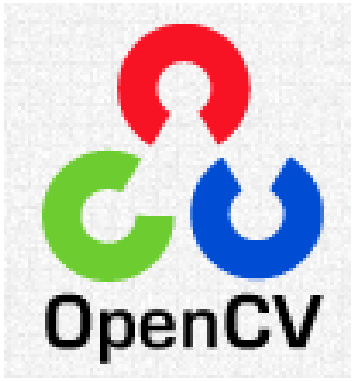


2020 겨울 OpenCV 워크샵

한동대학교 전산전자공학부
이강
email: yk@handong.edu

순서

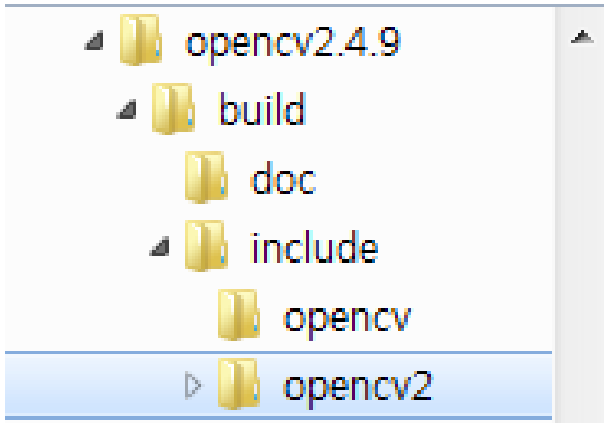
- Fundamentals On Images
- Grayscale Image Enhancement
  - 선형변환
  - 히스토그램 평활화
  - 필터링
- Color Image Enhancement
- **Threshholding and Binarization**
- Morphology 연산
- **Edge Detection and Line Detection**
- **Polygon Detection**
- Image Segmentation
  - connected component
  - K-means clustering
  - watershed
  - distance transform
- **OpenCV API 요약**
- **부록**
  - **Window/Mac/Linux 에서 OpenCV 설치**

OpenCV Concept

- openCV : open source computer vision library
  - Open source BSD-licensed library
  - openCV 2.x : C++ API
  - openCV 1.x : C API
- openCV structure
  - core : basic data structure for multi-dimentional array Mat and functions related
  - imgproc : image filter, geometrical transformation, color space conversion, histogram
  - cideo : motion estimation, background subtraction, object tracking
  - calib3d : multi-view geometry algorithms, single and stereo camera calibration, object pose estimation, 3D reconstructs
  - features2d : salient feature detectors, descriptors, and descriptor matchers
  - objdetect : detection of objects of predefined classes (faces, eyes, mugs, cars…)
  - highgui : interface to video capturing, image and video codec, simple UI
  - ml : machine learning, pattern matching
  - gpu : GPU-accelerated algorithms from different OpenCV modules
  - others : python, google wrappers

OpenCV 주요 헤더화일들

```
#include “opencv2/core/core.hpp”
#include “opencv2/imgproc/imgprog.hpp”
#include “opencv2/feature2d/feature2d.hpp”
#include “opencv2/objdetect/objdetect.hpp”
#include “opencv2/photo/photo.hpp”
#include “opencv2/video/video.hpp”
#include “opencv2/ml/ml.hpp”
#include “opencv2/calib3d/calib3d.hpp”
#include “opencv2/highgui/highu.hpp”
#include “opencv2/contrib/contrib.hpp”
```

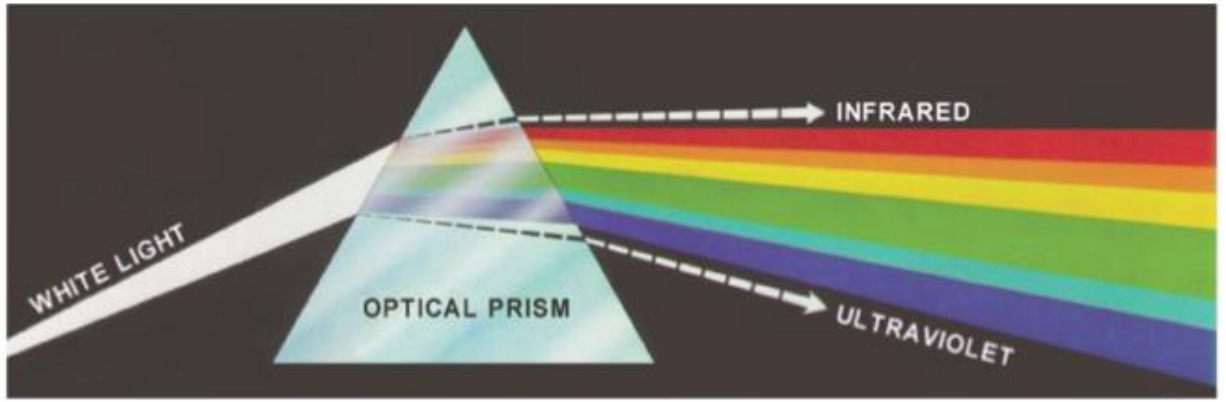



```
#include “opencv2/opencv.hpp”
```

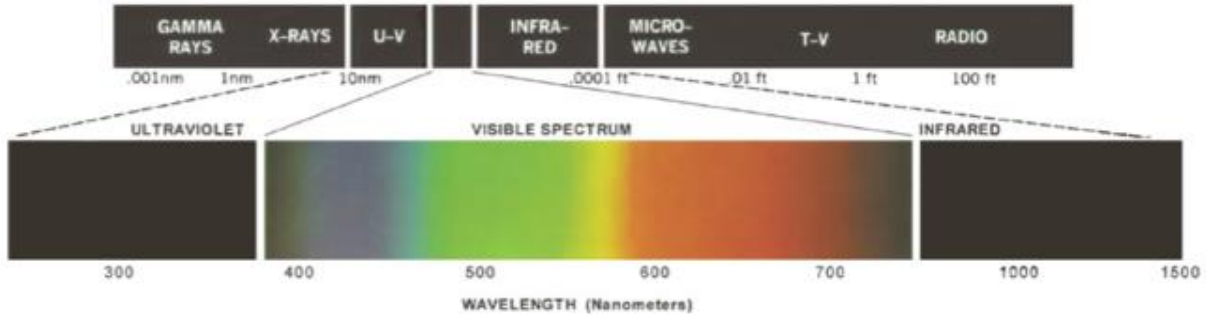
FUNDAMENTALS ON IMAGES

컬러의 기초

- 컬러 스펙트럼
  - 광선이 백색이 아닌 보라색부터 적색까지의 연속적인 스펙트럼



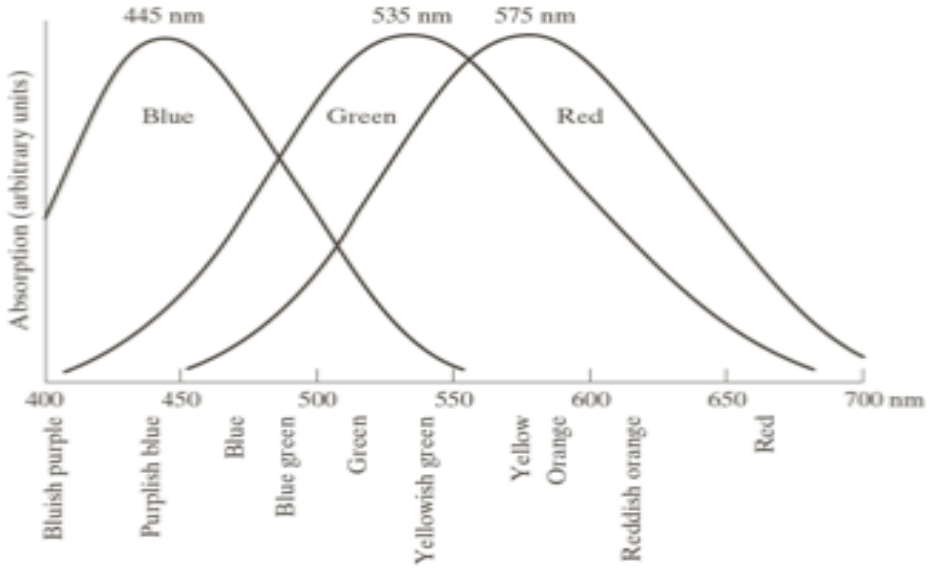
- 컬러의 인지
  - 해당 객체로부터 반사되는 빛의 성질에 의해 결정



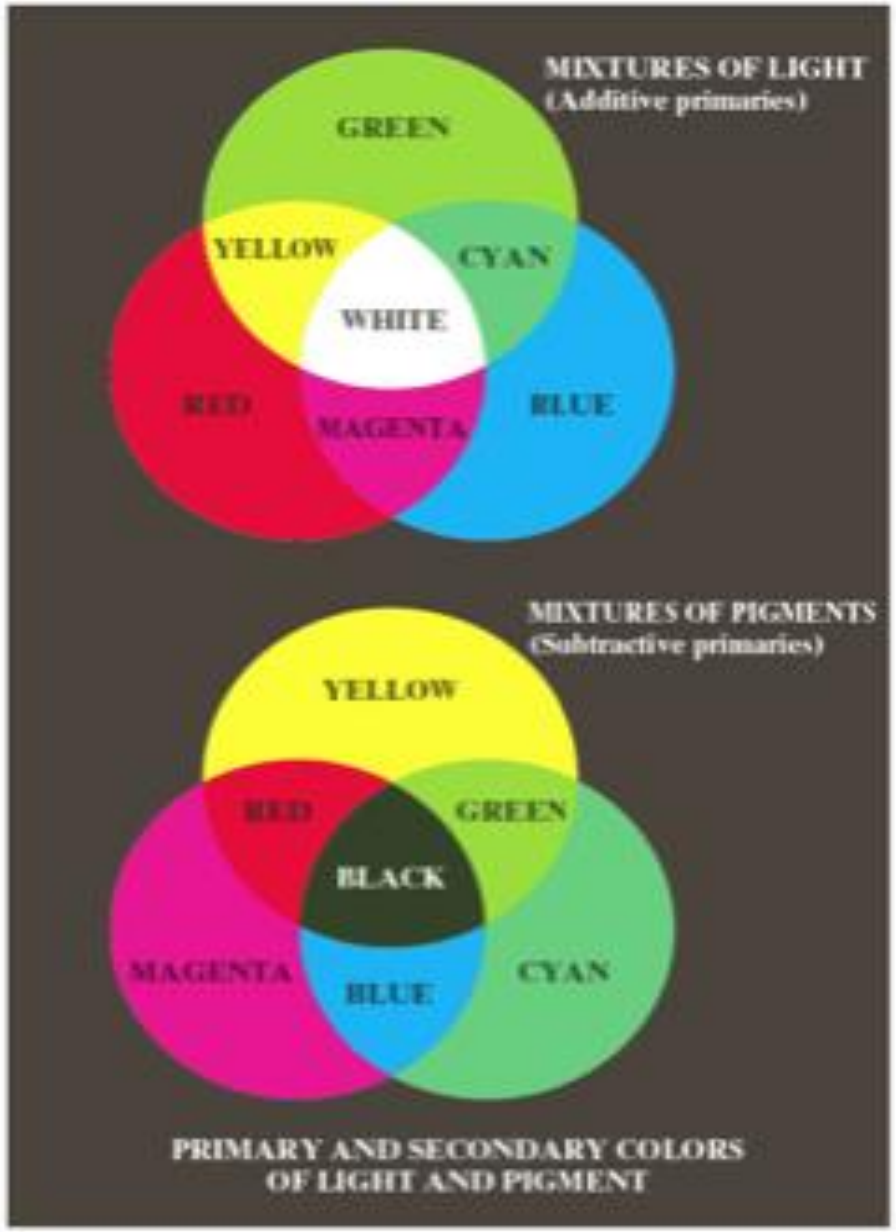
인간의 눈의 특성

- 추상체
  - 컬러 시각을 담당하는 눈의 센서,인간의 눈에는 약 6~7백만 개
  - 각각의 추상체가 적색, 녹색 혹은 청색을 지각
    - 약 65%: 적색 광에 민감
    - 약 33%: 녹색 광에 민감
    - 약 2% : 청색 광에 민감 (그러나 가장 예민)

- 적색, 녹색, 청색
  - 청색: 435.8nm, 녹색:546.1nm, 적색:700nm

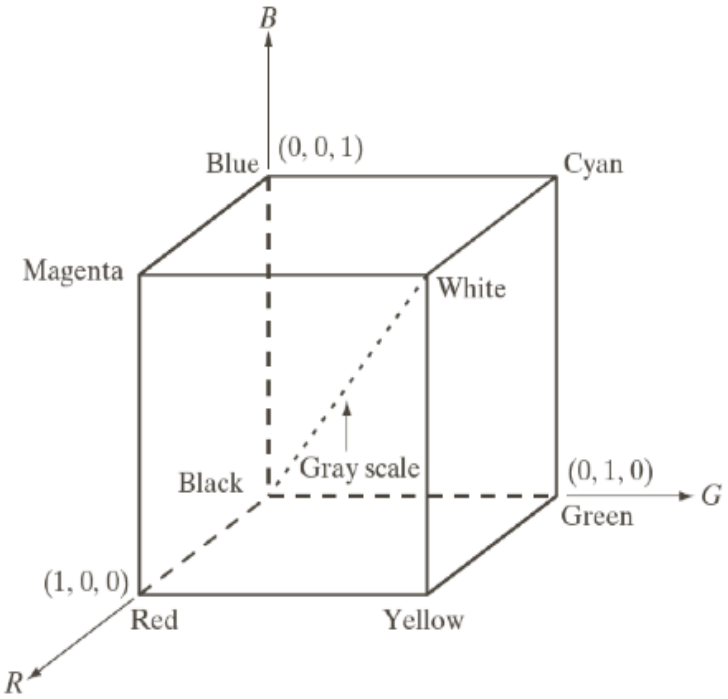


컬러의 합성


MIXTURES OF LIGHT
(Additive primaries)
GREEN
YELLOW
CYAN
WHITE
RED
MAGENTA
BLUE
MIXTURES OF PIGMENTS
(Subtractive primaries)
YELLOW
RED
GREEN
BLACK
MAGENTA
BLUE
CYAN
PRIMARY AND SECONDARY COLORS
OF LIGHT AND PIGMENT

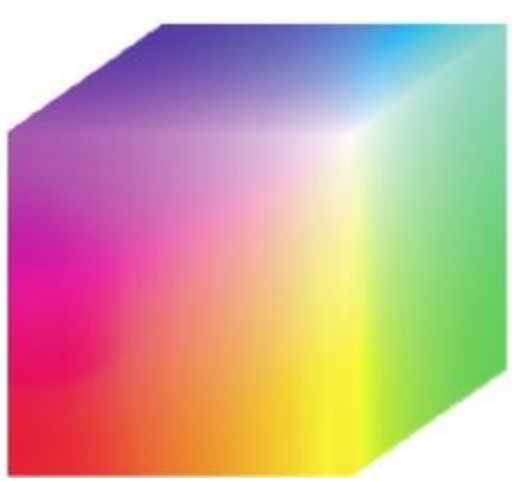
컬러 모델

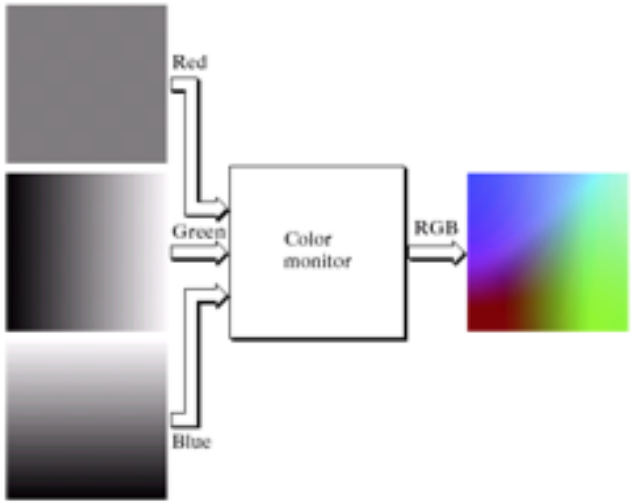
- RGB컬러 모델
  - 각 컬러는 적,녹,청의 원색 스펙트럼 성분들로 나타남
  - 데카르트 좌표 체계에 기반
  - 흑색은 원점, 백색은 원점에서 가장 먼 모퉁이
  - 그레이스케일은 흑색과 백색을 연결하는 선을 따라서 뻗쳐짐



컬러 모델

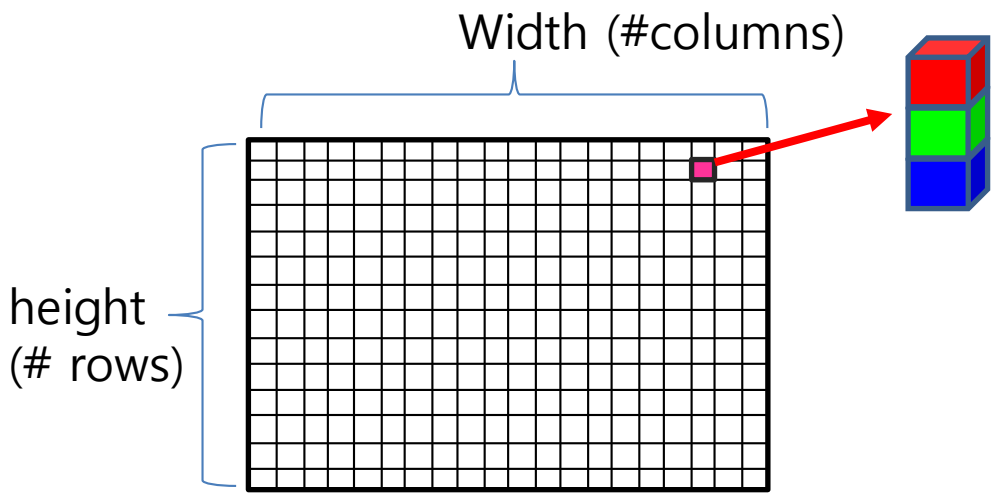
- 비트컬러
  - 각 화소를 표현하기 위해 사용되는 비트를 화소 깊이라고 부름
  - 각 RGB 영상이 8비트일 경우, 각 RGB컬러 화소는 24비트의 깊이
  - 24비트 RGB 영상에 있는 총 컬러 수는 $(2^8)^3$= 16,777,216
- 컬러 영상의 디스플레이 및 획득
  - 세 개의 개별 성분을 컬러 모니터에 입력하면 컬러 영상이 표시
  - 적, 녹, 청에 민감한 세 개의 필터로 각 컬러 정보를 획득

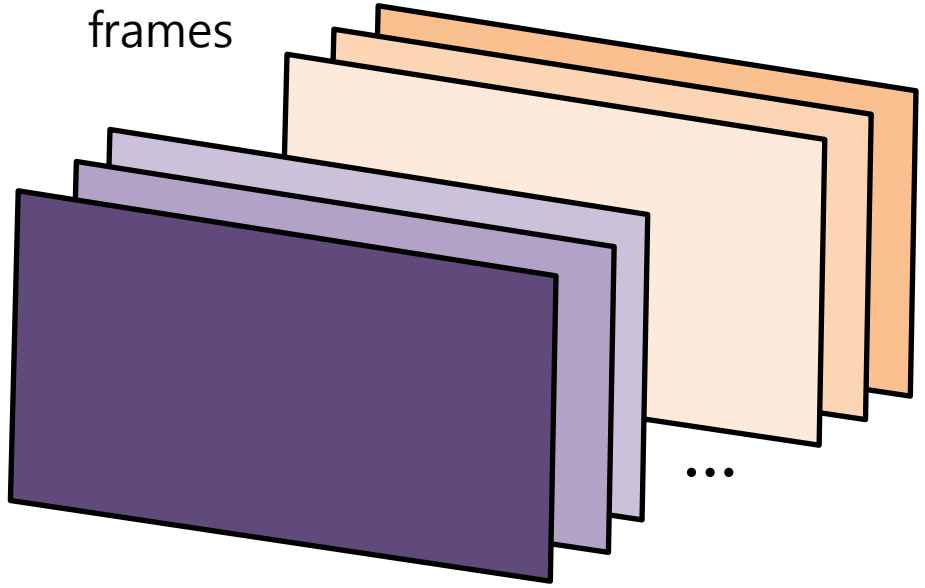




영상의 기본

- pixel = unit of picture element (디지털 영상을 구성하는 기본요소: 화소)
- 하나의 pixel은 여러 채널로 구성됨 (3 채널 = R,G,B)
- frame = one still image
- resolution of frame = width (#of pixels in horizoantal) x height (# pixels in vertical)
- 동영상의 경우, frame rate = number of frames per second





Mat 구조

Color Image Mat(n, m, CV_8U3C) 구조: 3 Channels

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

컬러가 (**B, G, R)**의 순서임에 유의

B+G=Cyan
G+R=Yellow
R+B=Magenta

Gray-scale image Mat(n,m) 구조: Single Channel

	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row ...	...,0	...,1	...	..., m
Row n	n,0	n,1	n,...	n, m

OpenCV 맛보기

```
#include  <opencv2/opencv.hpp>

using namespace std;
using namespace cv;

int main( ) {
    Mat img = imread("imag.png",  IMREAD_COLOR); // read an image file from disk
    imshow("output", img); // display the contents of a Mat onto the window
    waitKey(0);
    return 0;
}
```

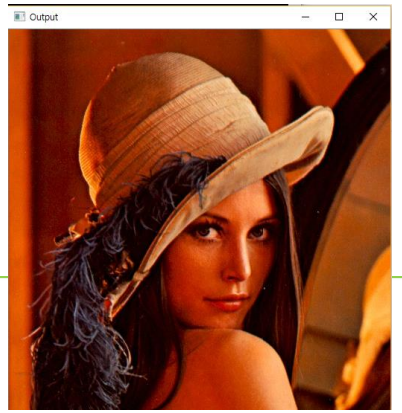
Image File name (bmp, jpg, png, tif)

color format to load into *Mat* object

- IMREAD_UNCHANGED
- IMREAD_GRAYSCALE
- IMREAD_COLOR
- IMREAD_ANYDEPTH
- IMREAD_ANYCOLOR
- IMREAD_LOAD_GDAL

Window name to display

Wait until any key is pressed.
Without the statement, image will just flash on the screen and disappear.



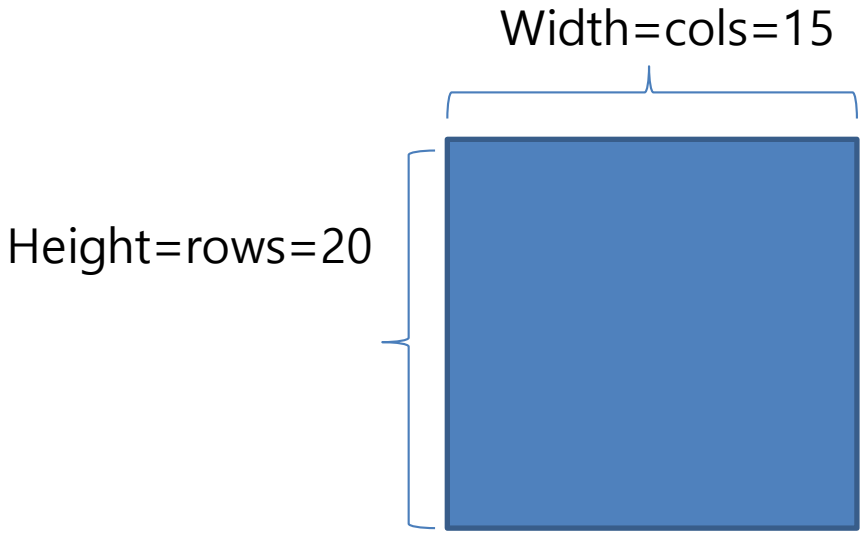
Mat : OpenCV에서 영상 저장을 위한 기본 자료구조

- Mat M;
- Mat M (***rows, colums***, *pixel_type*);
- Mat M (***rows, colums***, *pixel_type*, *initial_value*);
- Mat M (Size(***width, height***), *pixel_type*, *initial_value*);

예시

row, cols 순서에 주의

- Mat M1;
- Mat M2 (20,15, CV_8UC3);
- Mat M (20,15, CV_8UC3, Scalar(0,0,255));
- Mat M (Size(15,20), CV_U8C3, Scalar(0,0,255));



OpenCV 픽셀 Types

- Pixel Type Name

CV_[bit depth][**USF**][**C**(number of channel)]
예 CV_8UC3 ➔ unsigned 8-bit 3-channel

- CV_8U: uchar (8-bit unsigned) : 0 ~ 255
- CV_8S: schar(8-bit signed) : -128 ~ + 127
- CV_16U: ushort (16-bit unsigned) : 0 ~ +65,535
- CV_16S: short (16-bit signed int) : -32768 ~ +32767
- CV_32S: int (32-bit signed) : -2,147,483,648 ~ +2,147,483,647
- CV_32F: float (32-bits)
- CV_64F: double (64-bits)
- Multi-channel array: CV_8UC3, CV_8U(3), CV_64FC4, CV_64FC(4)

OpenCV의 다른 class들: Size, Scalar, Rect

- Size: 2개의 정수로 직사각형의 크기를 의미
  - Size constructor : Size(int width, int height)
  - typedef Size_<int> Size2i ;
  - typedef Size2i Size ;

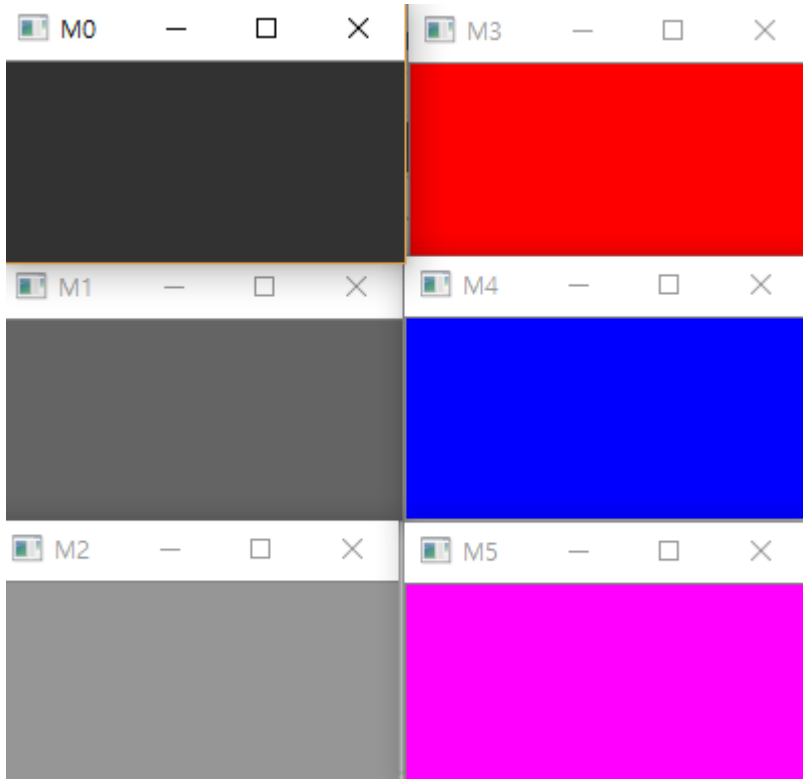
- Scalar: 1~4개의 double로 이루어진 숫자로, 하나의 픽셀값 의미
  - Scalar constructor: Scalar(int) or Scalar(int, int) or Scalar(int, int, int) or Scalar(int, int, int, int)
  - typedef Scalar_<double> Scalar;

- Rect : 직사각형의 위치를 (좌상꼭지점+크기) 또는 (좌상꼭지점+우하꼭지점)으로
  - Rect constructor: Rect(x, y, width, height) or Rect(Point(x,y), Point(x,y))

Mat 생성시 단색으로 초기화 시키기

```
#include <opencv2/opencv.hpp>

Using namespace cv;

int main()
{
  Mat M0(100, 200, CV_8UC1, Scalar(50));   // Gray scale = 50
  Mat M1(100, 200, CV_8UC1, Scalar(100));  // Gray scale = 100
  Mat M2(100, 200, CV_8UC1, Scalar(150));  // Gray scale = 150
  Mat M3(Size(200, 100), CV_8UC3, Scalar(0, 0, 255));    // color=red
  Mat M4(Size(200, 100), CV_8UC3, Scalar(255, 0, 0));    // color=blue
  Mat M5(Size(200, 100), CV_8UC3, Scalar(255 0, 255)); // color=red+blue

  imshow("M0", M0);
  imshow("M1", M1);
  imshow("M2", M2);
  imshow("M3", M3);
  imshow("M4", M4);
  imshow("M5", M5);

  waitKey(0);
  return 0;
}
```

- Color 값의 순서에 주의!
  - RGB (X)
  - BGR (O)

