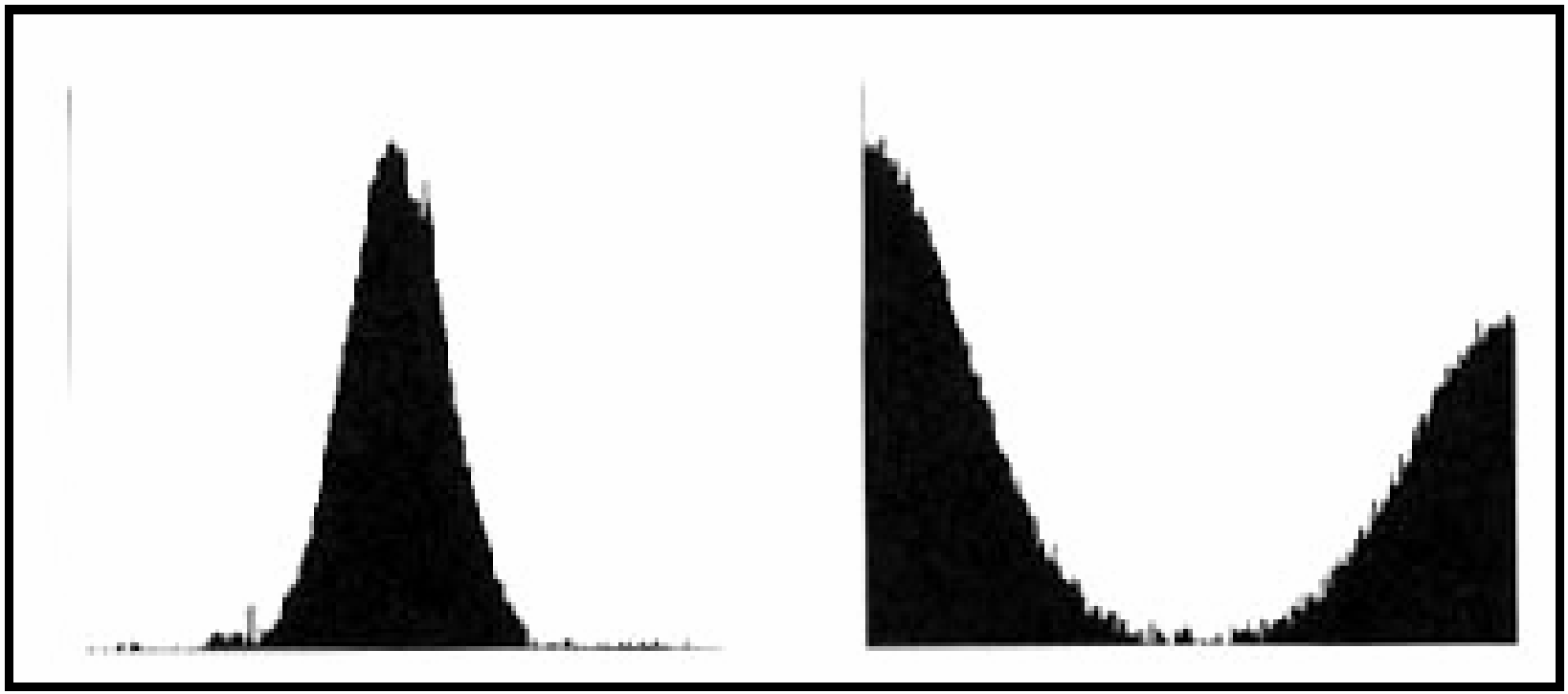
명암 대비 스트레칭
(Contrast Stretching)

명암 대비 스트레칭(Contrast Stretching)

- 명암대비 (contrast)
  - 밝거나 어두운 화소의 분포
- 낮은 명암 대비
  - 영상의 히스토그램에서 화소가 히스토그램의 왼쪽, 오른쪽 또는 중앙의 오른쪽에 밀집.
- 높은 명암 대비
  - 어둡고 밝은 영역을 모두 포함
  - 히스토그램은 두 개의 큰 마루를 가진다
    - 마루 중의 하나는 히스토그램의 왼쪽에, 나머지 하나는 오른쪽에 위치하는 특징을 갖는다
    - 예로 햇빛 밝은 창문 앞에 서 있는 사람의 영상은 높은 명암대비를 갖는다.

명암 대비 스트레칭(Contrast Stretching)



낮은 명암 대비와 높은 명암 대비를 가진 히스토그램

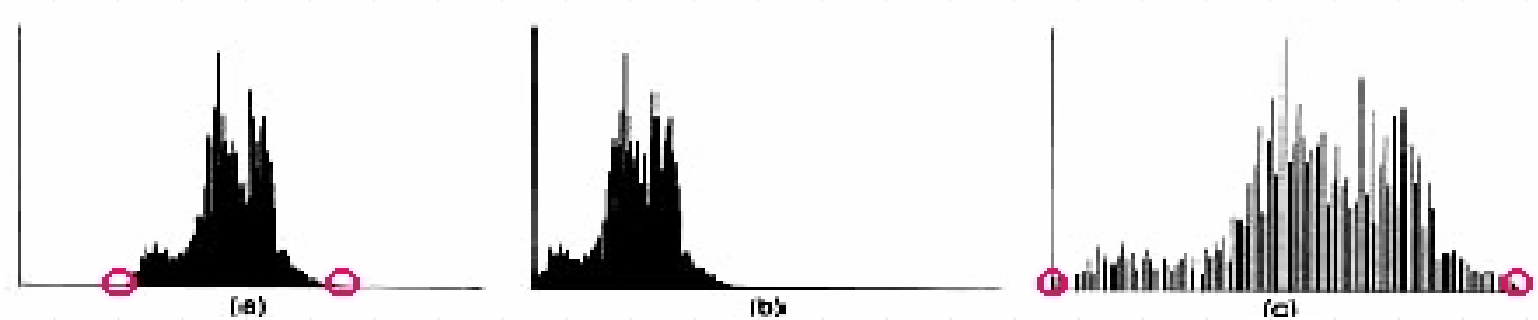
명암 대비 스트레칭(Contrast Stretching)

