Arithmetic 연산 및 Gray-level 변환을 사용한 영상 개선

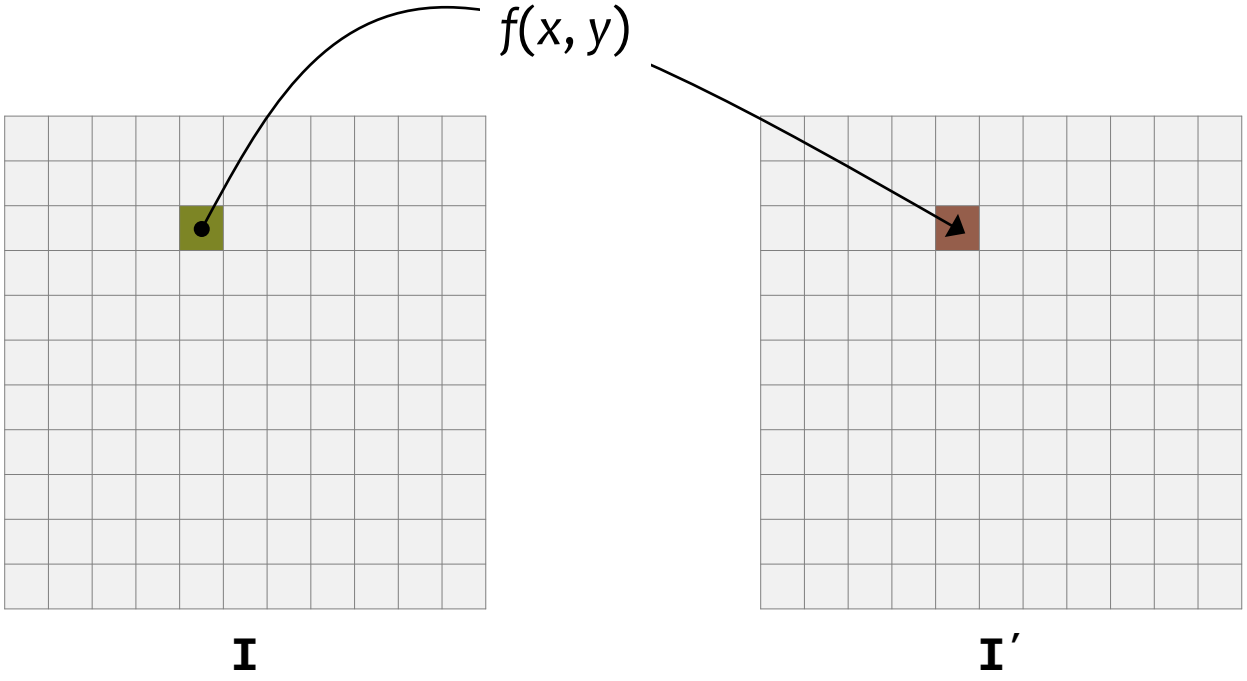
김성영교수
금오공과대학교
컴퓨터공학과

학습 내용

- 단일 픽셀 처리(point pixel processing) 개요
- Scalar and Image arithmetic operations
- Gray-level transformations

point pixel processing

- 이웃 픽셀과는 **독립**적으로 입력 영상의 각 픽셀 값을 변환한 후 결과 영상의 동일한 위치에 출력하는 연산



I = I′: in-place transformation

techniques

Arithmetic operations

Gray-level transformations

Histogram modifications

objective

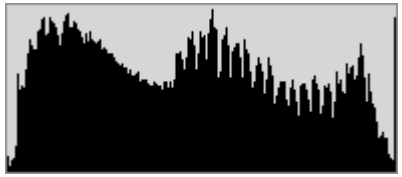
Improving *image **contrast*** and *brightness*

Image contrast: a measure of the distribution and range of the gray levels

the difference between the brightness and darkest pixel values, and
how the intermediate values are arranged