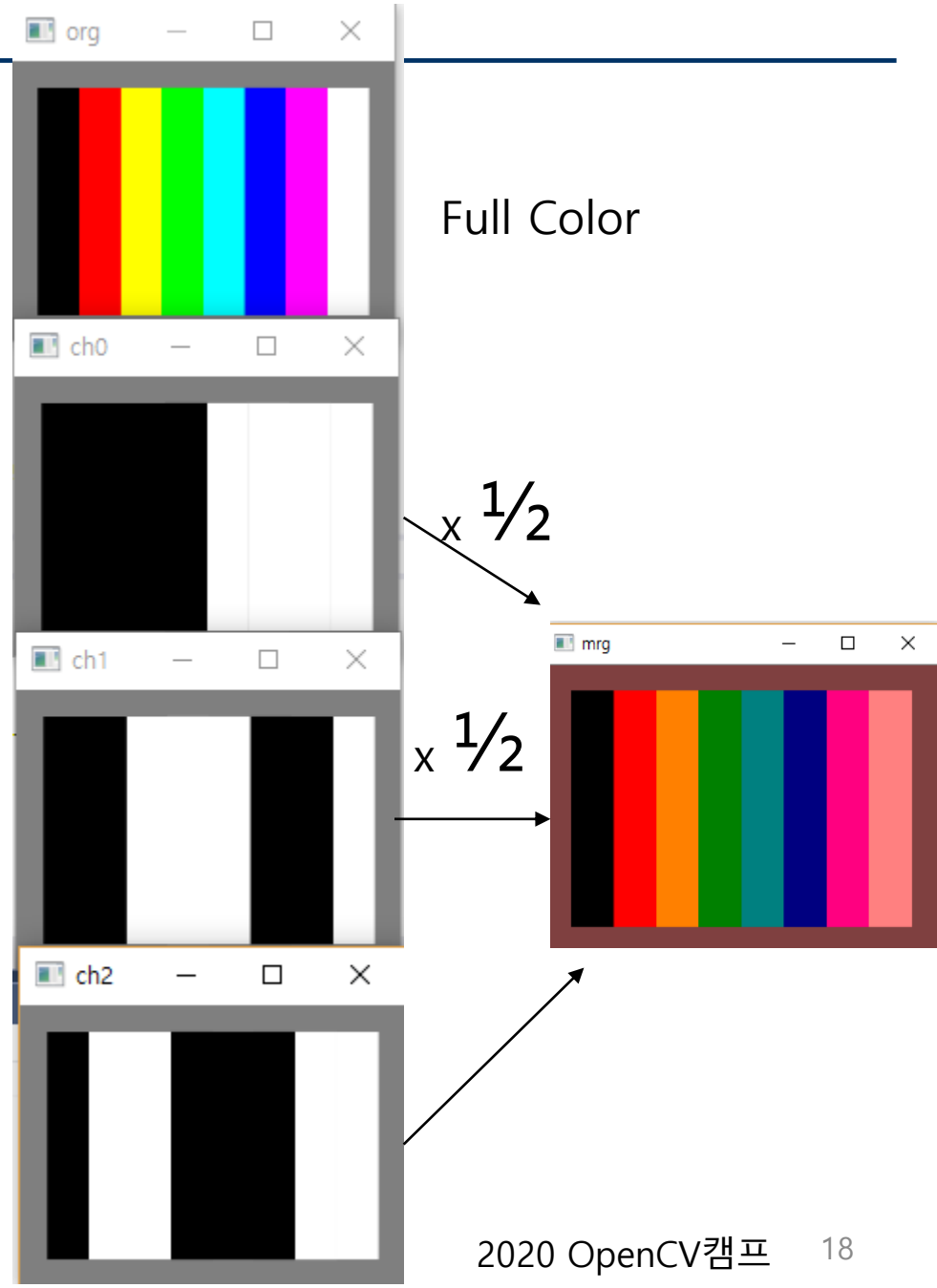


Mat의 채널 분리 및 병합, Mat에 대한 연산

```
void main()
{
    Mat img = imread(name, IMREAD_COLOR);

    resize(img, img, Size(200, 200.0 * img.rows / img.cols));
    imshow("org", img);

    vector<Mat> channel;
    split(img, channel);
    char wname[10];
    for (int i = 0; i < img.channels(); i++) {
        sprintf(wname, "ch%d", i);
        imshow(wname, channel[i]);
    }
    Mat merged;
    channel[0] = channel[0]/2;
    channel[1] = channel[1]/2;
    merge[channel, merged);
    imshow("mrg", merged);
    waitKey(0);
}
```

split (*Mat, vector<Mat>*);

merge(*vector<Mat>, Mat*);



컬러 모델

- HSI 컬러모델
  - 인간은 컬러 객체를 볼 때, 색상, 채도, 명도에 의해 컬러를 묘사
    - 명도: 밝기의 무색 개념
    - 색상: 관찰자에 의해 인지되는 지배적인 컬러
    - 채도: 상대적 순도 혹은 어떤 색상과 혼합된 백색광의 양
  - HSI 모델은 컬러 영상에서 밝기 성분을 컬러 수반정보로 부터 분리함
- RGB 컬러공간 <==> H,S,I컬러 공간

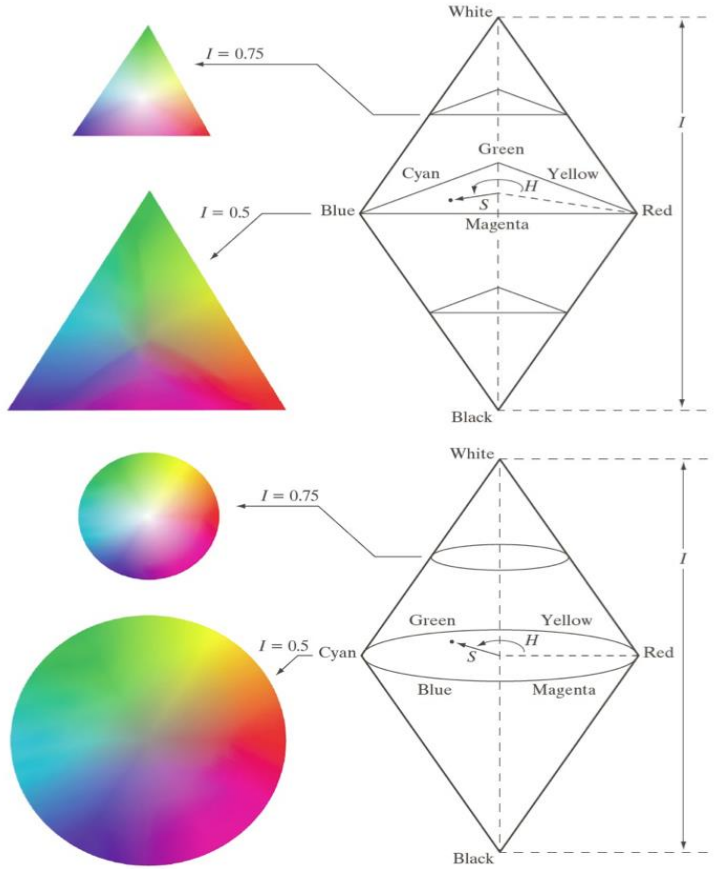
$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & otherwise \end{cases} \qquad \theta = cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$I = \frac{1}{3}(R+G+B)$$

컬러 모델

- 다양한 HSI 컬러모델
  - HSI 컬러 공간의 중요한 성분들은 수직 밝기 축, 컬러 점까지의 벡터의 길이, 이 벡터와 적색 축 간의 각도
  - 육각형, 삼각형, 혹은 원으로 정의 가능



컬러 모델

- YUV 컬러 모델
  - PAL(Phase Alternating Line), NTSC(National Television Standards Committee) 등의 컬러 비디오 표준에 사용
  - 1950년대 컬러 TV가 개발될 때, 그 전까지 사용하던 흑백 TV에서도 컬러 TV 방송을 흑백 TV에서도 수신하여 디스플레이 할 수 있도록 하기 위해서 만들어진 색상 공간
  - 구성
    - Y:밝기 성분
    - U:파란색에서 밝기 성분을 뺀 정보
    - V:빨간색에서 밝기 성분을 뺀 정보
  - 장점
    - 색상 성분은 밝기 성분에 비해 사람이 덜 민감하게 반응
    - U와 V 성분을 방송에서 송신할 때 적은 비트 할당
    - 전송할 신호의 양을 줄일 수 있음

컬러 모델간의 전환

- RGB로부터 YUV로의 컬러 전환
  - 기본 식

$$Y = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}$$
$$U = -0.147\mathrm{R} - 0.289\mathrm{G} + 0.436\mathrm{B}$$
$$V = 0.615\mathrm{R} - 0.515\mathrm{G} - 0.100\mathrm{B}$$

  - 기본 식을 사용하면 U의 범위는 -112~112,V의 범위는 -157~157
  - 변형된 식

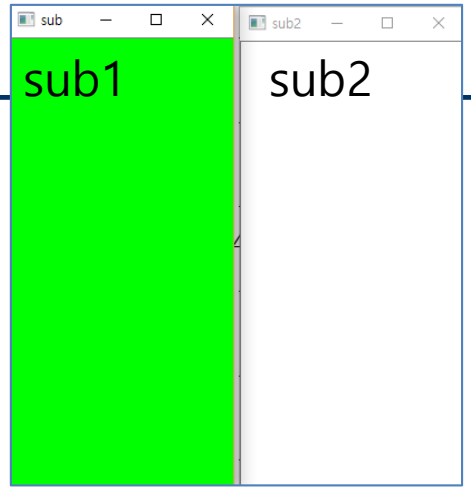
$$Y = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}$$
$$U = -0.169\mathrm{R} - 0.331\mathrm{G} + 0.500\mathrm{B} + 128$$
$$V = 0.500\mathrm{R} - 0.419\mathrm{G} - 0.081\mathrm{B} + 128$$

OpenCV에서 Color Space 변환하기

- cvtColor (input, output, flags)
  - flags =
    - COLOR_BGR2GRAY
    - COLOR_BGR2YUV
    - COLOR_BGR2HSV
    - COLOR_GRAY2BGR
    - COLOR_BGR2RGB

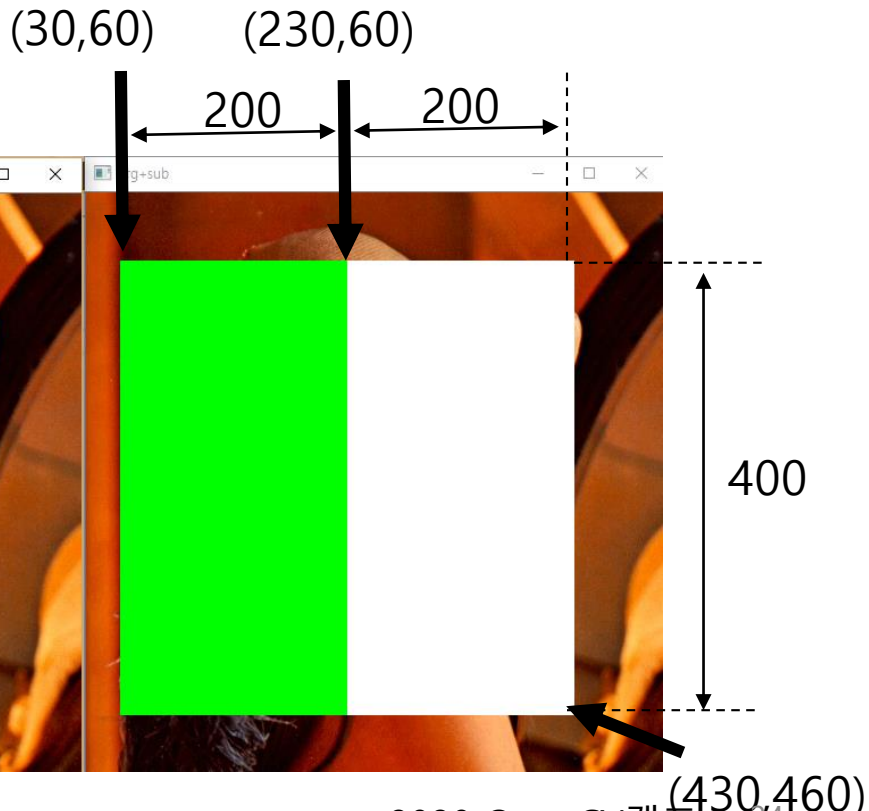
```
Mat img = imread ( “Lenna.jpg” , IMREAD_COLOR);
Mat dst, dst2, dst3;

cvtColor(img, dst, COLOR_BGR2GRAY);
cvtColor(img, dst2, COLOR_BGR2HSV);
cvtColor(img dst3,  COLOR_BGR2YUV);
imshow( “img” , img);
imshow( “GRAY", dst);
imshow( “HSV", dst2);
imshow( “YUV", dst3);
```

기존 Matrix의 일부로 sub Matrix 나타내기(Rect 사용)



- Mat M(*Mat*, *Rect*);
- Mat M (M0, Rect(30, 60, 200, 400));

```
Mat img = imread ( “Lenna.jpg” , IMREAD_COLOR);

Mat sub1(img, Rect(30, 60, 200, 400));
Mat sub2(img, Rect(Point(230, 60), Point(430,460)));
sub1 = Scalar(0, 255, 0); // sub1← Green
sub2 = Scalar(255, 255, 255); // sub2 ← White

imshow("sub", sub1);
imshow("sub2", sub2);
imshow("org+sub", img);
```

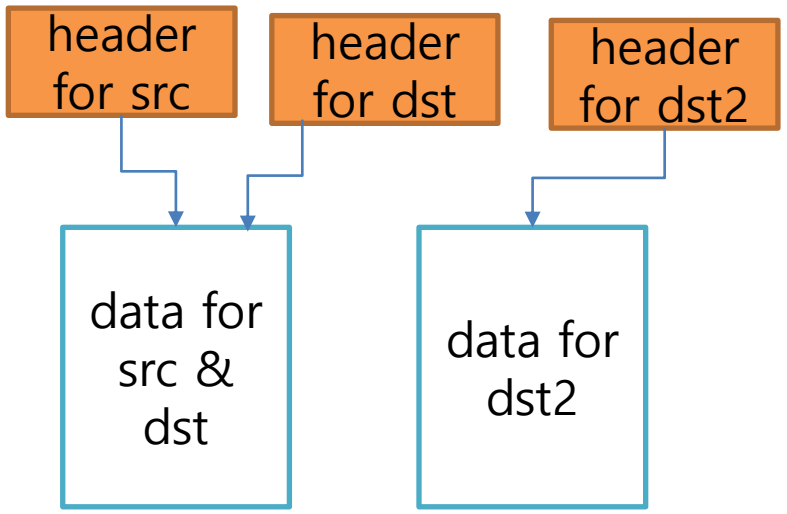


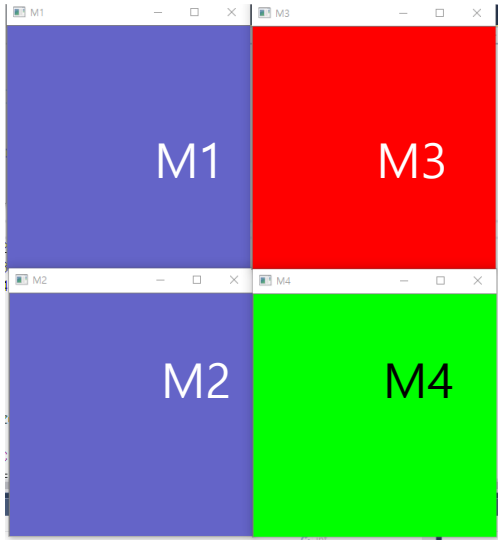
➔ Sub, sub2는 원 영상(img)와 데이터를 공유함
➔ img, sub, sub2 중 하나의 값을 바꾸면 모두변화

* Mat Data 메모리 **공유 vs 복사하기**

- Mat dst = src;
- Mat dst(src);
  모두 dst와 src의 Data 공유 (shallow copy)

- 원영상 데이터를 공유하지 않는 독립적으로 할당하는 방법 필요 (deep copy)
  - **기억장소 추가 할당** + 이미지 **데이터 복사**
  - 방법1: Mat member 함수 **clone()** : dst2 = src.clone()
  - 방법2: Mat member 함수 **copyTo**(Mat &*dst*) : src.copyTo(dst2);

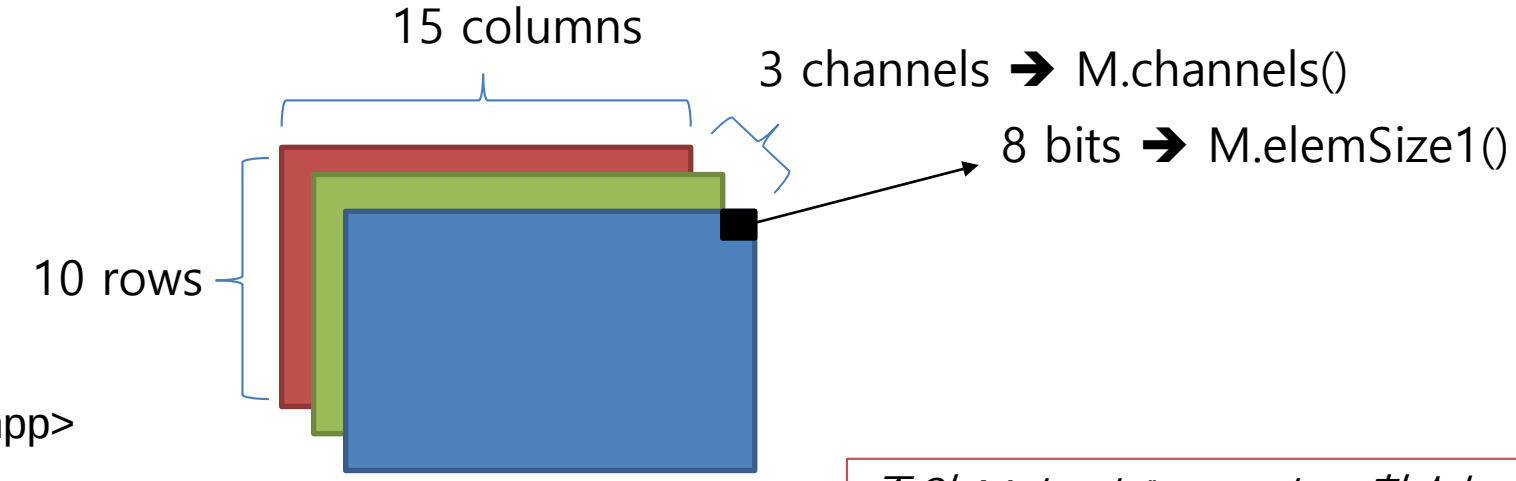





```
void basic_test3(char *name)
{
        Mat img = imread(name, IMREAD_COLOR);
        if (img.empty())
        return;

        resize(img, img, Size(300, 300.0 * img.rows/img.cols));
        imshow("org", img);

        Mat M1(img); // share the whole image area
        Mat M2, M3, M4;

        M2 = img;
        M3 = img.clone();
        img.copyTo(M4);

        M2 = Scalar(200, 100, 100);
        M3 = Scalar(0, 0, 255);
        M4 = Scalar(0, 255, 0);

        imshow("M1", M1);
        imshow("M2", M2);
        imshow("M3", M3);
        imshow("M4", M4);
        waitKey(0);
}
```

Mat Data size

- Size of Mat data matrix(bits) = #columns * #rows * #channels * channel depth
  - Mat M (10, 15, CV_8UC3)의 크기 = 10 rows * 20 cols * 3ch/pixel * 8 bits/ch = 4800 bits = 600 Bytes




```
#include <opencv2/opencv.hpp>
void main()
{
   cv::Mat M (10, 15, CV_8UC3);
   int imagesize = M.cols * M.rows * M.channels() * M.elemSize1();
   std::cout << “total size =“ << imagesize << “ bits” << std::endl;
}
```

주의 M.depth() member 함수는 bit수 외에도 unsigned와 signed, int, float를 구분

8U=0, 8S=1, 16S=2, 16U=3, 32U=4, 32F=5 로 값을 서로 구분함 (채널수에는 무관).

* M.elemSize1 () 함수: 한픽셀의 채널 당 비트수,
* M.elemSize () 함수: 한픽셀 당 (채널수 * 채널당 비트수) = M.elemSize1() * M.channels

Traversing Mat element (1)

```
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
void main(){
  Mat M = imread("Lenna.tif",0);
  int num_cols = M.cols;
  int num_rows = M.rows * M.channels();

  int dark_cnt = 0;
  if( M.isContinuous()) {
    uchar *p =  M.data;
    for(int i = 0; i < num_rows * num_cols; i++)
          dark_cnt += (*p++ < 50)?1:0;
   } else {
     for(int i = 0; i < num_rows; i++) {
        uchar *row_ptr = M.ptr<uchar>(i);
        for (int j = 0; j < num_cols; j++)
          dark_cnt += (row_ptr[j] < 50)? 1: 0;
     }
  }
  cnt << dark_cnt << "Dark pixels << endl;
}
```

==

```
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
void main(){
  Mat M = imread("Lenna.tif",0);
  int num_cols = M.cols;
  int num_rows = M.rows * M.channels();
  if (M.isContinuous( )) {
      num_cols = num_cols * num_rows;
      num_rows = 1;
  }

  int dark_cnt = 0;;
  for(int i = 0; i < num_rows; i++) {
     uchar *row_ptr = M.ptr<uchar>(i);
     for (int j = 0; j < num_cols; j++)
         dark_cnt += (row_ptr[j] < 50)? 1: 0;
  }
  cnt << dark_cnt << "Dark pixels << endl;
}
```

Traversing Mat elements (2)

- Mat 클래스의 변수들
  - M.data 멤버 변수 : 데이터가 저장된 전체 메모리의 시작 주소
- Mat 클래스의 함수들
  - M.ptr(row) 멤버 함수: 지정된 하나의 row를 가리키는 포인터를 돌려줌
  - M.at(i,j)멤버 함수 : 1xN 행렬일 경우는 at(i) 가능. Debug에서 느림
  - Iterator
    - MatIterator_<uchar> M.begin<uchar>()
    - MatIterator_<uchar> M.end<uchar>()

    - MatIterator_<Vec3b> M.begin<Vec3b>()
    - MatIerator_<Vec3b> M.bed<Vec3b>()

Traversing Mat element(3)

```
int dark_cntB = 0; dark_cntG=0; dark_cntR = 0;


for( int i = 0; i < M.rows; ++i)
      for( int j = 0; j < M.cols; ++j ) {
         zero_cntB += M.at<Vec3b>(i,j)[0] < 50)?1:0;
         zero_cntG += M.at<Vec3b>(i,j)[1] < 50)?1:0;
         zero_cntR += M.at<Vec3b>(i,j)[2] < 50)?1:0;
      }
}
cout << "Blue="<<zero_cntB << " Green= "
<< zero_cntG << " Red= "<< zero_cntR <<end;
```

==

```
int dark_cntB = 0; dark_cntG=0; dark_cntR = 0;
Mat_<Vec3b> Mv3 = M;

for( int i = 0; i < M.rows; ++i)
      for( int j = 0; j < M.cols; ++j ) {
         zero_cntB += Mv3.at(i,j)[0] < 50)?1:0;
         zero_cntG += Mv3.at(i,j)[1] < 50)?1:0;
         zero_cntR += Mv3.at(i,j)[2] < 50)?1:0;
      }
}
cout << "Blue="<<zero_cntB << " Green= "
<< zero_cntG << " Red= "<< zero_cntR <<end;
```

Traversing Mat element(4): at, channels

```
imshow("orginal", M);
if(M.channels() == 1) {
  for( int i = 0; i < M.rows; ++i)
      for( int j = 0; j < M.cols; ++j )
         if(M.at(i,j) > 100)? M.at<i,j> =255;
}
else if(M.channels() == 3) {
  int B = 0; G=0; R = 0, tot =0;

  for( int i = 0; i < M.rows; ++i)
      for( int j = 0; j < M.cols; ++j ) {
         if(M.at<Vec3b>(i,j)[0] >100) { M.at<vec3b>[0] = 255;  B++;}
         if(M.at<Vec3b>(i,j)[1] >100) { M.at<vec3b>[1] = 255; G++;}
         if(M.at<Vec3b>(i,j)[2] >100) { M.at<vec3b>[2] = 255;  R++;}
         tot++;
      }
   }
   cout << "B=" << (float)B/tot << " G=" << (float)G/tot
        << " R= " << (float)R/tot <<endl;
}
imshow("changed",M);
waitKey(0);
```

Traversing Mat elements (5) : Iterator

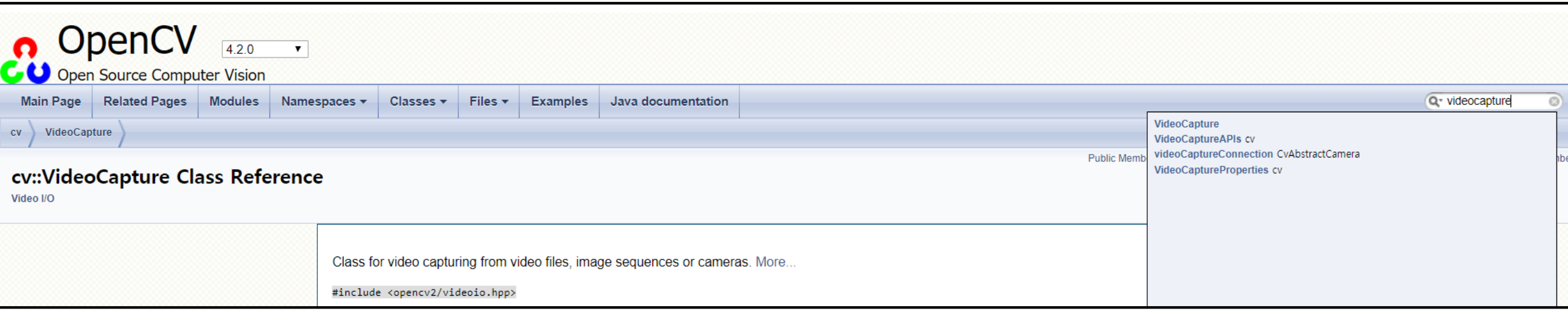
```
const int channels = I.channels();
switch(channels) {
  case 1: {
    MatIterator_<uchar> it, end;
    for( it = I.begin<uchar>(), end = I.end<uchar>(); it != end; ++it)
        if (*it > 100) *it = 255;
     break;
  }
  case 3: {
    MatIterator_<Vec3b> it, end;
    for( it = I.begin<Vec3b>(), end = I.end<Vec3b>(); it != end; ++it) {
        if((*it)[0] >100) (*it)[0]   = 255;
        if((*it)[1] >100) (*it)[1] = 255;
        if((*it)[2] >100) (*it)[2] = 255;
    }
    break;
   }
}
```

영상 READ 예제

Opencv reference 사이트

- docs.opencv.org 들어가서 opencv version 선택 (4.2.0)

opencv 함수명 / 객체명 등 검색



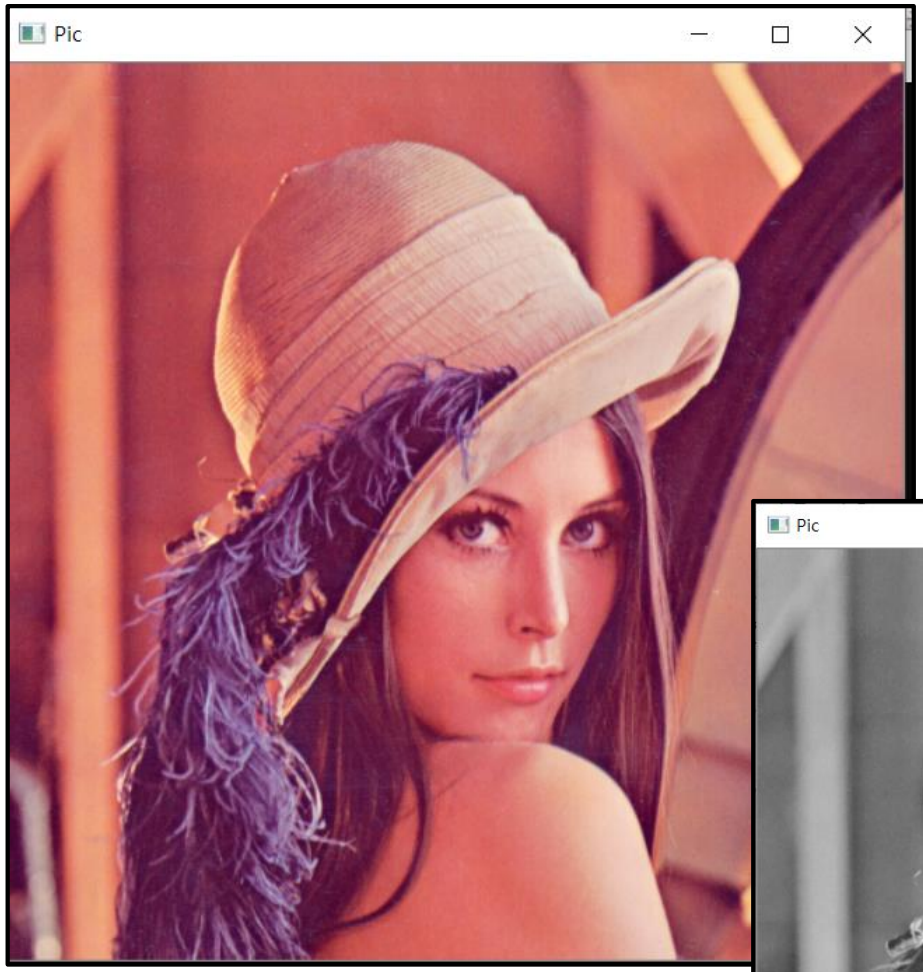
Mat class

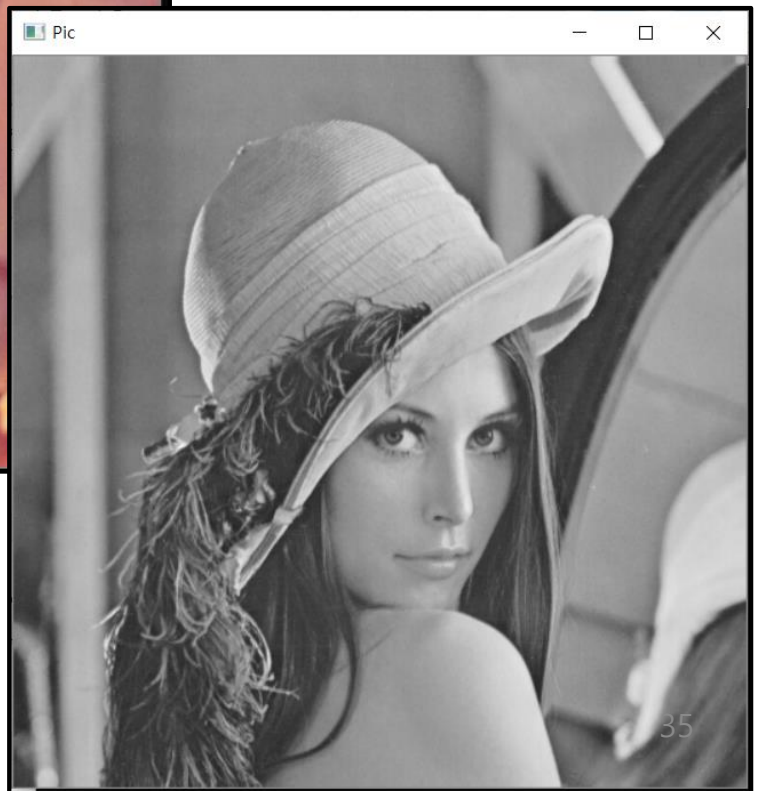
Public Member Functions

Mat ()
Mat (int **rows**, int **cols**, int **type**)
Mat (**Size size**, int **type**)
Mat (int **rows**, int **cols**, int **type**, const **Scalar** &s)
Mat (**Size size**, int **type**, const **Scalar** &s)
Mat (int ndims, const int *sizes, int **type**)
Mat (const std::vector< int > &sizes, int **type**)
Mat (int ndims, const int *sizes, int **type**, const **Scalar** &s)
Mat (const std::vector< int > &sizes, int **type**, const **Scalar** &s)
Mat (const **Mat** &m)
Mat (int **rows**, int **cols**, int **type**, void ***data**, size_t **step**=**AUTO_STEP**)
Mat (**Size size**, int **type**, void ***data**, size_t **step**=**AUTO_STEP**)
Mat (int ndims, const int *sizes, int **type**, void ***data**, const size_t *steps=0)
Mat (const std::vector< int > &sizes, int **type**, void ***data**, const size_t *steps=0)
Mat (const **Mat** &m, const **Range** &**rowRange**, const **Range** &**colRange**=**Range::all**())
Mat (const **Mat** &m, const **Rect** &roi)
Mat (const **Mat** &m, const **Range** *ranges)
Mat (const **Mat** &m, const std::vector< **Range** > &ranges)