- 좋은 명암 대비
  - 넓은 범위의 화소 값이 균일하게 분포를 나타내므로 히스토그램은 마루나 골이 부각되지 않는다.
- 명암대비 스트레칭
  - 명암 값 분포를 골고루 펼치기 위해 사용

명암 대비 스트레칭(Contrast Stretching)

- 기본 명암대비 스트레칭
  - 중앙에 명암값이 치우치는 히스토그램을 가진 영상에 잘 적용됨

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



(a) 입력영상 (b) 히스토그램 – low (c) (히스토그램-low)*255 /(high-low)

명암 대비 스트레칭(Contrast Stretching)

- 엔드인 탐색(End-in search)
  - 히스토그램의 특정부분에 화소들이 치우친 영상에 가장 잘 적용됨
  - 일정한 양의 화소를 흰색 또는 검정색을 갖도록 지정
  - **Low**
    - 낮은 범위의 임계값(Intensity)
    - 낮은 범위에서 지정한 양이 도달한 히스토그램값
  - **High**
    - 높은 범위의 임계값(Intensity)
    - 높은 범위에서 지정한 양이 도달한 히스토그램값

$$output(x) = \begin{cases} 0 & for\ x \le low \\ 255*(x-low)/(high-low) & for\ low < x \le high \\ 255 & for\ high \le x \end{cases}$$

명암 대비 스트레칭(Contrast Stretching)

```
void CTestDoc::Histo_Stretch()
{
	int histogram[256];
	unsigned char LUT[256];
	int lowthresh,highthresh;
	float scale_factor;//scaling factor for contrast stretch

	//intialalize thresholds
	lowthresh=0;
	highthresh=255;

	//히스토그램 명도 값의 빈도 수 조사
	for(int i=0; i<256; i++)
		histogram[i]=0;
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			histogram[(int)m_OpenImg[i][j]]++;
		}
	}
```

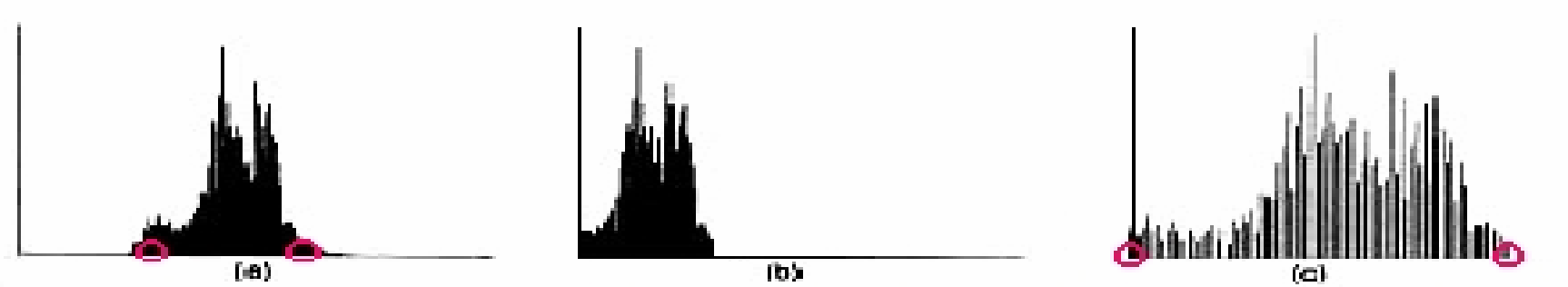
명암 대비 스트레칭(Contrast Stretching)

```
//히스토그램 중 가장 낮은 명도 값 계산.
for(i=0; i<256; i++){
        if (histogram[i]){
                lowthresh=i;
                break;
        }
}

//히스토그램 중 가장 높은 명도 값 계산.
for(i=255; i>=0; i--){
        if(histogram[i]){
                highthresh=i;
                break;
        }
}
```

