05장 히스토그램을 이용한 화소 점 처리

- 디지털 영상의 히스토그램
- 산술연산을 이용한 히스토그램에서의 이동
- 히스토그램 스트레칭
- 히스토그램 평활화
- 히스토그램 명세화

한빛미디어(주)

5장. 히스토그램을 이용한 화소 점 처리

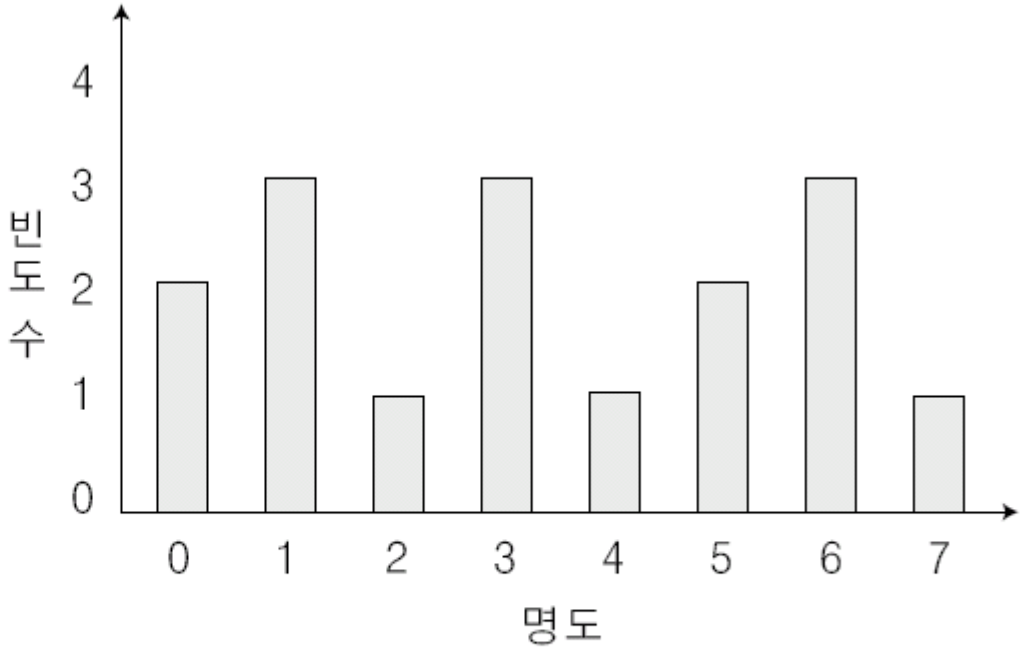
학습목표

- ✓ 디지털 영상의 히스토그램을 이해한다.
- ✓ 산술연산으로 히스토그램에서 명도와 명암 대비를 조정하는 방법을 이해한다.
- ✓ 히스토그램 스트레칭 기법의 원리와 효과를 학습한다.
- ✓ 히스토그램 평활화의 기본 원리를 익히고, 영상에서 보이는 효과를 알아본다.
- ✓ 히스토그램 명세화 원리를 익히고, 영상에서 보이는 효과를 알아본다.

Section 01 디지털 영상의 히스토그램

- 디지털 영상의 히스토그램
  - 관찰한 데이터의 특징을 한눈에 알아볼 수 있도록 데이터를 막대그래프 모양으로 나타낸 것
  - 디지털 영상에 대한 많은 정보를 제공함.