I) 정지 영상 read 예제

```
1  #include <iostream>
2  #include <opencv2/core/core.hpp>
3  #include <opencv2/highgui/highgui.hpp>
4
5  using namespace cv;
6  using namespace std;
7
8  int main(int argc, const char* argv[]) {
9      Mat img = imread("lena.png", 0);
10     if (img.empty()) {
11         cerr << "read fail!" << endl;
12         exit(-1);
13     }
14     imshow("image", img);
15     waitKey(0);
16
17     return 0;
18 }
```





1) 정지 영상 read 예제

◆ imread()

```
Mat cv::imread ( const String &  filename,
                 int             flags = IMREAD_COLOR
               )
```

Python:

```
retval = cv.imread( filename[, flags] )
```

```
enum  cv::ImreadModes {
        cv::IMREAD_UNCHANGED = -1,
        cv::IMREAD_GRAYSCALE = 0,
        cv::IMREAD_COLOR = 1,
        cv::IMREAD_ANYDEPTH = 2,
        cv::IMREAD_ANYCOLOR = 4,
        cv::IMREAD_LOAD_GDAL = 8,
        cv::IMREAD_REDUCED_GRAYSCALE_2 = 16,
        cv::IMREAD_REDUCED_COLOR_2 = 17,
        cv::IMREAD_REDUCED_GRAYSCALE_4 = 32,
        cv::IMREAD_REDUCED_COLOR_4 = 33,
        cv::IMREAD_REDUCED_GRAYSCALE_8 = 64,
        cv::IMREAD_REDUCED_COLOR_8 = 65,
        cv::IMREAD_IGNORE_ORIENTATION = 128
      }
      Imread flags. More...
```

1) 정지 영상 read 예제

◆ imshow()

void cv::imshow (const **String** & winname,
InputArray mat
)

Python:

None = cv.imshow(winname, mat)

```
#include <opencv2/highgui.hpp>
```

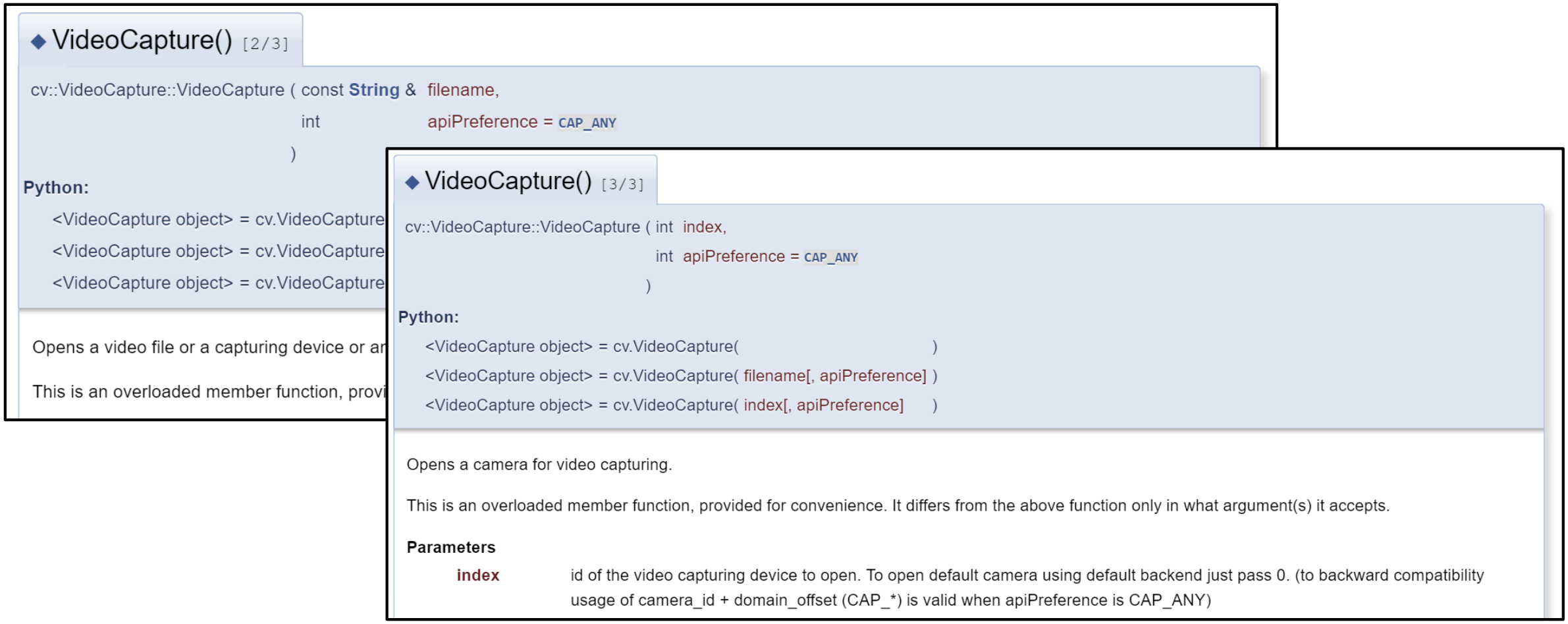
Displays an image in the specified window.

The function imshow displays an image in the specified window. If the window was created with the **cv::WINDOW_AUTOSIZE** flag, the image is shown with its original size, however it is still limited by the screen resolution. Otherwise, the image is scaled to fit the window. The function may scale the image, depending on its depth:

2) 동영상 파일 read 예제


```cpp
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace cv;
using namespace std;

int main(int argc, const char* argv[]) {
    Mat frame;
    VideoCapture cap("video.mp4");

    if (!cap.isOpened()) {
        cerr << "file open fail!" << endl;
        return -1;
    }
    while (1) {
        cap >> frame;
        if (frame.empty()) {
            cerr << "empty frame" << endl;
            break;
        }
        imshow("vid", frame);
        waitKey(33);
    }
    return 0;
}
```

2) 동영상 파일읽기 VideoCapture



◆ VideoCapture() [2/3]

cv::VideoCapture::VideoCapture (const String & filename,
int apiPreference = CAP_ANY
)

Python:

<VideoCapture object> = cv.VideoCapture
<VideoCapture object> = cv.VideoCapture
<VideoCapture object> = cv.VideoCapture

Opens a video file or a capturing device or ar

This is an overloaded member function, provi

◆ VideoCapture() [3/3]

cv::VideoCapture::VideoCapture (int index,
int apiPreference = CAP_ANY
)

Python:

<VideoCapture object> = cv.VideoCapture( )
<VideoCapture object> = cv.VideoCapture(filename[, apiPreference])
<VideoCapture object> = cv.VideoCapture(index[, apiPreference])

Opens a camera for video capturing.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

index	id of the video capturing device to open. To open default camera using default backend just pass 0. (to backward compatibility usage of camera_id + domain_offset (CAP_*) is valid when apiPreference is CAP_ANY)

2) 동영상 파일읽기 VideoCapture

virtual	**~VideoCapture** () Default destructor. More...
virtual double	**get** (int propId) const Returns the specified **VideoCapture** property. More...
String	**getBackendName** () const Returns used backend API name. More...
bool	**getExceptionMode** () query if exception mode is active More...
virtual bool	**grab** () Grabs the next frame from video file or capturing device. More...
virtual bool	**isOpened** () const Returns true if video capturing has been initialized already. More...
virtual bool	**open** (const **String** &filename, int apiPreference=**CAP_ANY**) Opens a video file or a capturing device or an IP video stream for video capturing. More...
virtual bool	**open** (int index, int apiPreference=**CAP_ANY**) Opens a camera for video capturing. More...
virtual **VideoCapture** &	**operator>>** (**Mat** &image) Stream operator to read the next video frame. More...
virtual **VideoCapture** &	**operator>>** (**UMat** &image)
virtual bool	**read** (**OutputArray** image) Grabs, decodes and returns the next video frame. More...
virtual void	**release** () Closes video file or capturing device. More...
virtual bool	**retrieve** (**OutputArray** image, int flag=0) Decodes and returns the grabbed video frame. More...
virtual bool	**set** (int propId, double value) Sets a property in the **VideoCapture**. More...
void	**setExceptionMode** (bool enable)

2) 동영상 파일 읽기

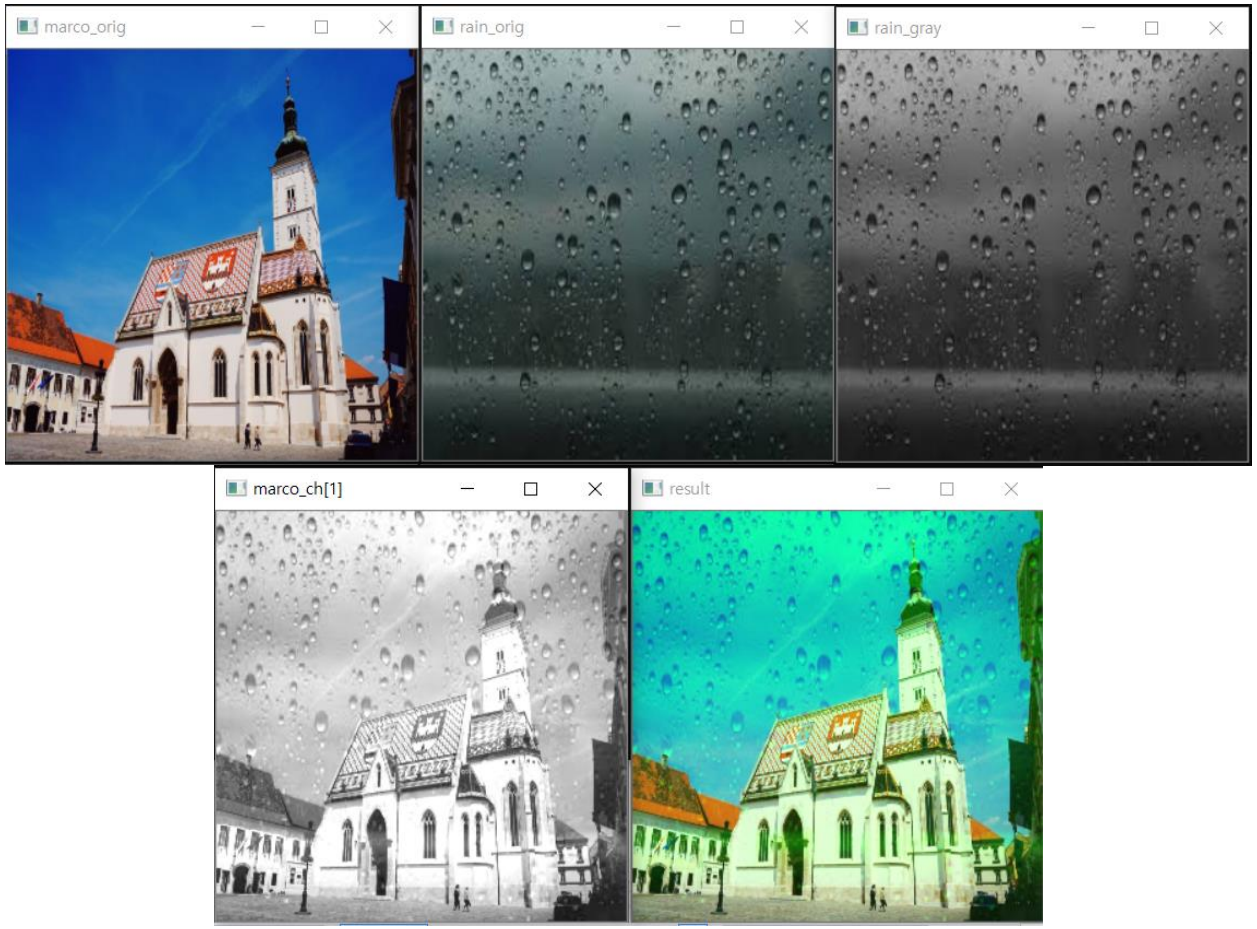
- 30fps : 초당 30frame

→1 frame 당 1/30초 = 0.0333초 = 33.3ms → waitKey(33);

SPLIT & MERGE 예제

Split & Merge 예제

#include <opencv2/imgproc.hpp>




```
9   int main(int argc, const char* argv[]) {
10      Mat img1 = imread("marco.jpg");
11      Mat img2 = imread("rain.jpeg");
12      Mat result;
13      Mat img2_gray;
14      vector<Mat> channels;
15
16      resize(img1, img1, Size(300, 300));
17      resize(img2, img2, Size(300, 300));
18      cvtColor(img2, img2_gray, COLOR_BGR2GRAY);
19      imshow("marco_orig", img1);
20      imshow("rain_orig", img2);
21      imshow("rain_gray", img2_gray);
22
23      split(img1, channels);
24      channels[1] += img2_gray;
25      imshow("marco_ch[1]", channels[1]);
26      merge(channels, result);
27      imshow("result", result);
28
29      waitKey(0);
30      return 0;
31  }
```

Split & Merge 예제

◆ cvtColor()

```
void cv::cvtColor ( InputArray  src,
                    OutputArray dst,
                    int         code,
                    int         dstCn = 0
                  )
```

Python:

```
dst = cv.cvtColor( src, code[, dst[, dstCn]] )
```

```
#include <opencv2/imgproc.hpp>
```

Converts an image from one color space to another.

The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

Split & Merge 예제

Enumerations

enum cv::ColorConversionCodes {

cv::COLOR_BGR2BGRA = 0,
cv::COLOR_RGB2RGBA = COLOR_BGR2BGRA,
cv::COLOR_BGRA2BGR = 1,
cv::COLOR_RGBA2RGB = COLOR_BGRA2BGR,
cv::COLOR_BGR2RGBA = 2,
cv::COLOR_RGB2BGRA = COLOR_BGR2RGBA,
cv::COLOR_RGBA2BGR = 3,
cv::COLOR_BGRA2RGB = COLOR_RGBA2BGR,
cv::COLOR_BGR2RGB = 4,
cv::COLOR_RGB2BGR = COLOR_BGR2RGB,
cv::COLOR_BGRA2RGBA = 5,
cv::COLOR_RGBA2BGRA = COLOR_BGRA2RGBA,
cv::COLOR_BGR2GRAY = 6,
cv::COLOR_RGB2GRAY = 7,
cv::COLOR_GRAY2BGR = 8,
cv::COLOR_GRAY2RGB = COLOR_GRAY2BGR,
cv::COLOR_GRAY2BGRA = 9,
cv::COLOR_GRAY2RGBA = COLOR_GRAY2BGRA,
cv::COLOR_BGRA2GRAY = 10,
cv::COLOR_RGB2YCrCb = 37,
cv::COLOR_YCrCb2BGR = 38,
cv::COLOR_YCrCb2RGB = 39,
cv::COLOR_BGR2HSV = 40,
cv::COLOR_RGB2HSV = 41,
cv::COLOR_BGR2Lab = 44,
cv::COLOR_RGB2Lab = 45,
cv::COLOR_BGR2Luv = 50,
cv::COLOR_RGB2Luv = 51,
cv::COLOR_BGR2HLS = 52,
cv::COLOR_RGB2HLS = 53,
cv::COLOR_HSV2BGR = 54,
cv::COLOR_HSV2RGB = 55,
cv::COLOR_Lab2BGR = 56,
cv::COLOR_Lab2RGB = 57,
cv::COLOR_Luv2BGR = 58,
cv::COLOR_Luv2RGB = 59,
cv::COLOR_HLS2BGR = 60,
cv::COLOR_HLS2RGB = 61,
cv::COLOR_BGR2HSV_FULL = 66,
cv::COLOR_RGB2HSV_FULL = 67,

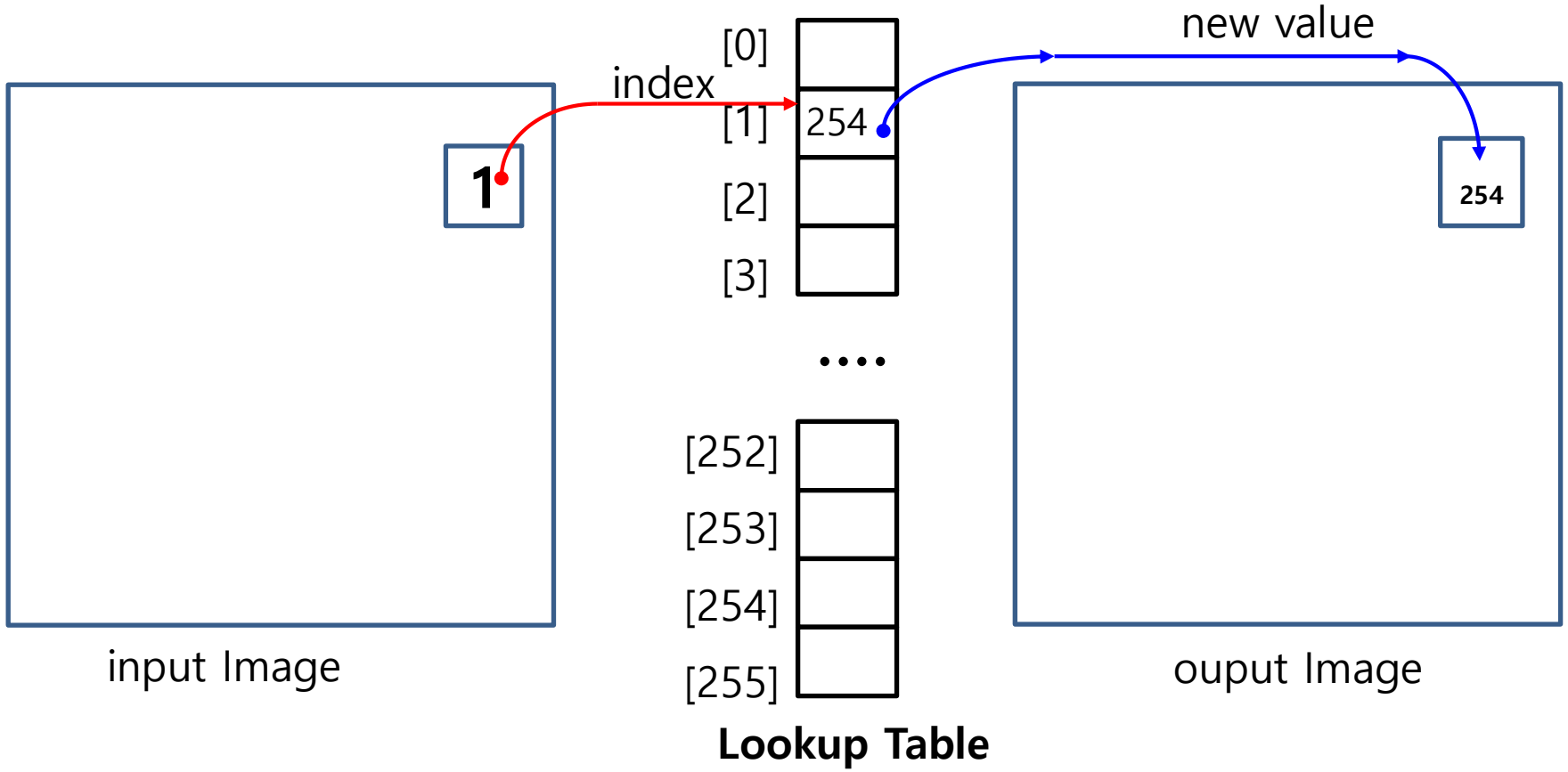
GRAYSCALE IMAGE ENHANCEMENT

선형 변환 : Negative Transform

- 영상 네거티브 (0.. L-1 까지 밝기 단계가 있을때)
  $s = (L - 1) - r$
  - 흑색 영역의 면적 비중이 클 때, 희색이나 그레이디테일을 개선시키는 데 적합

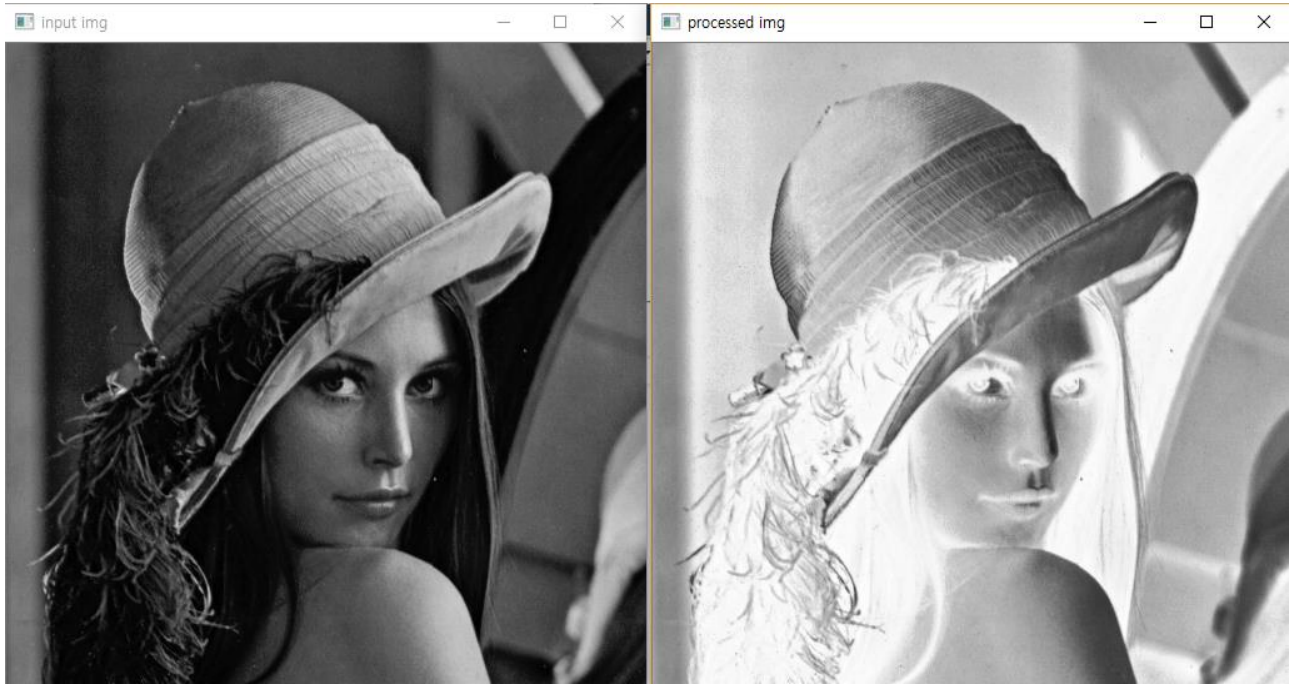
Lookup Table

- precompute the table for input to output mapping
  - Note number of vaues of input is only 256 !
  - Compute once, apply many



negative 변환 예제 코드

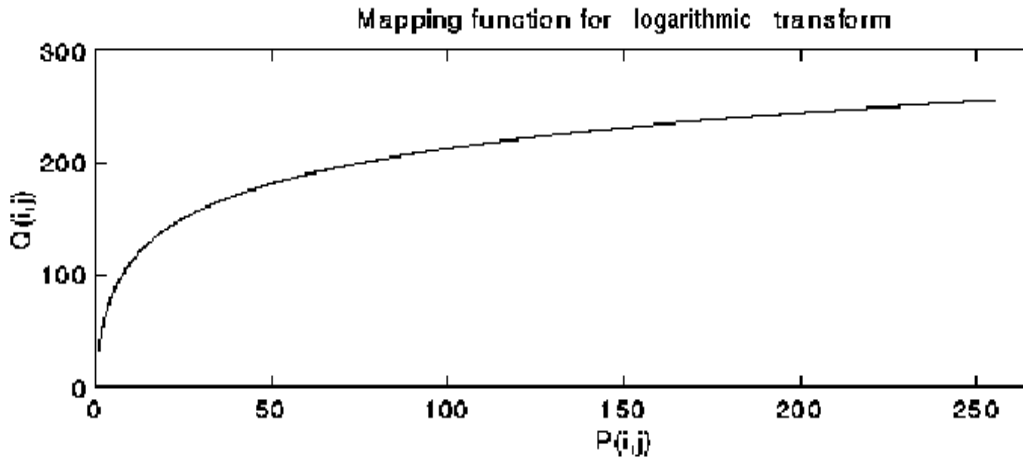
```
vector<uchar> getLUT( ) {
    vector<uchar> Table (256,0);
    for(int i =0; i < 256; ++i)
        Table[i] = (uchar)(255-i);
   return Table;
}

void process_ImageNegative(Mat &M) {
   vector<uchar> Table = get LUT ( );
   for(int i = 0; i < M.rows; ++i)
      for(int j = 0; j < M.cols; ++j)
          M.at<uchar>(i,j) = Table[M.at<uchar>(i,j)];
}

void main() {
     Mat img = imread("lena.tif", IMREAD_GRAYSCALE);
     Mat dst = img.clone();
     process_ImageNegative(dst);
     imshow("input img", img);
     imshow("processed img", dst);
     waitKey(0);
}
```



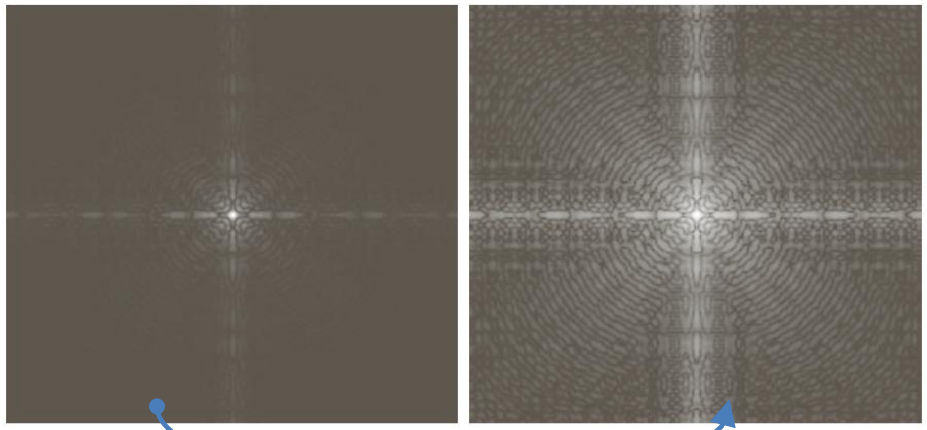
Log 변환

$$s = c \cdot log(1 + r)$$

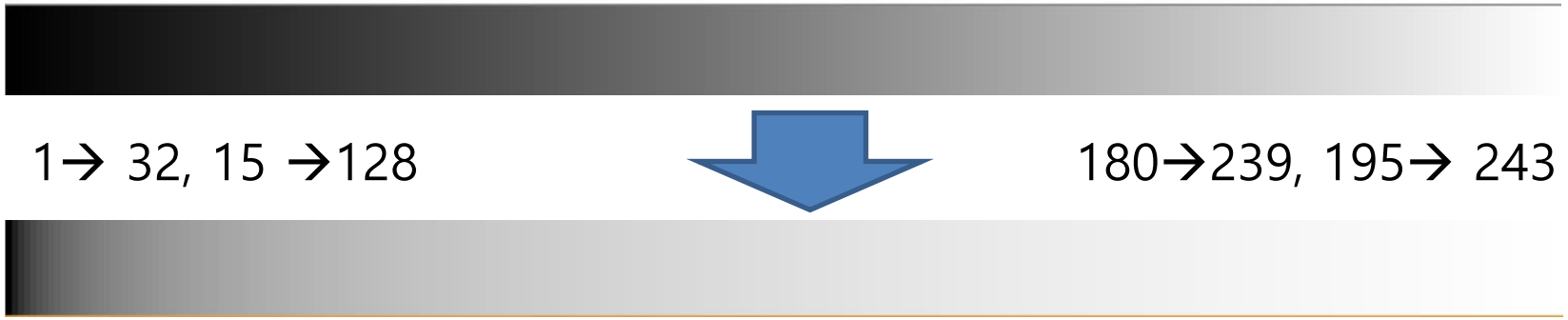
- 좁은 범위 내의 어두운 값들을 넓은 범위의 출력 레벨로 매핑(높은 밝기 값에 대해서는 반대)



c=255/log(1+255)=105.886



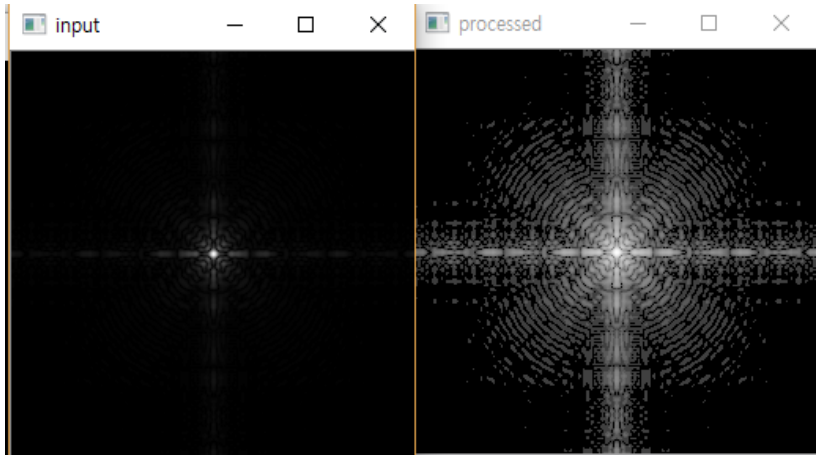
log 변환



이미지 밝기 log 변환 예제 코드

```
void get_logTable(uchar tab[ ])
{
   double rmax = 255;
   double C = 255.0 / log10(rmax + 1);
   for (int i = 0; i < 256; i++){
      double v = C * std::log10((float)i + 1.0);
      tab[i] = saturate_cast<uchar>(v);
    }
}
```

```
void process_ImageLog(char *name) {
   Mat img = imread(name,0);
   imshow("input", img);
   uchar table[256];
   get_logTable(table);

   for (int i = 0; i < img.rows; i++)
      for (int j = 0; j < img.cols; j++)
         img.at<uchar>(i, j) = table[img.at<uchar>(i, j)];
   imshow("processed", img);
   waitKey(0);
}
```