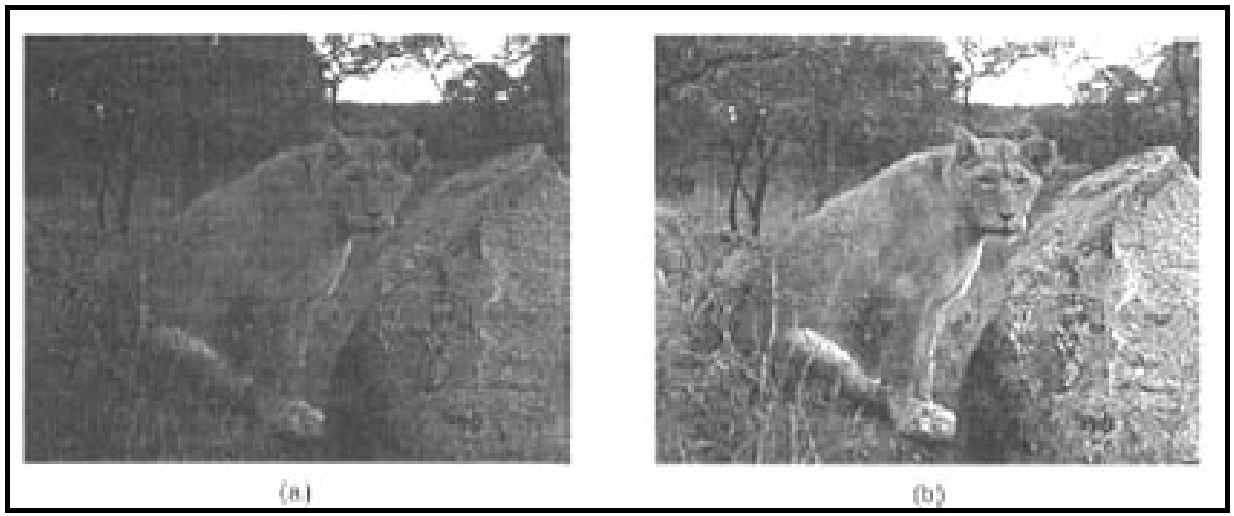
명암 대비 스트레칭(Contrast Stretching)

```
        //룩업 테이블 계산.
        for(i=0; i<lowthresh; i++)
                LUT[i]=0;
        for(i=255; i>highthresh; i--)
                LUT[i]=255;

        //스트레칭의 강도를 조절할 수 있는 factor계산.
        scale_factor=(float)(255.0/(highthresh-lowthresh));
        for(i=lowthresh; i<=highthresh; i++)
                LUT[i]=(unsigned char)((i-lowthresh)*scale_factor);

        //룩업 테이블에서 계산된 출력 값 할당.
        for(i=0; i<256; i++){
                for(int j=0; j<256; j++){
                        m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
                }
        }
}
```

명암 대비 스트레칭(Contrast Stretching)



(a) 입력영상 (b) 히스토그램 – low (c) 앤드인탐색기법적용

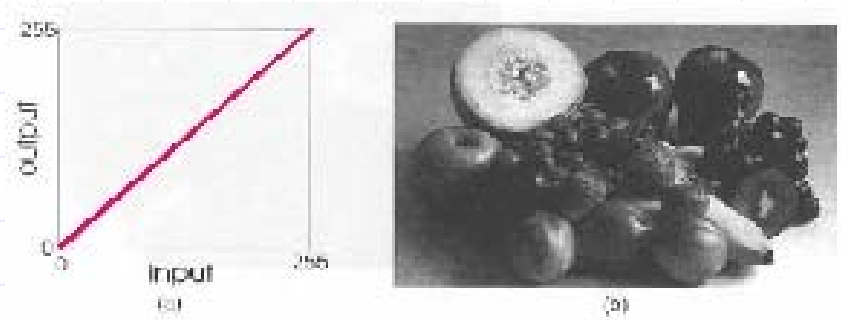


(a) 입력영상 (b) 명암대비가 스트레치 된 영상

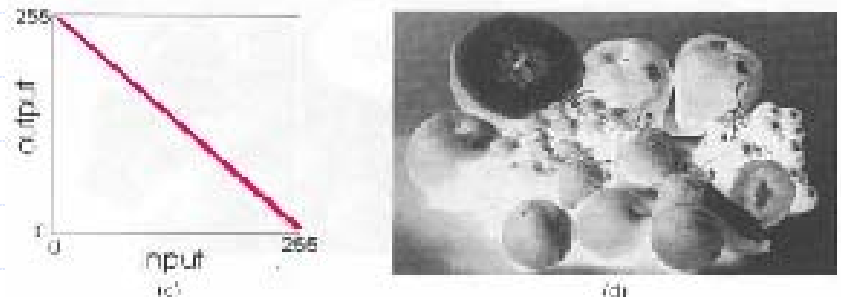
명암 변환

- 미리 저정된 함수를 기반으로 이전 화소를 새로운 화소로 변환하는 포인터 처리
- 널변환 : 이전화소 = 새로운 화소
- 역변환 : 영상을 역으로 변환(사진학적 역변환과 유사)