Image brightness: the overall average or mean pixel value in the image

Contrast & Brightness



Scalar arithmetic operation

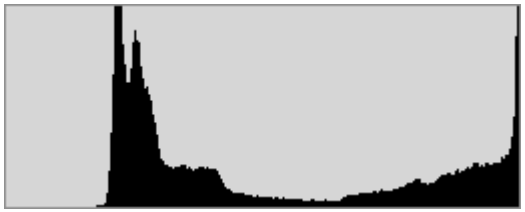
연산	구 현
덧셈	OutImage[x][y] = InImage[x][y]+C1
뺄셈	OutImage[x][y] = InImage[x][y]-C2
곱셈	OutImage[x][y] = InImage[x][y]*C3
나눗셈	OutImage[x][y] = InImage[x][y]/C4

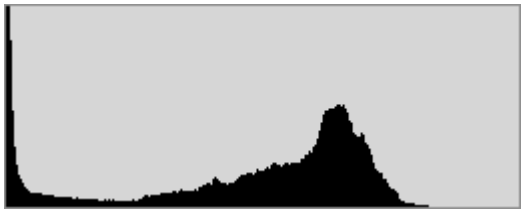
❖ **클리핑(clipping) 처리**

```
if( OutImage[x][y] > 255 )  OutImage[x][y] = 255;
if( OutImage[x][y] < 0 )  OutImage[x][y] = 0;
```

덧셈 연산 (+50)

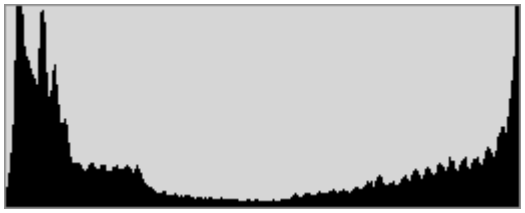
뺄셈 연산 (-50)

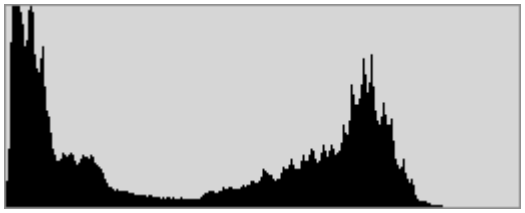




곱셈 연산 (*1.2)

나눗셈 연산 (/1.2)





```
void scalarAdd( Mat &image, Mat &result, int value )
{
    int numOfLines  = image.rows;  // number of lines in the image
    int numOfPixels = image.cols;  // number of pixels per a line

    result.create( image.rows, image.cols, image.type() );
    for( int r=0; r<numOfLines; r++ )
    {
        const uchar *data_in  = image.ptr<uchar>( r );
        uchar *data_out       = result.ptr<uchar>( r );
        for( int c=0; c<numOfPixels; c++ )
        {
            data_out[c] = saturate_cast<uchar>( data_in[c] + value );
        }
    }
}
```

