게 조절할 수 있음
- 감마 값이 1보다 크면 영상이 어두워지고, 1보다 작으면 영상이 밝아짐

(a) 입력 영상

(b) 감마 값(r) 0.8로 보정한 영상

(c) 감마 값(r) 1.2로 보정한 영상

(d) 감마 보정 변환 함수 그래프



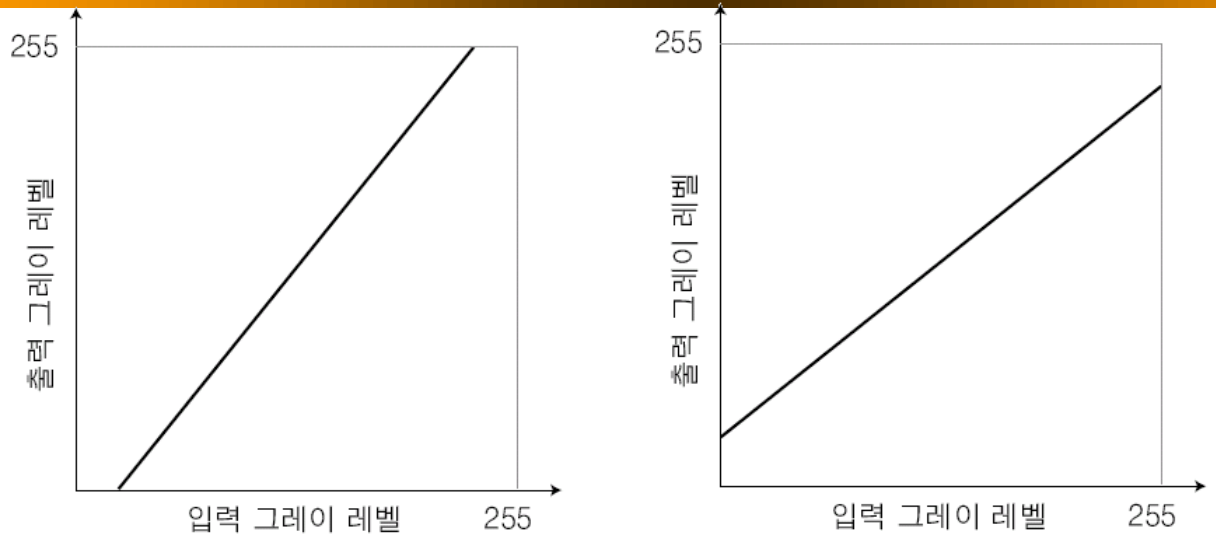
[그림 4-16] 감마 보정 영상과 감마 함수의 그래프

명암 대비 변환(Intensity Contrast Transform)

- 명암 대비 스트레칭(Intensity Contrast Stretch)
  - 밝기의 차이를 크게 하는 것
  - 영상의 가장 밝은 값을 최대 밝게, 가장 어두운 값을 최대 어둡게 설정하여 높은 명암 대비를 보이는 영상을 생성하는 것

- 명암 대비 압축(Intensity Contrast Compress)
  - 영상의 가장 어두운 값을 밝게, 가장 밝은 값을 어둡게 하여 밝기 차이를 줄임으로써 낮은 명암 대비를 보이는 영상을 생성하는 것

명암 대비 변환(Intensity Contrast Transform)(계속)



(a) 명암 대비 스트레칭 (b) 명암 대비 압축

[그림 4-17] 명암 대비 변환 함수 그래프

(a) 입력 영상 (b) 명암 스트레칭된 영상 (c) 명암 대비가 압축된 영상

[그림 4-18] 명암 대비 변환 결과 영상

경계 값을 이용한 처리

- 경계 값을 이용한 처리
  - 디지털 영상의 화소 값을 주어진 경계 값으로 그룹화하여 결국 화소 값의 수를 감소시키는 처리 방법

포스터라이징 (Posterizing)

- 영상에서 화소에 있는 명암 값의 범위를 경계 값으로 축소
- 경계 값 8개로 8비트 그레이 레벨 영상을 포스터라이징 처리하면, 명암 값 256개가 명암 값 8개로 변경됨.