널변환



역변환



명암 변환

```
void CTestDoc::Invert()
{

	unsigned char LUT[256];
	//역 변환 연산
	for(int i=0;i<256;i++)
	{
		LUT[i] = 255 - i;
	}
	//룩업 테이블에서 계산된 출력 값 할당.
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

명암 변환

- 영상의 명암은 감마 상관관계에 의해 조절될 수 있다.
  - 영상 센스 출력기 및 필름 등에서 발생하는 비선형적인 반응들을 보간하는 영상처리를 수행

$$output = input^{\frac{1}{\gamma}}$$

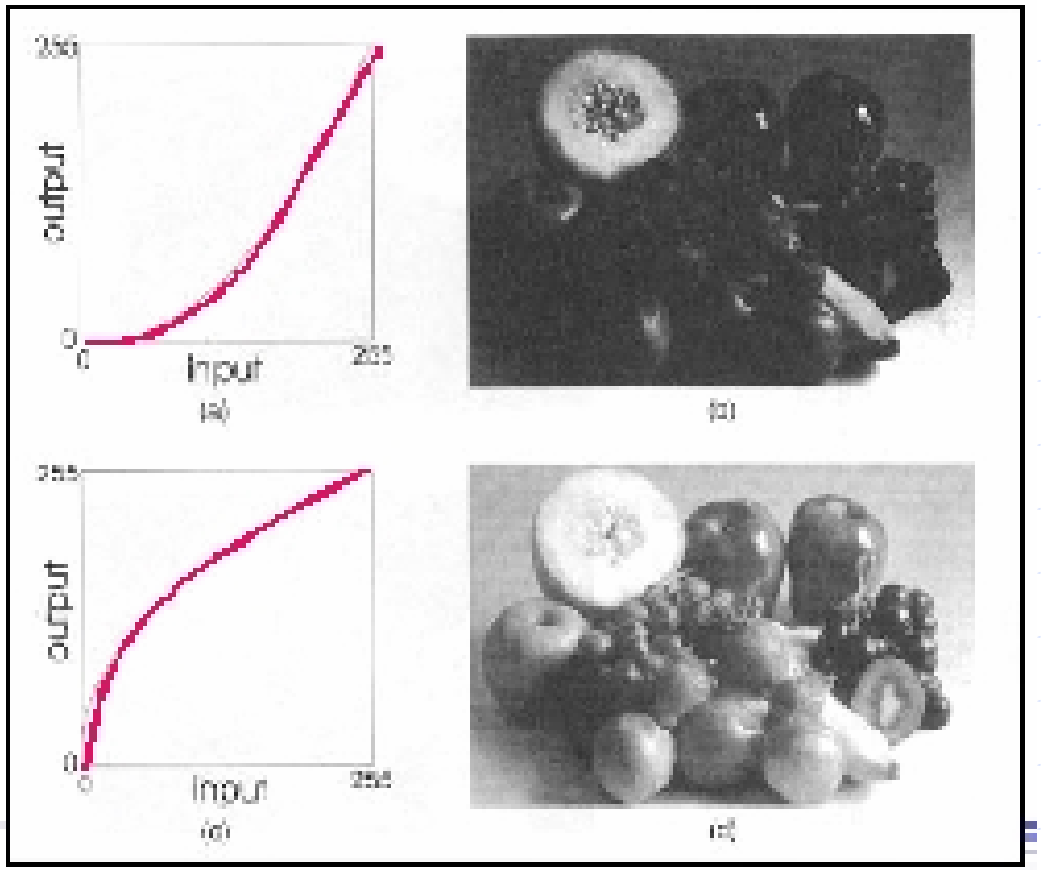
  - 감마값==1.0 : 널변환
  - 0.0 < 감마값 < 1.0 : 영상이 흐려짐
  - 감마값 > 1.0 : 영상이 밝아짐
- **GRB 모니터의 감마값 : 1.4~2.8**

명암 변환

- 감마값이 0.45일때와 2.2일때의 감마상관관계 변환

r=0.45

r=2.2



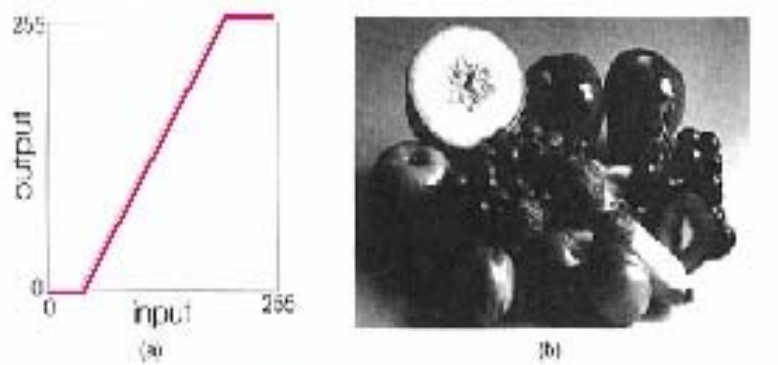
명암 변환

```
void CTestDoc::Gamma()
{
#define GAMMA       2.2
#define EXP                     1/(GAMMA)
#define transformation(x)       pow((double)(x) / 255.0, (EXP))

    unsigned char LUT[256];