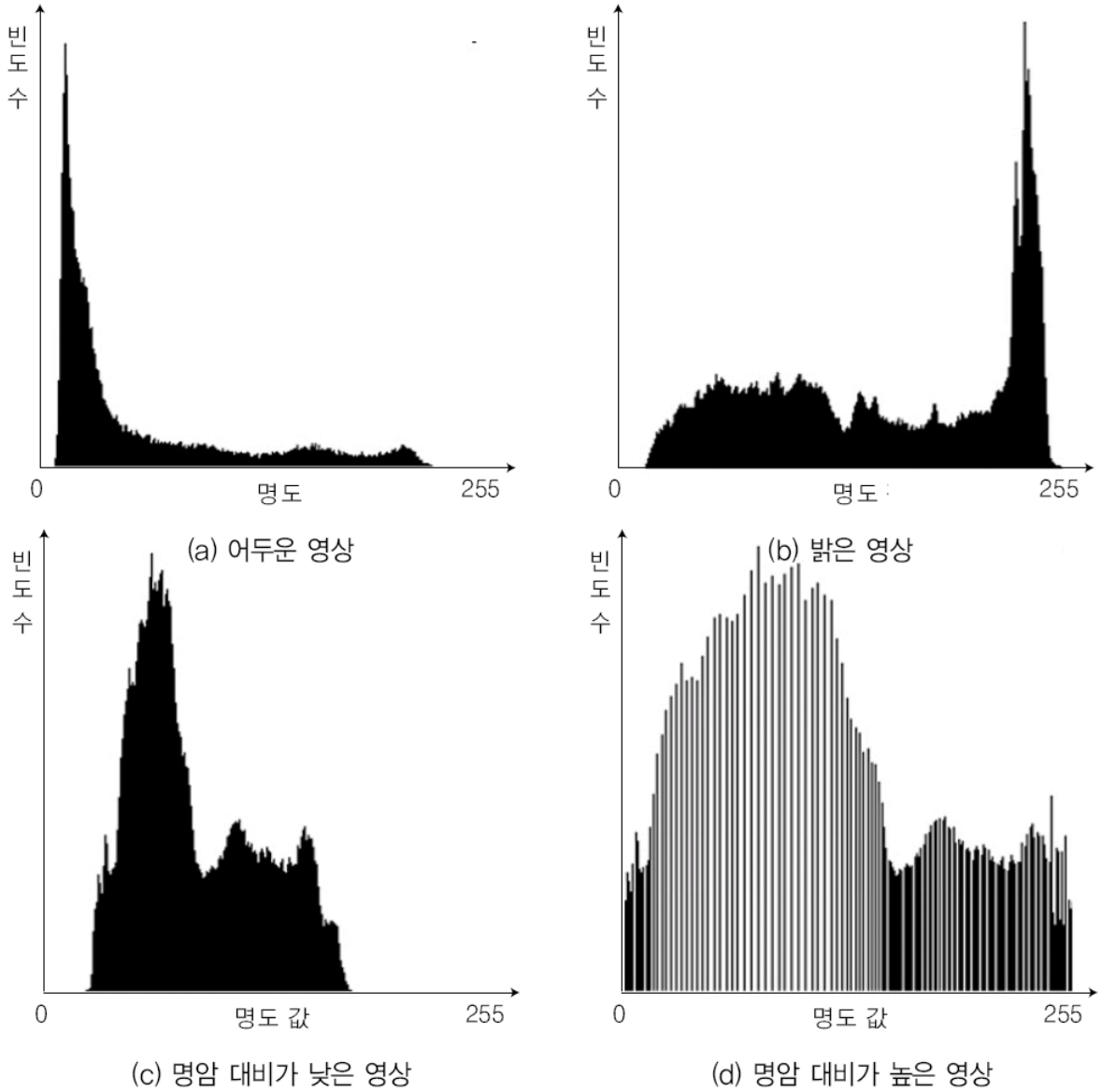
6	6	6	7
4	5	5	3
2	1	1	3
0	0	1	3



(a) 입력 영상 (b) 히스토그램

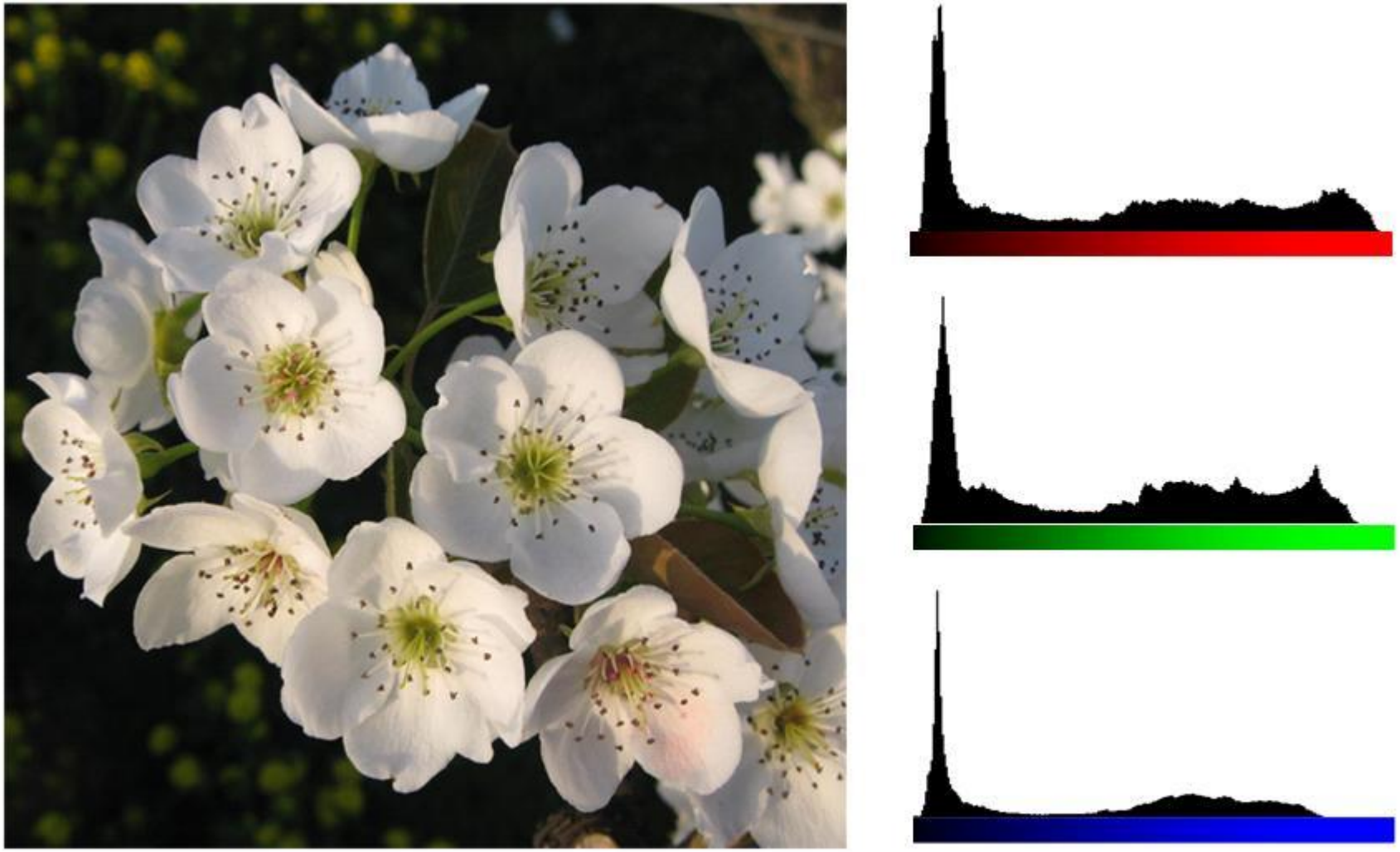
[그림 5-1] 이상적인 영상의 히스토그램

영상의 특성에 따른 히스토그램



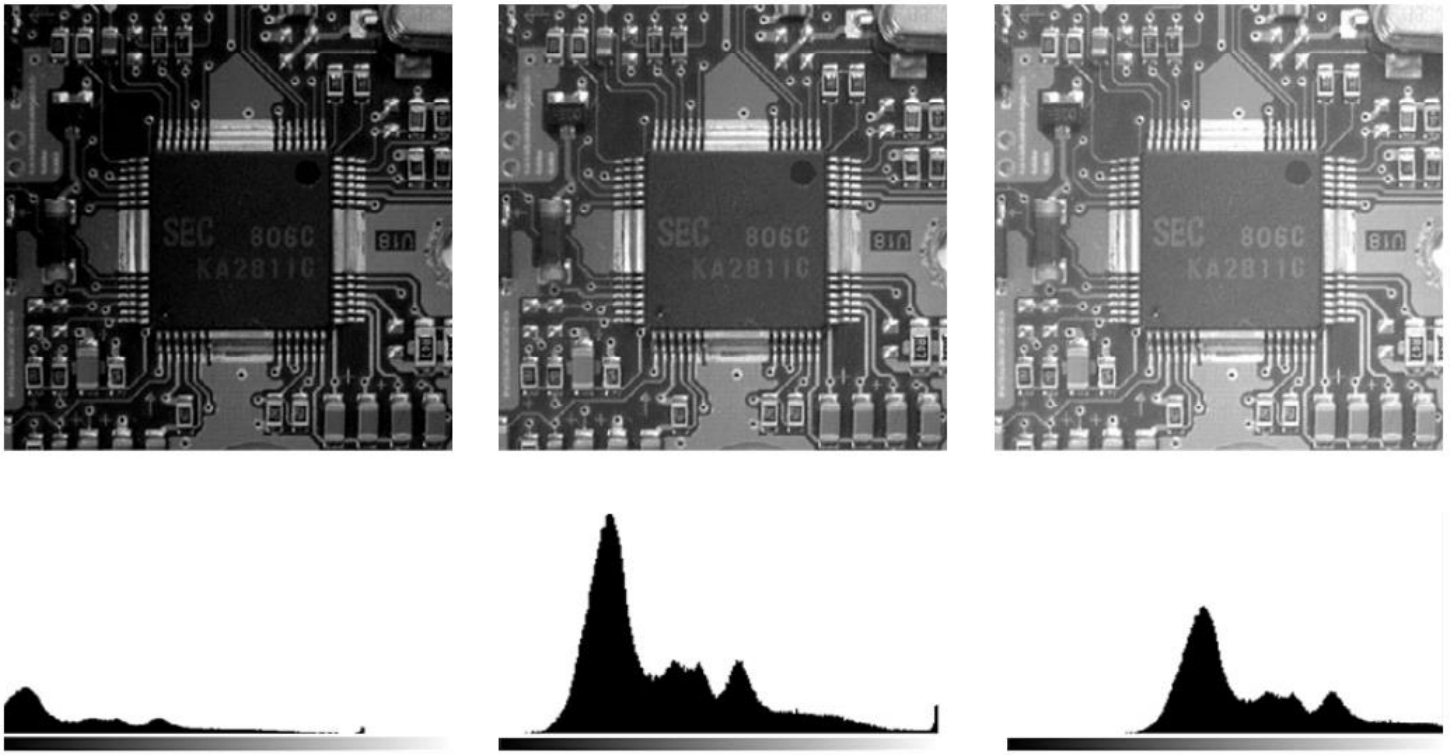
[그림 5-2] 영상의 특성에 따른 히스토그램

RGB 컬러 영상의 히스토그램



[그림 5-3] RGB 컬러 영상의 히스토그램

Section 02 산술 연산을 이용한 히스토그램 이동

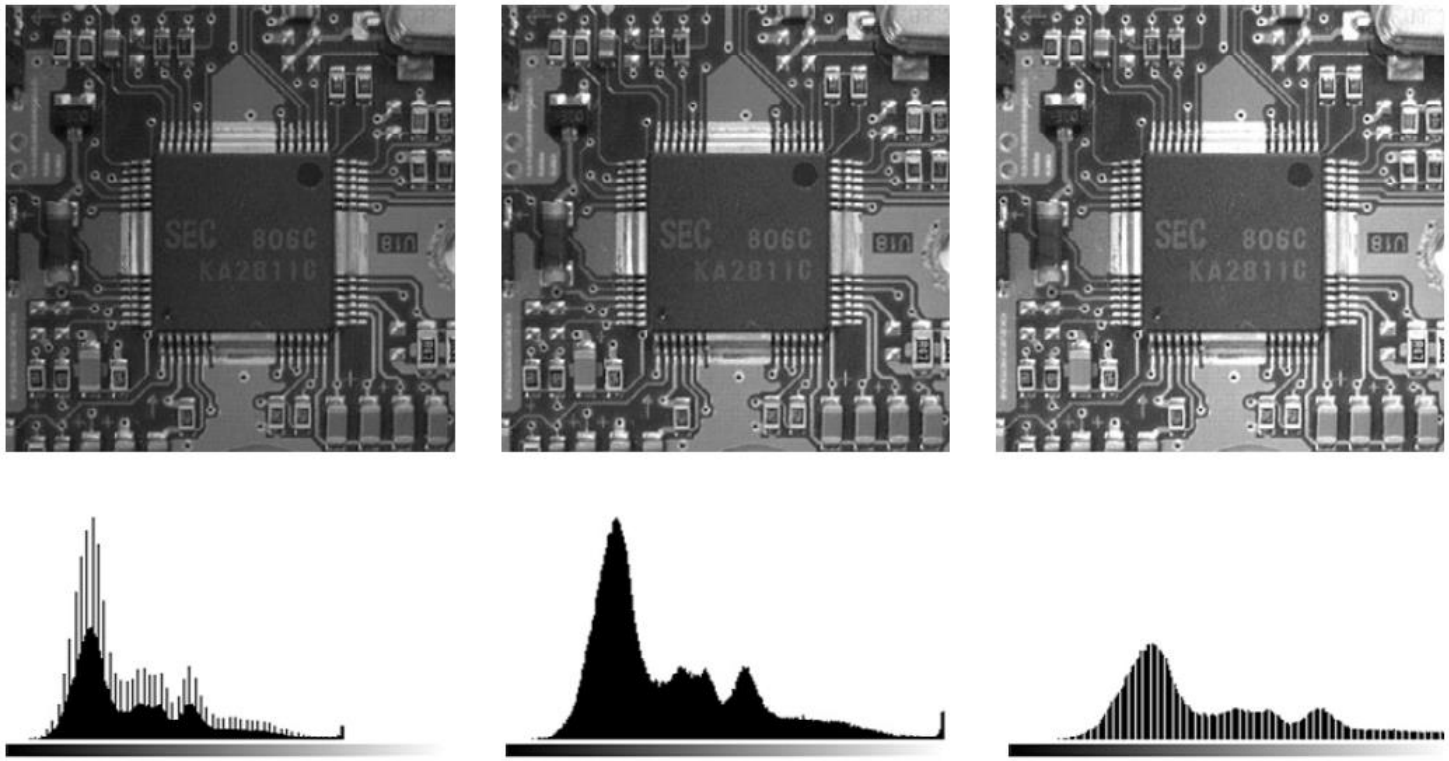


(a) 50을 뺄셈한 영상　　(b) 원본 영상　　(c) 50을 덧셈한 영상

[그림 5-4] 덧셈과 뺄셈연산으로 히스토그램 이동하기

- 덧셈연산: 명도 값을 증가시켜 밝게, 히스토그램의 기둥이 오른쪽으로 이동
- 뺄셈연산: 명도 값을 감소시켜 어둡게, 히스토그램의 기둥이 왼쪽으로 이동

Section 02 산술 연산을 이용한 히스토그램 이동(계속)



(a) 1.3을 나눗셈한 영상 (b) 원본 영상 (c) 1.3을 곱셈한 영상

[그림 5-5] 곱셈과 나눗셈연산으로 히스토그램 이동하기

- 곱셈연산: 명암 대비가 증가하여 히스토그램은 기둥의 분포 범위가 넓음.
- 나눗셈연산: 명암 대비가 감소하여 히스토그램의 분포 범위가 좁음.

Section 03 히스토그램 스트레칭

- 히스토그램 스트레칭(Histogram Stretching)
  - 명암 대비를 향상시키는 연산으로, 낮은 명암 대비를 보이는 영상의 화질을 향상시키는 방법
  - 명암 대비 스트레칭이라고도 함.
  - 히스토그램이 모든 범위의 화소 값을 포함하도록 히스토그램의 분포를 넓힘.
  - 기본 명암 대비 스트레칭과 앤드-인 탐색 기법이 대표적

기본 명암 대비 스트레칭

- 이상적이지 못한 히스토그램 분포 중에서 명암 대비가 낮은 디지털 영상의 품질을 향상시키는 기술
- 특정 부분이나 가운데에 집중된 히스토그램을 모든 영역으로 확장시켜서 디지털 영상이 모든 범위의 화소 값을 포함하게 함
- 기본 명암 대비 스트레칭 수행 공식

$$new\ pixel = \frac{old\ \ pixel - low}{high - low} \times 255$$

  - old pixel은 원 영상 화소의 명도 값
  - new pixel은 결과 영상 화소의 명도 값
  - low는 히스토그램의 최저 명도 값
  - high는 히스토그램의 최고 명도 값

기본 명암 대비 스트레칭(계속)

(a) 입력 영상　　(b) 히스토그램-low　　(c) (히스토그램-low)×255/(high-low)

[그림 5-7] 기본 명암 대비 스트레칭의 수행 과정

(a) 그레이 레벨 영상　　(b) 기본 명암 대비 스트레칭 영상

[그림 5-8] 영상에 기본 명암 대비 스트레칭을 적용한 결과 영상

앤드-인 탐색

- 일정한 양의 화소를 흰색이나 검정색으로 지정하여 히스토그램의 분포를 좀더 균일하게 만듦
- 앤드-인 탐색 수행 공식
  - 두 개의 임계 값(low, high) 사용