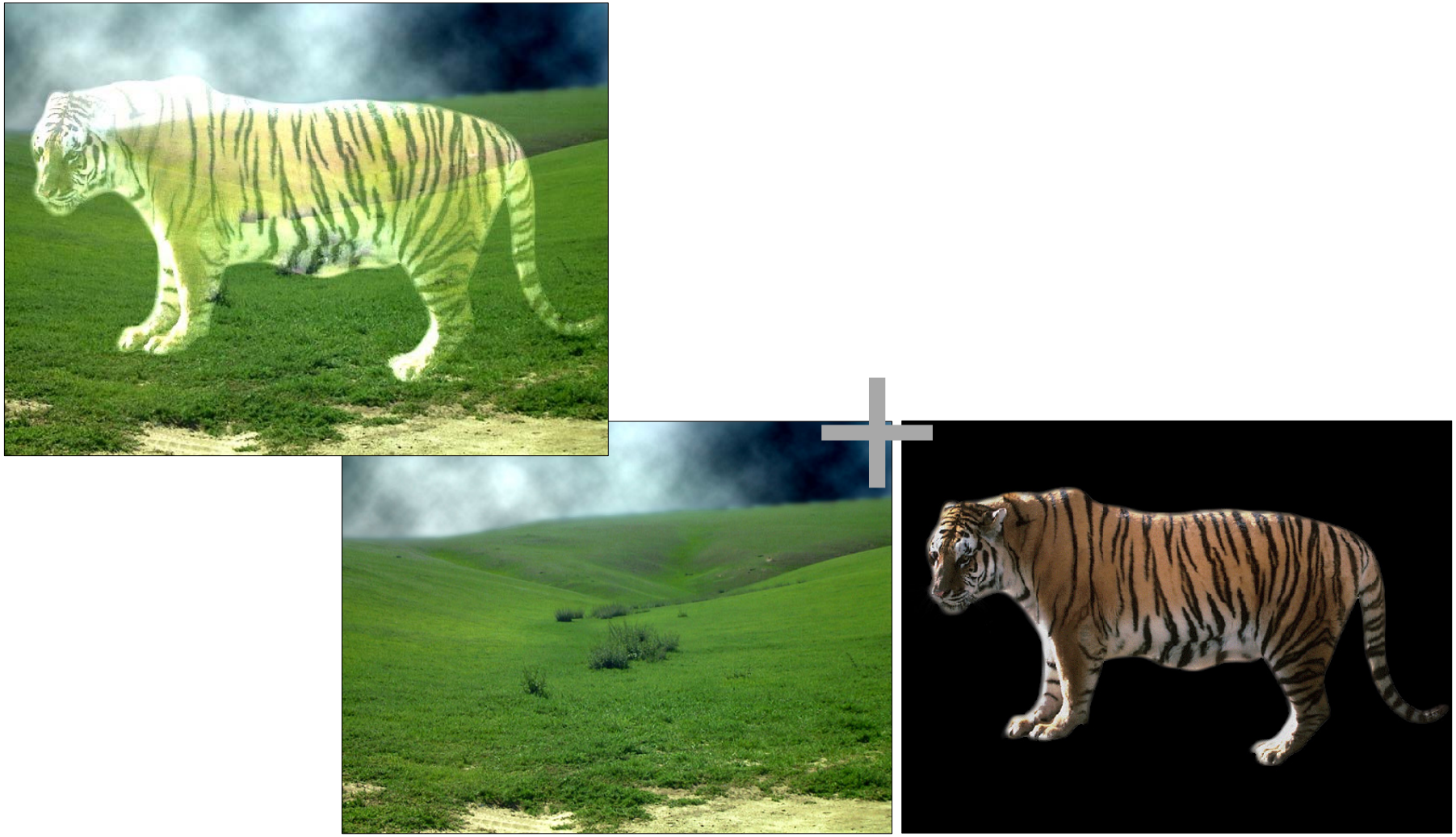
```
cv::add( image, cv::Scalar(value), result )
result = image + value

cv::subtract( image, cv::Scalar(value), result )
result = image - value

cv::multiply( image, cv::Scalar(value), result )
result = image * value

cv::divide( image, cv::Scalar(value), result )
result = image / value
```

Image arithmetic operation



```
void imageAdd( Mat &image1, Mat &image2, Mat &result )
{
    int numOfLines  = image1.rows;  // number of lines in the image
    int numOfPixels = image1.cols;  // number of pixels per a line

    result.create( image1.rows, image1.cols, image1.type() );
    for( int r=0; r<numOfLines; r++ )
    {
        const uchar *data1_in  = image1.ptr<uchar>( r );
        const uchar *data2_in  = image2.ptr<uchar>( r );
        uchar *data_out        = result.ptr<uchar>( r );
        for( int c=0; c<numOfPixels; c++ )
        {
            data_out[c] = saturate_cast<uchar>( data1_in[c] + data2_in[c] );
        }
    }
}
```