    //감마 상관관계 연산
    for(int i=0;i<256;i++)
    {
        LUT[i] = (unsigned char)transformation(i);
    }
    //룩업 테이블에서 계산된 출력 값 할당.
    for(i=0; i<256; i++){
        for(int j=0; j<256; j++){
            m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
        }
    }
}
```

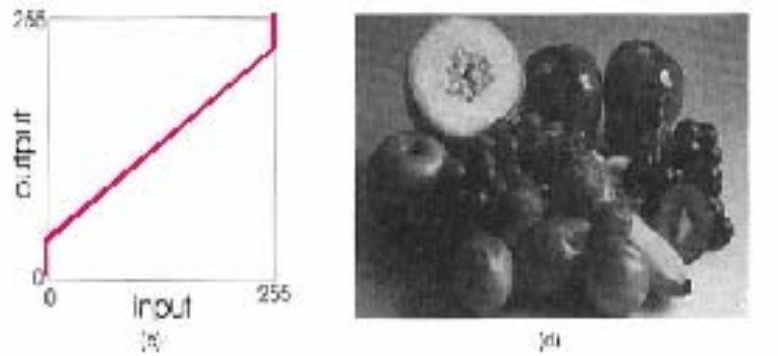
명암 변환

- 명암대비 스트레칭 / 압축
  - 높은 명암대비를 가진 영상의 명암분포를 향상시키는 방법
  - 가장 밝은 값은 어둡게, 가장 어두운 값을 밝게 처리

명암대비 스트레칭

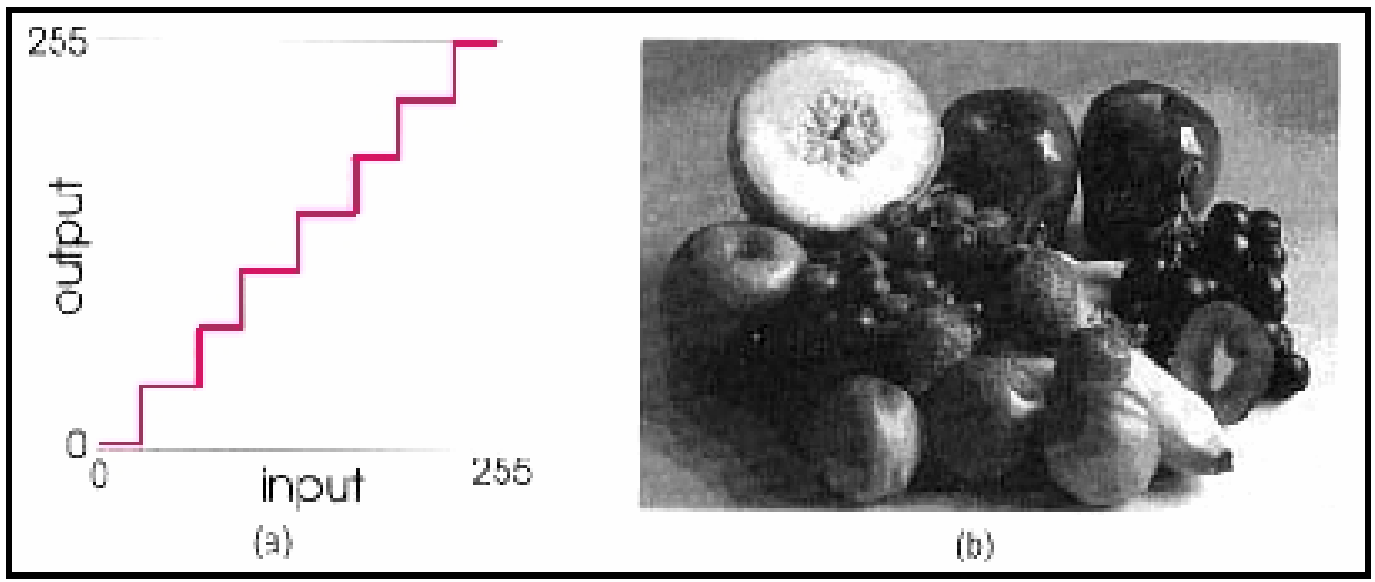


명암대비 압축



명암 변환

- Posterizing
  - 화소가 가질 수 있는 명암값의 범위를 축소



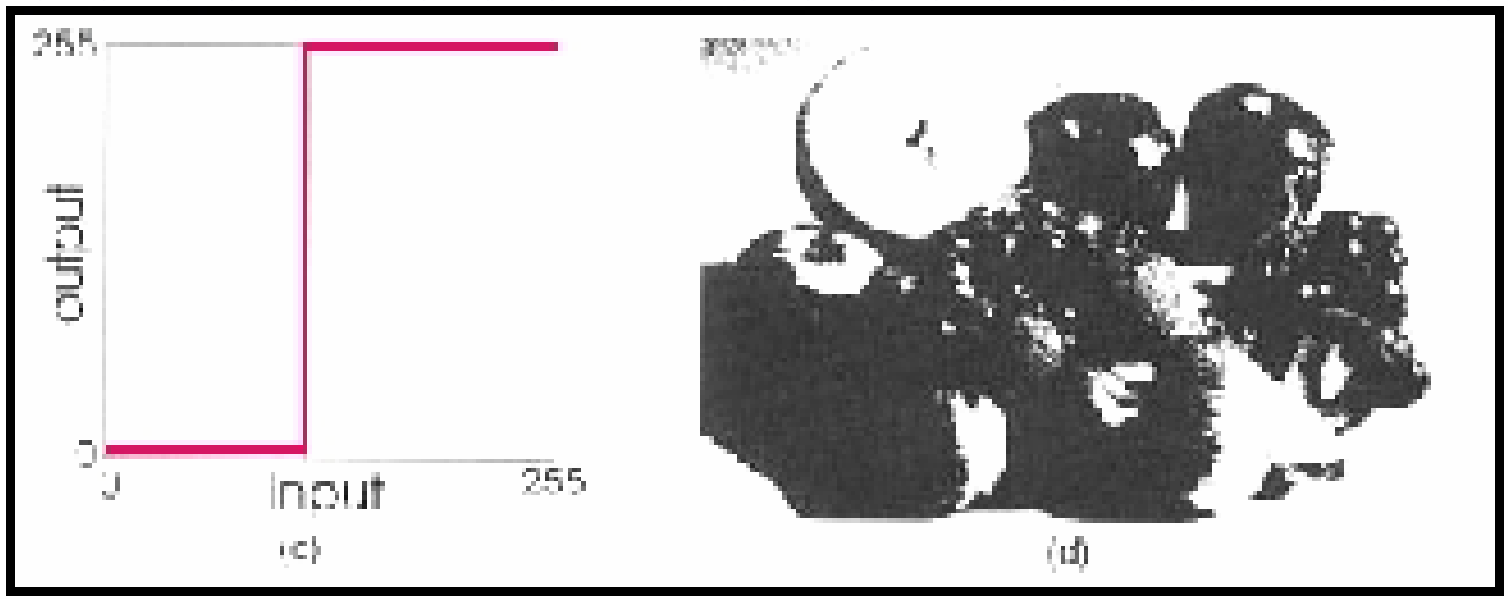
8 레벨 포스터라이징 변환

명암 변환