$$new\ pixel = \begin{cases} 0 & old\ pixel \le low \\ \dfrac{old\ pixel - low}{high - low} \times 255 & low \le old\ pixel \le high \\ 255 & high \le old\ pixel \end{cases}$$

앤드-인 탐색(계속)

(a) 그레이 레벨 영상

(b) 엔드-인 탐색 영상

[그림 5-9] 엔드-인 탐색이 적용된 결과 영상

[실습하기 5-1] 히스토그램 스트래칭의 프로그램

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

ID	ID_HISTO_STRETCH
Caption	히스토그램 스트레칭

② [MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 히스토그램 스트레칭을 실행하는 함수 추가

Class Name	Function Type	Function Name
View Class	void	OnHistoStretch
Doc Class	void	OnHistoStretch

③ Doc 클래스에 다음 프로그램 추가

[실습하기 5-1] 히스토그램 스트래칭의 프로그램

```
void CImageProcessingDoc::OnHistoStretch()
{
   int i;
   unsigned char LOW, HIGH, MAX, MIN;

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

   LOW = 0;
   HIGH = 255;

   MIN = m_InputImage[0]; // 최소값을 찾기 위한 초기값
   MAX = m_InputImage[0]; // 최대값을 찾기 위한 초기값

   // 입력 영상의 최소값 찾기
   for(i=0 ; i<m_size ; i++){
      if(m_InputImage[i] < MIN)
         MIN = m_InputImage[i];
   }

   // 입력 영상의 최대값 찾기
   for(i=0 ; i<m_size ; i++){
      if(m_InputImage[i] > MAX)
         MAX = m_InputImage[i];
   }

   m_OutputImage = new unsigned char[m_Re_size];

   // 히스토그램 stretch
   for(i=0 ; i<m_size ; i++)
      m_OutputImage[i] = (unsigned char)((m_InputImage[i] -
           MIN)*HIGH / (MAX-MIN));
   }
```

[실습하기 5-1] 히스토그램 스트래칭의 프로그램

④ View 클래스에 다음 프로그램 추가

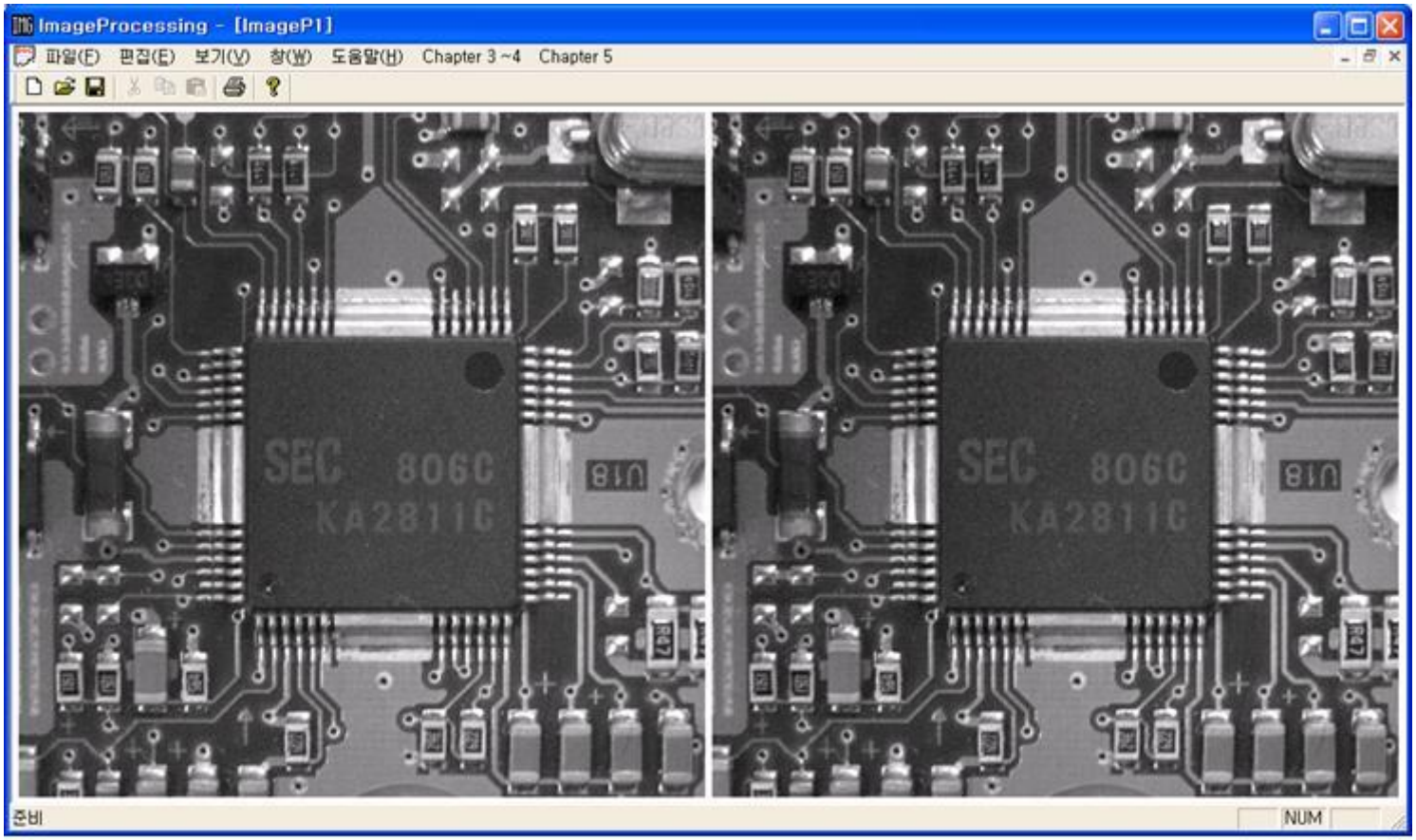
```
void CImageProcessingView::OnHistoStretch()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnHistoStretch();

    Invalidate(TRUE);
}
```