subtract

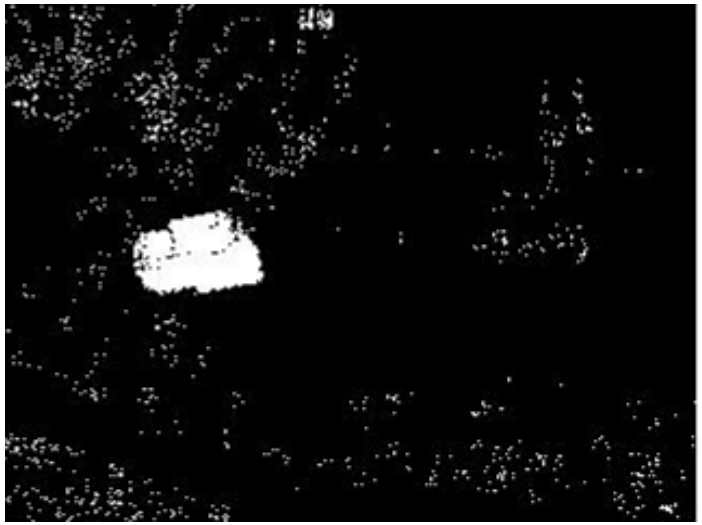
diff

```
void imageDiff( Mat &image1, Mat &image2, Mat &result )
{
    int numOfLines  = image1.rows;  // number of lines in the image
    int numOfPixels = image1.cols;  // number of pixels per a line

    result.create( image1.rows, image1.cols, image1.type() );
    for( int r=0; r<numOfLines; r++ )
    {
        const uchar *src1 = image1.ptr<uchar>( r );
        const uchar *src2 = image2.ptr<uchar>( r );
        uchar *dst        = result.ptr<uchar>( r );
        for( int c=0; c<numOfPixels; c++ )
        {
            dst[c] = saturate_cast<uchar>( abs(src1[c] - src2[c]) );
        }
    }
}
```


11:03:50AM


11:03:39AM



```
cv::addWeighted( src1, alpha, src2, beta, gamma, dst )
dst = src1*alpha + src2*beta + gamma
```

ex) result = image1*0.7 + image2*0.5 + 0.0

ex) result = image1*1.0 + image2*(−1.0) + 0.0

```
cv::scaleAdd( src1, alpha, src2, dst )
dst = src1*alpha + src2
```