```
void CTestDoc::Posterizing()
{
	unsigned char LUT[256];
	//Posterizing 연산
	for (int j=0;j<7;j++){
		for(int i=j*32;i<j*32+32;i++)
		{
			LUT[i] = j;
		}
	}
	//룩업 테이블에서 계산된 출력 값 할당.
	for(int i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

명암 변환

- 임계값을 이용한 변환
  - 명암값의 범위를 2로 축소시킬 때 사용

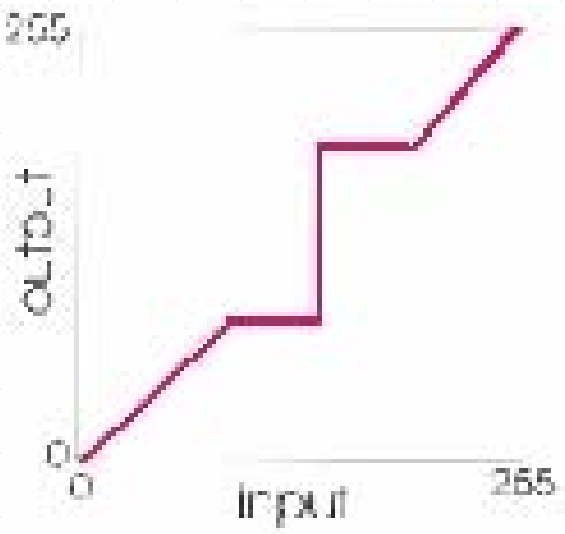


명암 변환

```
void CTestDoc::Boundary()
{
	unsigned char LUT[256];
	//범위가 2인 입계값 연산
	for(int i=0;i<128;i++)
	{
		LUT[i] = 0;
	}
	for(i=128;i<256;i++)
	{
		LUT[i] = 255;
	}

	//룩업 테이블에서 계산된 출력 값 할당.
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

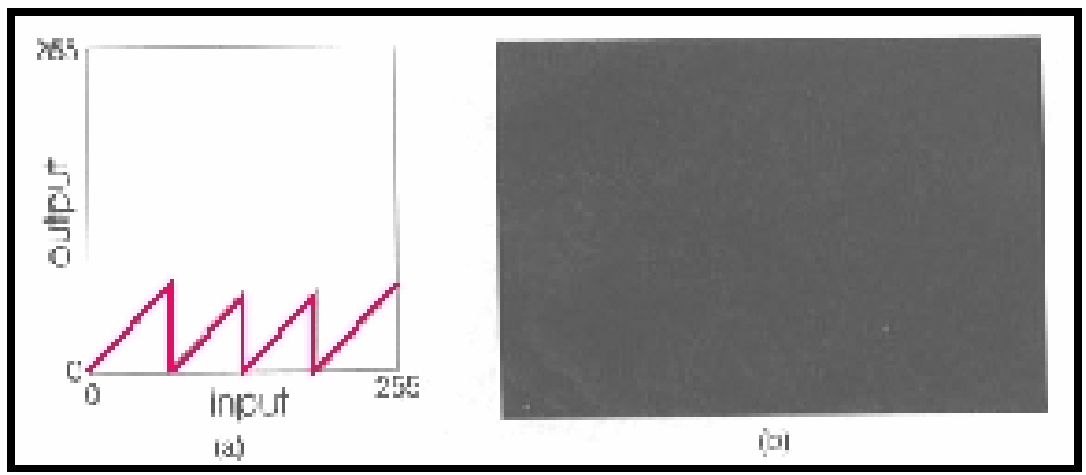
명암 변환

- 범위가 주어진 임계값 방법
  - 임계값을 제한된 범위로 축소하고 다른 입력 화소들은 널변환



명암 변환

- 비트클리핑(bit-Clipping)
  - 화소의 최상이 비트 중 일정부분을 0으로 설정
  - 넓은 범위의 명암값을 가진 영상을 동일한 명암주기를 가진 몇몇의 부영역으로 분할하는 효과
  - 2Bit-Clipping
    - 0~255까지의 명암값 범위 -> 0~63까지의 범위를 가진 4개의 부 구간으로 분할

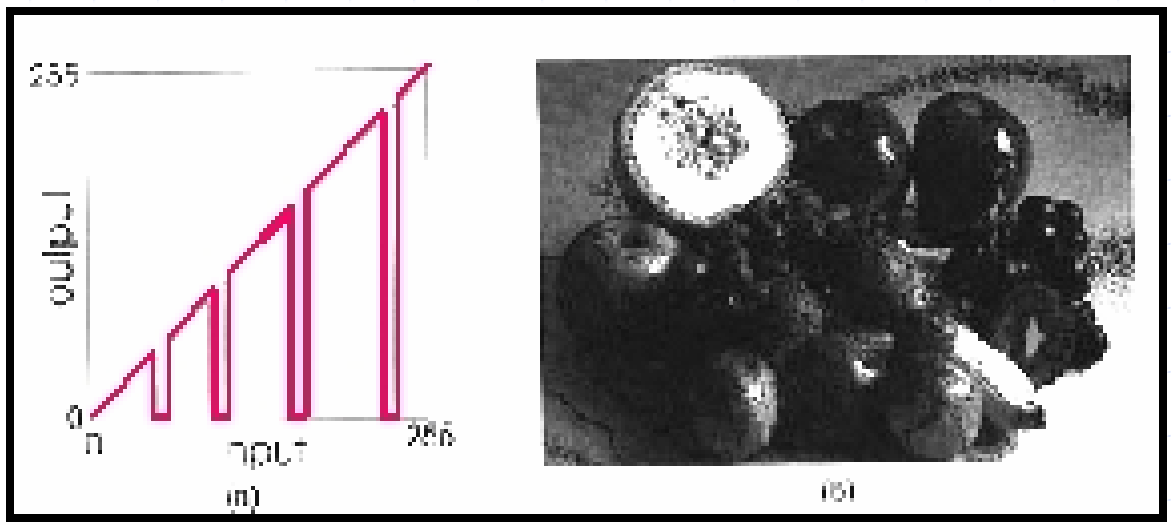


명암 변환