Using LUT core function for replacing the values

- OpenCV API LUT for replacing pixel value by Lookup Table
  1. Prepare Lookup Table mapping for 256 inputs : 0.. 255 → 0 .. 255
  2. Apply Lookup Table to Image

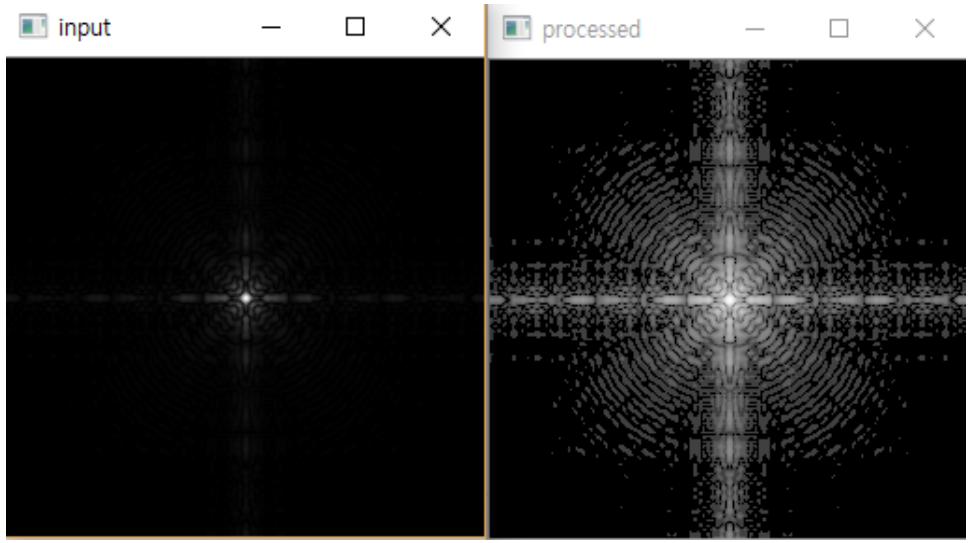
```
// preparing lookup table

Mat Table(1, 256, CV_8U);

uchar *p = Table.data;

double C = 255.0 / log10(255 + 1);
for(int i =0; i < 256; i++)
   p[i] = saturate_cast<uchar>(C * log10(i+1.0)); // option 1
   // Table.at<uchar>(0,i) = saturate_cast<uchar>(C * log10(i+1.0)); // option 2
```

```
// Applying the lookup table
// input : img
// output : trans

LUT (img, Table, trans);
```

Using Mathematical operator overloaded for Mat

- OpenCV 에서 주어지는 Mat의 연산자 및 함수만 이용하여 가능

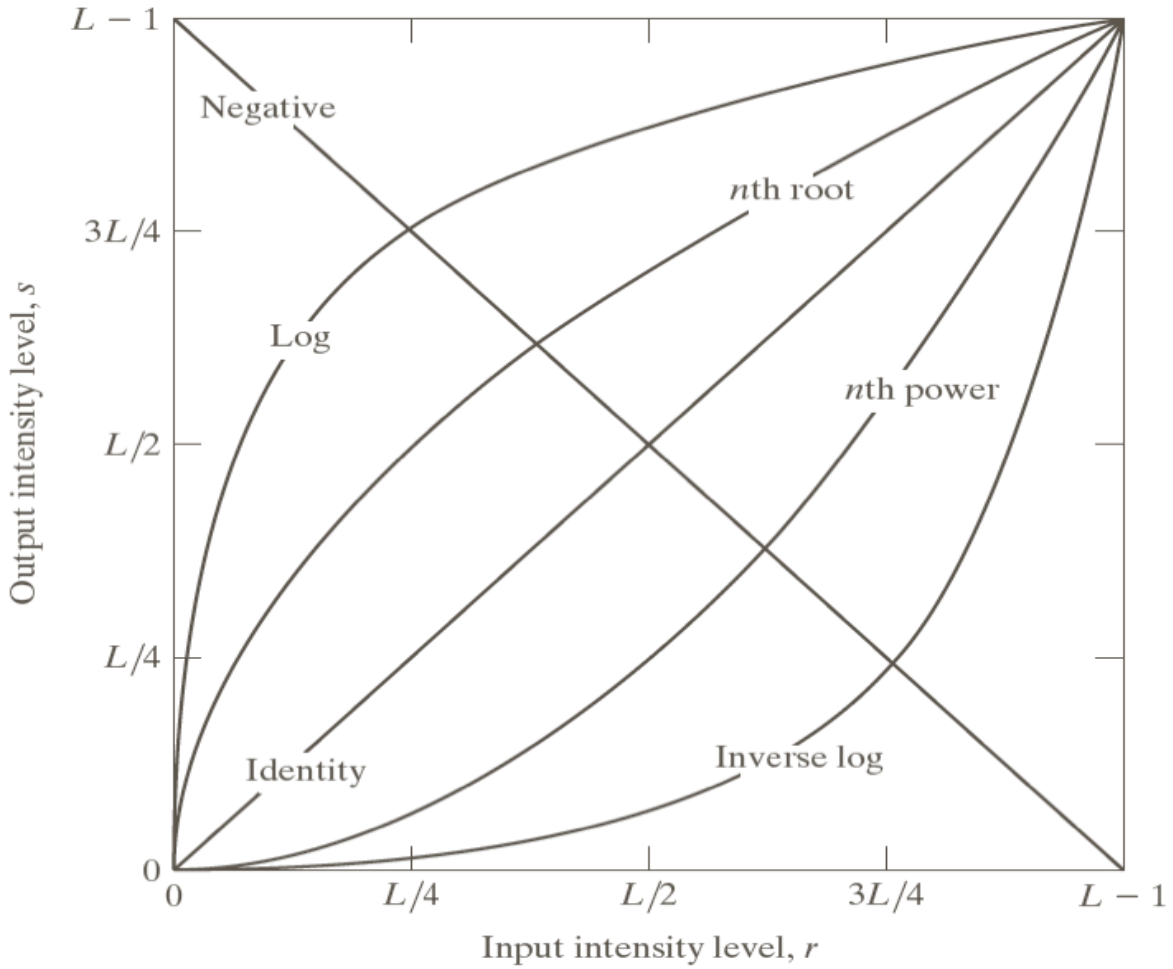
$$s = c \cdot log(1 + r)$$




```
void process_ImageLog_math(char *name)
{
    Mat img = imread(name, 0);
    Mat dst;

    img.convertTo(dst, CV_32F); // 8U --> 32F
    dst = dst + 1;  // r+ 1
    log(dst, dst);  // log(r+1)
    normalize(dst, dst, 0, 255, NORM_MINMAX);// C * log(1+r)
    convertScaleAbs(dst, dst); // 32F --> 8U

    imshow("input", img);
    imshow("processed", dst);
    waitKey(0);
}
```

그레이스케일 영상에서의 영상 보정

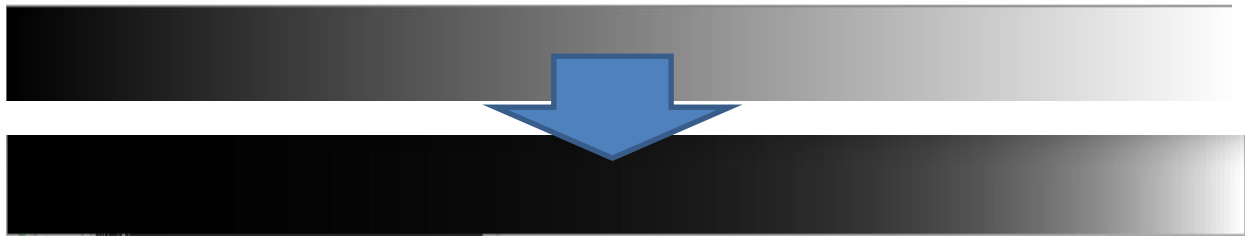
- 기본적 밝기 변환 함수들



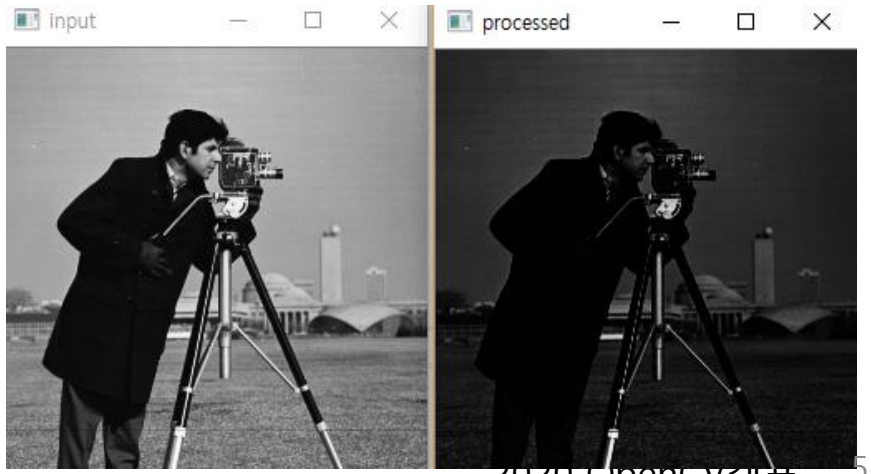
Exponential or Inverse-log transform

- $s = T(r) = \ c\ (e^r - 1)$
- log 함수의 역함수
- C = 255 /$(e^{max} - 1)$

```
void get_expTable(uchar tab[ ])
{
   double rmax = 255;
   const double BASE = 1.02;
   double C = 255.0 / (pow(BASE, rmax ) – 1);
   for (int i = 0; i < 256; i++){
      double v = C * pow(BASE, i) - 1.0);
      tab[i] = saturate_cast<uchar>(v);
    }
}
```



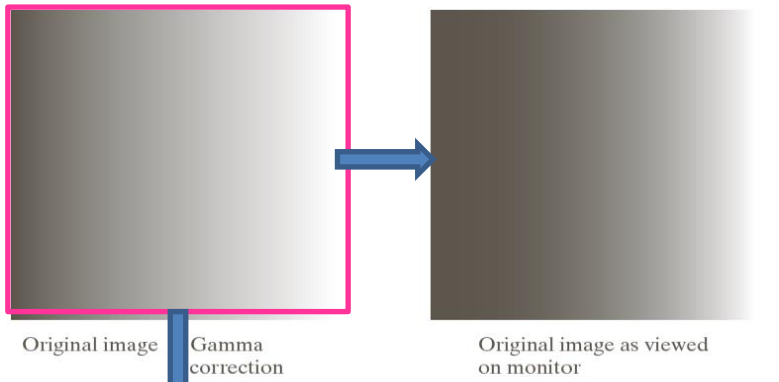
```
void process_Image_exp(char *name)
{
  Mat img = imread(name, IMREAD_GRAYSCALE);
  imshow("input", img);
  double maxval;
  minMaxLoc(img, NULL, &maxval);
  uchar table[256];
  get_expTable(table, maxval);
  Mat table_mat(1, 256, CV_8UC1, table);

  LUT(img, table_mat, img);
  imshow("processed", img);
  waitKey(0);
}
```



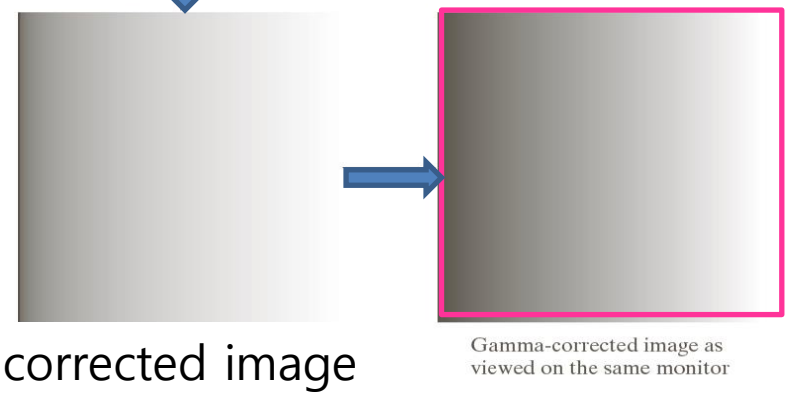
그레이스케일 영상에서의 영상 보정

- 감마 변환
  - $s = c\,\tau^{\gamma}$
  - 영상 촬영, 인쇄, 표시 등에 사용되는 다양한 장치들은 거듭제곱-법칙의 특성을 따르며, 이러한 반응 현상을 보정하기 위해 활용
  - 만약 $\gamma = 2.5$ 인 장치의 경우 입력 영상을 모니터에 입력하기 전에 $s = \tau^{1/2.5} = \tau^{0.4}$ 변환필요

original image



original image
viewed on monitor

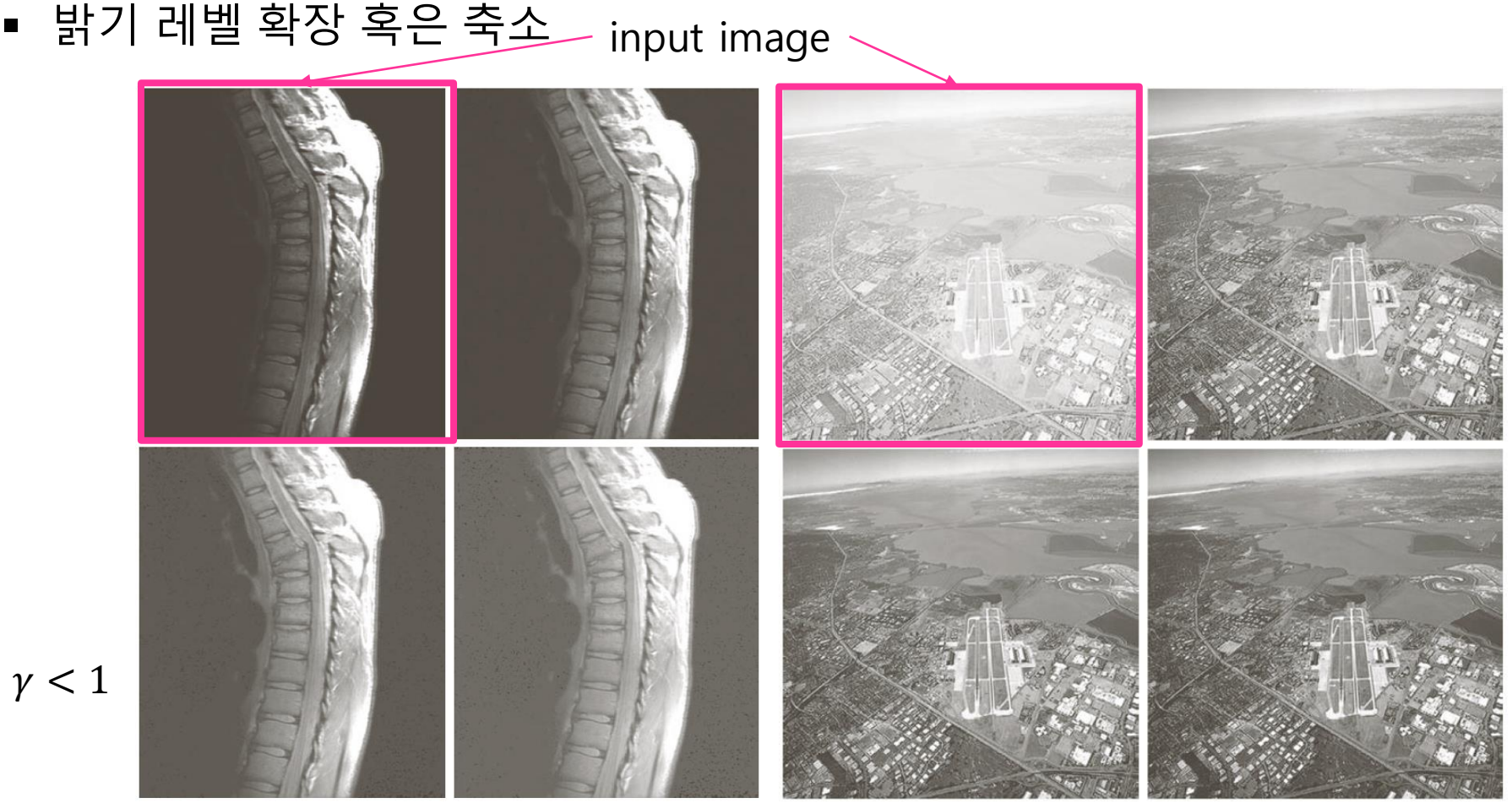


gamma-corrected image

gamma-corrected image
viewed on monitor

그레이스케일 영상에서의 영상 보정

- 감마 변환
  - 콘트라스트 조작에도 유용
  - 밝기 레벨 확장 혹은 축소



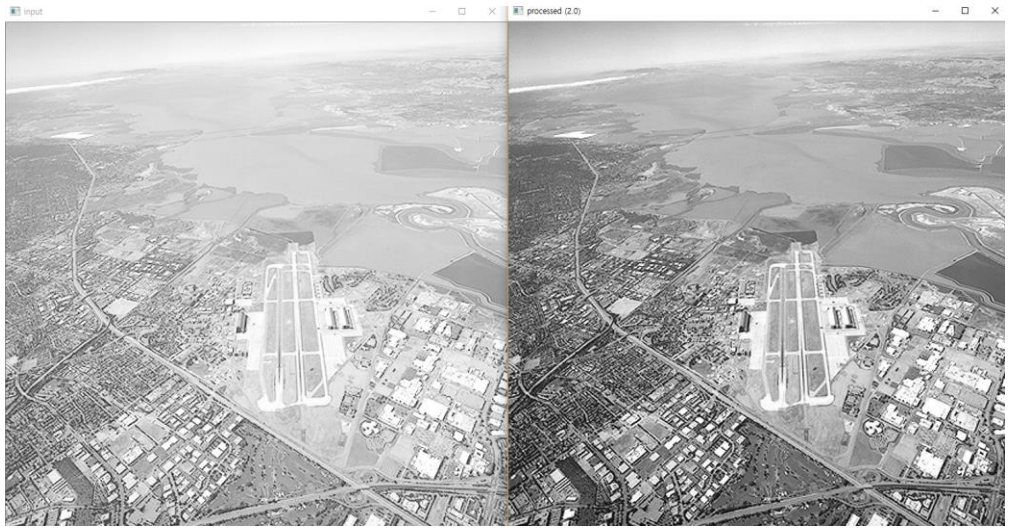
감마 변환 예제 코드

- LUT-based approach

```
void get_gamma_LUT(uchar table[], double gamma)
{
    double C = 255.0 / pow(255, gamma);
    for (int i = 0; i < 256; i++)
    table[i] = saturate_cast<uchar>(C * pow(i, gamma));
}
```

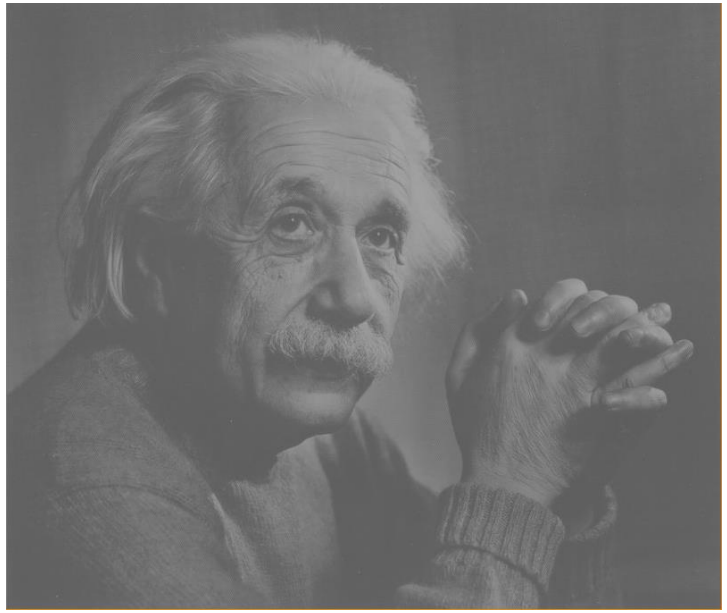
```
void process_Gamma_LUT(char *name, double gamma)
{
    Mat img = imread(name, 0);
    imshow("input", img);
    Mat dst;
    uchar table[256];
    get_gamma_LUT(table, gamma);
    Mat Table(1, 256, CV_8UC1, table);
    LUT(img, Table, dst);
    imshow("processed (0.5)", dst);

    waitKey(0);
}
```

$\gamma = 0.5$



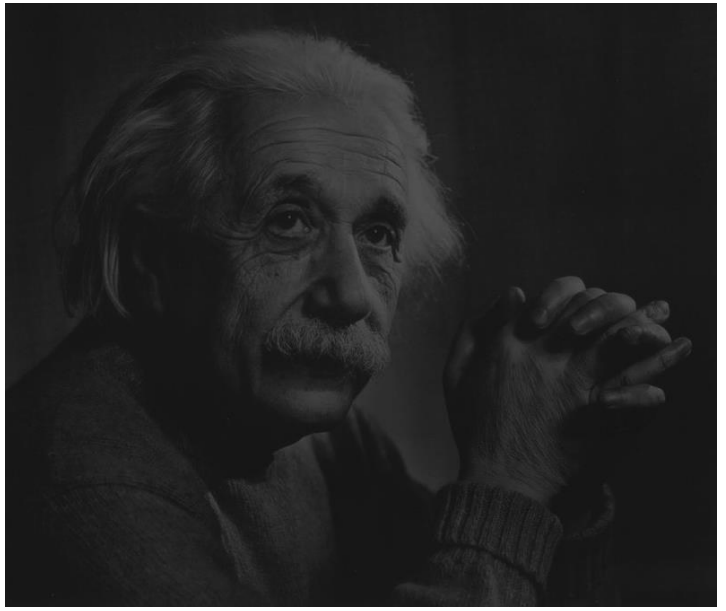
$\gamma = 2.0$

Gamma Transform Examples



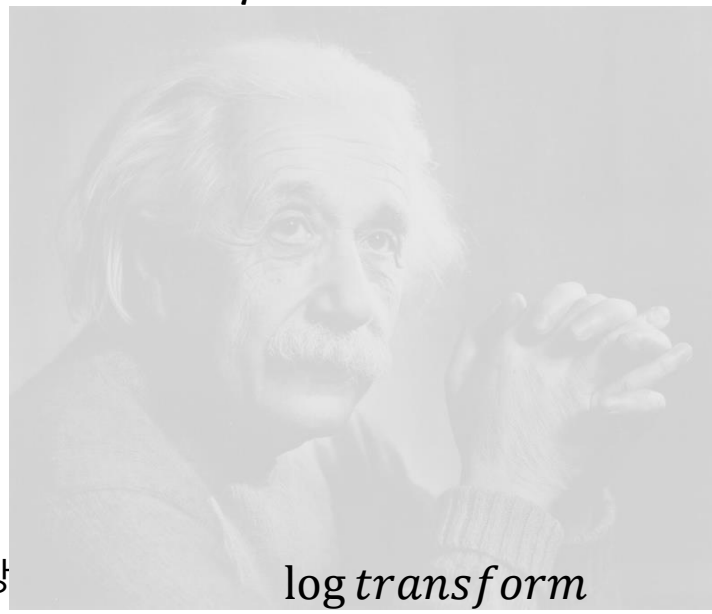
$input\ image$

$\gamma = 0.2$



$\gamma = 2.5$

- Gamma or Log Transforms cannot be enchance the Einsten.tif image

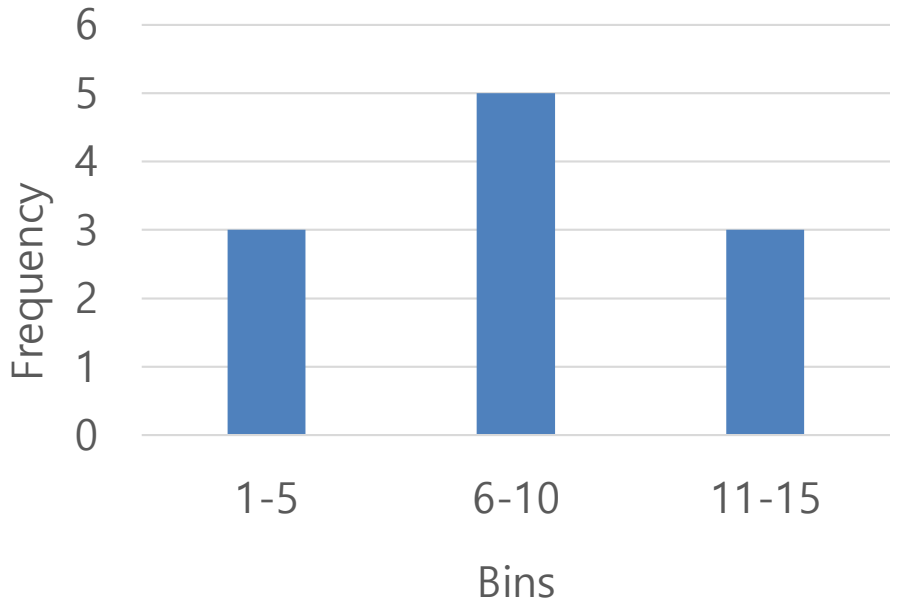


$\log transform$

Histogram

- 히스토그램이란?
  - A form of descriptor of images
  - 영상에서 (채널별) 픽셀 밝기의 분포 정보
  - 일반적 히스토그램의 의미 (예시):
    - D= {2,7,1,5,6,9,14,11,8,10,13}

Bins	frequence (빈도)
Bin1 (1~5)	3
Bin2 (6~10)	5
Bin3 (11~15)	3



- 히스토그램 용어
  - Histogram Size : Bin의 갯수, 히스토그램 x축의 눈금 수 (위 경우 size= 3)
  - Range : 입력 데이터 값의 최소, 최대값 (위 경우, {1,15})
  - Dimension : 히스토그램의 갯수 (위 경우 Dim=1)

Grayscale 이미지 Hisogram 직접 구하기

```
Mat computeHistogram(Mat & input_img)
{   // bin_size=256, range={0,255}, dim=1
    Mat hist = Mat::zero(256,1, CV_32S);
    for(int i = 0; i < input_img.rows; i++)
       for(int j = 0; j < input_img.cols; j++) {
           int binIdx = (int) input_img.at<uchar>(i,j);
           hist.at<int>(binIdx,0) += 1;
       }
    return hist;
}
```

```
void main() {
    Mat input = imread("images/lenna.tif");
    Mat hist = computeHistogram(input);
    for(int i = 0; i < host.rows; i++)
          cout << i << " : " << hist.at<int>(i,0) << endl;
}
```

OpenCV에서 Histogram 구하기 코드

- 히스토그램 구하는 OpenCV 함수

calcHist(&img, n, channels, mask, hist, dim, histSize, &histRange, uniform, accum)

- **img : input source image의 포인터 (Mat *) : 여러개의 이미지의 배열이 주어질 수있다.**
- **n : number of input images = 1**
- channels : 입력 image마다의 채널 수 (int *):n=1일 경우, 1개의 정수
- mask: 영상 전체가 대상일 경우 = NULL (Mat())
- dim: 출력 히스토그램의 갯수
- hist : 히스토그램 계산 결과가 저장되는 Mat
- histSize : 각 입력별 히스토그램 bin 갯수
- histRange: 각 입력별 히스토그램 X 축의 최소 및 최대값
- uniform = true, accum=false

1개의 Grayscale 입력 영상의 경우

```
calcHist(&img, 1, channels, Mat(), hist, 1, histSize, &histRange);
```