ex) result = image1*0.7 + image2

Gray-level transformations

Improving image contrast and brightness
by using **mapping function**

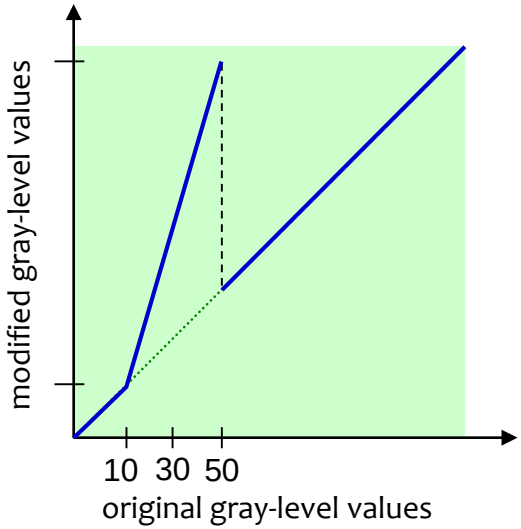
$$O(x, y) = M[I(x, y)]$$

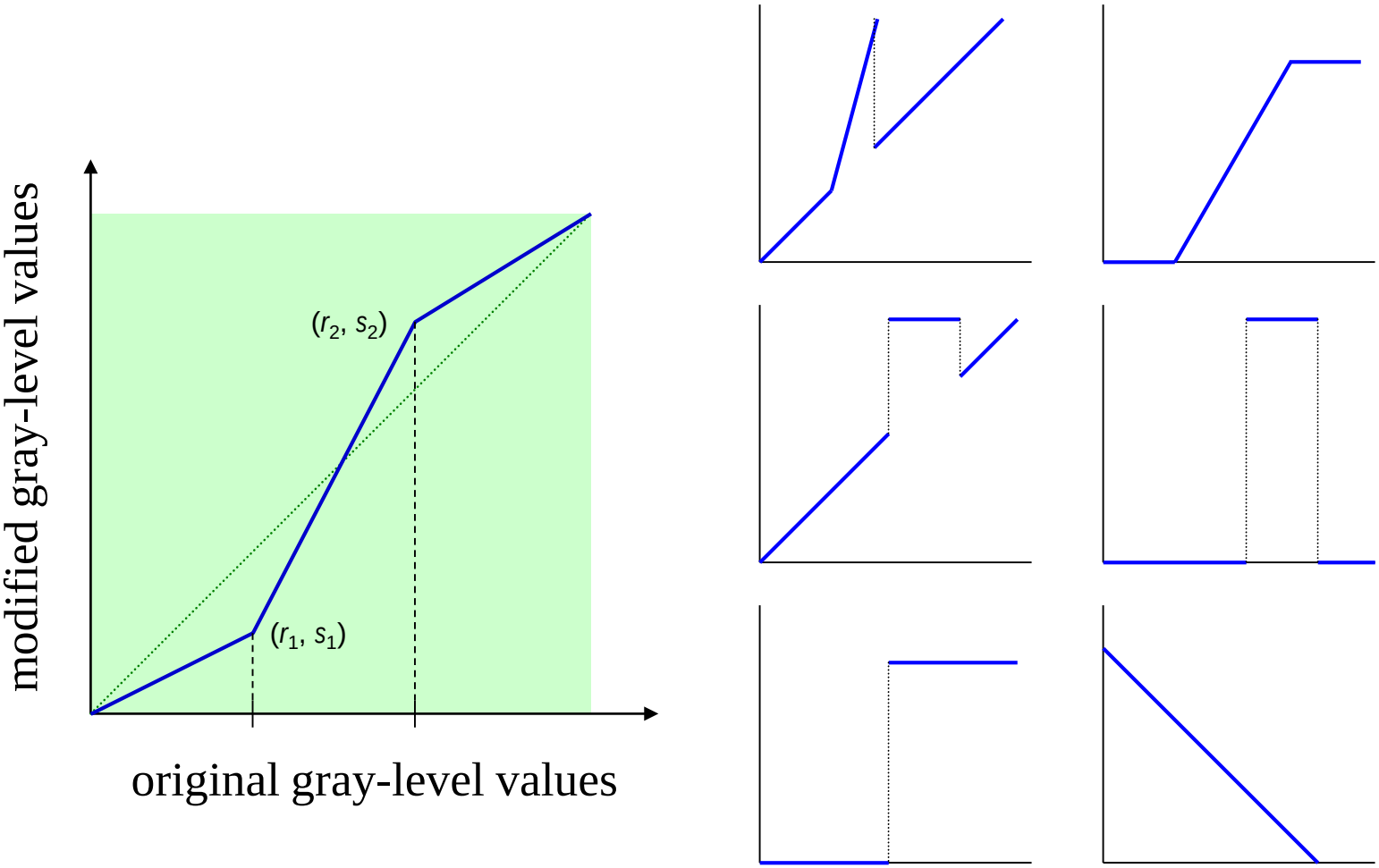
gray-level scaling or gray-scale modification

example

(10,50) 범위의 gray level을 (10,250) 범위로 확장

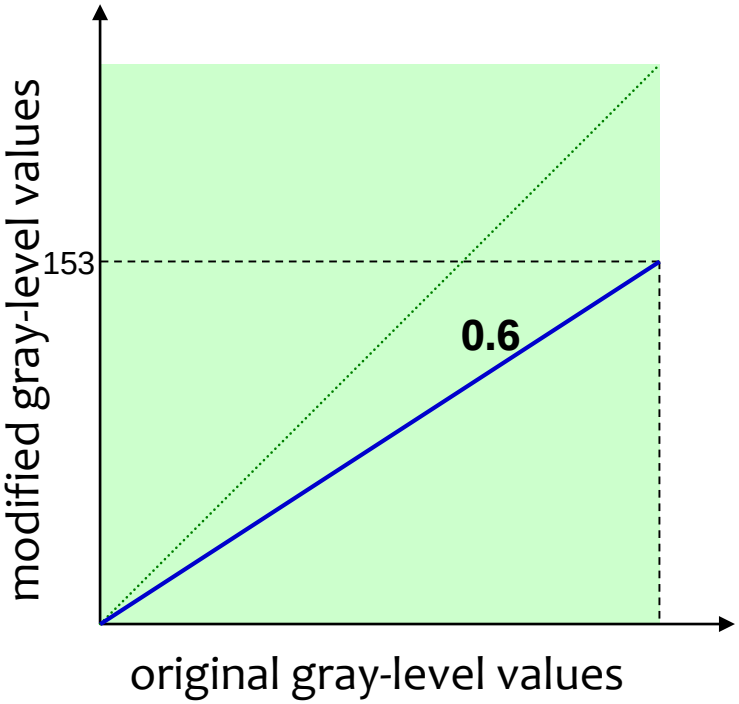
$$M[I(x,y)] = \begin{cases} I(x,y) & for\ 0 \le I(x,y) < 50 \\ 6[I(x,y)] - 50 & for\ 10 \le I(x,y) \le 50 \\ I(x,y) & for\ 50 < I(x,y) \le 255 \end{cases}$$