[실습하기 5-1] 히스토그램 스트래칭의 프로그램

⑤ 프로그램 실행 결과 영상



히스토그램 스트레칭 프로그램을 구현한 결과 영상

[실습하기 5-2] 앤드-인 탐색 프로그램

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

ID	ID_END_IN_SEARCH
Caption	End-in search

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 앤드-인 탐색을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnEndInSearch
Doc Class	void	OnEndInSearch

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 5-2] 앤드-인 탐색 프로그램

```
void CImageProcessingDoc::OnEndInSearch()
{
   int i;
   unsigned char LOW, HIGH, MAX, MIN;

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

   LOW = 0;
   HIGH = 255;

   MIN = m_InputImage[0];
   MAX = m_InputImage[0];

   for(i=0 ; i<m_size ; i++){
      if(m_InputImage[i] < MIN)
         MIN = m_InputImage[i];
   }

   for(i=0 ; i<m_size ; i++){
      if(m_InputImage[i] > MAX)
         MAX = m_InputImage[i];
   }

   m_OutputImage = new unsigned char[m_Re_size];
```

[실습하기 5-2] 앤드-인 탐색 프로그램

```
    for(i=0 ; i<m_size ; i++){
        // 원본 영상의 최소값보다 작은 값은 0
        if(m_InputImage[i] <= MIN){
            m_OutputImage[i] = 0;
        }

        // 원본 영상의 최대값보다 큰 값은 255
        else if(m_InputImage[i] >= MAX){
          m_OutputImage[i] = 255;
        }
        else
          m_OutputImage[i] = (unsigned char)((m_InputImage[i] -
            MIN)*HIGH / (MAX-MIN));
    }
}
```

[실습하기 5-2] 앤드-인 탐색 프로그램

④ View 클래스에 다음 프로그램 추가

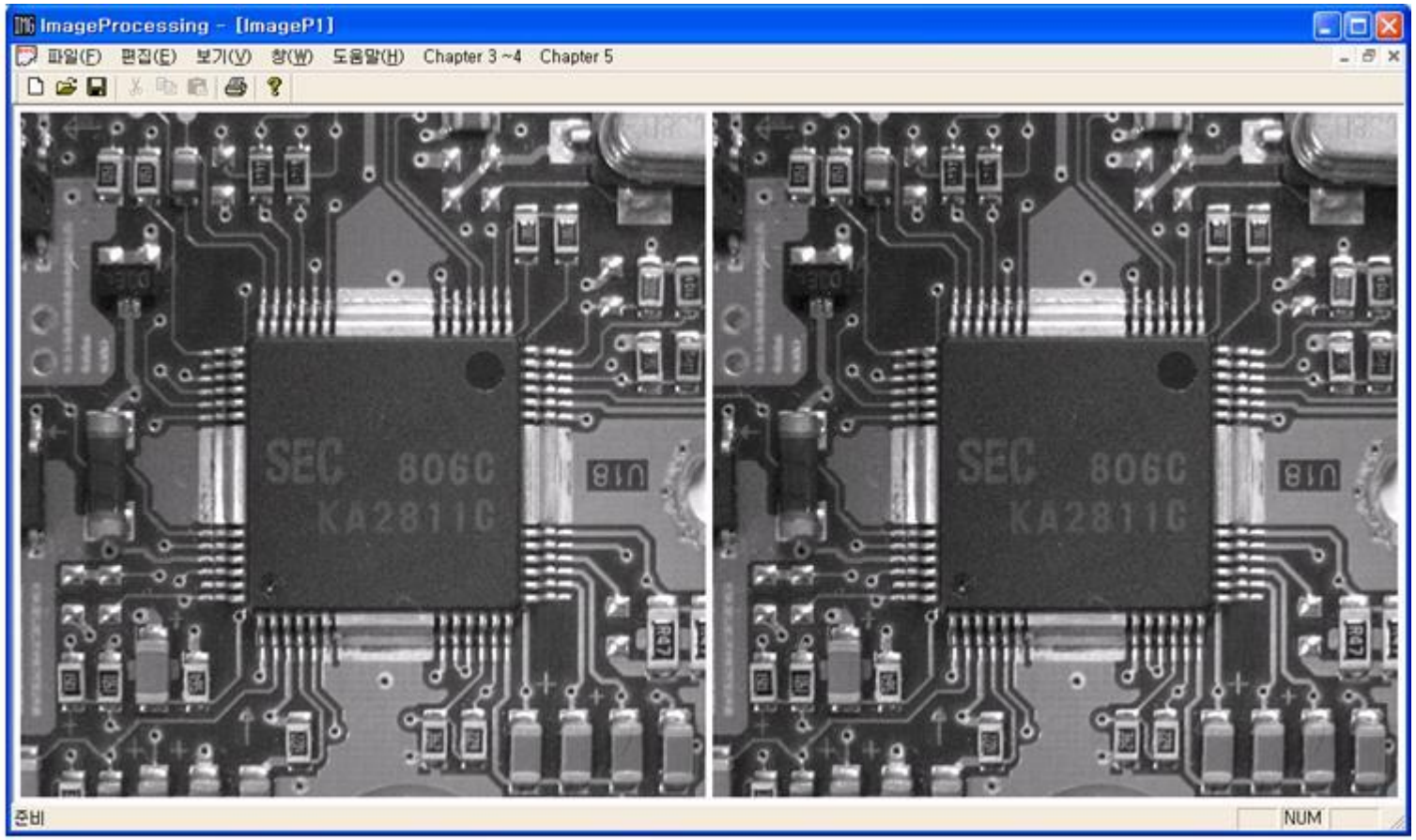
```
void CImageProcessingView::OnEndInSearch()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnEndInSearch();

    Invalidate(TRUE);
}
```

[실습하기 5-2] 앤드-인 탐색 프로그램

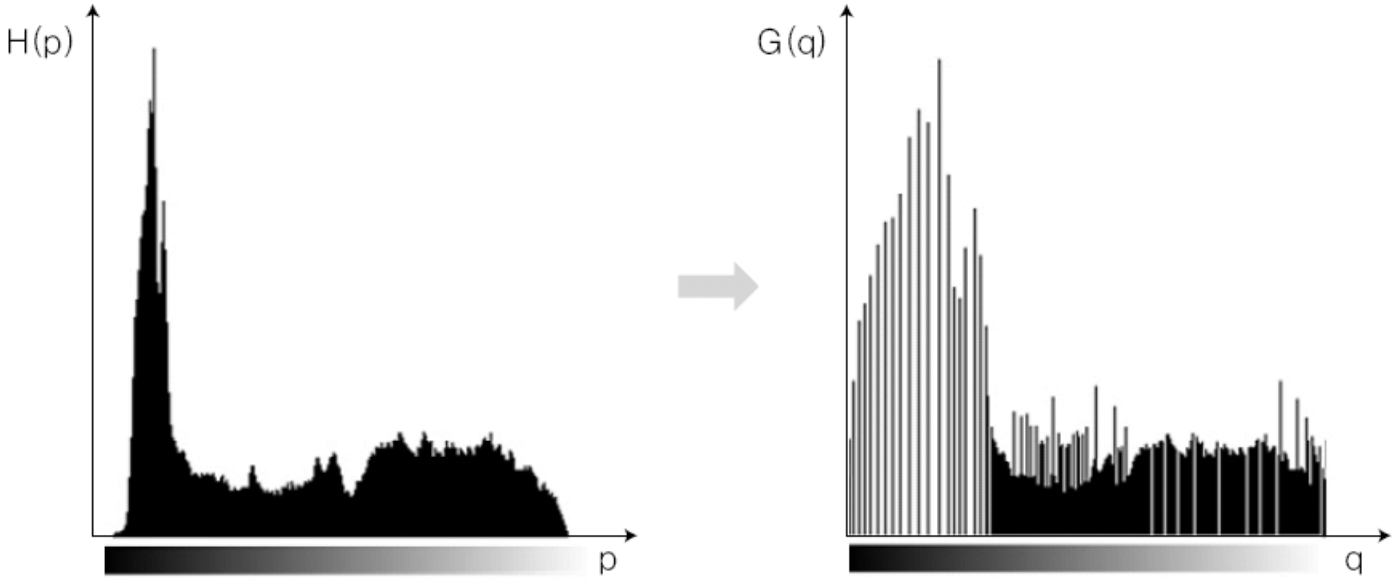
⑤ 프로그램 실행 결과 영상



엔드-인 탐색 프로그램을 구현한 결과 영상

Section 04. 히스토그램 평활화

- 히스토그램 평활화 기법(Histogram Equalized)
  - 어둡게 촬영된 영상의 히스토그램을 조절하여 명암 분포가 빈약한 영상을 균일하게 만들어 줌.
  - 영상의 밝기 분포를 재분배하여 명암 대비를 최대화
  - 명암 대비 조정을 자동으로 수행
  - 각 명암의 빈도는 변경하지 않음.
  - 검출 특성이 좋은 영상만 출력하지는 않지만 영상의 검출 특성을 증가시킴



[그림 5-10] 히스토그램 평활화를 수행한 뒤 변화된 히스토그램 모습

히스토그램 평활화의 3단계

- 1단계
  - 명암 값 j의 빈도 수 hist[j]를 계산해 입력 영상의 히스토그램 생성
- 2단계
  - 각 명암 값 i에서 0~i까지의 누적 빈도 수(누적합)를 계산

$$sum[i] = \sum_{j=0}^{i} hist[j]$$

- 3단계
  - 2단계에서 구한 누적 빈도 수를 정규화(정규화 누적합)

$$n[i] = sum[i] \times \frac{1}{N} \times I_{\max}$$

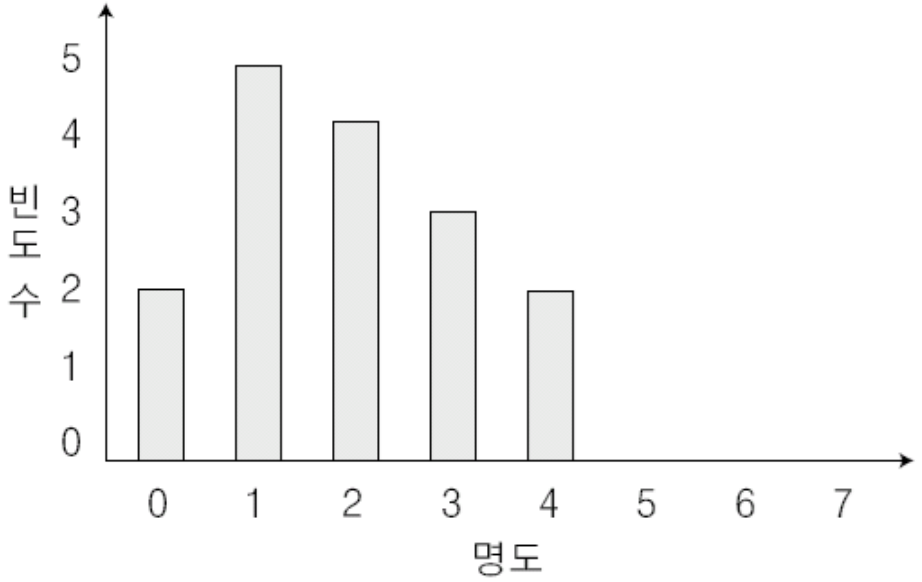
  - N은 화소의 총 수, Imax는 최대 명도 값
  - 3단계에서 얻은 정규화된 값 n[i]로 입력 영상의 화소 값 i를 변환하면 평활화된 결과 영상 생성