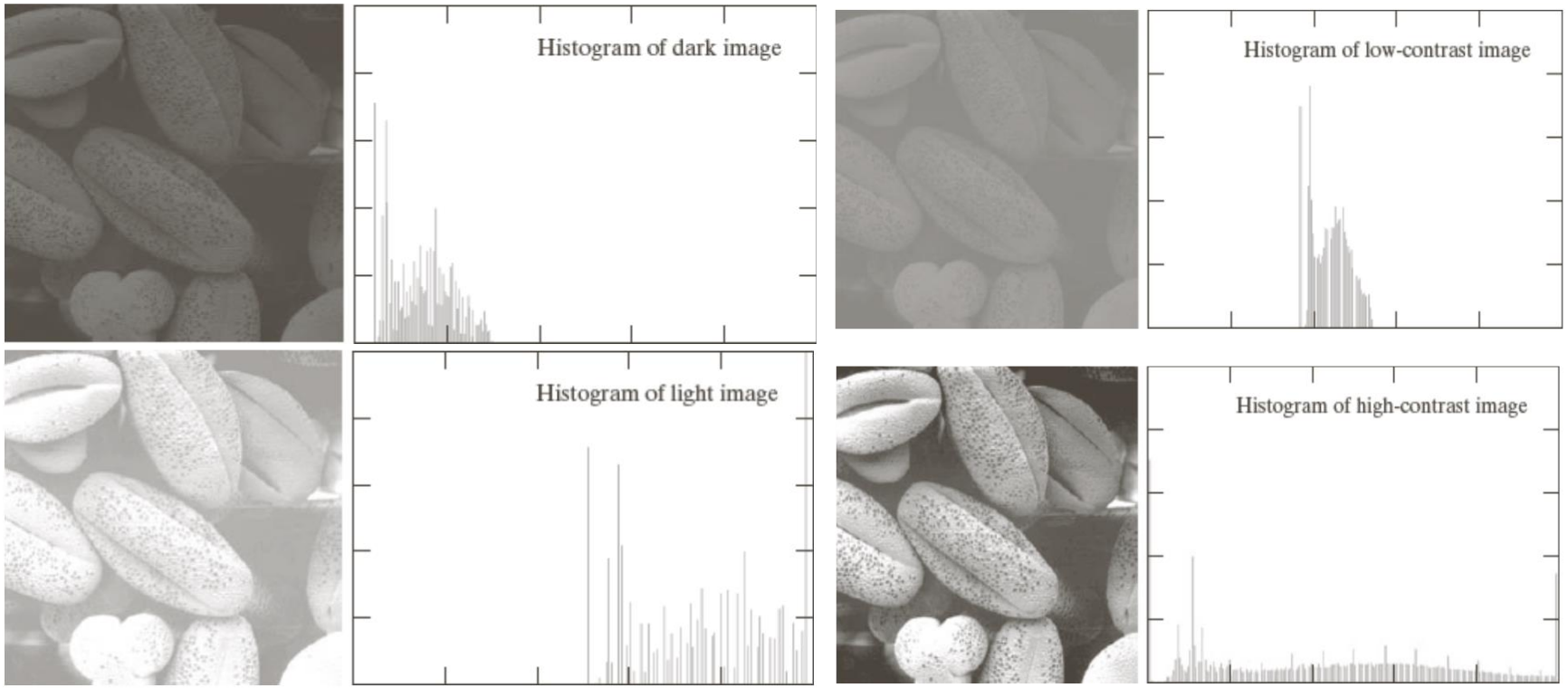
히스토그램 구하고 그리기 코드 예제

```
void show_hist(Mat & img, char *wname) {
  Mat hist;
  float range[] = { 0, 255 };
  const float * histRange = { range };
  const int histSize[] = { 256 };
  bool uniform = true, accum = false;
  const int channels[] = { 0 };
  calcHist(&img, 1, channels, Mat(), hist, 1, histSize, &histRange, uniform, accum);

  double maxVal = 0;
  minMaxLoc(hist, 0, &maxVal, 0, 0);
  int scale = 2;
  Mat histImg = Mat::zeros(256 * scale, 256 * scale, CV_8UC3);

  for (int v = 0; v < 256; v++){
     float binVal = hist.at<float>(v);
     int  num = cvRound(binVal * 255.0 / maxVal);
     rectangle(histImg, Point(v*scale, num*scale), Point((v + 1)*scale - 1, 0), Scalar(0, 0, 100), CV_FILLED);
  }
  flip(histImg, histImg, 0);
  imshow(wname, histImg);
}
```

그레이스케일 Histogram을 이용한 영상 개선

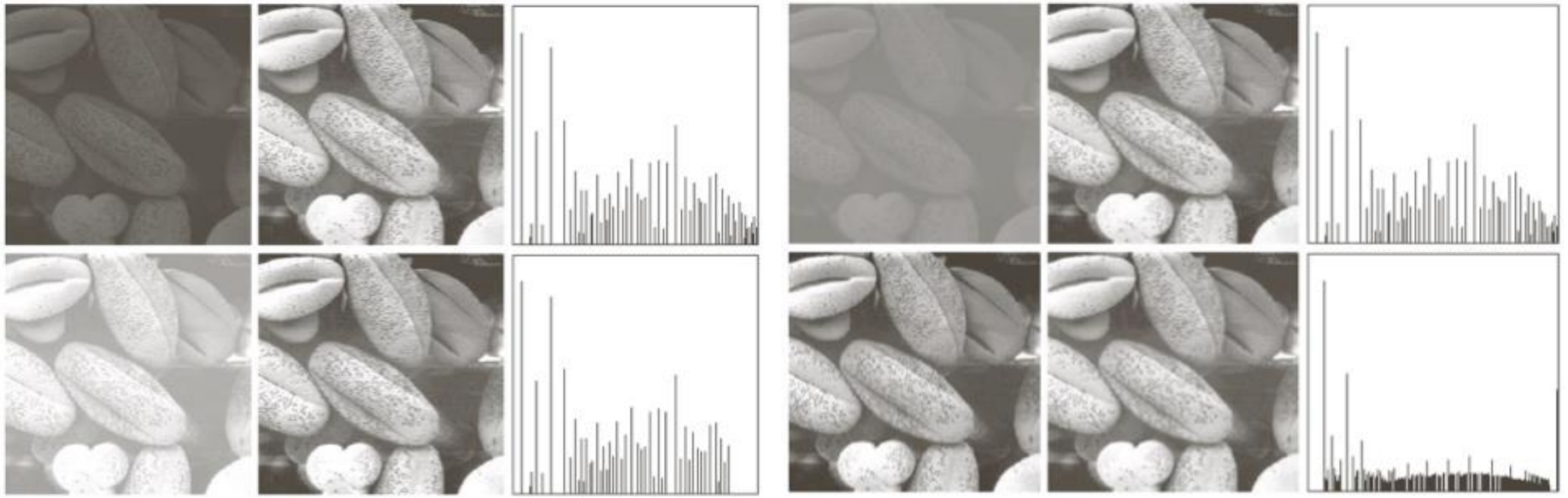
- 히스토그램
  - 많은 공간 도메인 처리 기법들을 위한 기초
  - 영상 개선, 영상 압축, 분할 같은 분야에서 응용 가능

- 다양한 그레이 영상의 히스토그램



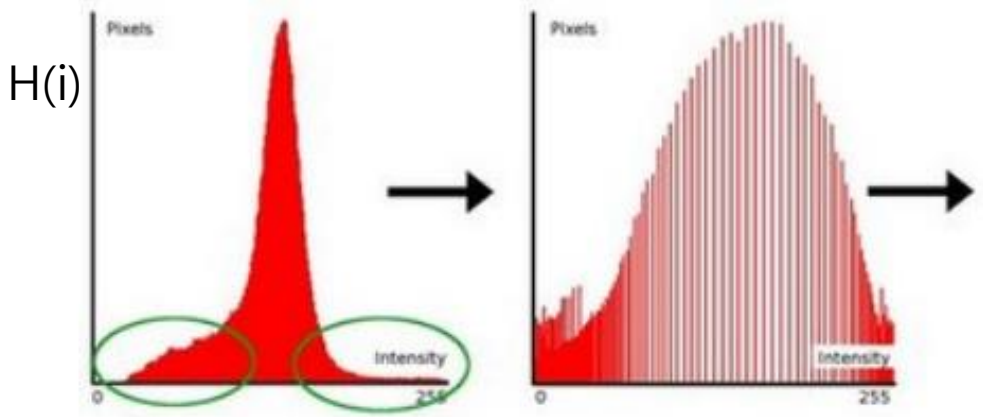
그레이스케일 영상에서의 영상 보정

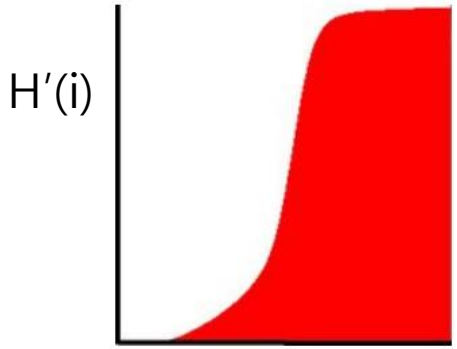
- 히스토그램 평활화
  - 히스토그램 평활화의 이유
    - 히스토그램의 분포가 균일할 수록 높은 콘트라스트와 매우 다양한 그레이톤을 가질 것으로 추정할 수 있음



히스토그램 평활화 원리

- Equalization implies *mapping* one distribution (the given histogram) to another distribution (a wider and more uniform distribution of intensity values) so the intensity values are spreaded over the whole range.





$$H'(i) = \sum_{0 \le j < i} H(j)$$

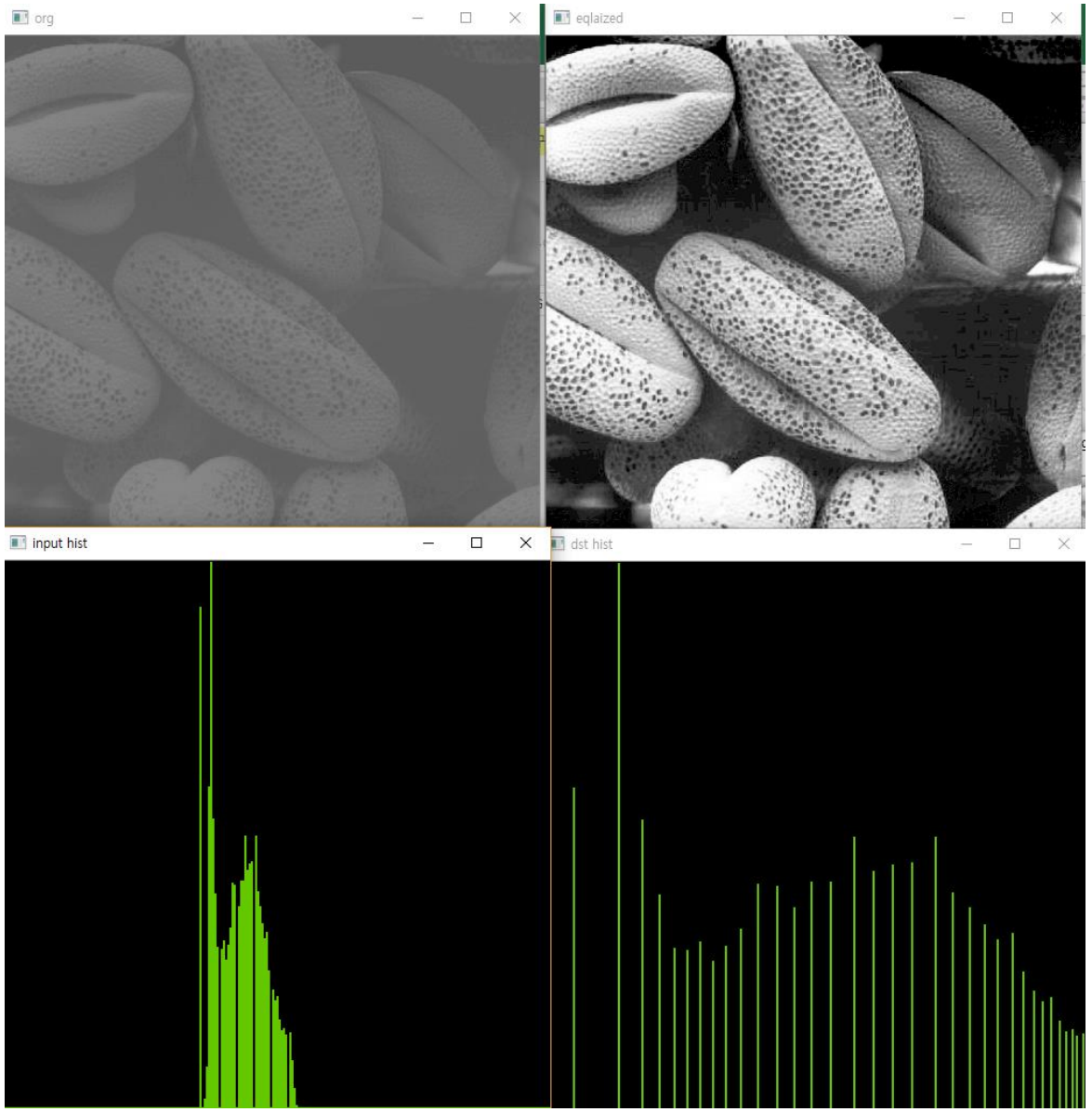
$$equalized(x,y) = H'(src(x,y))$$

Histogram Equalization 예제 코드

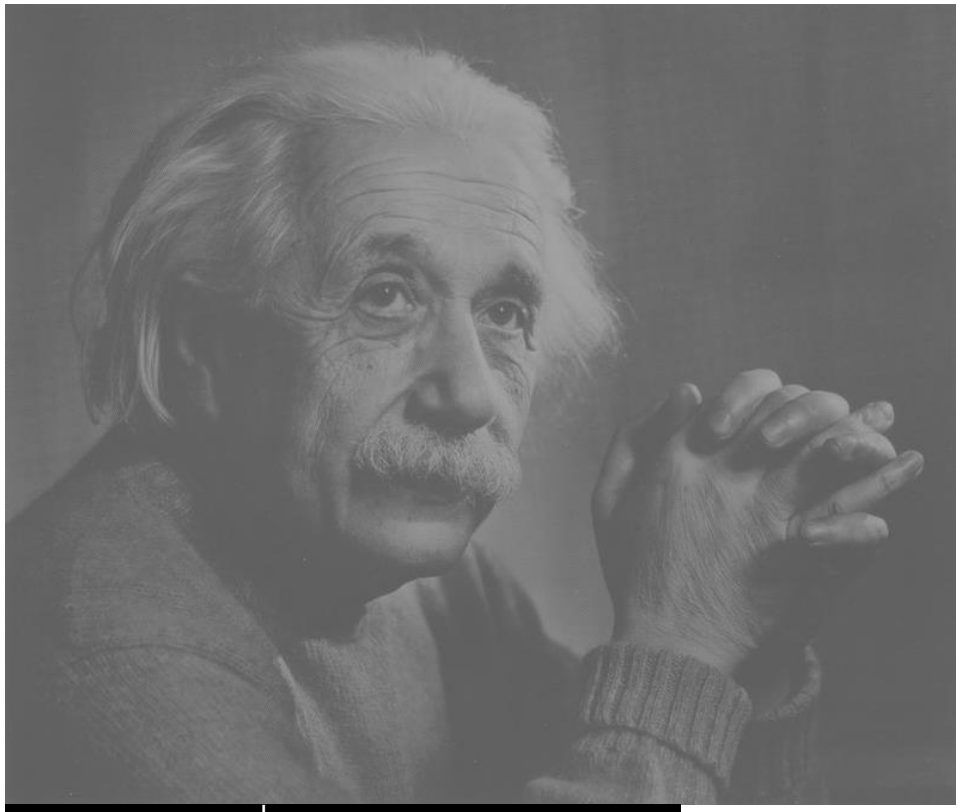
- OpenCV 함수

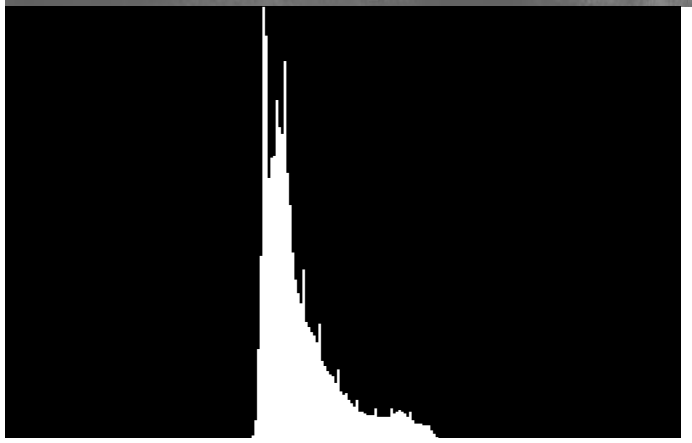
  equalizeHist(inputMat, outputMat);

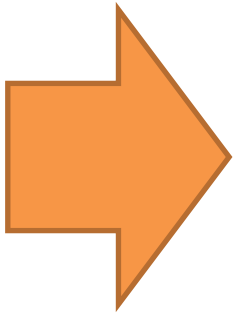
```
void histogram_Test(char *name)
{
    Mat img = imread(name,0);
    Mat dst;
    imshow("org", img);
    show_hist(img, "input hist");

    equalizeHist(img, dst);

    show_hist(dst, "dst hist");
    imshow("eqlaized", dst);

    waitKey(0);
}
```

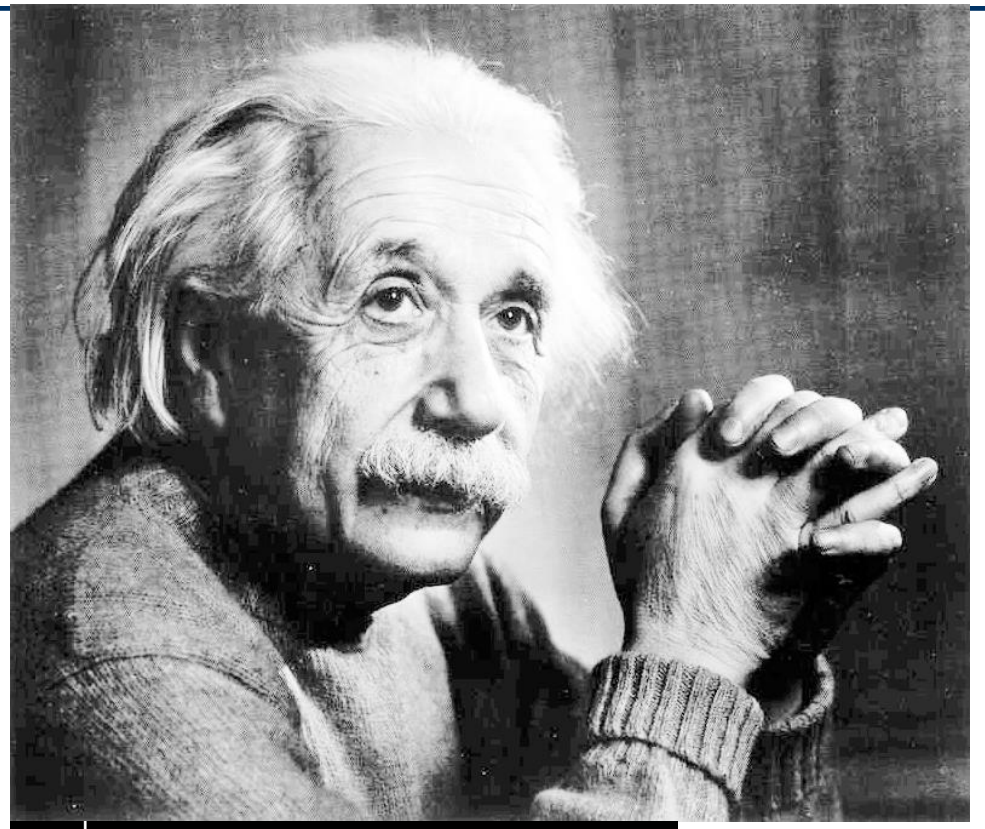


Einstein Image Enhancement by Histogram Equalization









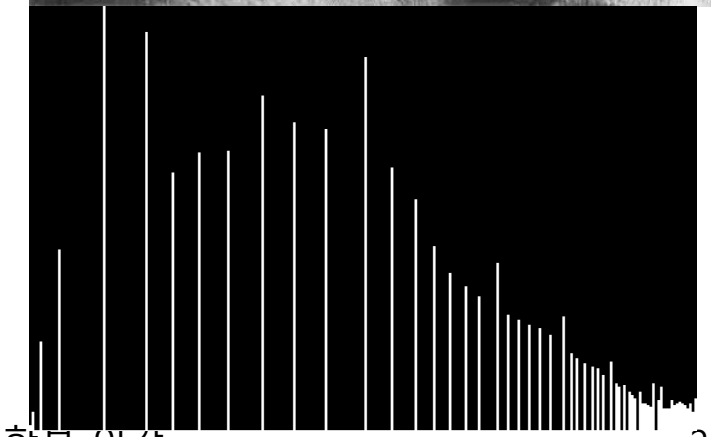
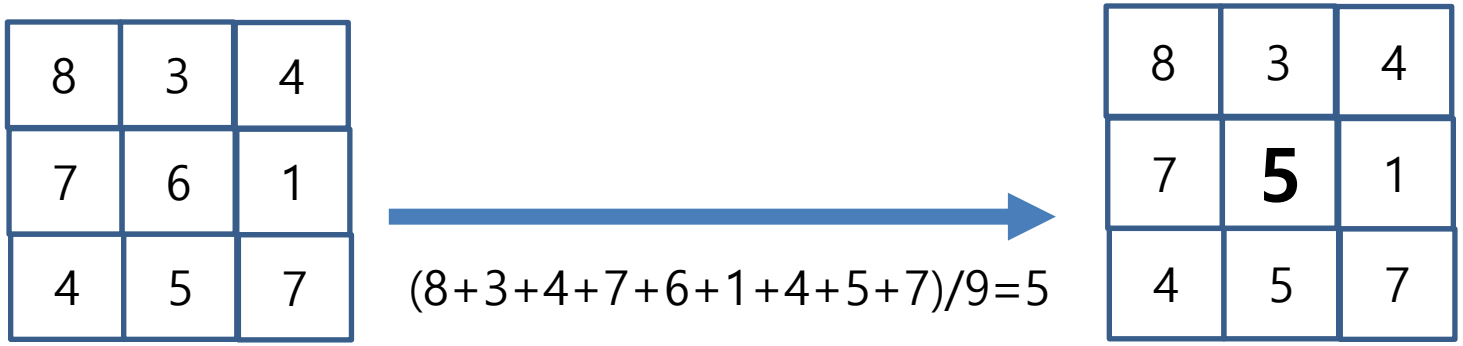


Image Filtering에 의한 영상 보정

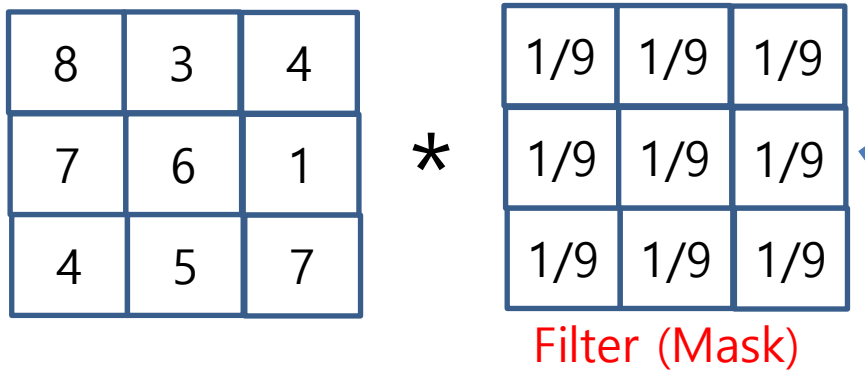
- Neighborhood of a pixel in (x,y)

(x-1, y-1)	(x, y-1)	(x+1, y-1)
(x-1, y)	(x,y)	(x+1, y)
(x-1, y+1)	(x, y+1)	(x+1, y+1)

- Image Averaging



- image Filtering
  - Convolution

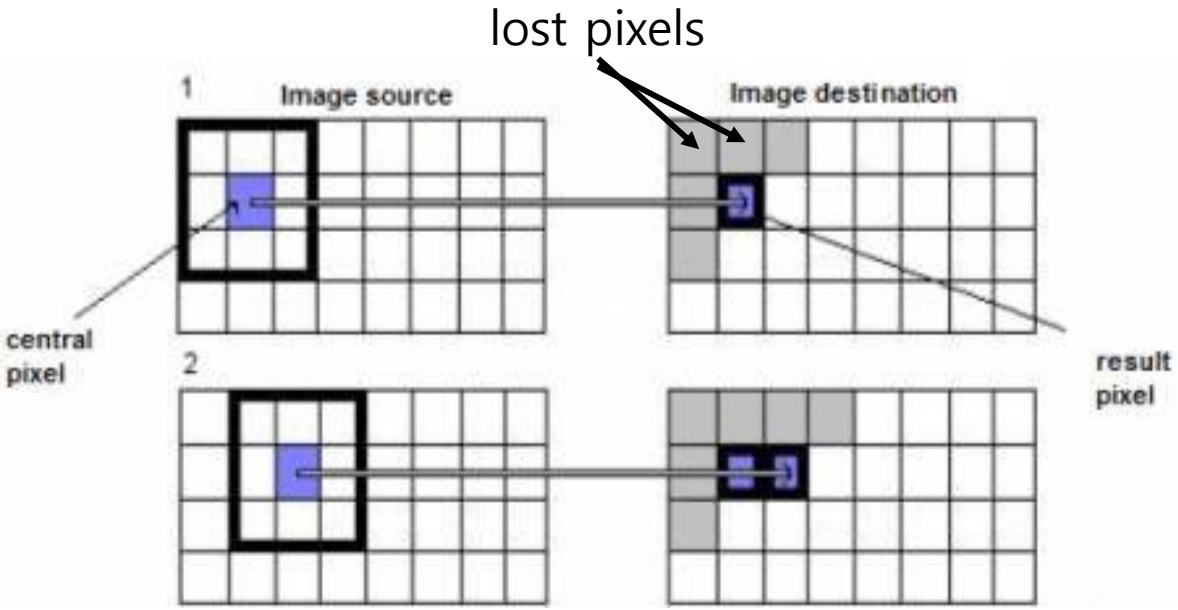


Filtering에 의한 영상 보정

- Filter의 성질
  1. The same –operation has to be performed at all pixel location
  2. The operation at any location must be linear

- 선형 공간 Filtering
  - 영상의 임의의 점 (x, y)에서의 필터 응답 $g(x, y)$는 필터 계수들과 필터가 덮고 있는 영상 화소들의 곱들의 합이다.

- 경계선 지역의 Filter 적용 문제
  - 경계선 픽셀의 Filter 적용을 위해서 바깥을 가상의 값으로 채워야 함
  - 상수 값, 경계선 값의 반복, 기타,...



0	0	0	0	0	0	0	0	0	0	0
0	1	1	2	5	6	3	6	7	3	0
0	2	3	4	6	7	5	1	8	4	0
0	8	7	6	5	7	6	3	3	4	0
0	2	3	5	6	7	8	2	7	3	0
0	4	5	3	2	1	6	8	7	2	0
0	1	4	5	3	2	6	7	8	1	0
0	2	3	4	5	6	8	9	2	1	0
0	0	0	0	0	0	0	0	0	0	0

상수값으로 채움

1	1	1	2	5	6	3	6	7	3	3
1	1	1	2	5	6	3	6	7	3	3
2	2	3	4	6	7	5	1	8	4	4
8	8	7	6	5	7	6	3	3	4	4
2	2	3	5	6	7	8	2	7	3	3
4	4	5	3	2	1	6	8	7	2	2
1	1	4	5	3	2	6	7	8	1	1
2	2	3	4	5	6	8	9	2	1	1
2	2	3	4	5	6	8	9	2	1	1

경계선 값 반복

OpenCV Mean Filter Functions

- **boxFilter**(input_image, output, depth ksize, anchor, normalize, border_type);
  - depth : output image pixel type (CV_32F, CV_8U), -1 ➔ the same depth as input
  - ksize: filter size such as Size (3,3), Size(5,5), ...
  - anchor : coordinate within filter, center of kernel = Point(-1,-1)
  - normailize (bool): if true, output is average of inputs
    else output is sum of pixels

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{vs.} \quad \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

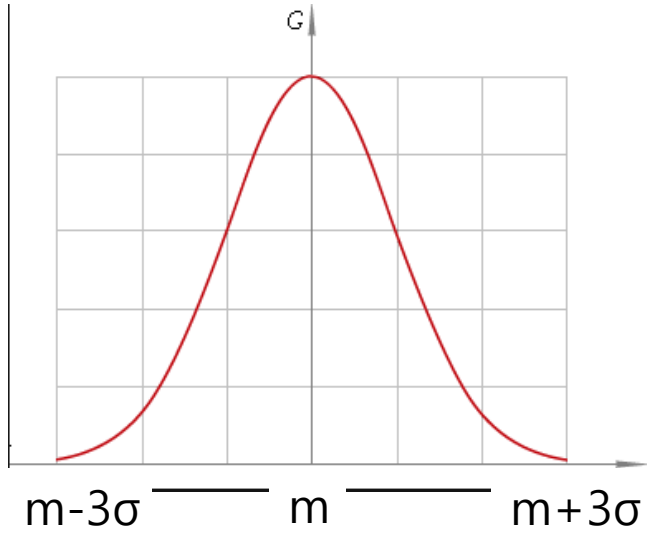
  - border_type :
    - BORDER_REPLICATE: aaaaaa abcdefgh hhhhhh
    - BORDER_REFLECT: fedcba abcdefgh hgfedcb
    - BORDER_WRAP : cdefgh abcdefgh abcdefg
    - BORDER_CONSTANT iiiiii abcdefgh iiiiii

- **blur**(input, output, ksize, anchor, border)
  - boxFilter(input,output, *-1*, ksize, anchor, *true*, border_type)과 동일

Gaussian Filtering

- Gaussian curve

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

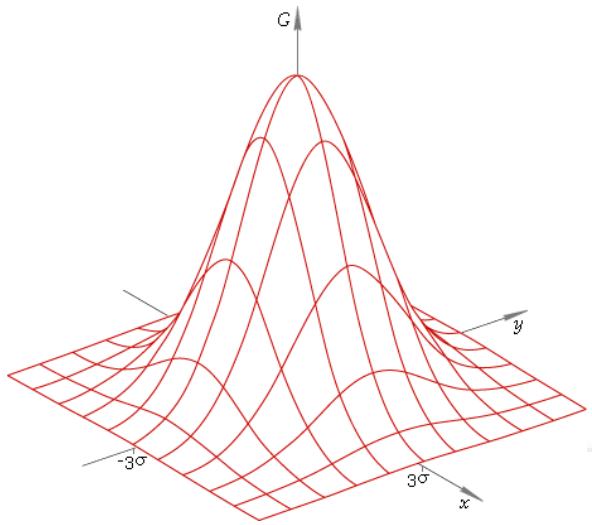



```
Gaussian Kernel size=7 sigma=
0.00443305  0.0540056  0.242036  0.39905  0.242036  0.0540056  0.00443305
Gaussian Kernel size=7 sigma=
0.0701593  0.131075  0.190713  0.216106  0.190713  0.131075  0.0701593
Gaussian Kernel size=7 sigma=
0.106289  0.140321  0.16577  0.17524  0.16577  0.140321  0.106289
```

```
void print_Gaussian(double size, double sigma) {
  Mat kernel = getGaussianKernel(size, sigma);
  int row = kernel.rows;
  int col = kernel.cols;

  for (int i = 0; i < row; i++) {
      double *row_ptr = kernel.ptr<double>(i);
      for (int j = 0; j < col; j++)
         cout << row_ptr[j] << "  ";
      cout << endl;
   }
}

void main() {
   print_Gaussian(7, 1.0);
   print_Gaussian(7, 2.0);
   print_Gaussian(7, 3.0);
}
```