(a) 입력 영상 (b) 포스터라이징 변환된 영상 (c) 포스터라이징 변환 함수 그래프

[그림 4-19] 포스터라이징의 결과 영상과 함수 그래프

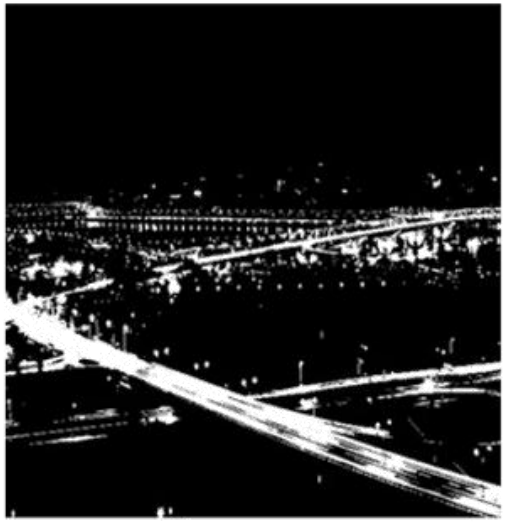
이진화 (Binarization)

- 경계 값을 이용해 값이 두 개만 있는 영상으로 변환해 주는것
- 보통 그레이 레벨 영상을 이진 영상으로 변환할 때 사용
- 값이 두 개뿐이라서 영상을 쉽게 분석할 수 있고, 명암 대비가 매우 낮은 영상에서는 배경과 물체를 확실하게 구분할 수 있게 해줌.

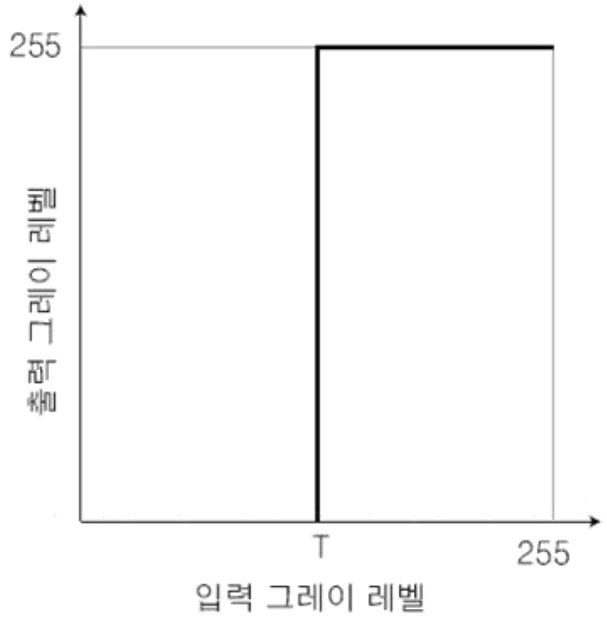
$$Output(q) = \begin{cases} 255 & Input(p) \geq T \\ 0 & Input(p) < T \end{cases}$$

(a) 입력 영상

(b) 이진화된 영상



(c) 이진화 함수 그래프



[그림 4-20] 이진화 함수 그래프와 이진화 영상

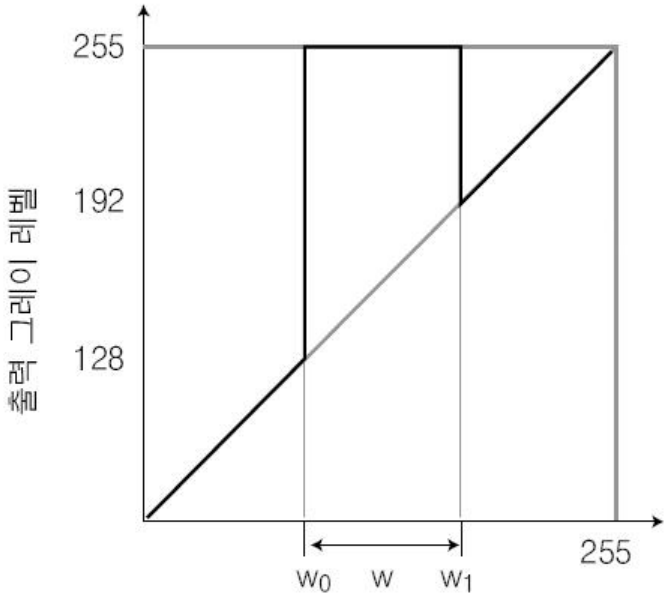
범위 강조 변환

- 영상에서 한 부분의 화소는 원 상태를 그대로 유지한 채 일정 범위의 화소만 강조하는 변환
- 원하는 부분의 화소 값이 더 커지거나 작아져 다른 부분과 비교해서 더욱 도드라져 보임.