히스토그램 평활화_1단계

1단계

- 빈도 수 hist[j]에서의 히스토그램 생성

2	4	4	3
2	1	3	3
1	0	1	2
0	1	1	2



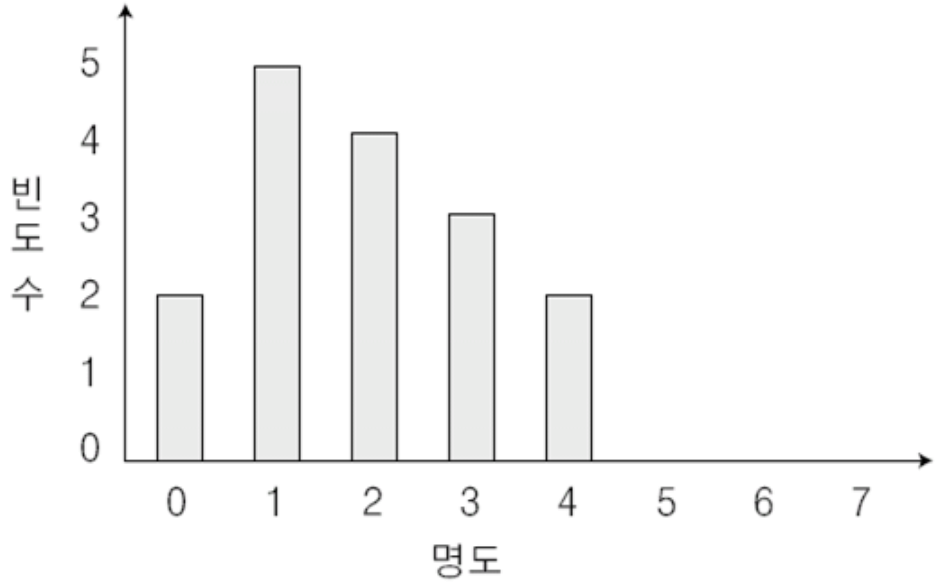
(a) 입력 영상 (b) 히스토그램

[그림 5-11] 히스토그램의 생성

- 화소의 명도 값 0은 2개, 1은 5개, 2는 4개, 3은 3개, 4는 2개
- 가장 큰 명도 값이 4이므로 전체적으로 왼쪽으로 치우침.

히스토그램 평활화_2단계

- 2단계
  - 누적합 sum[i] 생성



(a) 히스토그램

명도	누적합
0	2
1	7
2	11
3	14
4	16
5	16
6	16
7	16

(b) 누적합

[그림 5-12] 히스토그램에서 누적합 계산

- 화소의 명도 0번까지의 누적합은 2, 1번까지는 2+5=7, 2번까지는 2+5+4=11, 3번까지는 2+5+4+3=14, 4번까지는 2+5+4+3+2=16
- 나머지 명도 값은 영상에는 없으므로 누적합은 16

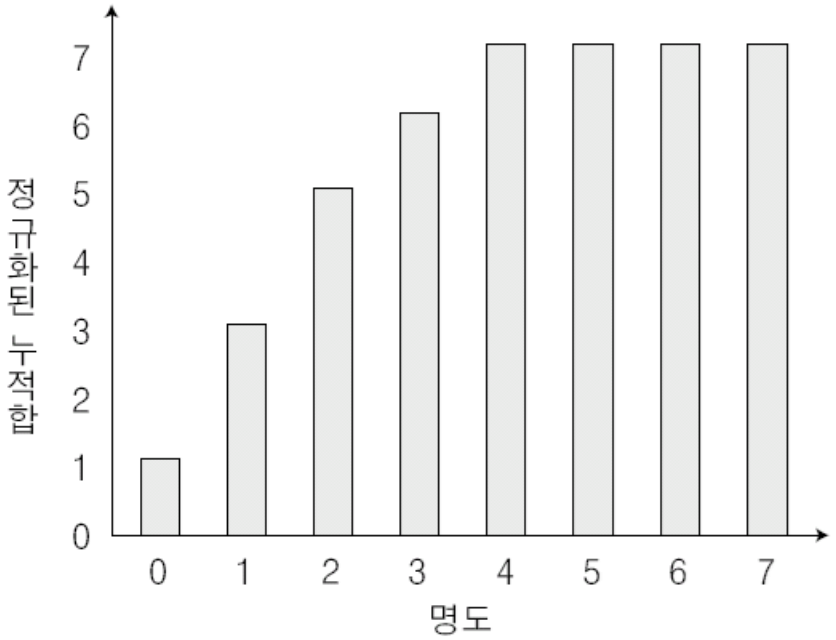
히스토그램 평활화_3단계

3단계

- **n[i]=sum[i]*(1/16)*7**

명도 (i)	누적합 (sum[i])	정규화된누적합 (n[i])
0	2	0.875
1	7	3.0625
2	11	4.8125
3	14	6.125
4	16	7
5	16	7
6	16	7
7	16	7

(a) 정규화

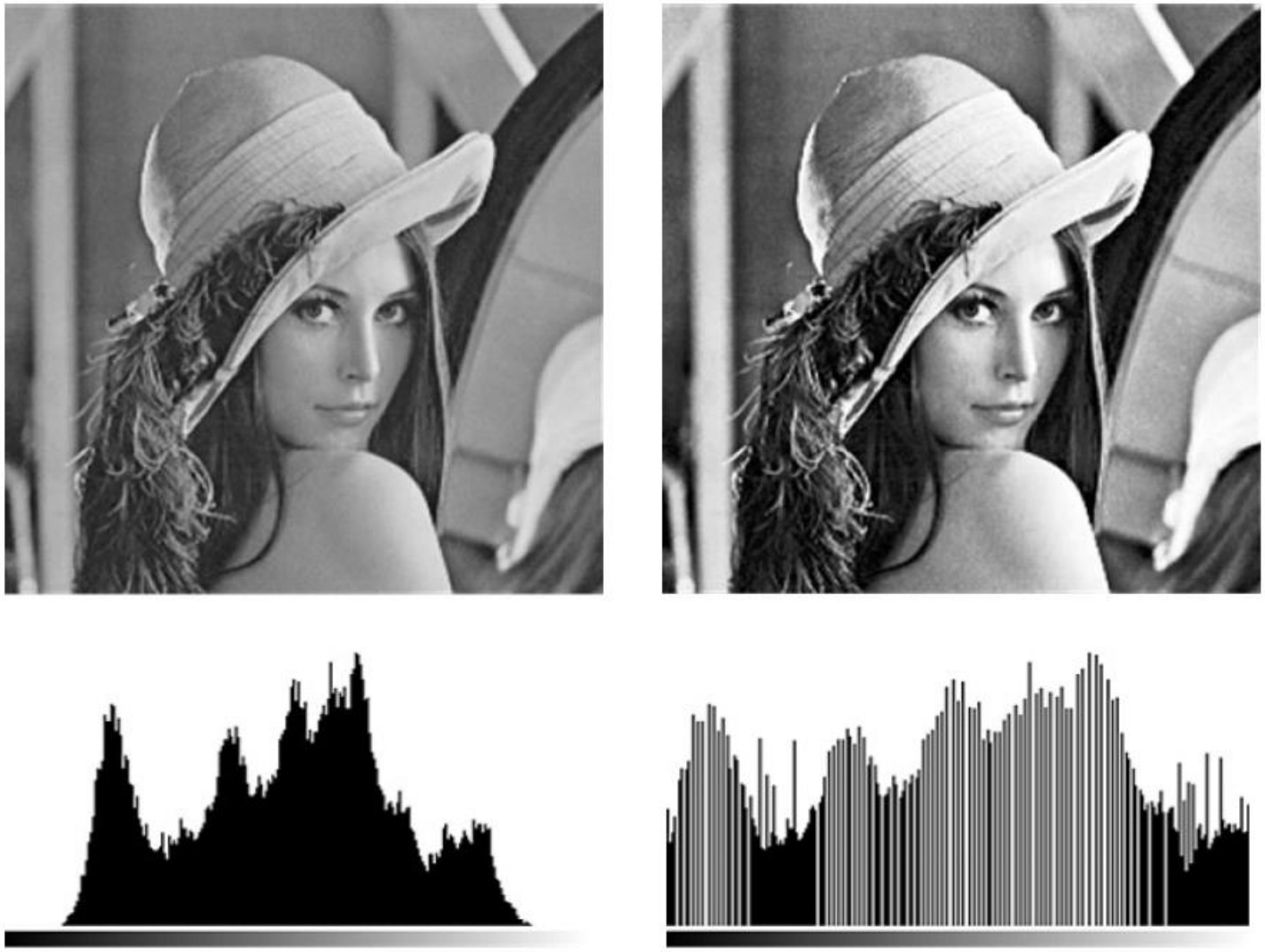


(b) 정규화된 히스토그램

[그림 5-13] 누적합에서 정규화

- **n[0]은 2*(1/16)*7=0.875, n[1]은 7*(1/16)*7=3.0625**
- **n[2]는 11*(1/16)*7=4.8125, n[3]은 14*(1/16)*7=6.125**
- **n[4]와 n[5], n[6], n[7]은 16*(1/16)*7=7**

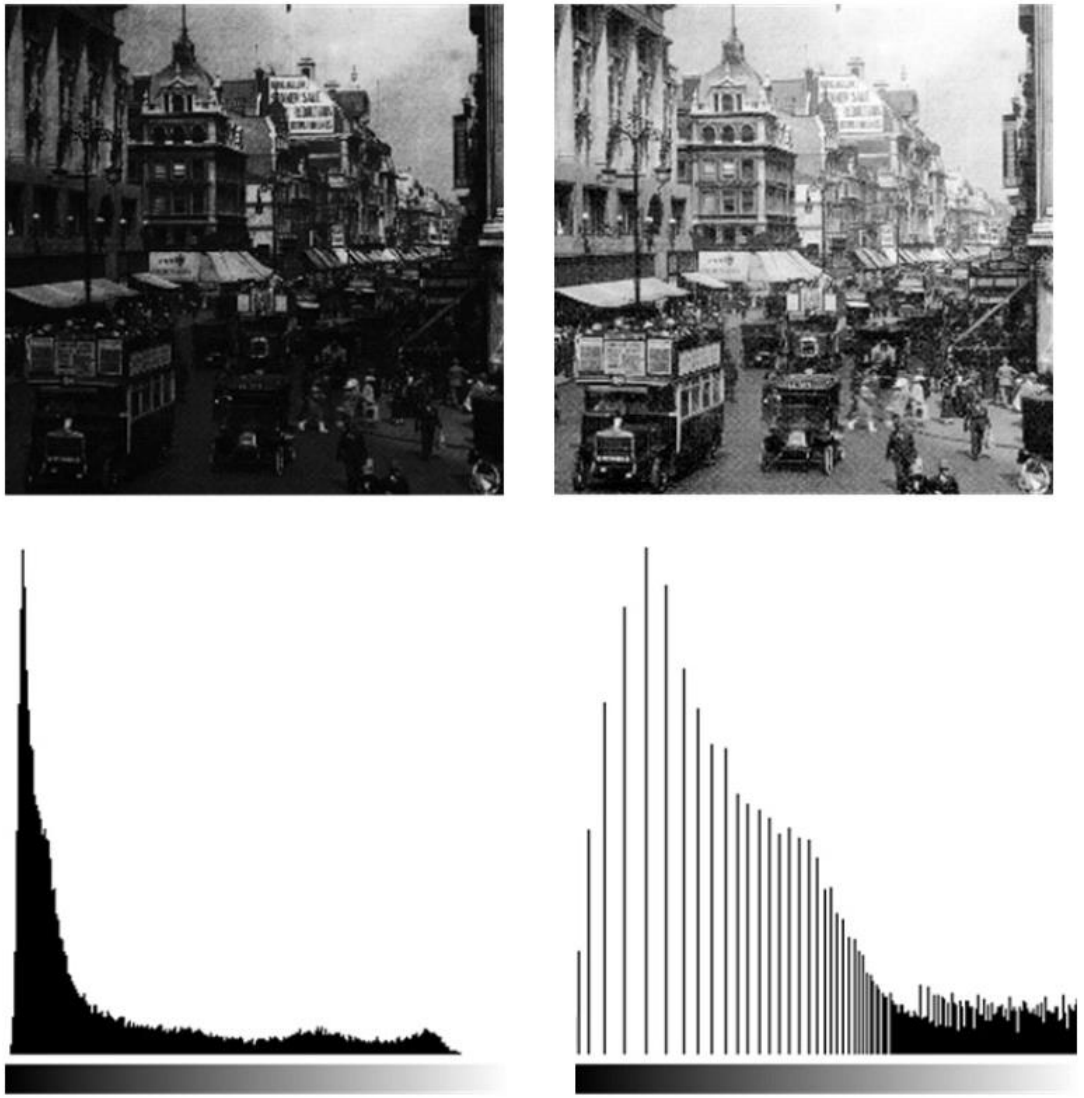
히스토그램 평활화를 적용한 영상



(a) 원본 영상 (b) 평활화 영상

[그림 5-15] 레나(Lenna) 영상의 평활화와 평활화한 영상의 히스토그램

히스토그램 평활화를 적용한 영상(계속)