2D Gaussian Filter

- 2-D Gaussian Curve $$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\left(x^2+y^2\right)/\left(2\sigma^2\right)}$$



- 2-D Gaussian Filter

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

OpenCV Gaussian Filter function

GaussianBlur(input, output, ksize, dev_x, dev_y border)

- ksize: Filter size (예: Size(7,7))
- dev_x : x 방향의 표준편차 σ_x
- dev_y : y 방향의 표준편차 σ_y

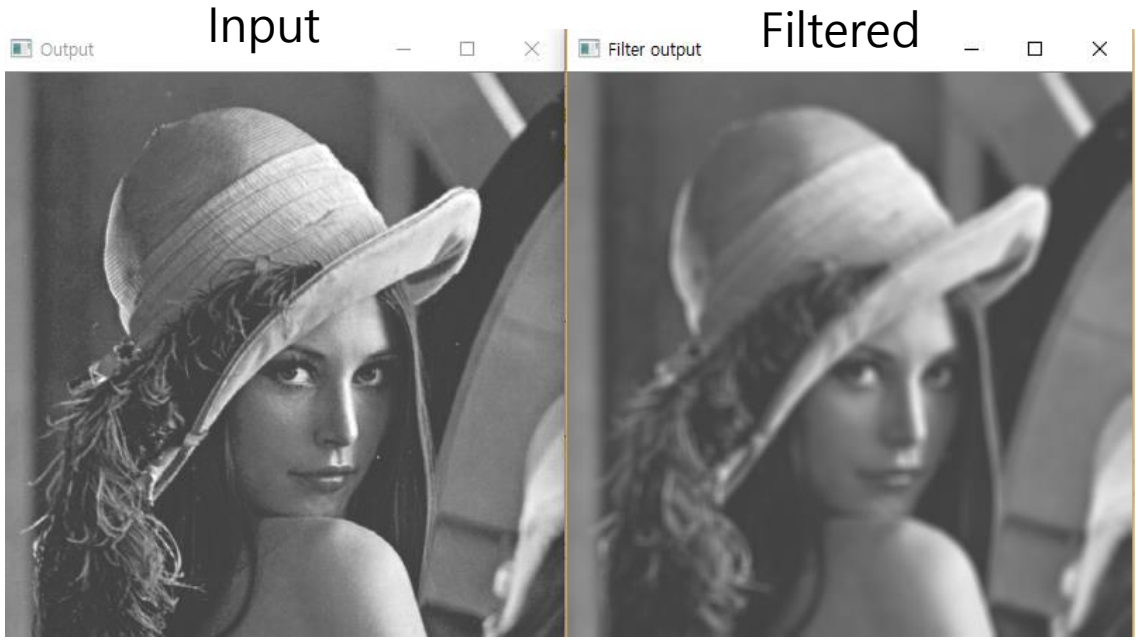
```
void main() {
  Mat img = imread( "lenna_gray.tif" , 0);
  Mat dst;
  imshow("Output", img);

  GaussianBlur(img, dst, Size(7, 7), 1.0, 1.0);
  imshow("1.0", dst);
  GaussianBlur(img, dst, Size(7, 7), 2.0, 2.0);
  imshow("2.0", dst);
  GaussianBlur(img, dst, Size(7, 7), 3.0, 3.0);
  imshow( "3.0", dst);
  GaussianBlur(img, dst, Size(7, 7), 4.0, 4.0);
  imshow( "4.0", dst);
  waitKey(0);
}
```

General Filter Design

$$f'(x,y) = \frac{1}{k \times h}\left[\sum_{i=-k/2}^{k/2}\sum_{j=-h/2}^{h/2} f(x+i, y+j)g(i,j)\right]$$

- filter2D (input, output, depth, kernel, anchor, delta, border_type)
  - depth: output Mat depth
  - kernel: user-defined 2D Mat
  - anchor: Point(-1,-1) for center location of kernel
  - delta : added to all output element after filtering

```
const int KSIZE = 7;
Mat img = imread(name, 0);
Mat dst;
imshow( "Input", img);

Mat kernel = Mat::ones(KSIZE, KSIZE, CV_32F);
kernel = kernel / (float)(KSIZE*KSIZE);
filter2D(img, dst, -1, kernel);
imshow("Filter output", dst);
waitKey(0);
```



General Filter Design

- << 연산자를 이용한 Mat 초기화로 kernel 정의

```
void main( ) {
  const int KSIZE = 3;
  Mat img = imread("lenna_gray.tif", 0);
  Mat dst;
 Mat kernel(KSIZE, KSIZE, CV_32F);

  kernel = Mat_<float>(3, 3) << 1.0, 2.0, 1.0, 2.0, 4.0, 2.0, 1.0, 2.0, 1.0);
  kernel = kernel / 16.0;

  cout << kernel;
  filter2D(img, dst, -1, kernel);
  imshow("Filter output", dst);

  waitKey(0);
}
```

그레이스케일 영상에서의 영상 보정

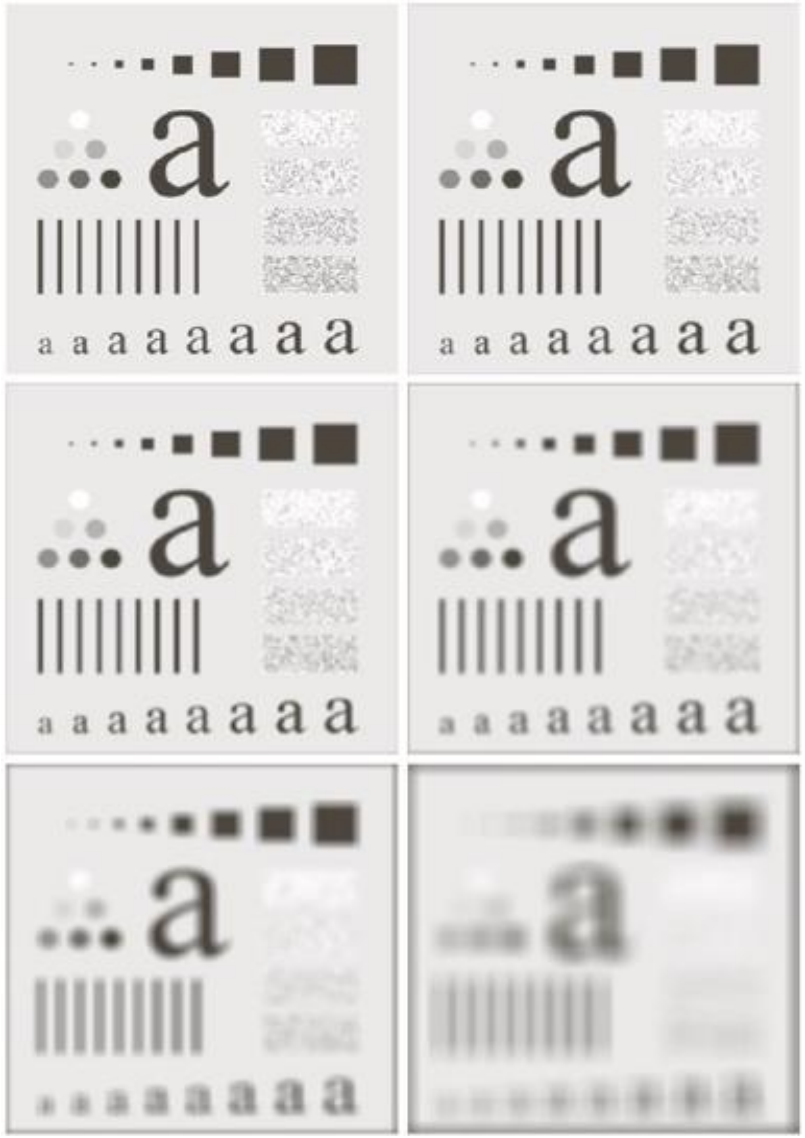
- 스무딩 (Smoothing)
  - 스무딩 필터의 활용
    - **블러링과 노이즈 감소에 사용**
    - 밝기에서의 가파를 변화를 감소
    - 무의미한 디테일의 축소
  - 다양한 형태의 스무딩 필터
    - 표준 평균, 가중 평균
    - 필터의 크기가 클 수록 스무딩 효과가 큼

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

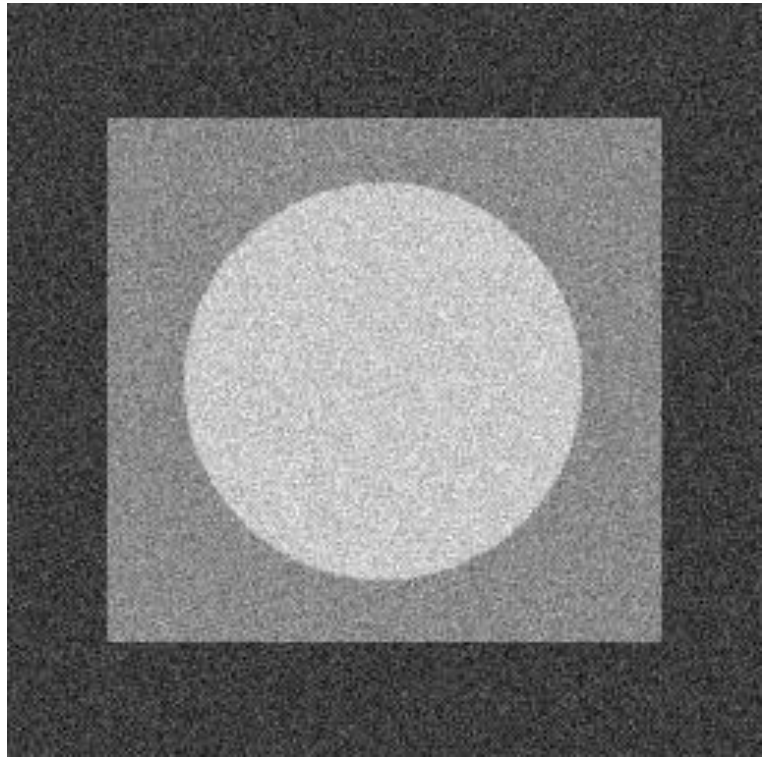
$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1



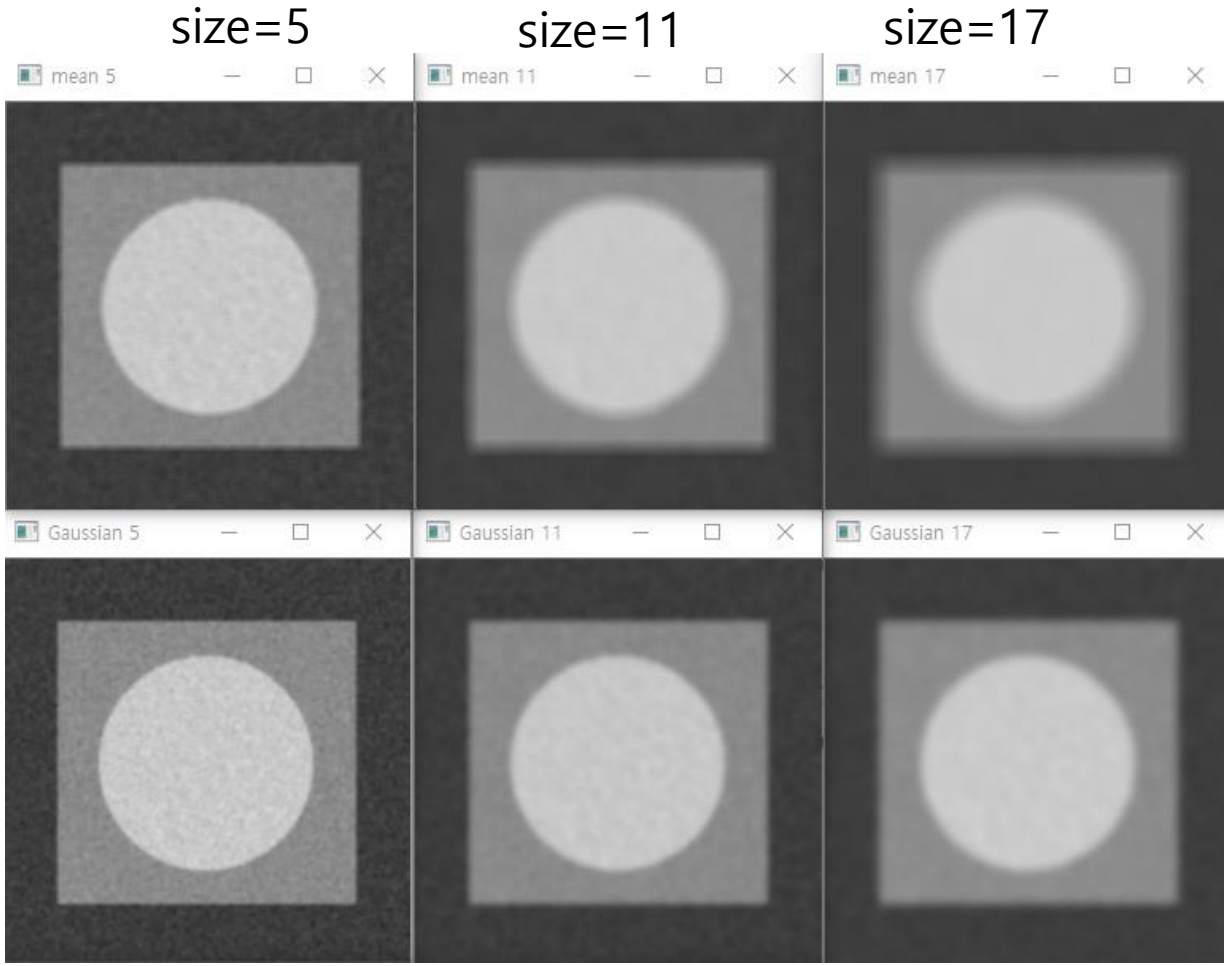
Blurring에 의한 노이즈 제거: Gaussian Noise

```
for(int i = 5; i < 20; i += 6) {
  blur(img, blurred, Size(i, i));
  GaussianBlur(img, blurred, Size(i, i), 0, 0);
  medianBlur(img, blurred, i);
  bilateralFilter(img, blurred, i, i * 2, i / 2);
}
```

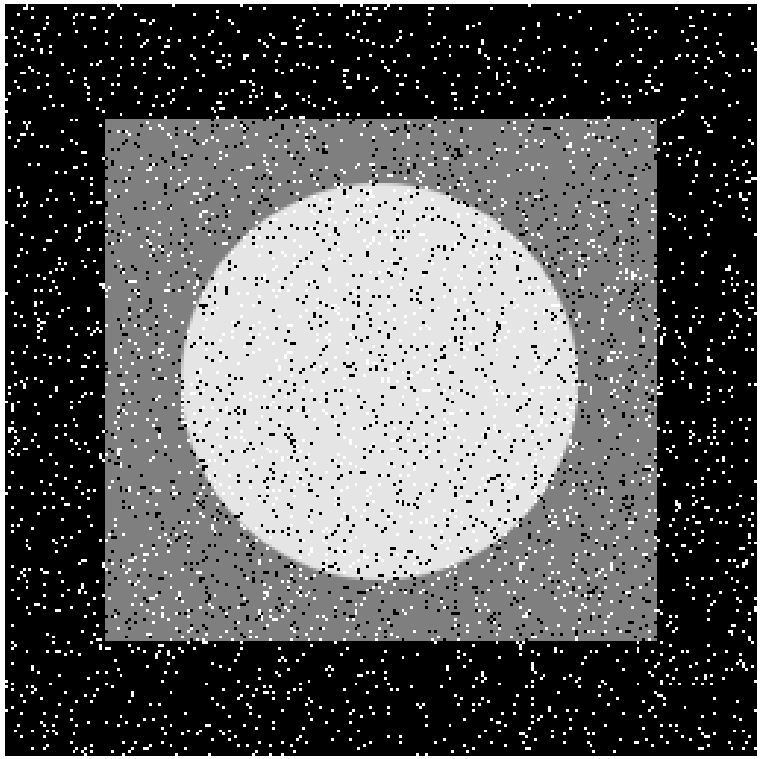


Mean

Gaussian



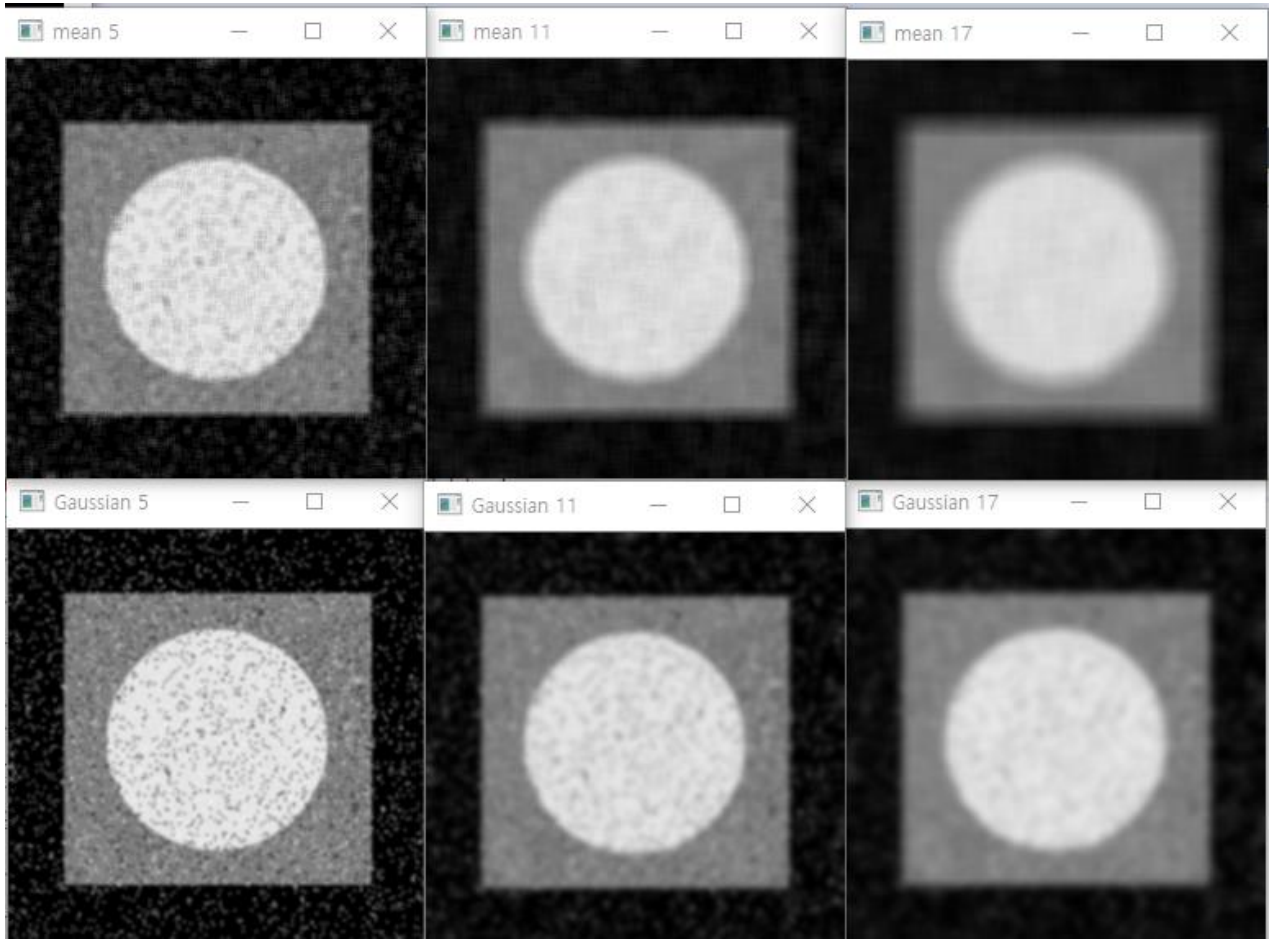
Blurring에 의한 노이즈 제거: Salt & Pepper Noise



size=5 size=11 size=17

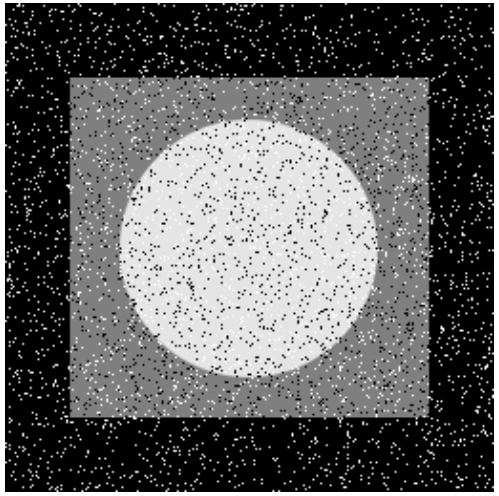
Mean

Gaussian



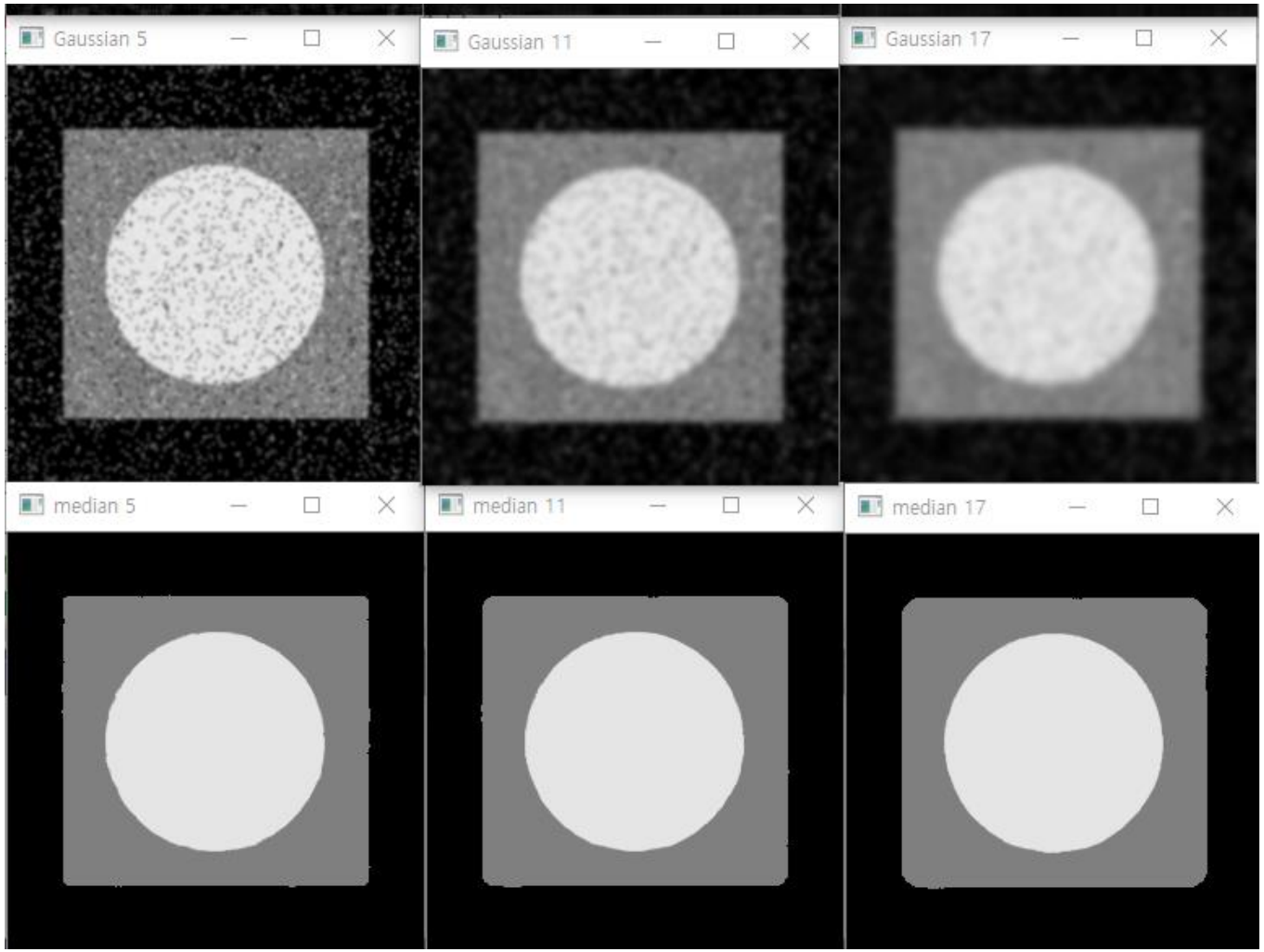
Non-Linear Filter: Median Filter

- kernel 내 픽셀값들을 정렬한 뒤 가운데(median)값을 취함
- Salt and Pepper Noise에 매우 강함



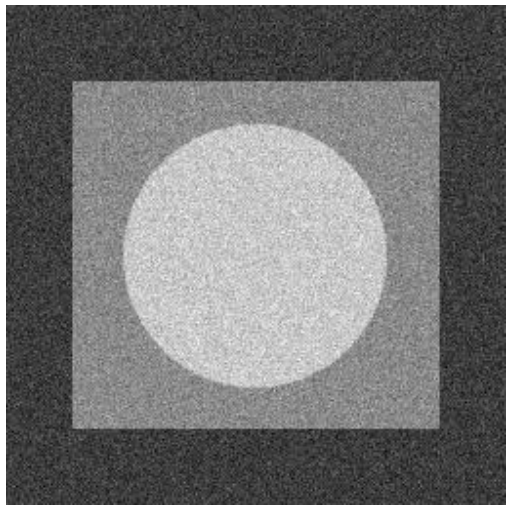
Gaussian

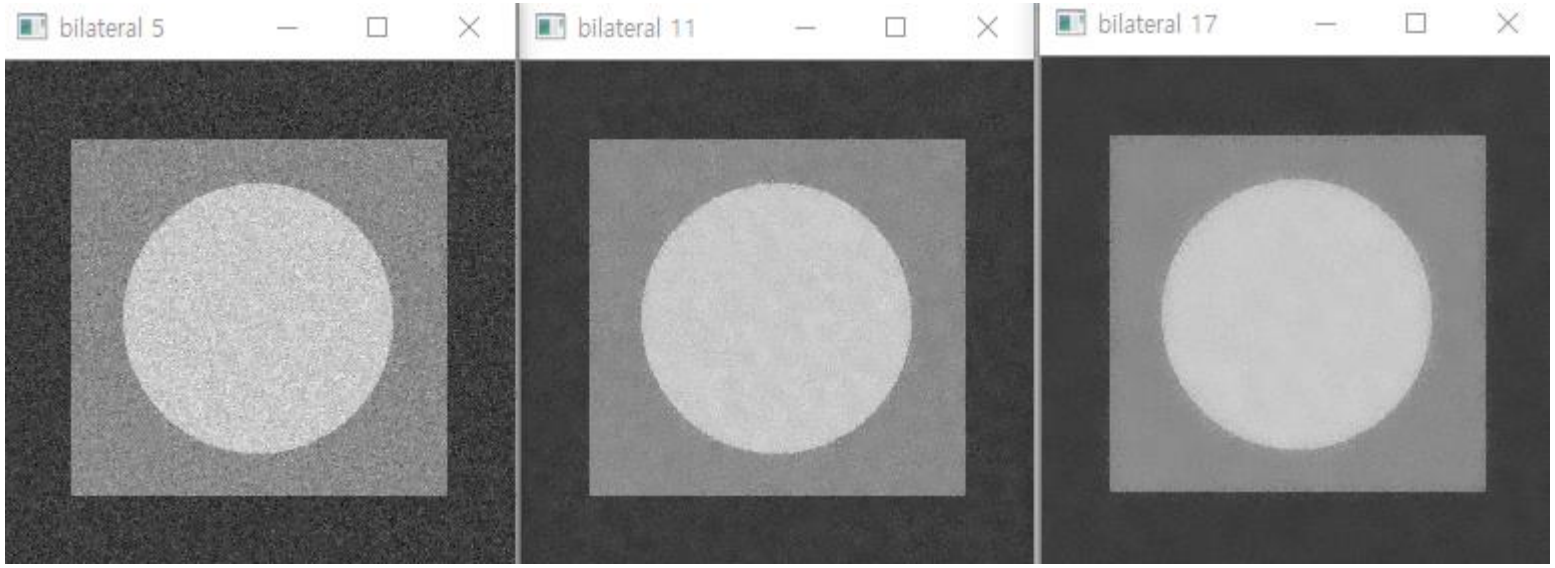
Median

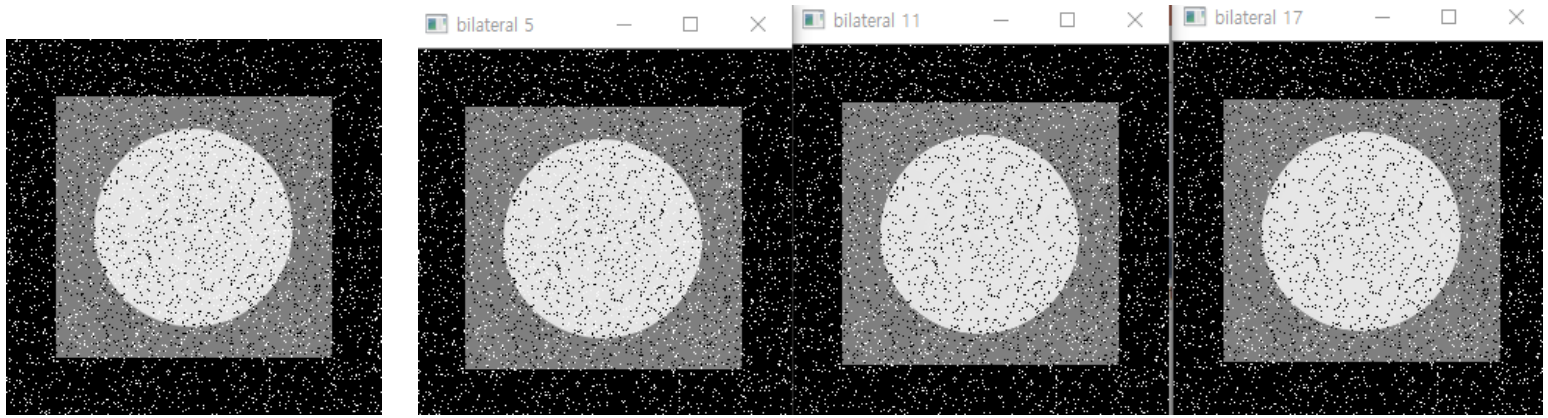


Special Non-linear Filter: Bilateral Filter

- 에지는 살리고, 노이즈는 제거
- Salt and pepper Noise에는 취약





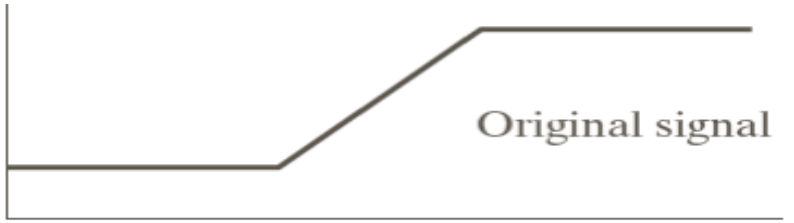


그레이스케일 영상에서의 영상 보정

- 샤프닝
  - 활용
    - 밝기의 변화를 강조
  - 개념
    - 영상 블러링은 적분과 유사
    - 영상 샤프닝은 미분과 유사
  - 방법
    - 1) 원래 영상을 블러링시킨다.
    - 2) 블러링된 영상을 원래 영상으로부터 뺀다.
    - 3) 이 마스크를 원래 영상에 더한다.