(a) 입력 영상

(b) 범위 강조된 영상

(c) 범위 강조 함수 그래프



[그림 4-21] 범위 강조 변환된 영상과 함수 그래프

[실습하기 4-9] 영상을 반전시키는 프로그램

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

ID	ID_NEGA_TRANSFORM
Caption	영상 반전 변환

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnNegaTransform
Doc Class	void	OnNegaTransform

[실습하기 4-9] 영상을 반전시키는 프로그램

③ **Doc 클래스의 OnNegaTransform 함수에 다음 프로그램 추가**

```
void CImageProcessingDoc::OnNegaTransform()
{
   int i;

   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];

   for(i=0 ; i<m_size ; i++)
      m_OutputImage[i] = 255 - m_InputImage[i]; // 영상 반전을 수행
}
```

[실습하기 4-9] 영상을 반전시키는 프로그램

④ View 클래스의 OnNegaTransform 함수에 다음 프로그램 추가

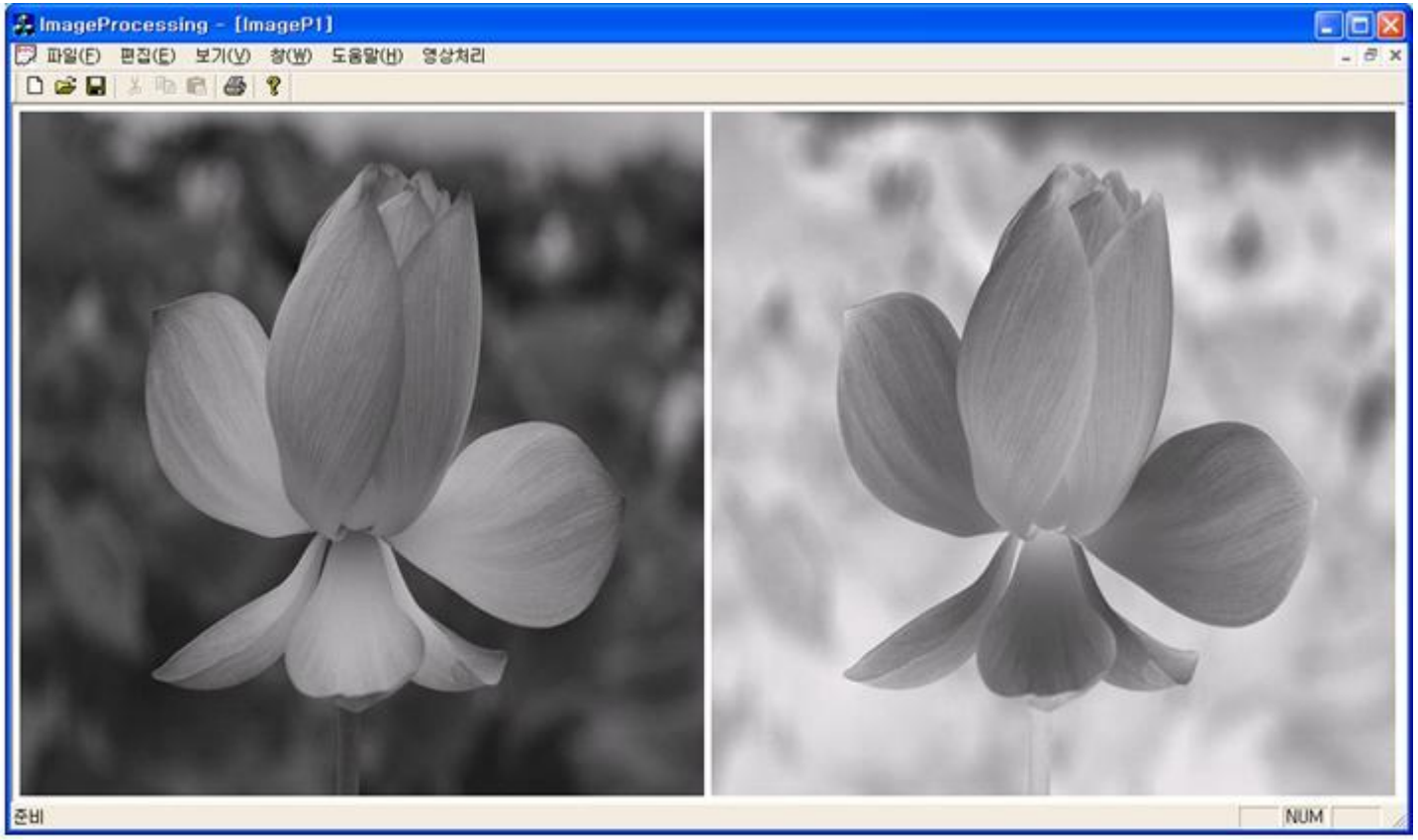
```
void CImageProcessingView::OnNegaTransform()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnNegaTransform();

    Invalidate(TRUE);
}
```