```
void CTestDoc::BitClipping()
{
	unsigned char LUT[256];
	//2Bit Clipping 변환
	for(int i=0;i<4;i++)
	{
		int k = 0;
		for(int j=i*64;j<i*64+64;j++)
		{
			LUT[j] = k++;
		}
	}
	//룩업 테이블에서 계산된 출력 값 할당.
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

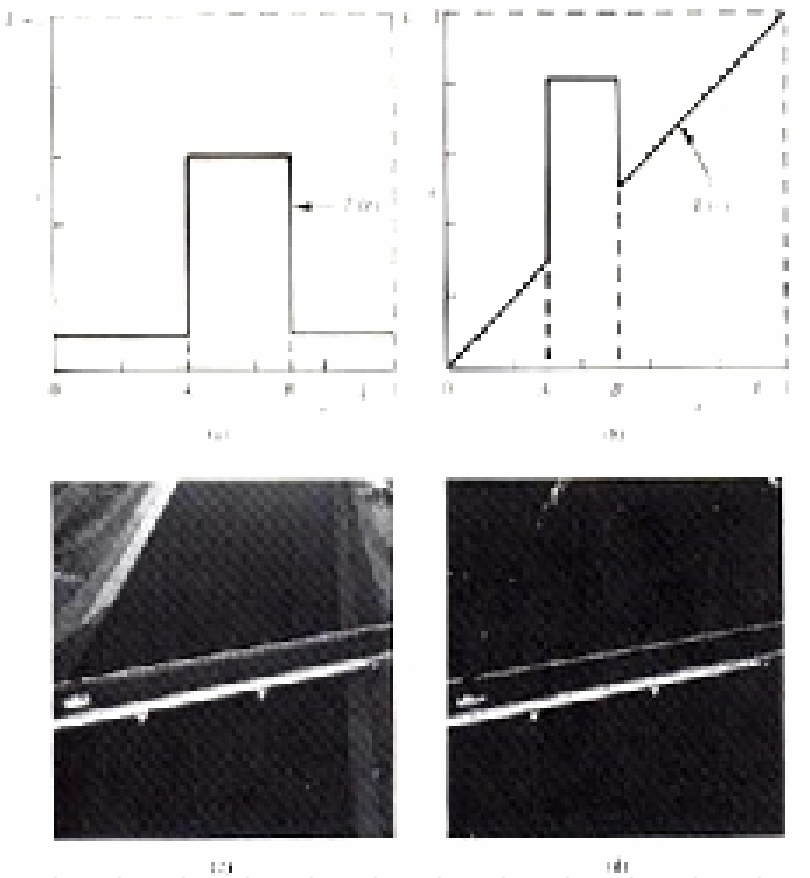
명암 변환

- 특정한 입력 명암값을 흰색 또는 검정색으로 지정
- 특정 구간에 존재하는 영상의 윤곽선을 찾기 위해 사용
- 몇몇의 지정된 명암값만을 변형하고 대부분의 입력 영상 자료는 원래대로 유지