$$g(x,y) = f(x,y) + k \times g_{mask}(x,y)$$

f



g_{mask}

g

Sharpening

- addWeighted(src1, alpha, src2, beta, gamma, dst, dtype)
  - src1: input image 1
  - alpha: weight of src1
  - src2: input image 2
  - beta: weight of src2
  - dst : output image = (alpha * src1 + beta * src2 + gamma)
  - dtype : optional depth of output array (-1 default)

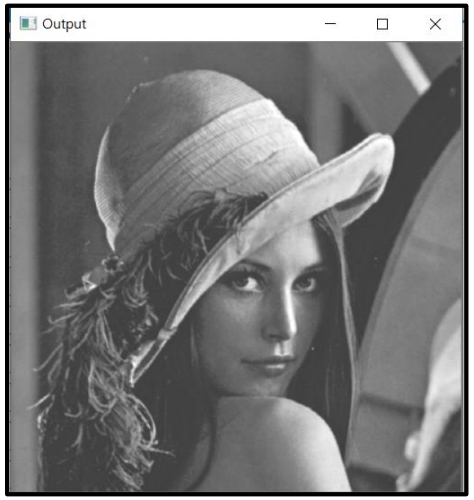
$$\texttt{dst}(I) = \texttt{saturate}(\texttt{src1}(I) * \texttt{alpha} + \texttt{src2}(I) * \texttt{beta} + \texttt{gamma})$$

```
void sharpening(char *name){
  Mat img = imread(name, 0);
  Mat blurred, dst;

  blur(img, blurred, Size(7, 7));
  addWeighted(img, 2.0, blurred, -1.0, 0.0, dst);
  imshow("org", img);
  imshow("blurred", blurred);
  imshow("shapen", dst);
}
```

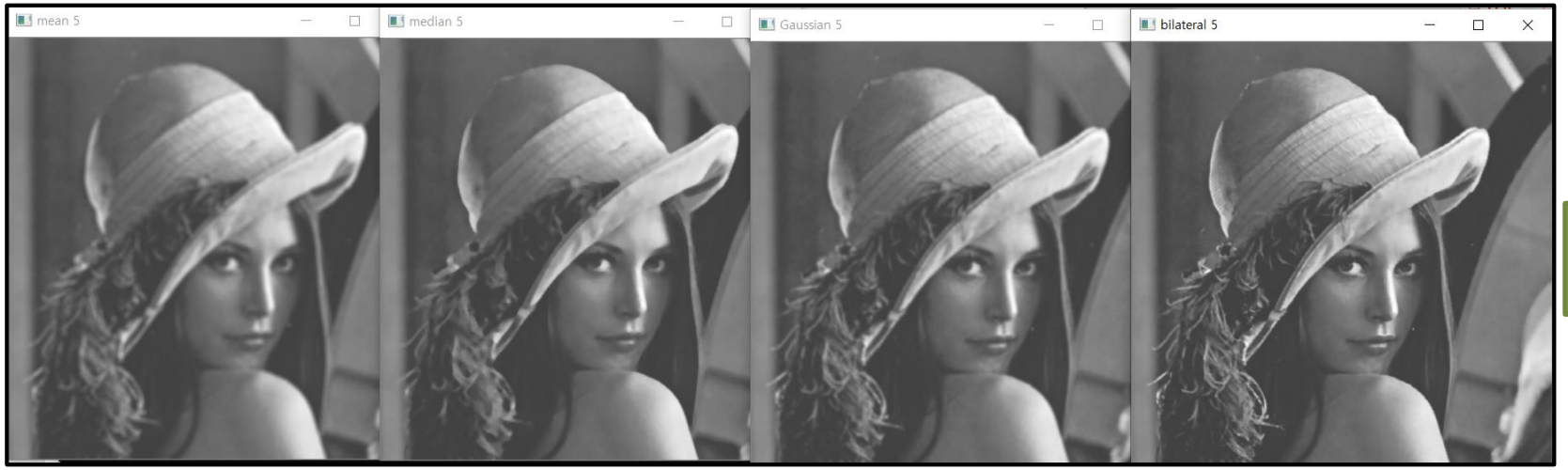
BLURRING 예제

Blurring 예제

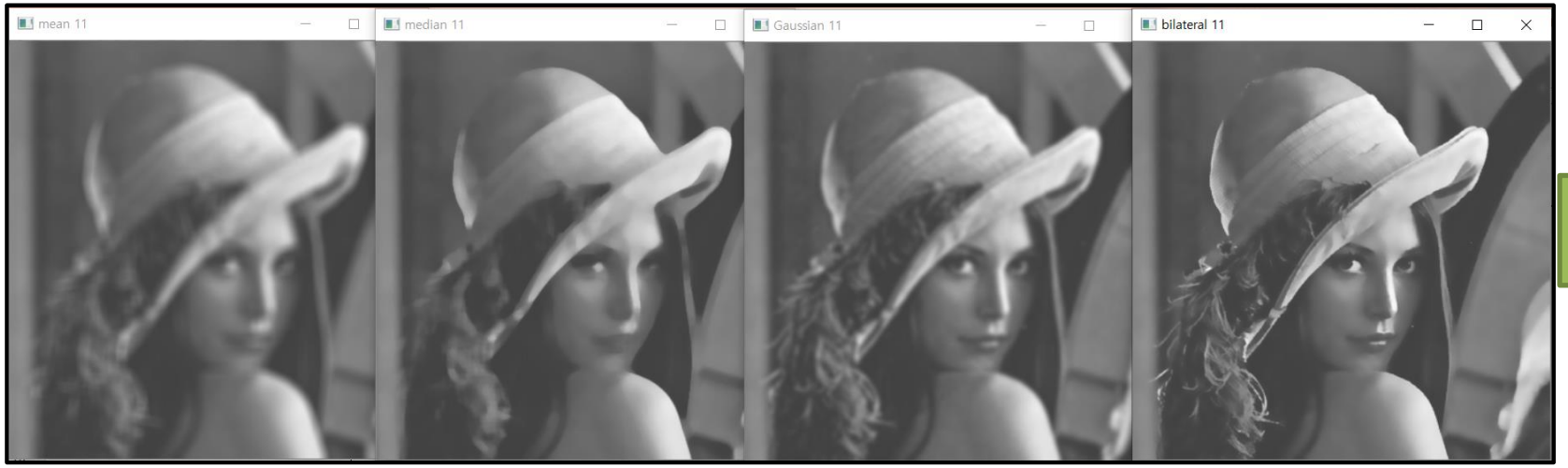
- Kernel 크기를 바꿔가며 다양한 blur 함수를 실행해 비교해보자.



원본



Kernel size (5, 5)



Kernel size (11, 11)

Blurring 예제

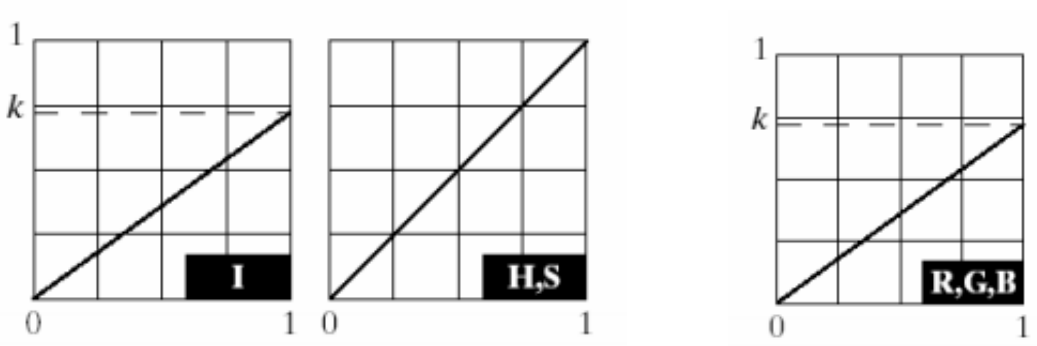
- Kernel 크기를 바꿔가며 다양한 blur 함수를 실행해 비교해보자.

```
7   void smoothing(char *name) {
8       Mat img = imread(name, 0);
9       Mat blurred, dst;
10      if (img.empty())
11      return;
12      imshow("Output", img);
13      char title[20];
14      for (int i = 5; i < 20; i += 6) {
15          blur(img, blurred, Size(i, i));
16          sprintf(title, "mean %d", i);
17          imshow(title, blurred);
18
19          medianBlur(img, blurred, i);
20          sprintf(title, "median %d", i);
21          imshow(title, blurred);
22
23          GaussianBlur(img, blurred, Size(i, i), 0, 0);
24          sprintf(title, "Gaussian %d", i);
25          imshow(title, blurred);
26
27          bilateralFilter(img, blurred, i, i * 2, i / 2);
28          sprintf(title, "bilateral %d", i);
29          imshow(title, blurred);
30      }
31      waitKey(0);
32  }
```

COLOR IMAGE ENHANCEMENT

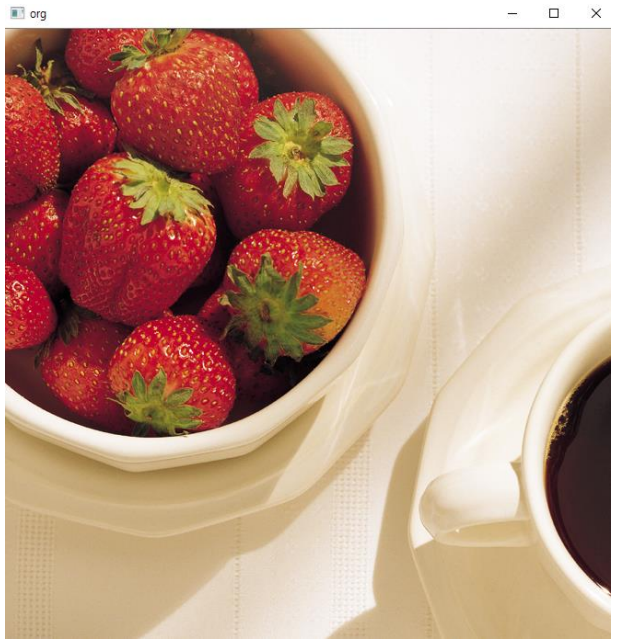
컬러영상에서 밝기 조정하기

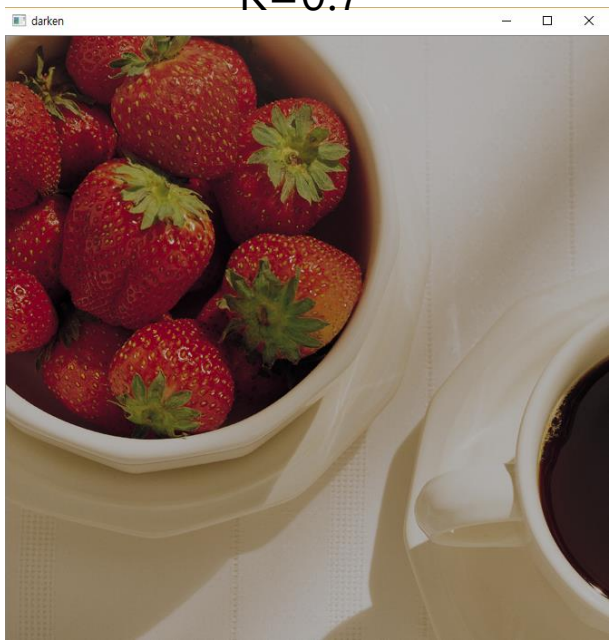
- HSI 에서 I채널 픽셀값들 k배로
- RGB에서 모든 채널의 픽셀값들을 K배로




```
void color_darken(char *name) {
    Mat img = imread(name, 1);
    Mat hsv, enhanced;
    imshow("org", img);

    cvtColor(img, hsv, COLOR_BGR2HSV);
    std::vector<Mat> channels(3);
    split(hsv, channels);

    channels[2] = 0.7* channels[2];
    merge(channels, enhanced);

    cvtColor(enhanced, enhanced, COLOR_HSV2BGR);
    imshow("darken", enhanced);
    waitKey(0);
}
```

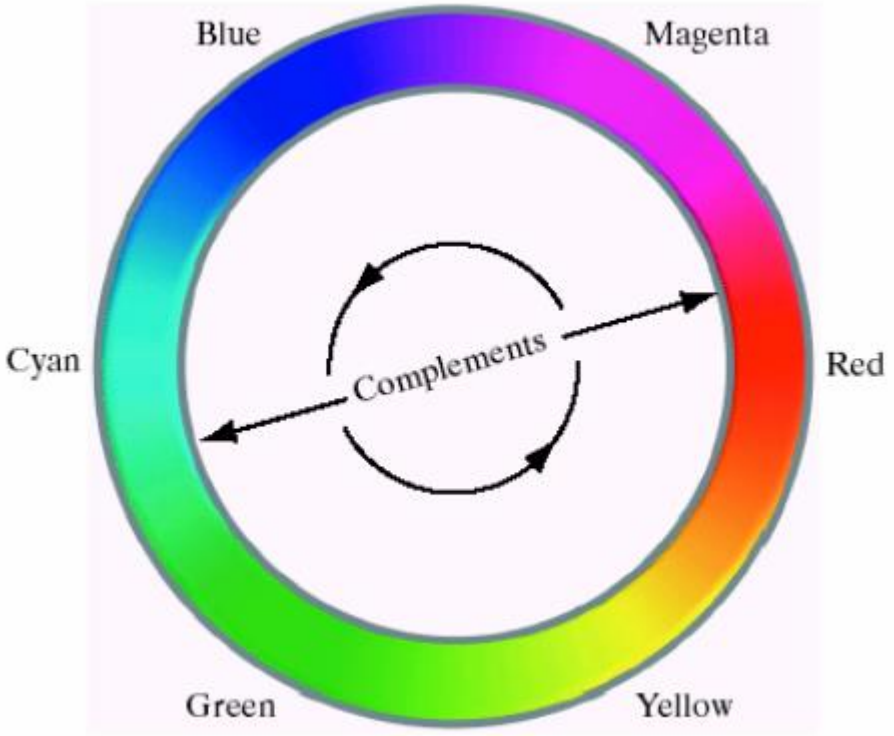
K=0.7

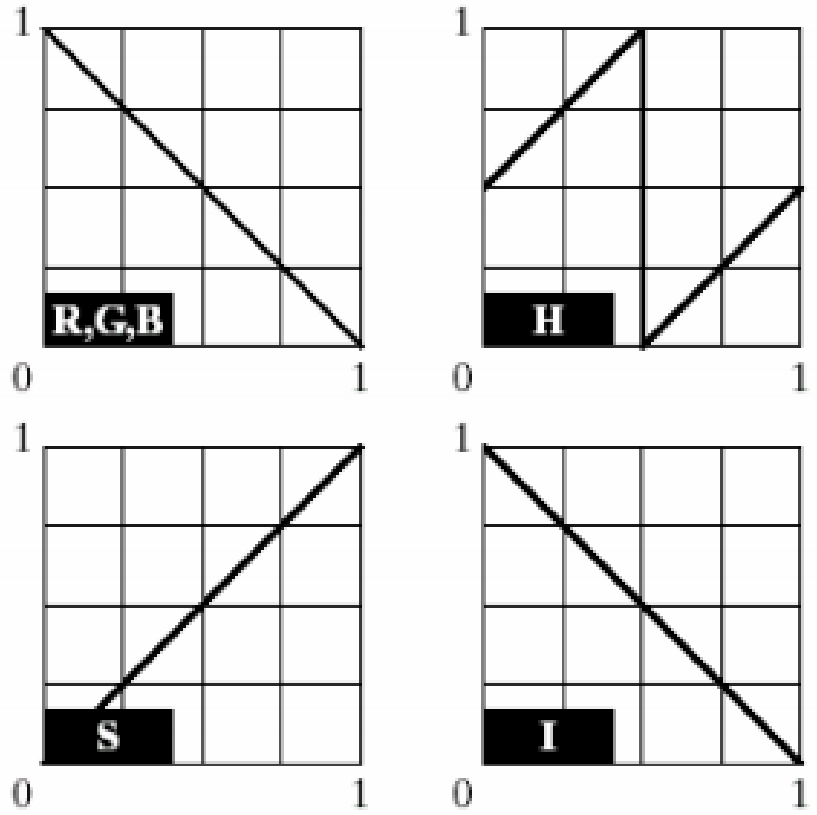




컬러 영상에서의 보색만들기

- 보색으로 만들기
  - 보색: 컬러원 상에서 서로 맞은 편에 있는 색상
  - RGB에서는 모든 채널을 1-채널값으로 변환
  - HSV에서는 H채널은 90을 더하고, S채널은 그대로, V 채널은 256-I 변환.

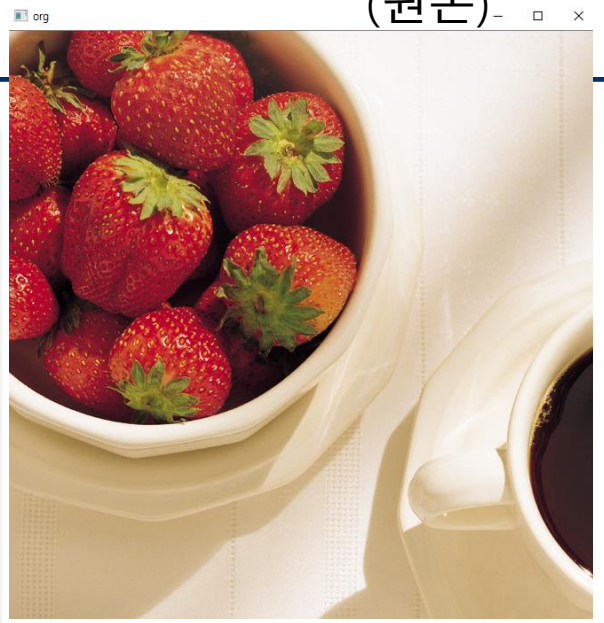




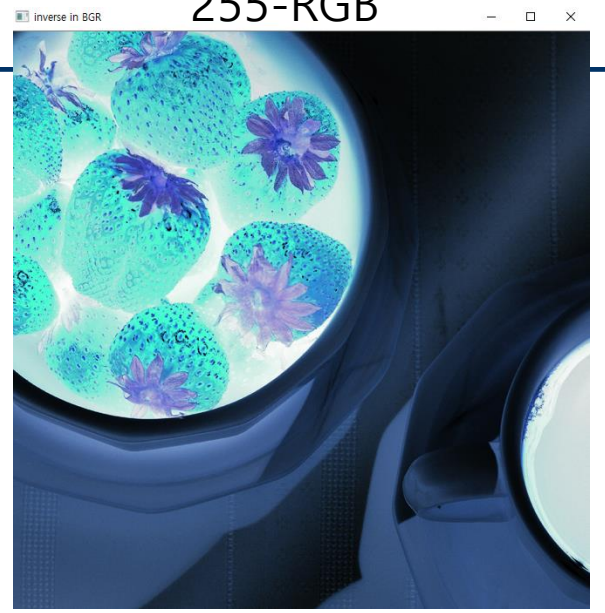
보색 만들기 예제

```
Mat hsv, blurr, enhanced;
imshow("org", img);

cvtColor(img, hsv, COLOR_BGR2HSV);

std::vector<Mat> channels(3);
split(img, channels);

unsigned char color_comp_tab[256];
get_color_comp_tab(color_comp_tab);
Mat mapping(1, 256, CV_8UC1, color_comp_tab);
LUT(channels[0], mapping, channels[0]); // H
//channels[0] =  (channels[0]+90) % 180;
channels[2] = 255 - channels[2]; // V

merge(channels, enhanced);
cvtColor(enhanced, enhanced, COLOR_HSV2BGR);
imshow("inverse", enhanced);
```

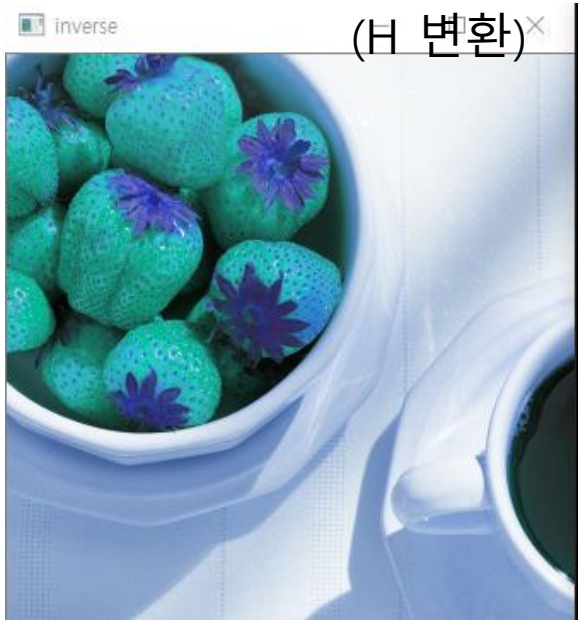
(원본)



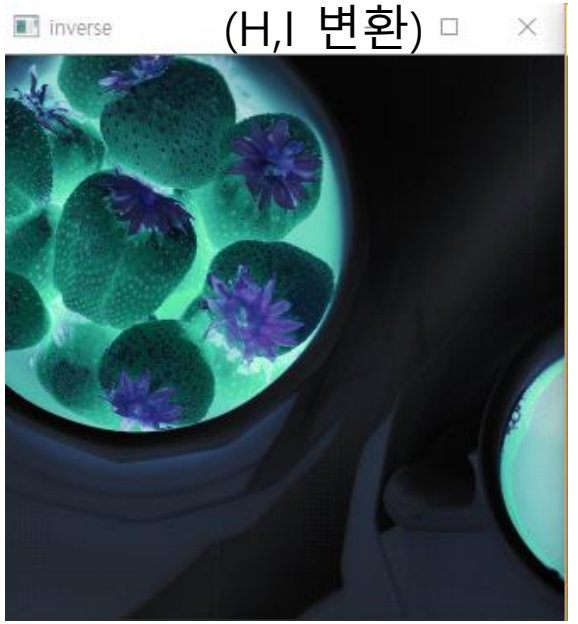
255-RGB



(H 변환)

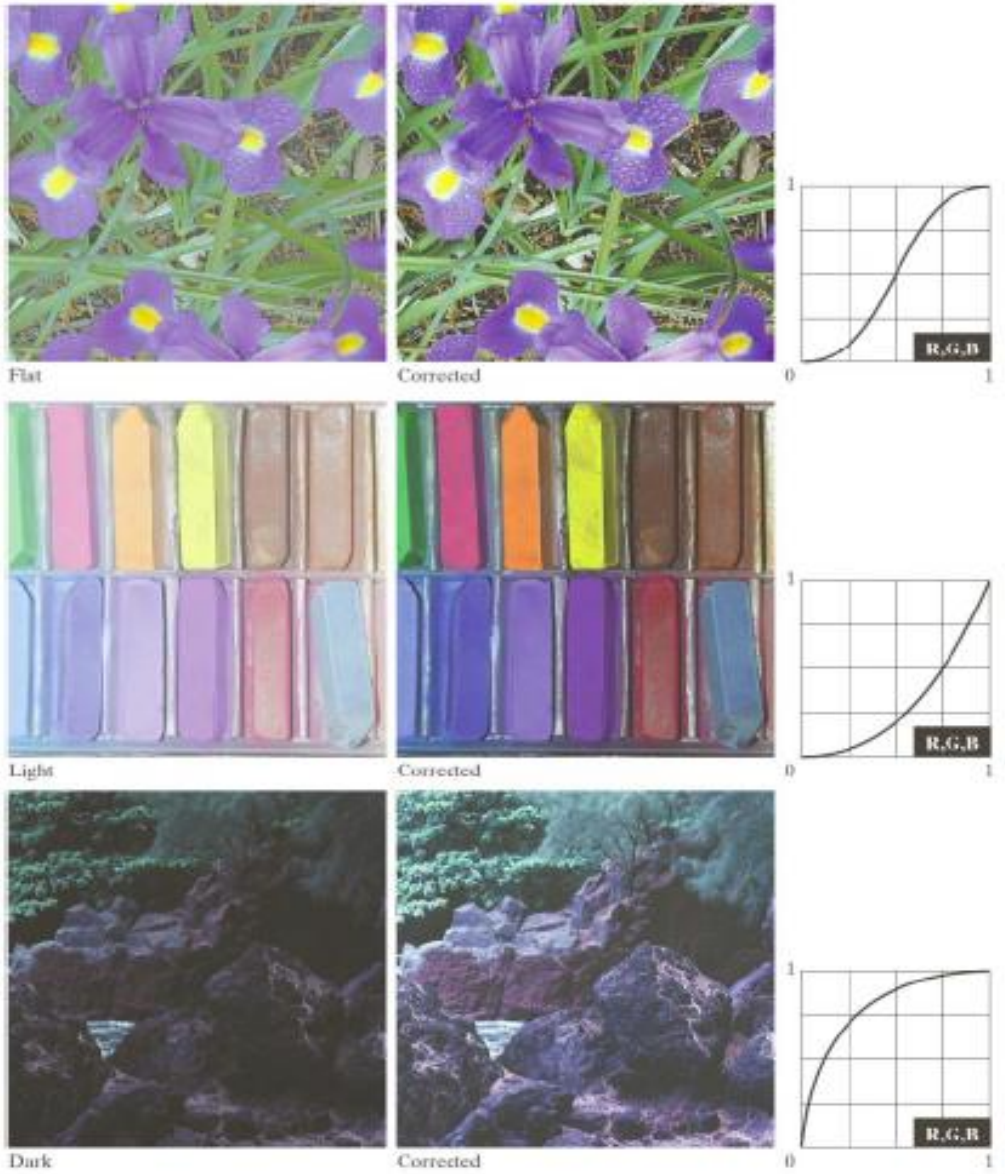


(H,I 변환)



컬러 영상에서의 보정

- 색조 및 컬러 보정
  - 정의
    - 명도와 콘트라스트를 실험에 의해 조절
    - S형 곡선: 콘트라스트를 높이는 데에 이상적
    - 또는 명암 영역을 어둡게 할 수도 있고, 지나친 콘트라스트를 보정할 수도 있음
  - 색조 보정 판단기준
    - 백색 영역을 통해
    - 피부색을 통해

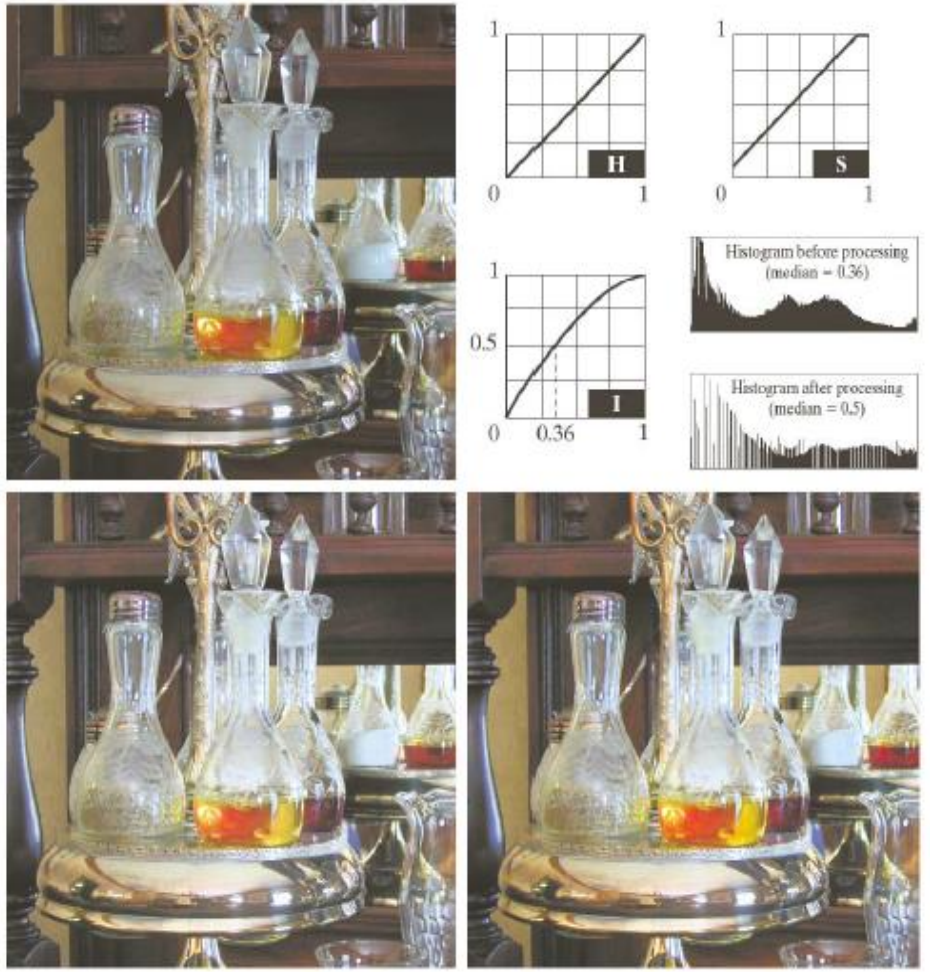


컬러교정 (gamma 변환)

```
void color_correction(char *name) {
    Mat img = imread(name, 1);
    uchar gamma_table[256];
    get_gamma_LUT(gamma_table,2.5);
    Mat gamma_mat(1, 256, CV_8UC1, gamma_table);

    vector<Mat> channels;
    split(img, channels);
    for (int i = 0; i < 3; i++)
        LUT(channels[i], gamma_mat, channels[i]);

    Mat res;
    merge(channels,res);

    imshow("src", img);
    imshow("res", res);
    waitKey(0);
}
```