[실습하기 4-9] 영상을 반전시키는 프로그램

⑤ 프로그램 실행 결과 영상



영상 반전 프로그램을 구현한 결과 영상

[실습하기 4-10] 감마 보정 프로그램

- 감마 보정하는 함수 : **Output(q) = [Input(p)](1/Γ)**
  - **Γ값은 영상의 산술연산에서 사용한 [상수 입력] 대화상자를 이용하여 입력받도록 함.**

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

ID	ID_GAMMA_CORRECTION
Caption	감마보정

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnGammaCorrection
Doc Class	void	OnGammaCorrection

[실습하기 4-10] 감마 보정 프로그램

③ **Doc 클래스의 OnGammaCorrection 함수에 다음 프로그램 추가**

```
void CImageProcessingDoc::OnGammaCorrection()
{
   CConstantDlg dlg;

   int i;
   double temp;

   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){
      for(i=0 ; i<m_size ; i++){
         temp = pow(m_InputImage[i], 1/dlg.m_Constant);
         // 감마 값 계산
      if(temp < 0)
         m_OutputImage[i] = 0;
      else if(temp > 255)
         m_OutputImage[i] = 255;
      else
         m_OutputImage[i] = (unsigned char)temp;
      }
   }
}
```

[실습하기 4-10] 감마 보정 프로그램

④ View 클래스의 OnGammaCorrection 함수에 다음 프로그램 추가

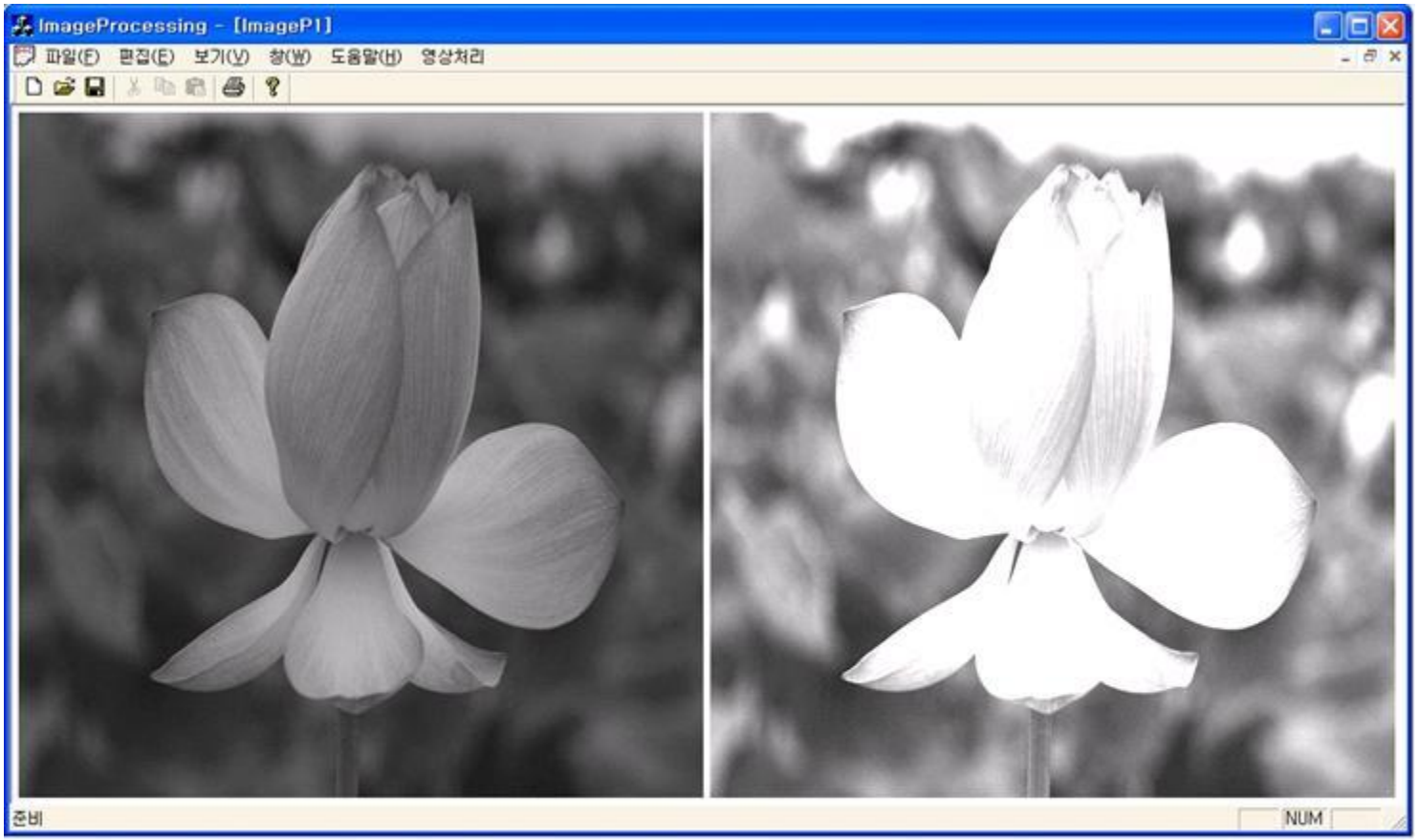
```
void CImageProcessingView::OnGammaCorrection()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnGammaCorrection();

    Invalidate(TRUE);
}
```