(a) 원본 영상 (b) 평활화 영상

[그림 5-16] 명암 대비가 낮은 영상의 평활화와 평활화한 영상의 히스토그램

[실습하기 5-3] 히스토그램 프로그램

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

ID	ID_HISTOGRAM
Caption	히스토그램

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

Class Name	Function Type	Function Name
View Class	void	OnHistogram
Doc Class	void	OnHistogram

③ 히스토그램에서 사용할 전역변수를 선언. 변수를 전역으로 선언하여 히스토그램과 히스토그램 평활화에서 공유

```
double m_HIST[256];
double m_Sum_Of_HIST[256];
unsigned char m_Scale_HIST[256];
```

[실습하기 5-3] 히스토그램 프로그램

```
void CImageProcessingDoc::OnHistogram()
{
   // 히스토그램의 값은 0~255
   // 히스토그램의 크기 값을 MAX=255로 정규화하여 출력
   // 히스트그램의 크기 : 256*256 지정

   int i, j, value;
   unsigned char LOW, HIGH;
   double MAX, MIN, DIF;

   m_Re_height = 256;
   m_Re_width = 256;
   m_Re_size = m_Re_height * m_Re_width;

   LOW = 0;
   HIGH = 255;

   // 초기화
   for(i=0 ; i<256 ; i++)
      m_HIST[i] = LOW;

   // 빈도 수 조사
   for(i=0 ; i<m_size ; i++){
      value = (int)m_InputImage[i];
      m_HIST[value]++;
   }

   // 정규화
   MAX = m_HIST[0];
   MIN = m_HIST[0];

   for(i=0 ; i<256 ; i++){
      if(m_HIST[i] > MAX)
         MAX = m_HIST[i];
   }

   for(i=0 ; i<256 ; i++){
      if(m_HIST[i] < MIN)
         MIN = m_HIST[i];
   }
```

[실습하기 5-3] 히스토그램 프로그램

```
   DIF = MAX - MIN;

   // 정규화된 히스토그램
   for(i=0 ; i<256 ; i++)
      m_Scale_HIST[i] = (unsigned char)((m_HIST[i] - MIN) * HIGH / DIF);

   // 정규화된 히스토그램 출력
   m_OutputImage = new unsigned char [m_Re_size + (256*20)];

   for(i=0 ; i<m_Re_size ; i++)
      m_OutputImage[i] = 255;

   // 정규화된 히스토그램의 값은 출력 배열에 검은 점(0)으로 표현
   for(i=0 ; i<256 ; i++){
      for(j = 0 ; j<m_Scale_HIST[i] ; j++){
         m_OutputImage[m_Re_width*(m_Re_height-j-1) + i] = 0;
      }
   }

   // 히스토그램을 출력하고 그 아래 부분에 히스토그램의 색을 표시
   for(i = m_Re_height ; i<m_Re_height + 5 ; i++){
      for(j=0 ; j<256 ; j++){
         m_OutputImage[m_Re_height * i + j] = 255;
      }
   }

   for(i = m_Re_height+5 ; i<m_Re_height + 20 ; i++){
         for(j=0 ; j<256 ; j++){
         m_OutputImage[m_Re_height * i + j] = j;
      }
   }

   m_Re_height = m_Re_height + 20;
   m_Re_size = m_Re_height * m_Re_width;
}
```

[실습하기 5-3] 히스토그램 프로그램

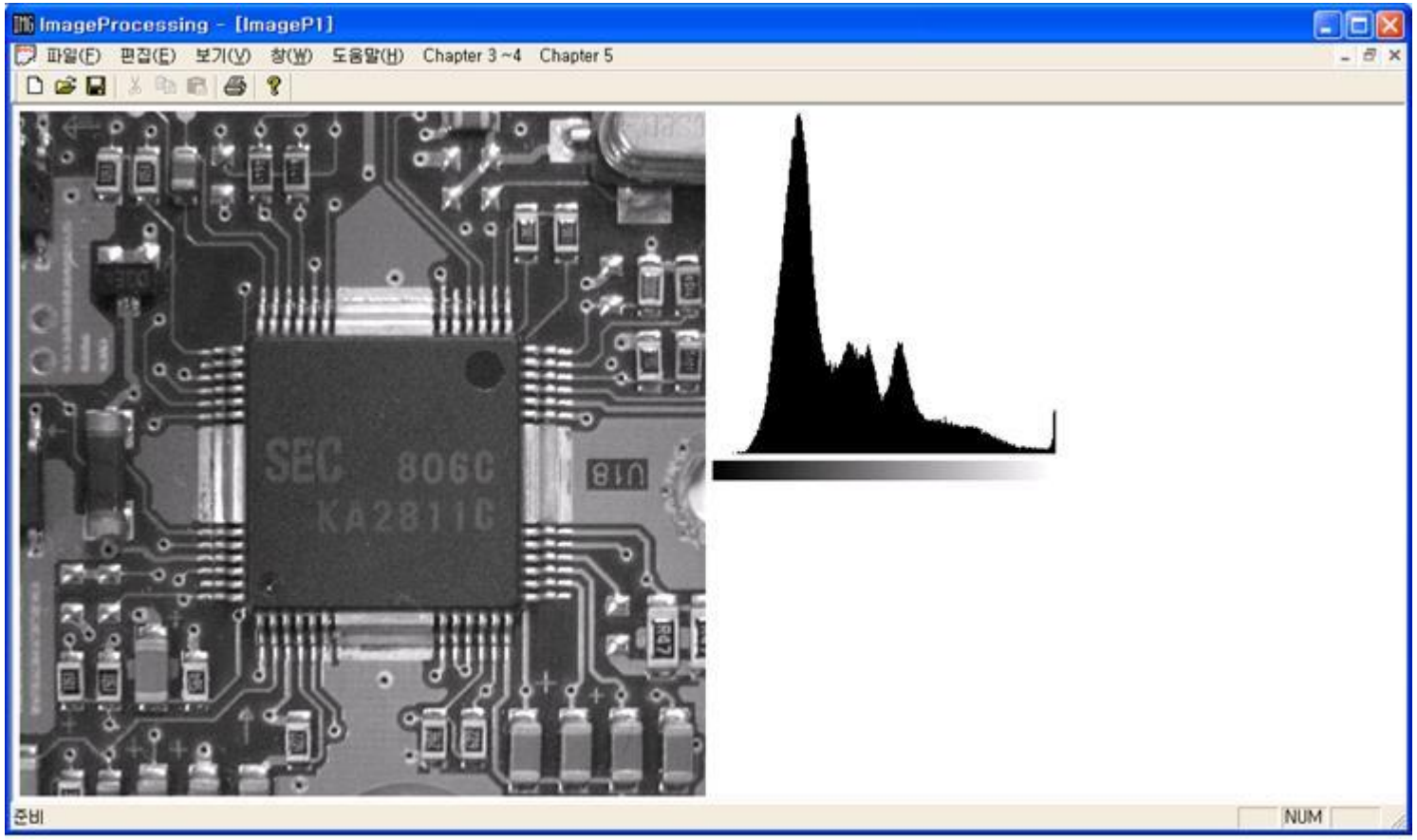
④ View 클래스에 다음 프로그램 추가

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

    pDoc->OnHistogram();
    Invalidate(TRUE);
}
```

[실습하기 5-3] 히스토그램 프로그램

⑤ 프로그램 실행 결과 영상



히스토그램 프로그램을 구현한 결과 영상

[실습하기 5-4] 히스토그램 평활화 프로그램

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

ID	ID_HISTO_EQUAL
Caption	히스토그램 평활화

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 히스토그램 평활화를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnHistoEqual
Doc Class	void	OnHistoEqual

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 5-4] 히스토그램 평활화 프로그램

```
void CImageProcessingDoc::OnHistoEqual()
{
   int i, value;
   unsigned char LOW, HIGH, Temp;
   double SUM = 0.0;

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

   LOW = 0;
   HIGH = 255;

   // 초기화
   for(i=0 ; i<256 ; i++)
      m_HIST[i] = LOW;

   // 빈도 수 조사
   for(i=0 ; i<m_size ; i++){
      value = (int)m_InputImage[i];
      m_HIST[value]++;
   }

   // 누적 히스토그램 생성
   for(i=0 ; i<256 ; i++){
      SUM += m_HIST[i];
      m_Sum_Of_HIST[i] = SUM;
   }

   m_OutputImage = new unsigned char[m_Re_size];

   // 입력 영상을 평활화된 영상으로 출력
   for(i=0 ; i<m_size ; i++){
      Temp = m_InputImage[i];
      m_OutputImage[i]=(unsigned char)(m_Sum_Of_ HIST[Temp]*HIGH/m_size);
   }
}
```

[실습하기 5-4] 히스토그램 평활화 프로그램

④ View 클래스에 다음 프로그램 추가

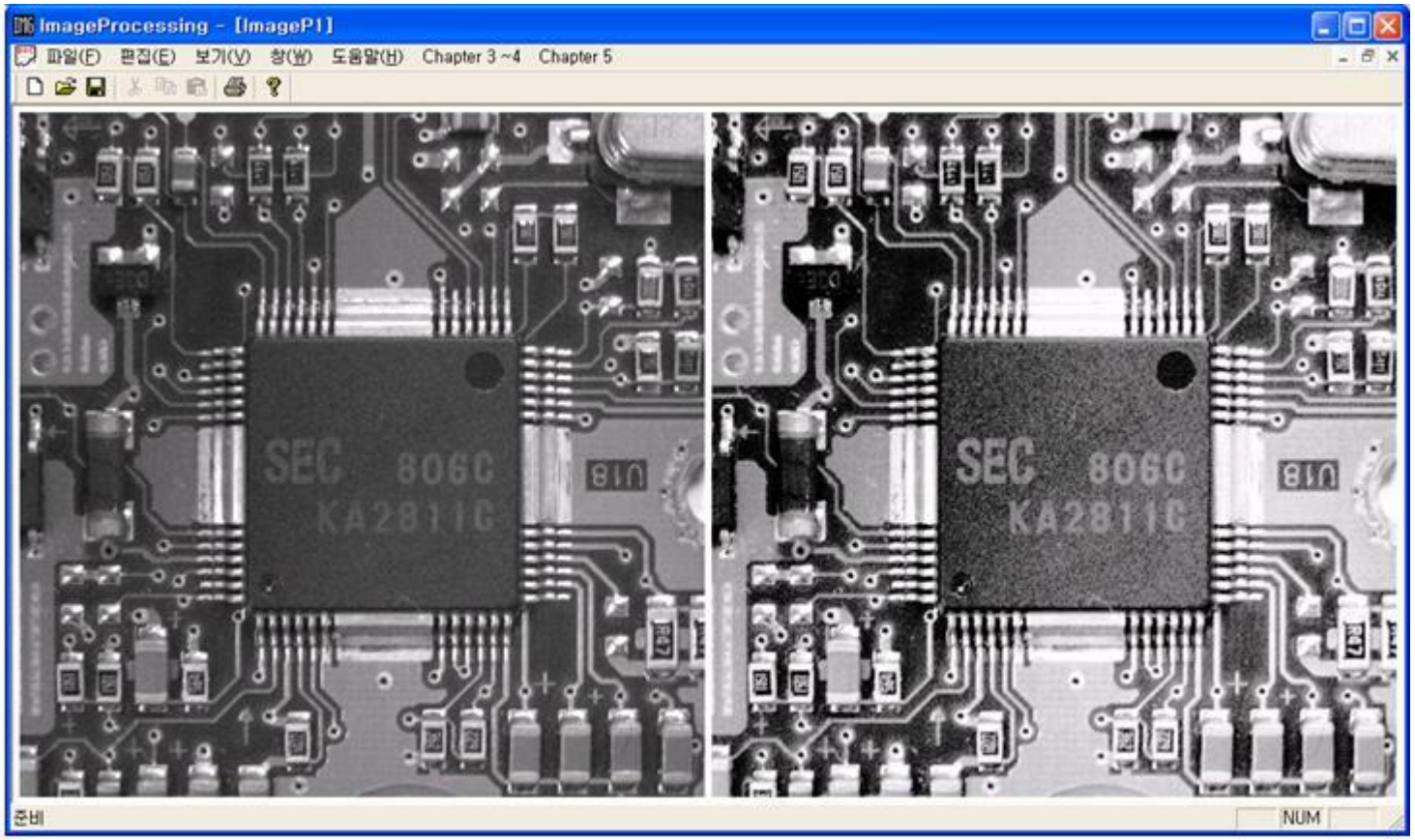
```
void CImageProcessingView::OnHistoEqual()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnHistoEqual();