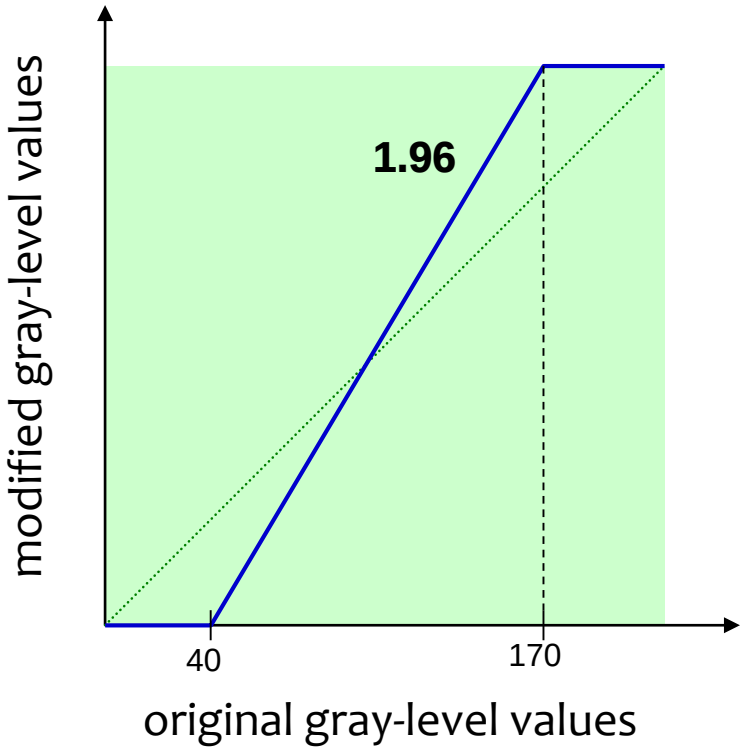


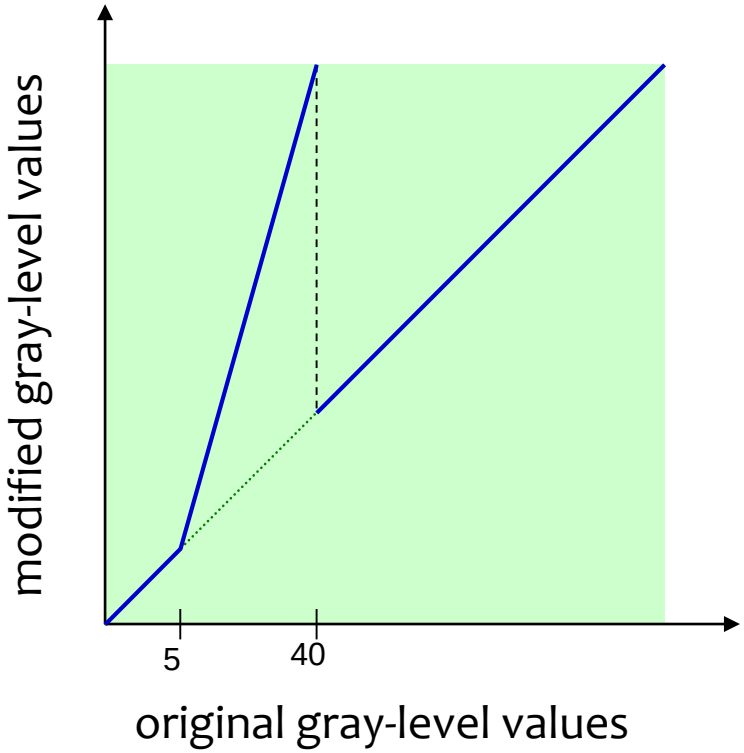
General Form of Gray-Scale Modification

Gray-scale Compression

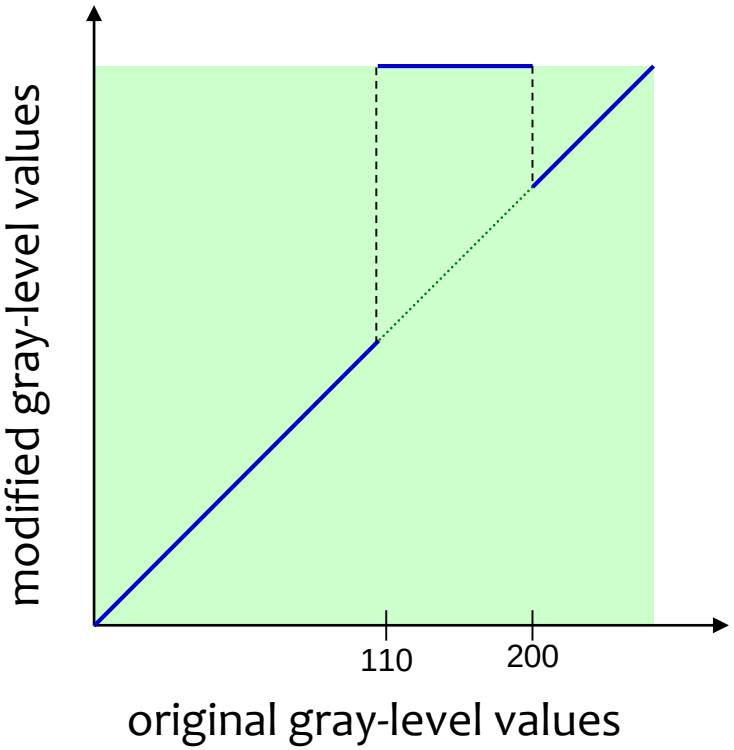


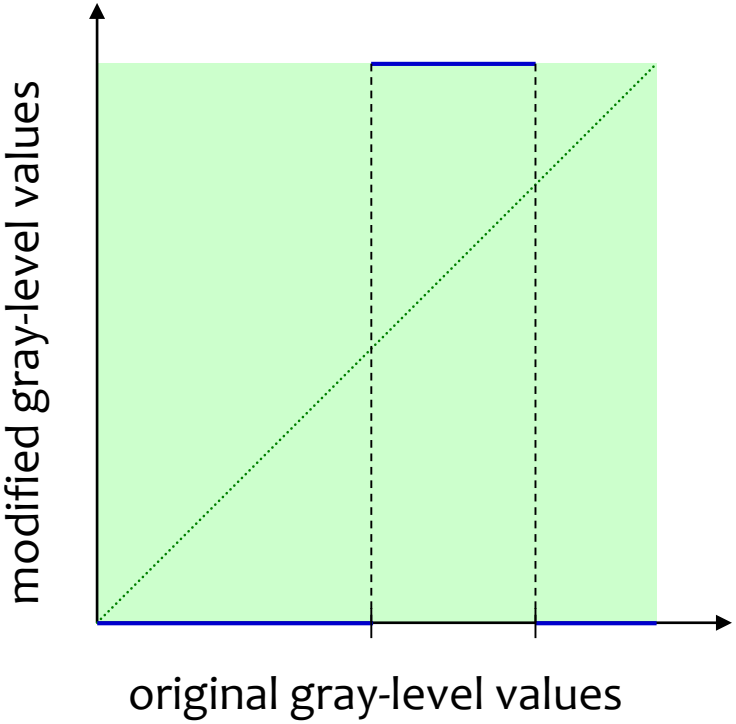
Gray-scale Stretching



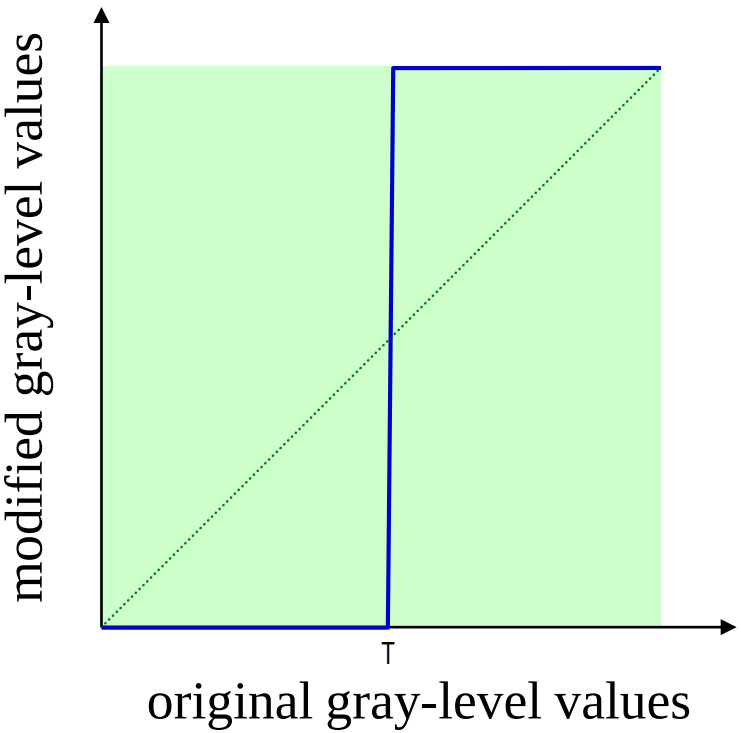

modified gray-level values
5
40
original gray-level values

Gray-level Slicing

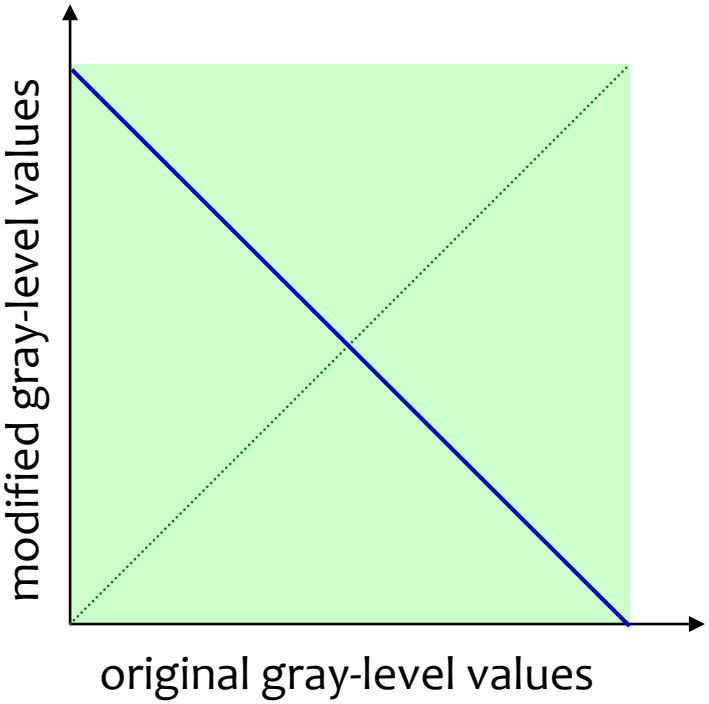



modified gray-level values
original gray-level values

Gray-level Thresholding



Gray-level Negative



요약

- point pixel processing
  - 이웃 픽셀과는 독립적으로 입력 영상의 각 픽셀 값을 변환한 후 결과 영상의 동일한 위치에 출력하는 연산
  - 목적: Improving image contrast and brightness
- Arithmetic operation
  - Scalar operation: 단일 영상에 상수를 더하거나 빼는 연산
  - Image operation: 영상 간에 더하거나 빼는 연산
- Gray-level transformations
  - Improving image contrast and brightness by using mapping function
  - Gray-scale Compression, Gray-scale Stretching, Gray-level Sliding, Gray-level Thresholding, Gray-level Negative 등

Reference

- Scott E Umbaugh, **Computer Imaging**, CRC Press, 2005
- R. Laganière, **OpenCV2 Computer Vision: Application Programming Cookbook,** PACKT Publishing, 2011
- G. Bradski and A. Kaebler, **Learning OpenCV: Computer Vision with the OpenCV Library**, O'REILLY, 2008
- 정성환, 이문호, **오픈소스 OpenCV를 이용한 컴퓨터 비전 실무 프로그래밍**, 홍릉과학출판사, 2007