[실습하기 4-10] 감마 보정 프로그램

⑤ 프로그램 실행 결과 영상



감마 보정 프로그램을 구현한 결과 영상(r=0.85)

[실습하기 4-11] 영상의 이진 프로그램

- 영상의 이진화는 임의의 경계 값(Threshold)을 이용
- 임의의 경계 값은 영상의 산술연산에서 사용한 [상수 입력] 대화상자를 이용해 입력받도록 함

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

ID	ID_BINARIZATION
Caption	영상 이진화

② **[MFC ClassWizard] 대화상자를 이용해 Doc 클래스와 View 클래스에 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnBinarization
Doc Class	void	OnBinarization

[실습하기 4-11] 영상의 이진 프로그램

③ **Doc 클래스의 OnBinarization 함수에 다음 프로그램 추가**

```
void CImageProcessingDoc::OnBinarization()
{
   CConstantDlg dlg;

   int i;

   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){
      for(i=0 ; i<m_size ; i++){
         if(m_InputImage[i] >= dlg.m_Constant)
            m_OutputImage[i] = 255; // 임계 값보다 크면 255 출력
         else
            m_OutputImage[i] = 0; // 임계 값보다 작으면 0 출력
      }
   }
}
```

[실습하기 4-11] 영상의 이진 프로그램

④ View 클래스의 OnBinarization 함수에 다음 프로그램 추가

```
void CImageProcessingView::OnBinarization()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnBinarization();

    Invalidate(TRUE);
}
```

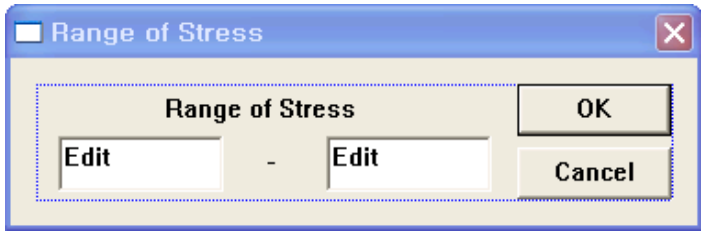
[실습하기 4-11] 영상의 이진 프로그램

⑤ 프로그램 실행 결과 영상

영상의 이진 프로그램을 실습한 결과 영상(경계 값=130)

[실습하기 4-12] 범위를 강조하는 변환 프로그램

① 화소 값의 범위를 선택하려고 다음과 같이 대화상자를 만듦.