    Invalidate(TRUE);
}
```

[실습하기 5-4] 히스토그램 평활화 프로그램

⑤ 프로그램 실행 결과 영상



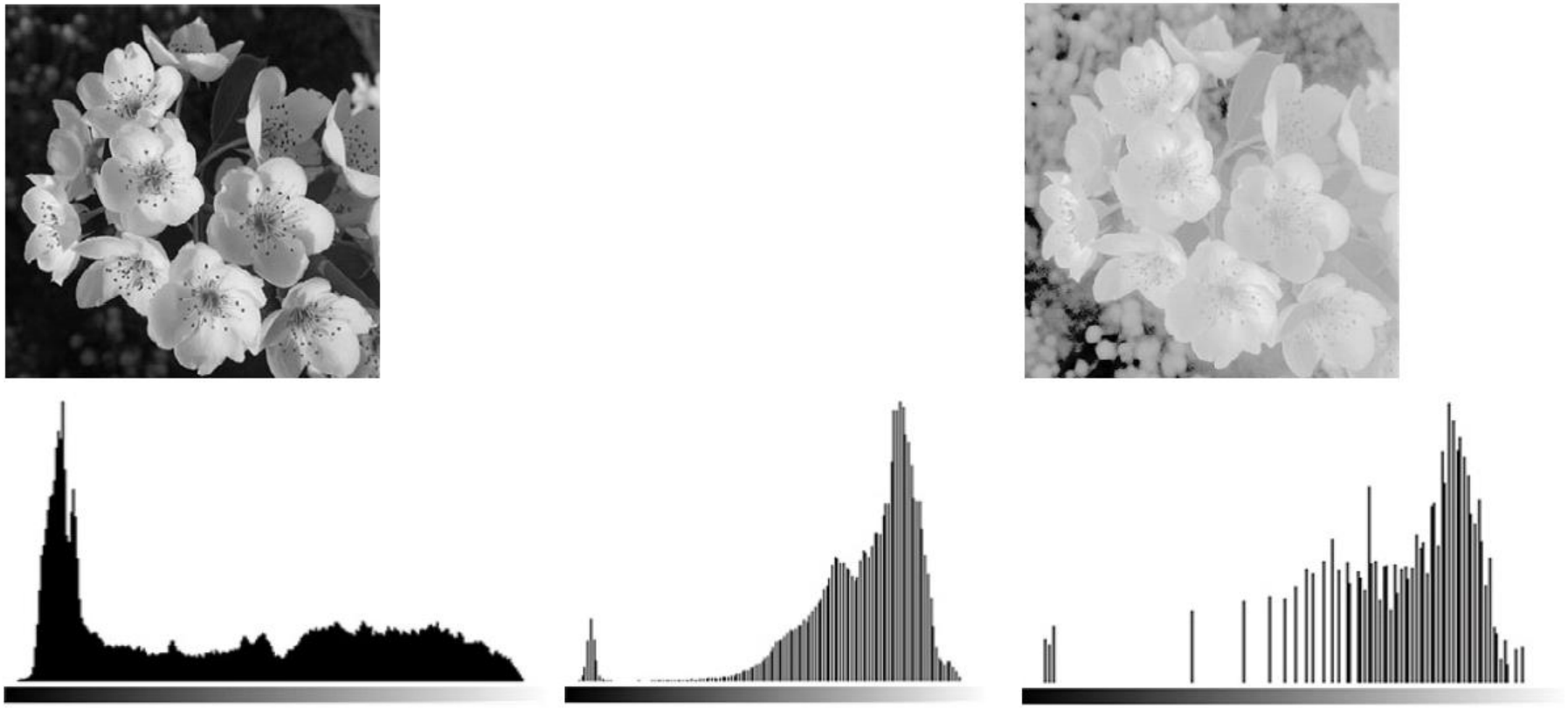
히스토그램 평활화 프로그램을 구현한 결과 영상

Section 05 히스토그램 명세화

히스토그램 명세화(Histogram Specification)

- 특정 모양의 히스토그램을 생성된 디지털 영상의 히스토그램에 포함하여 영상의 일부 영역의 명암 대비(콘트라스트)를 개선할 수 있는데, 이런 영상 처리 기법
- 입력 영상의 히스토그램을 원하는 히스토그램으로 변환한다고 해서 히스토그램 정합(Histogram Matching) 기법
- 명암 대비를 개선하는 것은 히스토그램 평활화와 같지만 특정 부분을 향상시키려고 원하는 히스토그램을 이용한 정합으로 일부 영역에서만 명암 대비를 개선한다는 점이 다름.
- 기본적으로 입력 영상을 원하는 히스토그램으로 평활화하고 역 히스토그램 평활화 수행 → 룩업테이블(lookup table)을 생성하고 평활화된 원 영상을 역 변환하여 원하는 히스토그램을 얻음.

히스토그램 명세화 개념



(a) 원본 영상과 히스토그램 (b) 원하는 히스토그램 (c) 명세화된 영상과 히스토그램

[그림 5-17] 히스토그램의 명세화 개념

히스토그램 명세화 과정

- 1단계
  - 입력 디지털 영상의 히스토그램 생성
- 2단계
  - 입력 디지털 영상의 히스토그램을 평활화하려고 정규화된 누적 빈도 수의 함수를 구한 뒤 다음 변환식을 얻음.

    $$q = T(P)$$

    - P는 원 영상의 화소 값, q는 평활화 값
  - 변환식를 바탕으로 평활화를 수행하여 균일 분포된 히스토그램을 얻음.
- 3단계
  - 원하는 히스토그램의 정규화된 누적 빈도 수 함수를 구하고, 역변환 함수가 있는 변환식을 구한 뒤 평활화 수행

    $$v = G(Z)$$

    - Z는 원하는 히스토그램의 명도 값, v는 평활화 값

히스토그램 명세화 과정(계속)

- 4단계
  - 평활화된 원하는 히스토그램을 역평활화하여 역변환 함수를 구함. 여기서 역변환 함수는 실제 룩업테이블이 됨

$$Z = G\text{-}1(v)$$

- 5단계
  - 4단계에서 구한 역변환 함수를 이용하여 평활화된 원 영상의 히스토그램을 원하는 히스토그램이 있는 영상으로 만듦.

$$Z = G\text{-}1(v) \Rightarrow Z = G\text{-}1(q) = G\text{-}1[T(P)]$$

- 최초의 입력 영상은 원하는 히스토그램이 아니지만 평활화되어 균일하게 분포. 따라서 역변환 함수는 평활화되어 균일하게 분포된 입력 영상도 원하는 히스토그램으로 만들어 줌.

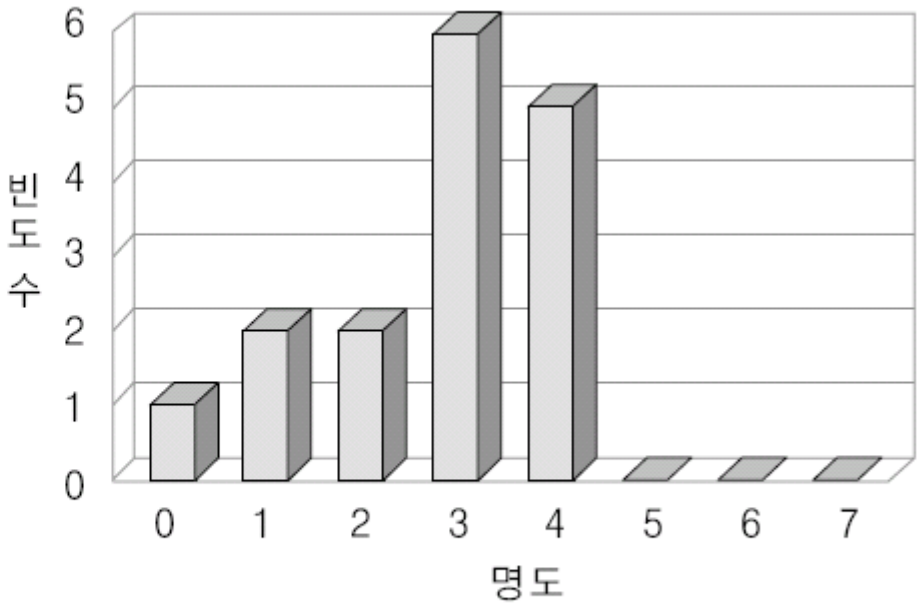
히스토그램 명세화_1단계

1 단계

- 히스토그램 생성

4	4	3	3
4	4	3	3
4	1	2	3
0	1	2	3

명도	빈도 수
0	1
1	2
2	2
3	6
4	5
5	0
6	0
7	0



[그림 5-18] 입력 영상에서 화소의 명도 값, 빈도 수, 히스토그램

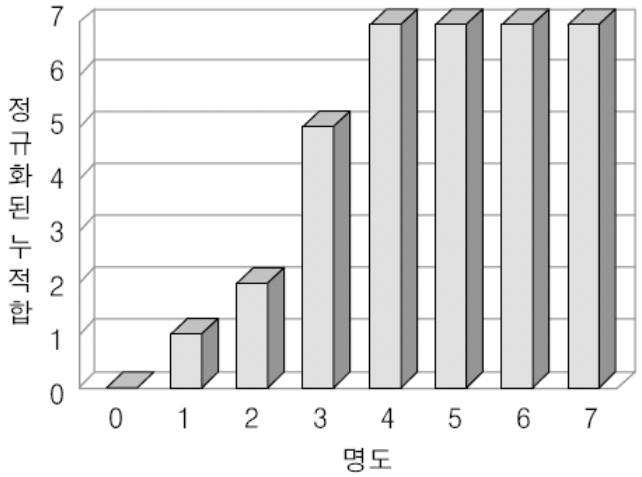
히스토그램 명세화_2단계

2 단계

입력 영상의 평활화

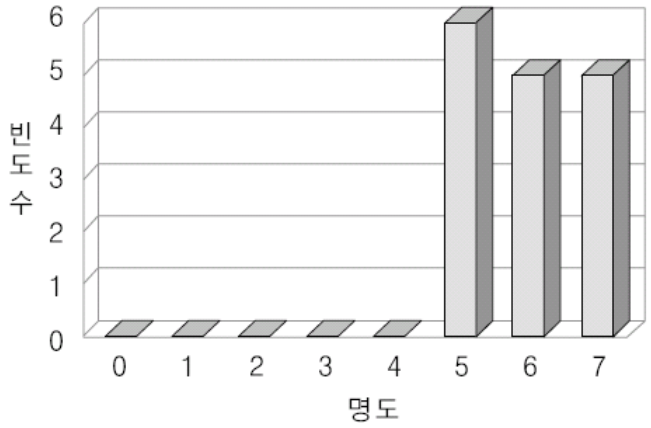
7	7	5	5
7	7	5	5
7	1	2	5
0	1	2	5

명도	누적합	정규화된 누적합
0	1	0.43
1	3	1.31
2	5	2.18
3	11	4.81
4	16	7
5	16	7
6	16	7
7	16	7



[그림 5-19] 입력 영상의 평활화

명도	빈도 수
0	0
1	0
2	0
3	0
4	0
5	6
6	5
7	5

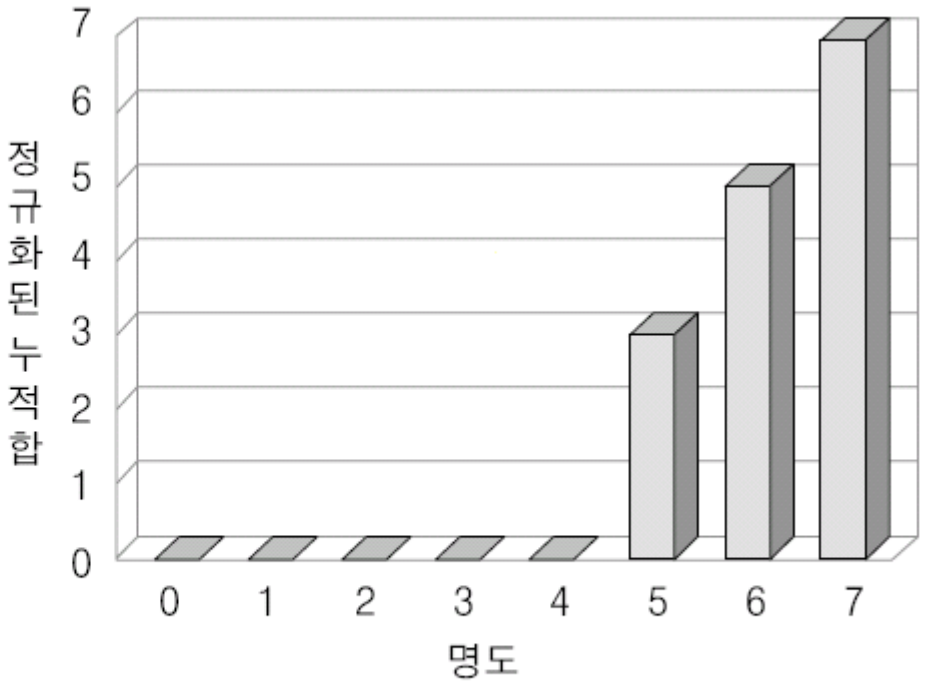


[그림 5-20] 원하는 히스토그램

히스토그램 명세화_3단계

- 3 단계
  - 원하는 히스토그램을 평활화하여 분포가 균일한 히스토그램을 만듦.

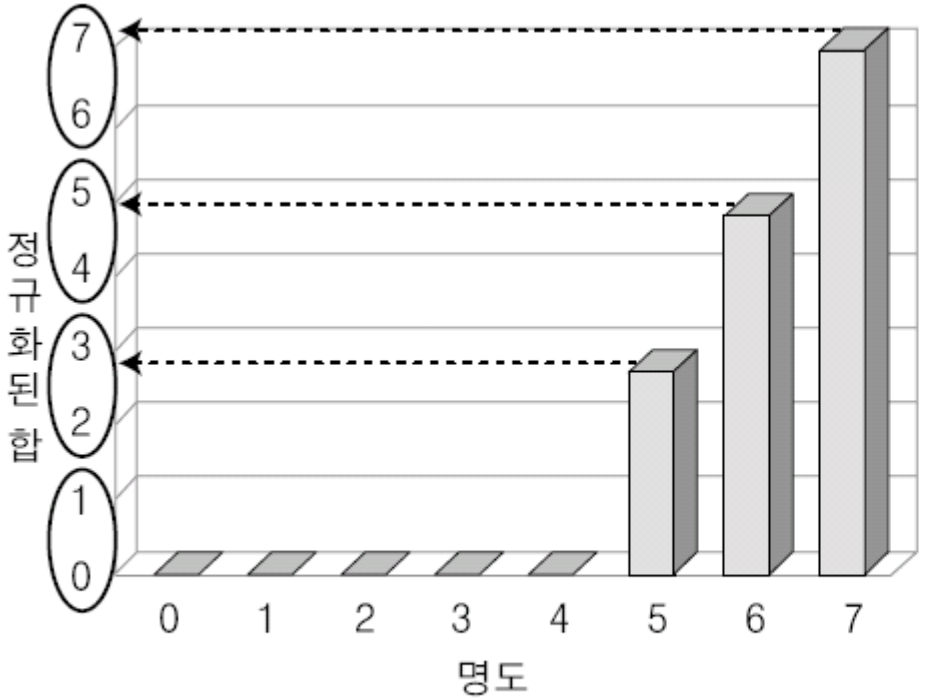
명도	누적합	정규화된 누적합
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	6	2.6
6	11	4.8
7	16	7.0



[그림 5-21] 원하는 히스토그램의 평활화

히스토그램 명세화_4단계

- 4 단계
  - 평활화된 히스토그램을 역 평활화하는 과정
  - 평활화와 반대로 정규화된 누적합이 명도 값이 되고, 명도 값은 역 평활화 값이 됨. 역 평활화 값이 역함수로서 룩업테이블로 사용됨.

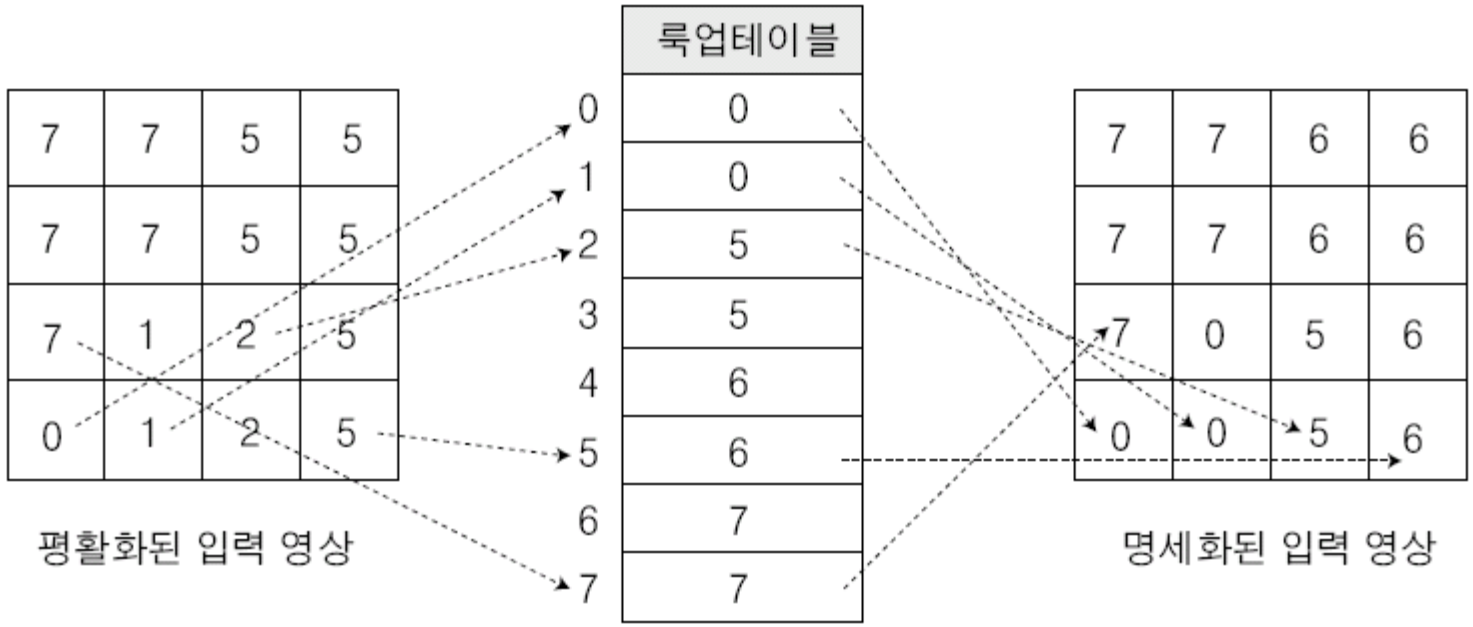


명도	누적합	평활화	역평활화
0	0	0	0
1	0	0	0
2	0	0	5
3	0	0	5
4	0	0	6
5	6	2.6	6
6	11	4.8	7
7	16	7.0	7

[그림 5-22] 원하는 히스토그램의 역평활화

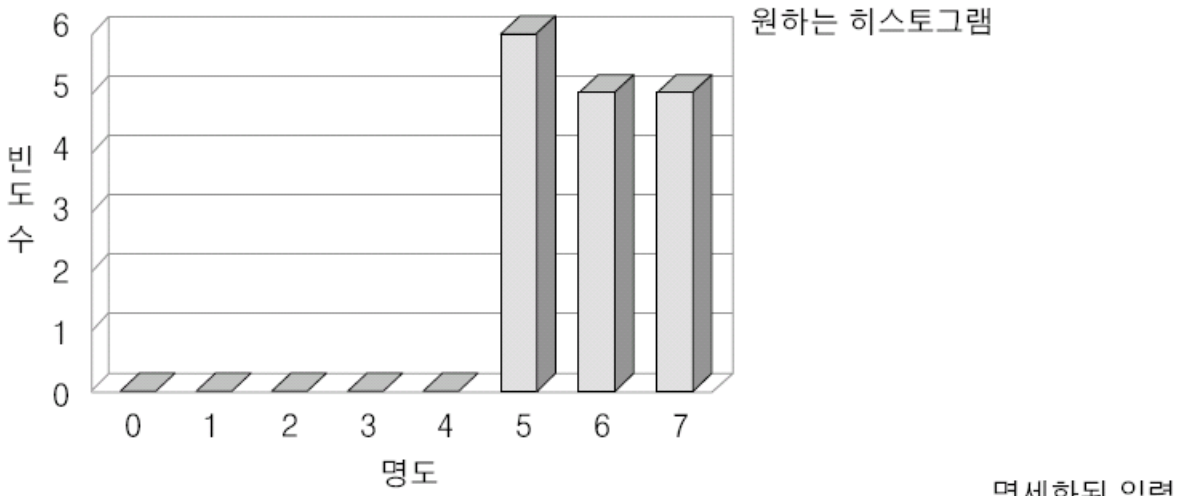
히스토그램 명세화_5단계

- 5 단계
  - 역변환 함수를 이용해 입력 영상을 원하는 히스토그램으로 만들어 줌.



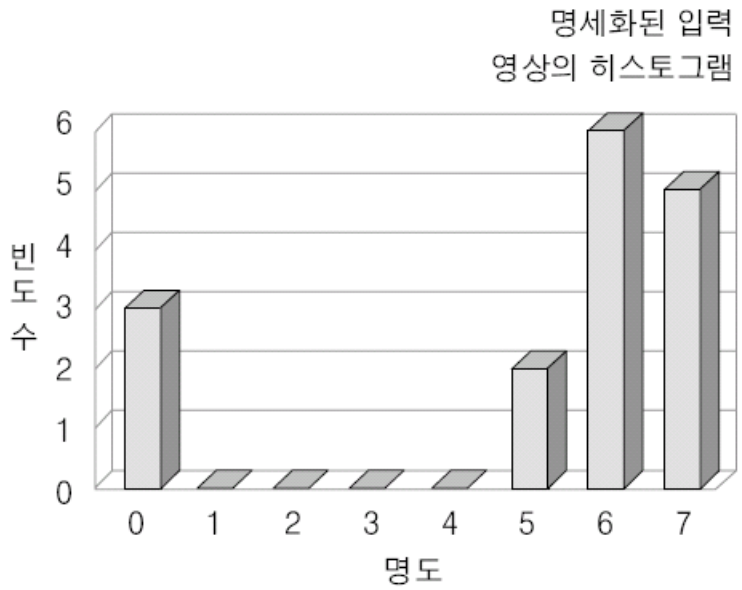
[그림 5-23] 룩업테이블을 이용한 명세화된 영상 생성

히스토그램 명세화_5단계(계속)



7	7	6	6
7	7	6	6
7	0	5	6
0	0	5	6

명세화된 입력 영상



[그림 5-24] 원하는 히스토그램과 명세화된 영상의 히스토그램 비교

[실습하기 5-5] 히스토그램 명세화 프로그램

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

ID	ID_HISTO_SPEC
Caption	히스토그램 명세화

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 히스토그램 명세화를 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnHistoSpec
Doc Class	void	OnHistoSpec

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 5-5] 히스토그램 명세화 프로그램