명암 변환

- 일정 밤위의 화소만 강조



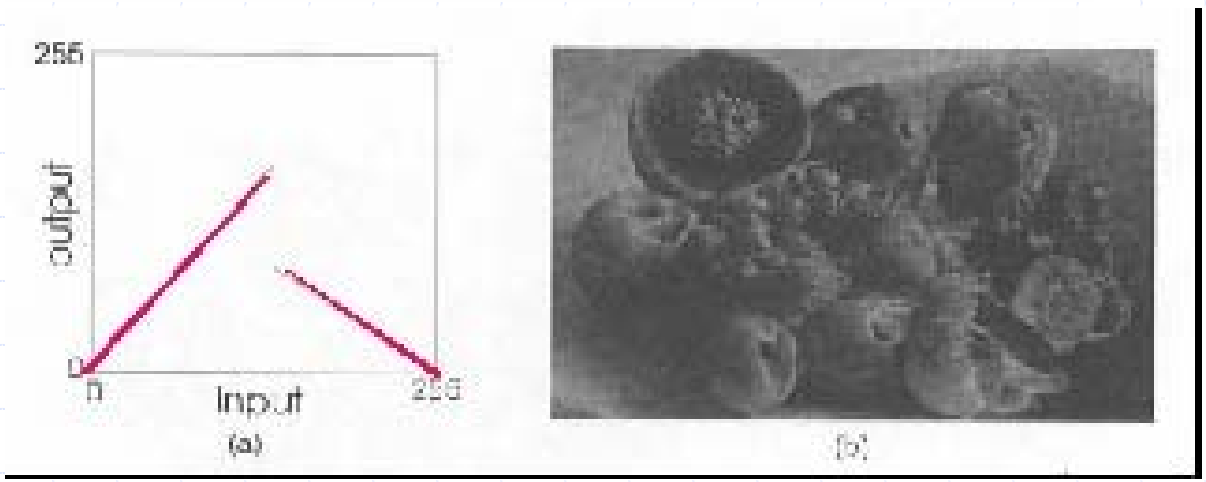
명암 변환

```
void CTestDoc::Spotright()
{
	unsigned char LUT[256];
	//범위 강조 변환
	for(int i=0;i<256;i++)
	{
		LUT[i] = i;
	}
	for(i=64;i<192;i++)
	{
		LUT[i] = 255;
	}
	//룩업 테이블에서 계산된 출력 값 할당.
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

명암 변환

- 솔라라이징(Solarizing)
  - th = threshold

$$output(x) = \begin{cases} x & for\ x \le th \\ 255 - x & for\ < th \end{cases}$$



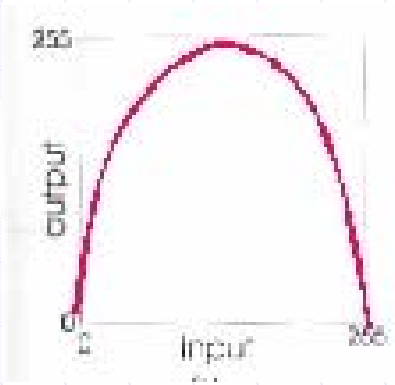
명암 변환

```
void CTestDoc::Solarizing()
{
#define Threshold      128
	unsigned char LUT[256];
	//솔라라이징 변환 강조 변환
	for(int i=0;i<Threshold;i++)
	{
		LUT[i] = i;
	}
	for(i=Threshold;i<256;i++)
	{
		LUT[i] = 255-i;
	}
	//룩업 테이블에서 계산된 출력 값 할당.
	for(i=0; i<256; i++){
		for(int j=0; j<256; j++){
			m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
		}
	}
}
```

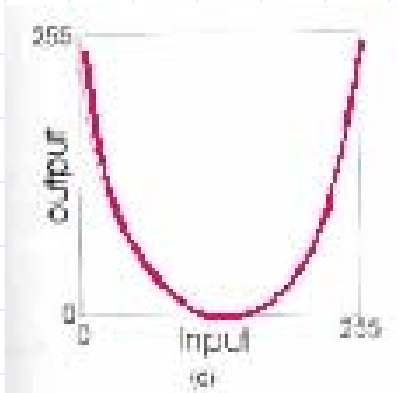
명암 변환

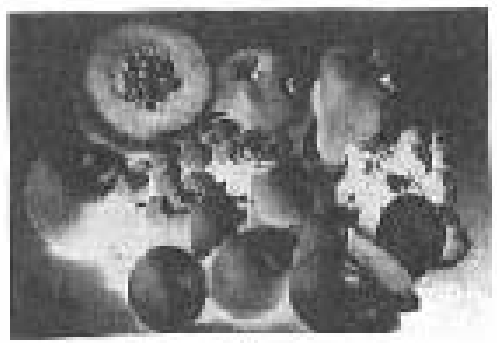
- 파라볼라(Parabola) 변환
  - 2nd Parabola

$$output(x) = 255 - 255(x/128 - 1)^2$$



$$output(x) = 255(x/128 - 1)^2$$





명암 변환

```
void CTestDoc::Parabola1()
{
    unsigned char LUT[256];
    //첫번째 파라볼라 변환
    for(int i=0;i<256;i++)
    {
        LUT[i] = 255 - 255 * pow((double)(i/128.0)-1.0, 2);
    }

    //룩업 테이블에서 계산된 출력 값 할당.
    for(i=0; i<256; i++){
        for(int j=0; j<256; j++){
            m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
        }
    }
}
```

명암 변환

```
void CTestDoc::Parabola2()
{
    unsigned char LUT[256];
    //두번째 파라볼라 변환
    for(int i=0;i<256;i++)
    {
        LUT[i] = 255 * pow((double)(i/128.0)-1.0, 2);
    }

    //룩업 테이블에서 계산된 출력 값 할당.
    for(i=0; i<256; i++){
        for(int j=0; j<256; j++){
            m_ResultImg[i][j]=LUT[(unsigned char)m_OpenImg[i][j]];
        }
    }
}
```