	ID	Caption
Dialog Properties	IDD_DIALOG5	Range of Stress
Text Properties	IDC_STATIC	Range of Stress
Edit Properties	IDC_EDIT1	
Edit Properties	IDC_EDIT2	

[실습하기 4-12] 범위를 강조하는 변환 프로그램

② [View]-[ClassWizard] 메뉴 클릭→ [MFC ClassWizard] 대화상자를 이용해 새로운 클래스 등록 → ConstantDlg.h 선언

```
#include "stdafx.h"
#include "ImageProcessing.h"
#include "ImageProcessingDoc.h"
#include "DowmSampleDlg.h"
#include "UpSampleDlg.h"
#include "QuantizationDlg.h"
#include "math.h"
#include "ConstantDlg.h"
#include "StressTransformDlg.h" // 범위 강조 대화상자를 위한 헤더 선언
```

[실습하기 4-12] 범위를 강조하는 변환 프로그램

③ 생성된 **edit box**에 변수를 할당. **[MFC ClassWizard]** 대화상자의 **Object IDs** 항목에서 **CStressTransformDlg Class**을 선택 → **[Member Variables]** 탭 클릭 → **Control Ids** 항목에서 **ID_EDIT1**과 **ID_EDIT2**를 선택하여 각각 **[Add Variable]** 버튼 클릭→ **[Add Member Variable]** 대화상자를 이용해 정수형 변수를 선언 → **[OK]** 버튼 클릭

Control IDs	Member variable name	Category	Variable type	Minimum Value	Maximum Value
IDC_EDIT1	m_StartPoint	Value	int	0	255
IDC_EDIT2	m_EndPoint	Value	int	0	255

④ **ResourceView** 창에서 **[Menu]-[IDR_IMAGETYPE]** 폴더를 더블클릭하여 메뉴 추가

ID	ID_STRESS_TRANSFORM
Caption	범위 강조 변환

[실습하기 4-12] 범위를 강조하는 변환 프로그램

⑤ **[MFC ClassWizard]** 대화상자를 이용해 범위 강조 변환을 수행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnStressTransform
Doc Class	void	OnStressTransform

[실습하기 4-12] 범위를 강조하는 변환 프로그램

⑥ Doc 클래스의 OnStressTransform 함수에 다음 프로그램 추가

```
void CImageProcessingDoc::OnStressTransform()
{
   CStressTransformDlg dlg;

   int i;

   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){
      for(i=0 ; i<m_size ; i++){
         // 입력 값이 강조 시작 값과 강조 종료 값 사이에 위치하면 255 출력
         if(m_InputImage[i] >= dlg.m_StartPoint &&
            m_InputImage[i] <= dlg.m_EndPoint)
            m_OutputImage[i] = 255;
         else
            m_OutputImage[i] = m_InputImage[i];
      }
   }
}
```

[실습하기 4-12] 범위를 강조하는 변환 프로그램

⑦ View 클래스의 OnStressTransform 함수에 다음 프로그램 추가

```
void CImageProcessingDoc::OnStressTransform()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc-> OnStressTransform ();

    Invalidate(TRUE);
}
```

[실습하기 4-12] 범위를 강조하는 변환 프로그램

⑤ 프로그램 실행 결과 영상

범위 강조하여 변환한 결과 영상(범위 : 100~150)

요약

- 화소 점 처리
  - 원 화소의 값이나 위치를 바탕으로 단일 화소 값이 각각 독립적으로 변경되는 기술
- 컬러 영상에는 다양한 색상이 있으므로 화소는 색의 밝기를 표현함
  - RGB 영상에서는 빨간색의 R 채널, 초록색의 G 채널, 파란색의 B 채널 영상이 각 화소를 8비트로 표현함.
- AND 연산과 OR 연산
  - 특정 비트를 0으로 바꾸려면 특정 비트 위치가 0으로 구성된 이진 데이터와 AND 연산 수행
  - 특정 비트를 1로 구성하려면 특정 비트 위치가 1로 설정된 이진 데이터와 OR 연산 수행
- XOR
  - 비교연산.
  - 화소 비트와 임의의 이진 비트를 XOR해서 대응되는 비트의 값이 같으면 화소의 해당 비트를 0으로 구성
- NOT 연산
  - 화소 비트를 반전시킴(검정색은 흰색으로, 흰색은 검정색으로)

요약

- 명암 대비 스트레칭
  - 높은 명암 대비를 보이는 영상을, 명암 대비 압축은 낮은 명암 대비를 보이는 영상을 생성하는 것
- 포스터라이징
  - 영상에서 화소에 있는 명암 값의 범위를 경계 값으로 축소하는 기법
- 이진화
  - 영상의 화소 값을 경계 값을 이용해 값이 두 개만 있는 영상으로 변환해 주는 것
- 범위 강조 변환
  - 영상에서 한 부분의 화소는 원 상태를 그대로 유지한 채 일정 범위의 화소만 강조하는 변환

Thank you