```
void CImageProcessingDoc::OnHistoSpec()
{
   int i, value, Dvalue, top, bottom, DADD;
   unsigned char *m_DTEMP, m_Sum_Of_ScHIST[256], m_TABLE[256];
   unsigned char LOW, HIGH, Temp, *m_Org_Temp;
   double m_DHIST[256], m_Sum_Of_DHIST[256], SUM = 0.0, DSUM = 0.0;
   double DMAX, DMIN;

   top = 255;
   bottom = top - 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_Org_Temp = new unsigned char [m_size];

   CFile File;
   CFileDialog OpenDlg(TRUE);
```

[실습하기 5-5] 히스토그램 명세화 프로그램

```
  // 원하는 히스토그램이 있는 영상을 입력받음
   if(OpenDlg.DoModal() == IDOK){
      File.Open(OpenDlg.GetFileName(), CFile::modeRead);

      if(File.GetLength() == (unsigned)m_size){
         m_DTEMP = new unsigned char[m_size];
         File.Read(m_DTEMP, m_size);
         File.Close();
      }
      else{
         AfxMessageBox("Image size not matched");
         // 같은 크기의 영상을 대상으로 함
         return;
      }
}

   LOW = 0;
   HIGH = 255;

   // 초기화
   for(i=0 ; i<256 ; i++){
      m_HIST[i] = LOW;
      m_DHIST[i] = LOW;
      m_TABLE[i] = LOW;
   }
```

[실습하기 5-5] 히스토그램 명세화 프로그램

```
// 빈도 수 조사
for(i=0 ; i<m_size ; i++){
   value = (int)m_InputImage[i];
   m_HIST[value]++;
   Dvalue = (int)m_DTEMP[i];
   m_DHIST[Dvalue]++;
}

// 누적 히스토그램 조사
for(i=0 ; i<256 ; i++){
   SUM += m_HIST[i];
   m_Sum_Of_HIST[i] = SUM;
   DSUM += m_DHIST[i];
   m_Sum_Of_DHIST[i] = DSUM;
}
// 원본 영상의 평활화
for(i=0 ; i<m_size ; i++){
   Temp = m_InputImage[i];
   m_Org_Temp[i]=(unsigned char)(m_Sum_Of_HIST[Temp]*HIGH/m_size);
}

// 누적 히스토그램에서 최소값과 최대값 지정
DMIN = m_Sum_Of_DHIST[0];
DMAX = m_Sum_Of_DHIST[255];
```

[실습하기 5-5] 히스토그램 명세화 프로그램

```
    // 원하는 영상을 평활화
    for(i=0 ; i<256 ; i++){
       m_Sum_Of_ScHIST[i]=(unsigned char)((m_Sum_Of_DHIST[i]
       -DMIN)*HIGH/(DMAX - DMIN));
    }


    // 룩업테이블을 이용한 명세화
    for( ; ; ){
       for(i=m_Sum_Of_ScHIST[bottom] ;
       i <= m_Sum_Of_ScHIST [top] ; i++){
       m_TABLE[i] = top;
    }
    top = bottom;
    bottom = bottom - 1;

    if(bottom < -1)
    break;
    }

    for(i=0 ; i<m_size ; i++){
       DADD = (int)m_Org_Temp[i];
       m_OutputImage[i] = m_TABLE[DADD];
    }
}
```

[실습하기 5-5] 히스토그램 명세화 프로그램

④ View 클래스에 다음 프로그램 추가

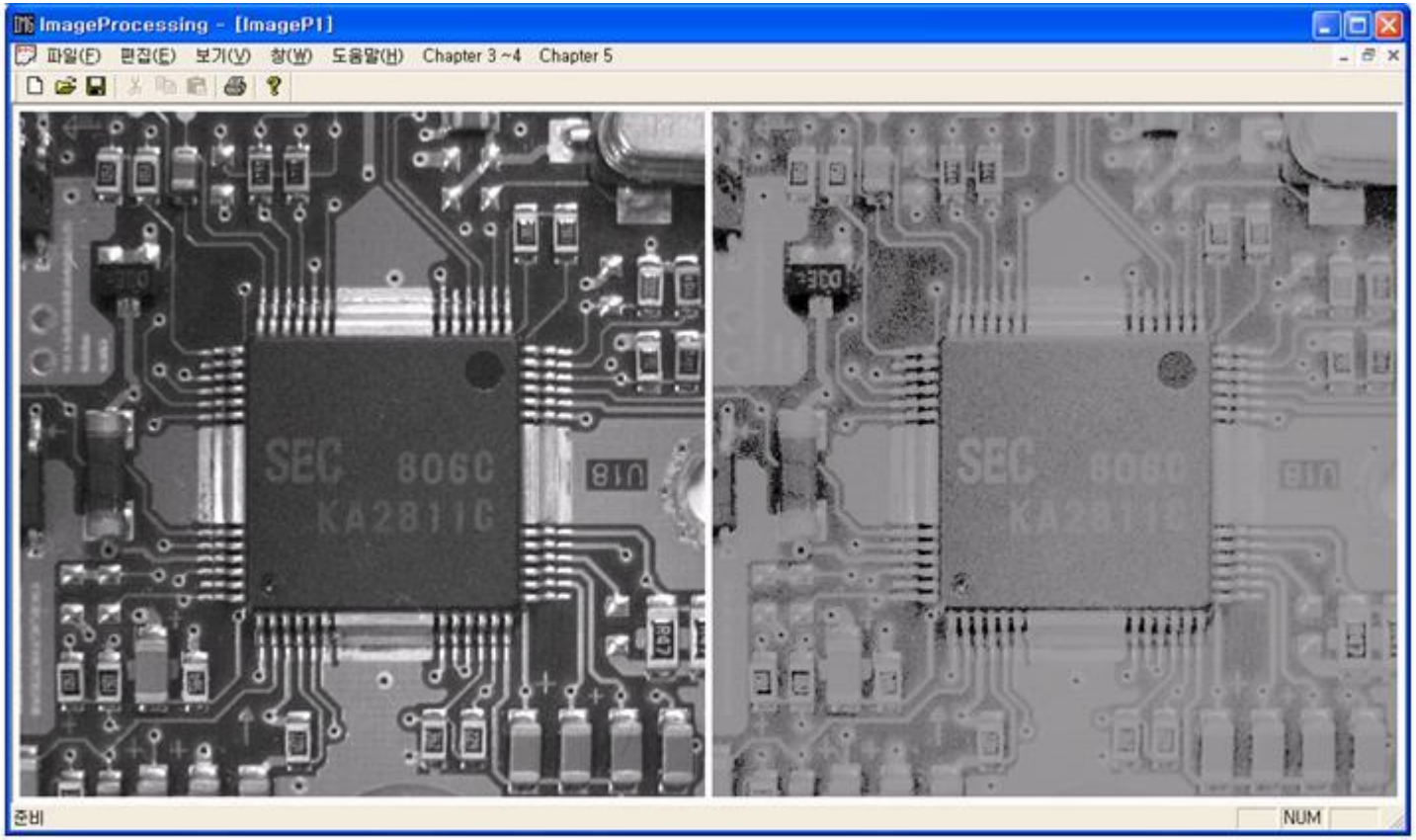
```
void CImageProcessingView::OnHistoSpec
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnHistoSpec

    Invalidate(TRUE);
}
```

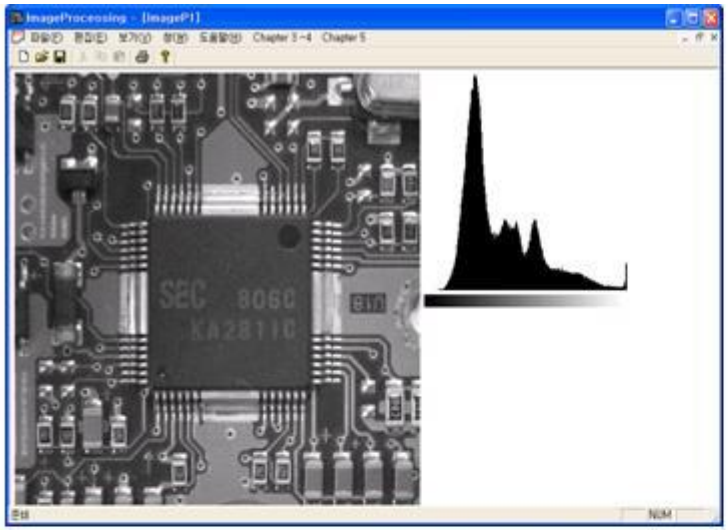
[실습하기 5-5] 히스토그램 명세화 프로그램

⑤ 프로그램 실행 결과 영상

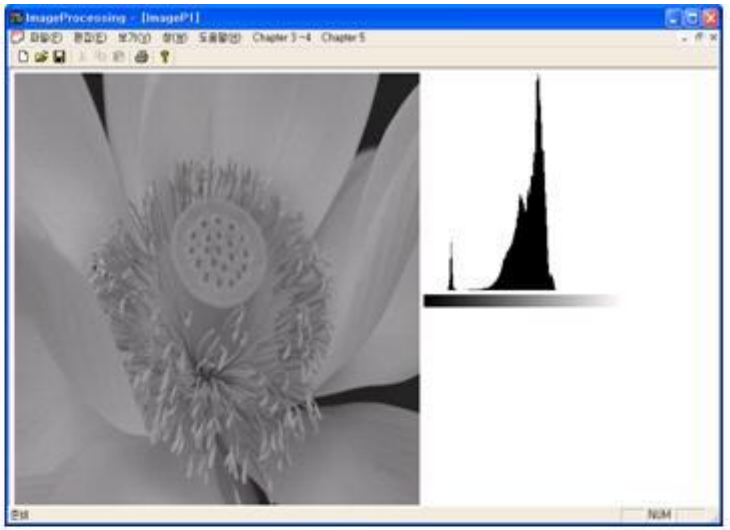


히스토그램 명세화 프로그램을 구현한 결과 영상(semiconduct512 영상의 히스토그램을 flower 512 영상의 히스토그램으로 명세화)

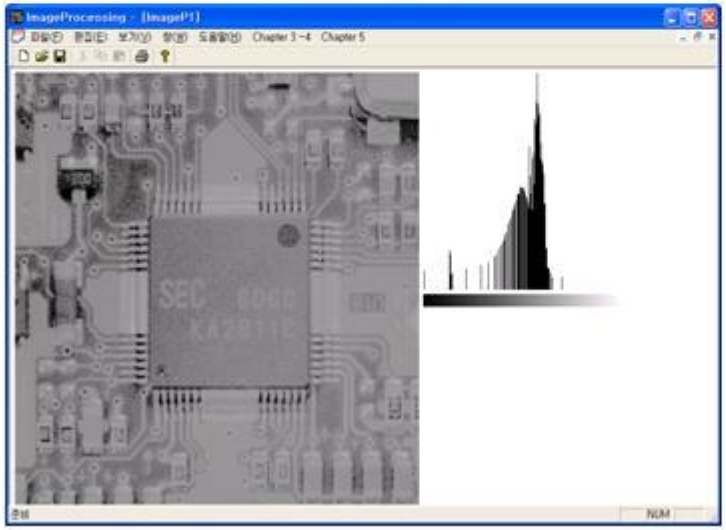
[실습하기 5-5] 히스토그램 명세화 프로그램



semiconduct512 영상의 히스토그램



flower512 영상의 히스토그램



flower512 영상의 히스토그램으로 명세화된 semiconduct512 영상과 히스토그램

요약

- **RGB 컬러 영상의 히스토그램**
  - 각 채널에서 히스토그램을 생성.
  - **R, G, B** 채널에는 채널별로 각각 히스토그램이 있음.
- 디지털 영상에서 산술연산: 히스토그램의 기둥을 왼쪽, 오른쪽으로 이동시키거나 기둥의 폭을 조절함
  - 덧셈연산: 히스토그램의 기둥을 오른쪽으로 이동시킴.
  - 뺄셈연산: 히스토그램의 기둥을 왼쪽으로 이동시킴.
  - 곱셈연산: 수행한 영상의 히스토그램은 기둥의 분포가 넓음
  - 나눗셈연산: 최대 명도 값과 최소 명도 값의 차이가 작아져 명암 대비가 감소하고 히스토그램의 분포도 좁음.
- 히스토그램 스트레칭
  - 명암 대비를 향상시키는 연산으로, 낮은 명암 대비를 보이는 영상의 품질을 향상시키는 기법
  - 명암 대비 스트레칭이라고도 함.
  - 히스토그램 스트레칭을 수행한 디지털 영상은 모든 범위의 화소 값을 포함하며, 히스토그램은 이상적인 형태인 전 구간에 걸쳐 분포가 균일

요약

- 엔드-인(end-in) 탐색 기법
  - 히스토그램이 전 구간에 분포하지만 특정 부분에 집중되며, 최저와 최고의 명도 값 부근은 아주 빈약한 영상의 품질을 향상시킬 수 있음.
  - 일정한 양의 화소를 흰색 또는 검정색을 갖도록 지정하여 히스토그램의 분포를 좀 더 균일하게 함
- 히스토그램 평활화 기법
  - 명암 분포가 빈약한 영상을 분포가 균일한 영상으로 만듦.
  - 즉, 개략적인 모습은 원 영상 히스토그램과 유사하게 하면서 명암도의 분포를 좀 더 균일화하는 작업
  - 특정 모양의 히스토그램을 생성된 디지털 영상의 히스토그램에 포함하여 영상의 일부 영역의 명암 대비(콘트라스트)를 개선시키는 기술
- 히스토그램 명세화 과정
  - 기본적으로 입력 영상을 원하는 히스토그램으로 평활화하고 역 히스토그램 평활화를 수행
  - 그런 뒤 룩업테이블(lookup table)을 생성하고 평활화된 원 영상을 역 변환하여 원하는 히스토그램을 얻음.

Thank you