컬러 영상에서의 영상 보정

- 히스토그램 평활화
  - 컬러 자체는 변하지 않으면서 컬러 농도를 균등하게 펼쳐야 함
  - HSI 공간이 이상적
  - I성분에 대한 히스토그램 평활화 + 채도 성분 증가



- 스무딩
  - RGB공간에서 각 컬러를 별도로 스무딩
  - HSI공간에서는 I 채널만 스무딩함

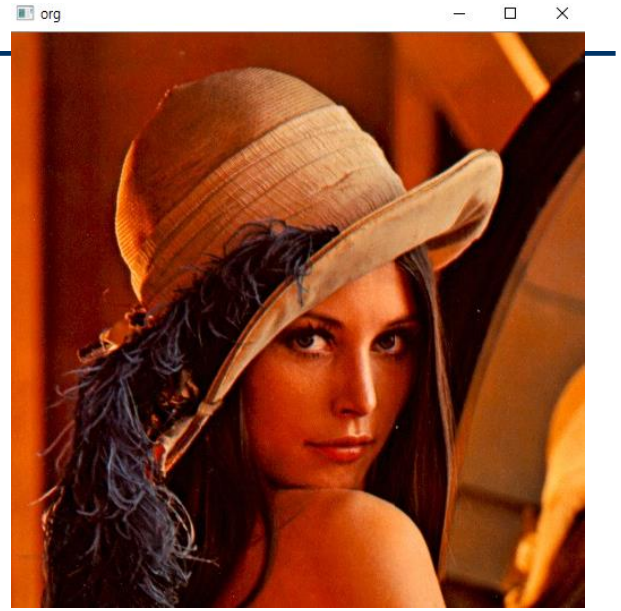
- 샤프닝
  - RGB에서는 각 컬러를 샤프닝
  - HSI 공간의 경우 I채널만을 샤프닝
  - Laplacian 변환후 그 결과를 원 영상에 add함

HSV에서 V를 히스토그램 평활화

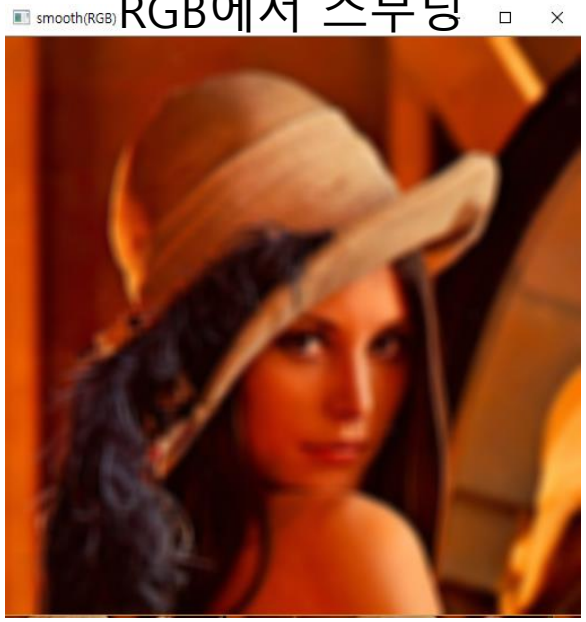
```
Mat hsv, enhanced;
imshow("org", img);

cvtColor(img, hsv, COLOR_BGR2HSV);
std::vector<Mat> channel(3);
split(hsv, channel);
equalizeHist(channel[2], channel[2]);
merge(channel, enhanced);

cvtColor(enhanced, enhanced, COLOR_HSV2BGR);
imshow("enhanced", enhanced);
```

HSI에서 V 채널만 smothing

```
Mat img = imread( “lenna.tif” ,1);

imshow("org", img);

blur(img, smooth, Size(11, 11));
imshow("smooth (RGB)", smooth);

cvtColor(img, hsv, COLOR_BGR2HSV);
std::vector<Mat> channel(3);
split(hsv, channel);

blur(channel[2], channel[2], Size(11, 11));

merge(channel, hsv);
cvtColor(hsv, img, COLOR_HSV2BGR);
imshow("smoothing(V ch)", img);
```



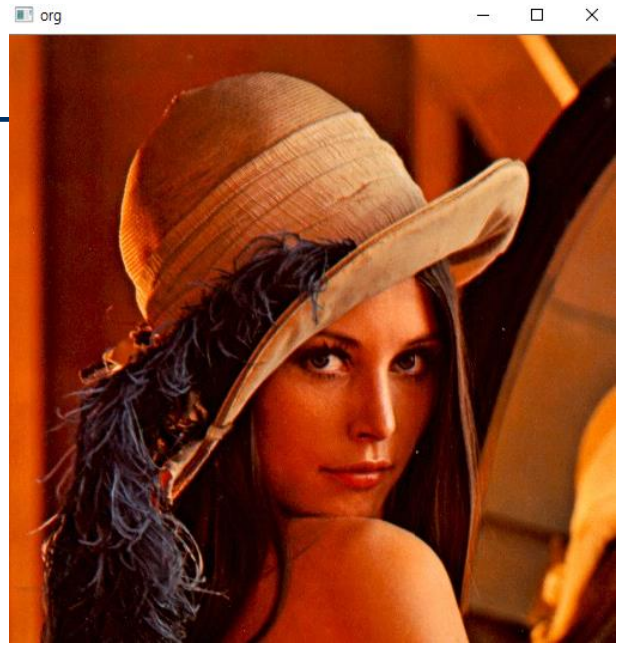
RGB에서 스무딩

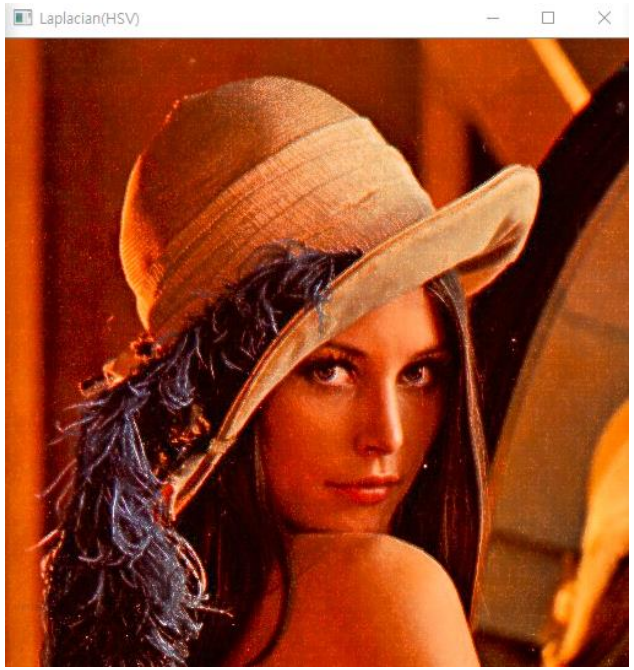


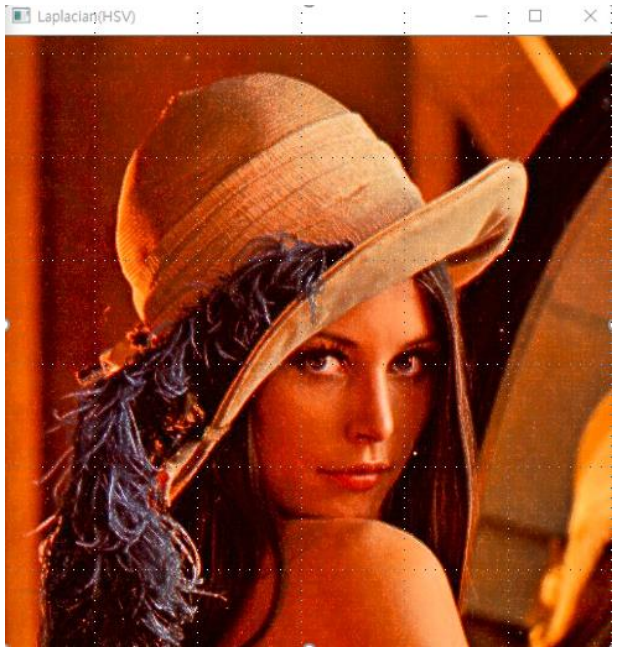
HSI에서 I채널만 스무딩

HSI에서 I채널 Lapalician 변환

```
Mat img = imread(name, 1);
vector<Mat> rgb_ch(3);
split(img, rgb_ch);
for (int i = 0; i < 3; i++) {
    Laplacian(rgb_ch[i], filtered, CV_16S);
    convertScaleAbs(filtered, filtered);
    rgb_ch[i] = rgb_ch[i] + 1.5* filtered;
}
merge(rgb_ch, rgb_filtered);
imshow("Laplacian (RGB)", rgb_filtered);

cvtColor(img, hsv, COLOR_BGR2HSV);
std::vector<Mat> channel(3);
split(hsv, channel);
Laplacian(channel[2], filtered, CV_16S);
convertScaleAbs(filtered, filtered);
channel[2] = channel[2] + 1.5* filtered;
merge(channel, hsv);
cvtColor(hsv, img, COLOR_HSV2BGR);
imshow("Laplacian(HSV)", img);
```







THRESHOLDING AND BINARIZATION

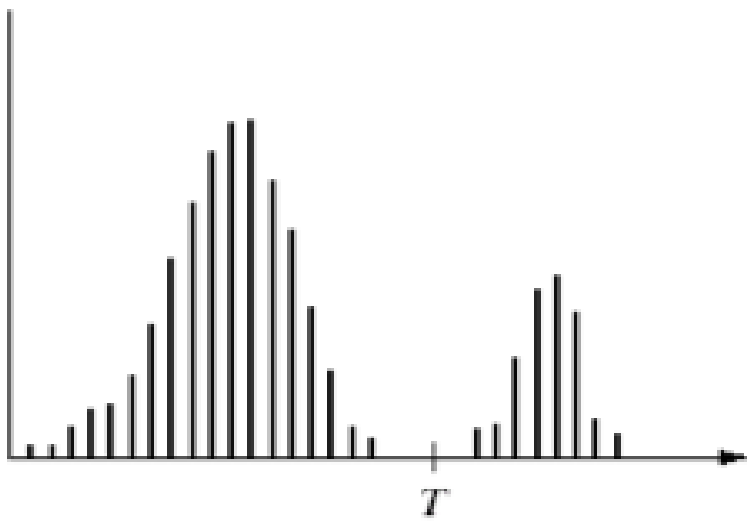
이진화 :Threshold

- 이진 영상
  - 영상 전체의 픽셀 값이 두가지만 존재 (0과 1)
  - OpenCV에서는 채널 타입을 Grayscale 영상과 같은 CV_8UC1 임
  - ➔ 8비트 : 0과 1대신, 0 (Black) 과 255 (White)로 구분
- 이진화의 목적
  - 다수의 computer vision 알고리즘에서 중요한 전처리 단계
  - 영상을 2개의 영역으로 구분 : 예) 전경과 배경, 물체 경계와 아닌 부분
- Thresholding을 통한 영상 이진화
  - 특정 threshold 값을 기준으로 모든 픽셀의 값을 0 또는 1 중 하나로 매핑
  - threshold 값의 결정
    - global threshold: 영상 전체에 단일 threshold 값 적용
    - adaptive threshold : 각 픽셀별로 threshold 값을 각 픽셀 주변픽셀값을 기반으로 계산

문턱치 처리

- 단일 문턱치 처리
  - 어두운 배경으로 밝은 객체로 구성된 영상이 있는 경우
  - 객체를 추출하는 방법은 문턱치를 활용하는 것임

$$g(x,y) = \begin{cases} 1 & f(x,y) > T \\ 0 & otherwise \end{cases}$$



Single threshold

  - 전역적 문턱치: 영상 전체에서 문턱치 값이 같을 경우
  - 가변적 문턱치: 영역에 따라 문턱치 값이 다를 경우

OpenCV threshold 함수

- Threshold (input_image, output_image, threshold_value, max_val, threshold_type)
  - threshold_value: 이 값을 기준으로 영상을 이진화를 시킨다.
    입력 픽셀이 이 값 미만이면 0 이 되고, 이상이면 max_val(255)가 된다.
  - max_valu : 입력 픽셀이 threshold 이상일때 가지게 되는 값 (255)
  - threshold_type : 7가지 종류
    - THRESHOLD_BINARY : 대표적인 방법
    - THRESHOLD_BINARY_INV : THRESHLD_BINARY 결과를 반전시킴
    - THRESHOLD_TRUNC : 입력값이 threshold_value를 초과하면 threshold_value로 만들고, 그 이하는 원래값을 그대로 유지함
    - THRESHOLD_TOZERO: 입력값이 threshold_value 이하이면 0으로 만들고, 초과이면 그대로 원래값을 유지함
    - THRESHOLD_TOZERO_INV : THRESHOLD_TOZERO를 반전시킴
    - THRESHOLD_OTSU : 영상 전체 히스토그램에서 두개의 peak를 구분하는 threshold값을 찾음

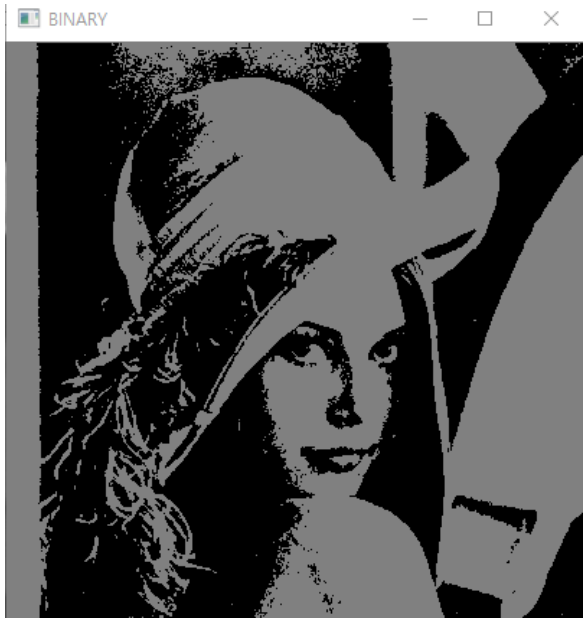
Threshold 예제

입력 ORG

- threshold 값 조정
  - 100, 120, 150, 180
- max_value 조정
  - 255, 128

threshold=100

threshold=120

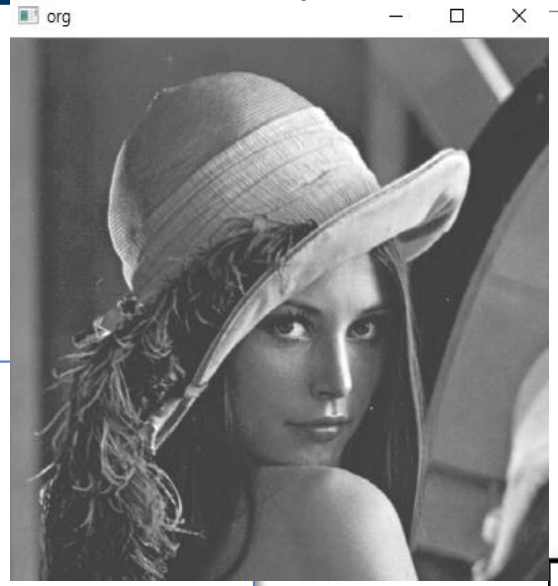


Threshold 적용 예제

threshold=100 적용

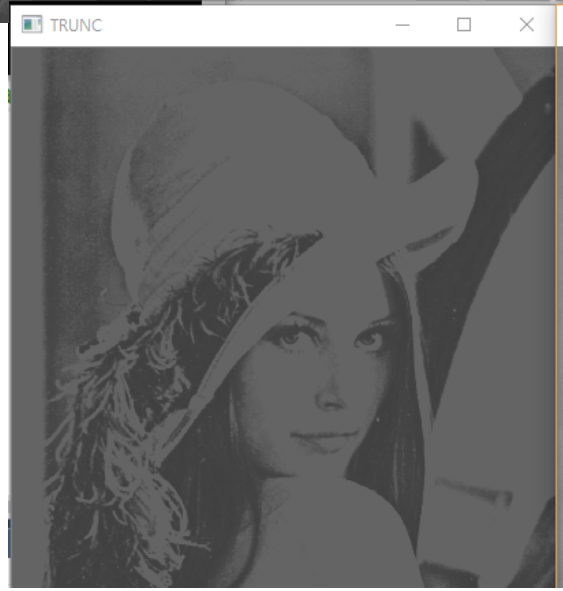
```
void thresh_test(char *name) {
   Mat img = imread(name, IMREAD_GRAYSCALE);
   Mat dst;

   imshow("org", img);
   threshold(img, dst, 100, 255,  THRESH_BINARY);
   imshow("BINARY", dst);
   threshold(img, dst, 100, 255, THRESH_BINARY_INV);
   imshow("INV", dst);
   threshold(img, dst, 100, 255,  THRESH_TRUNC);
   imshow("TRUNC", dst);
   threshold(img, dst, 100, 255, THRESH_TOZERO);
   imshow("TOZERO", dst);
   waitKey(0);
}
```

입력 ORG

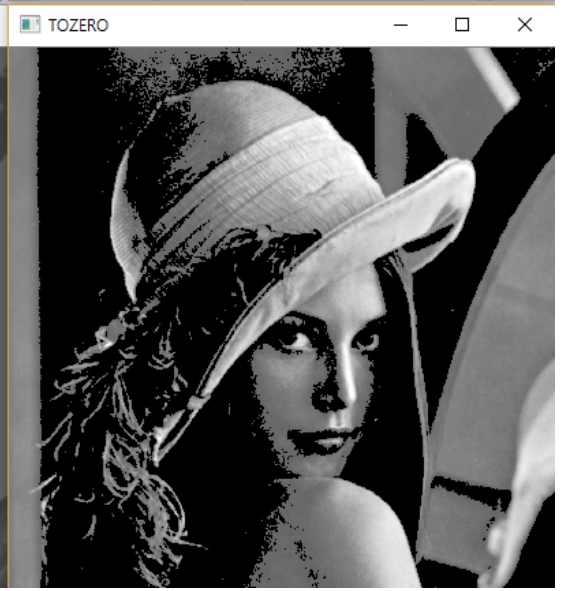


BINARY

BINARY_INV



TRUNC



TOZERO

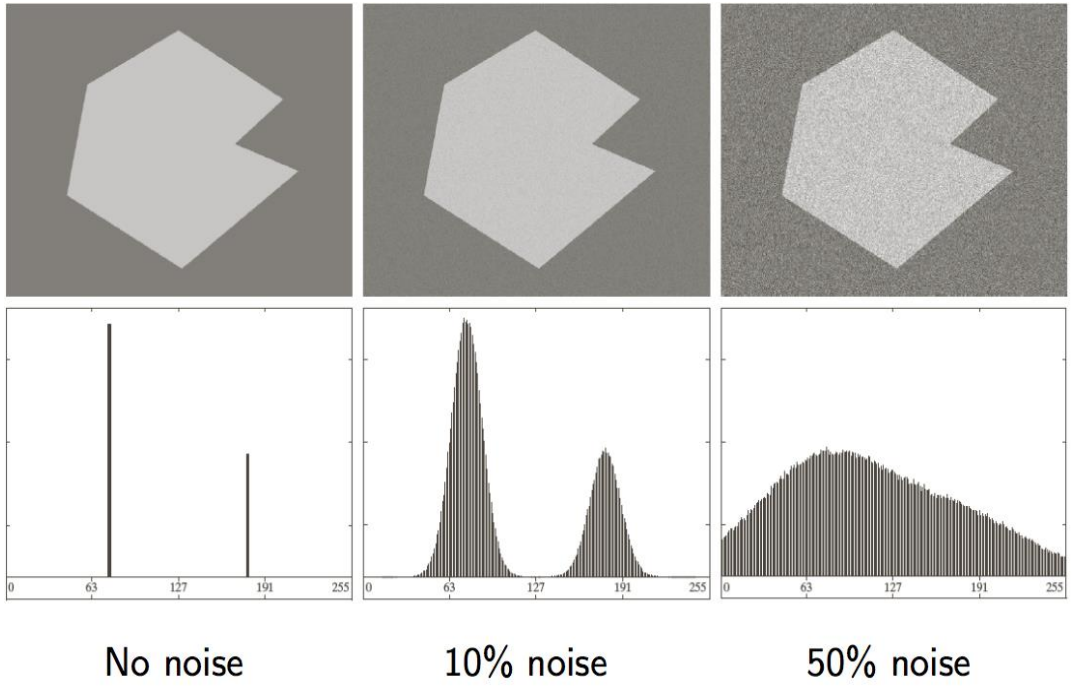
OTSU Threshold

- 이미지 전체가 두개의 밝기 성분으로 구성되었을때 매우 효과적임
- 히스토그램을 분석하여 두개의 peak 값을 잘 나눌 수 있는 문턱치 자동 계산
- threshold 함수의 threshold_type 인자에 THRESH_OTSU flag 설정으로 실행

예시) threshold(img, dst, 0, 255, THRESH_OTSU);

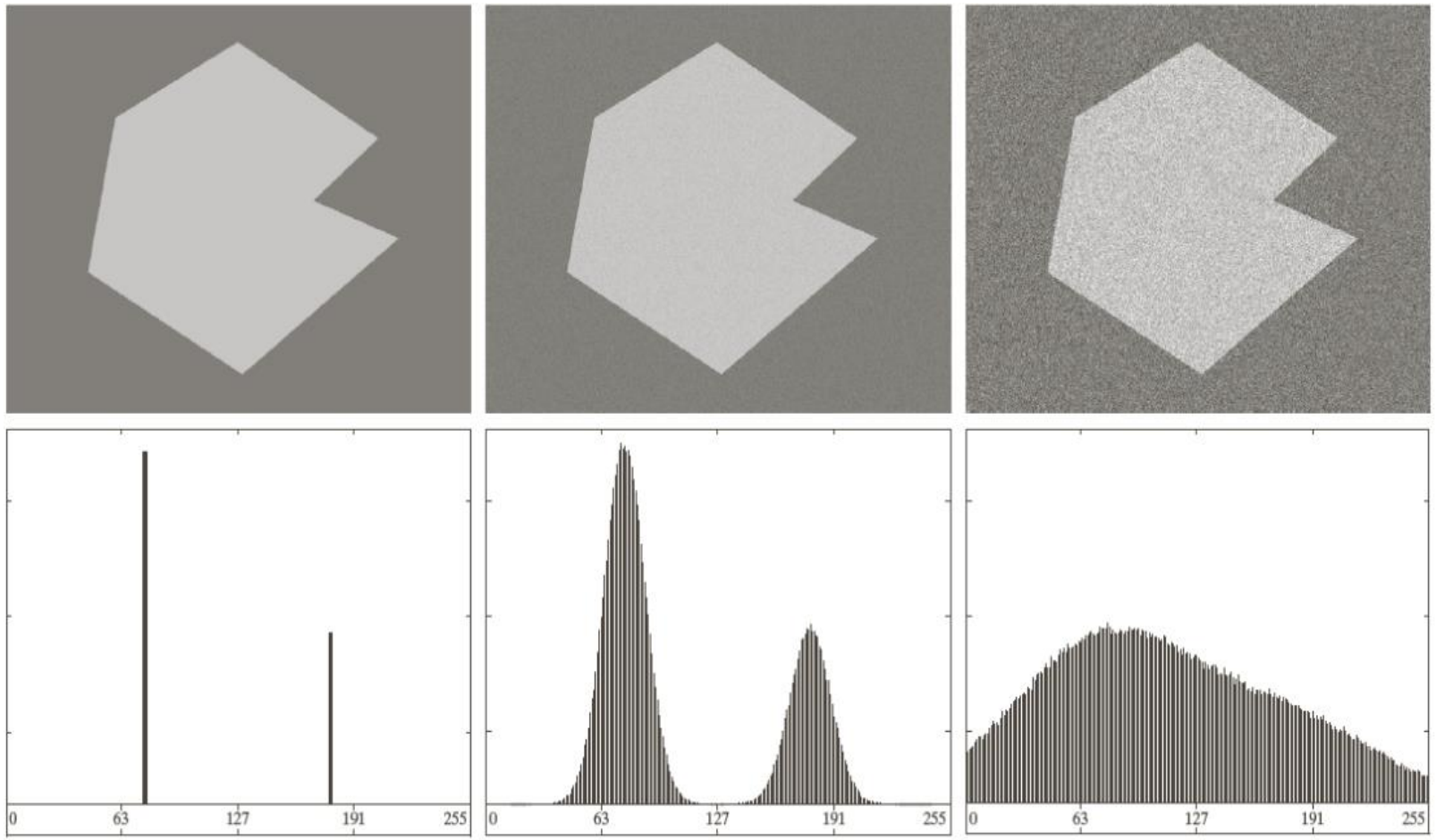
문턱치 결정 문제

- 문턱치 처리에서 고려해야 할 요소
  - 노이즈
    - 스무딩을 통해 노이즈를 제거
    - 객체들과 배경 사이의 에지 또는 그 근처의 화소들만을 고려
  - 조명 및 반사
    - 음영패턴을 고침
    - 가변적 문턱치 처리를 이용하는 방안



과제) 노이즈가 있는 이미지 이진화

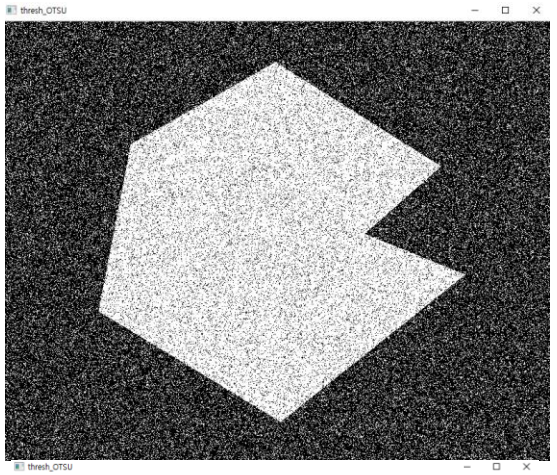
- OTSU 이진화로 나눌 수 있는 예제에 노이즈가 있는 경우 해결방안



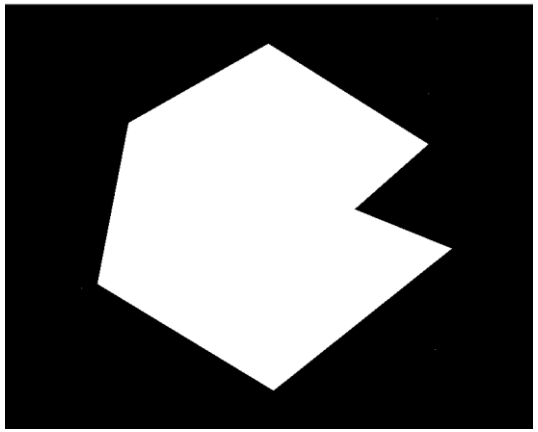

```
threshold(img, dst, 0, 255, THRESH_OTSU);
```

노이즈가 많으면 OTSU 이진화가 작동X

high noise



low noise



Adaptive Threshold

- thrshold를 pixel 마다 자동으로 따로 구함
- threshol의 기준은 이웃한 픽셀들의 (가중) 평균값-C 를 이용

adaptiveThreshold(input, output, max_value, adaptive_threshold_type, threshold_type, ksize, C)

- adaptive_threshold_type: ADAPTIVE_THRESH_MEAN_C, ADAPTIVE_THRESH_GAUSSIAN_C
- ksize : 이웃한 픽셀의 범위 결정 Size(ksize, ksize) 영역
- threshold_type
  - THRESH_BINARY : 픽셀값이 평균치 – C 보다 크면 max_val, 아니면 0
  - THRESH_BINARY_INV: 픽셀값이 평균치 – C 이하이면 max_val, 아니면 0
  - .....

예제) adaptiveThreshold(img, dst, 255, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, 19, 0);

Adaptive Threshold

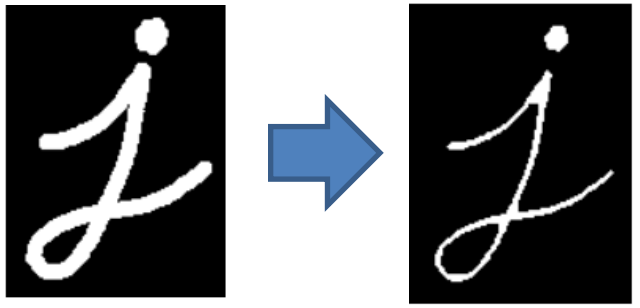
```
adaptiveThreshold(img, dst, 255,
                  ADAPTIVE_THRESH_MEAN_C,
                  THRESH_BINARY, 19, 0);

imshow("MEAN_C 19", dst);

adaptiveThreshold(img, dst, 255,
                  ADAPTIVE_THRESH_GAUSSIAN_C,
                  THRESH_BINARY, 19, 0);

imshow("GAUSSIAN 19", dst);
```

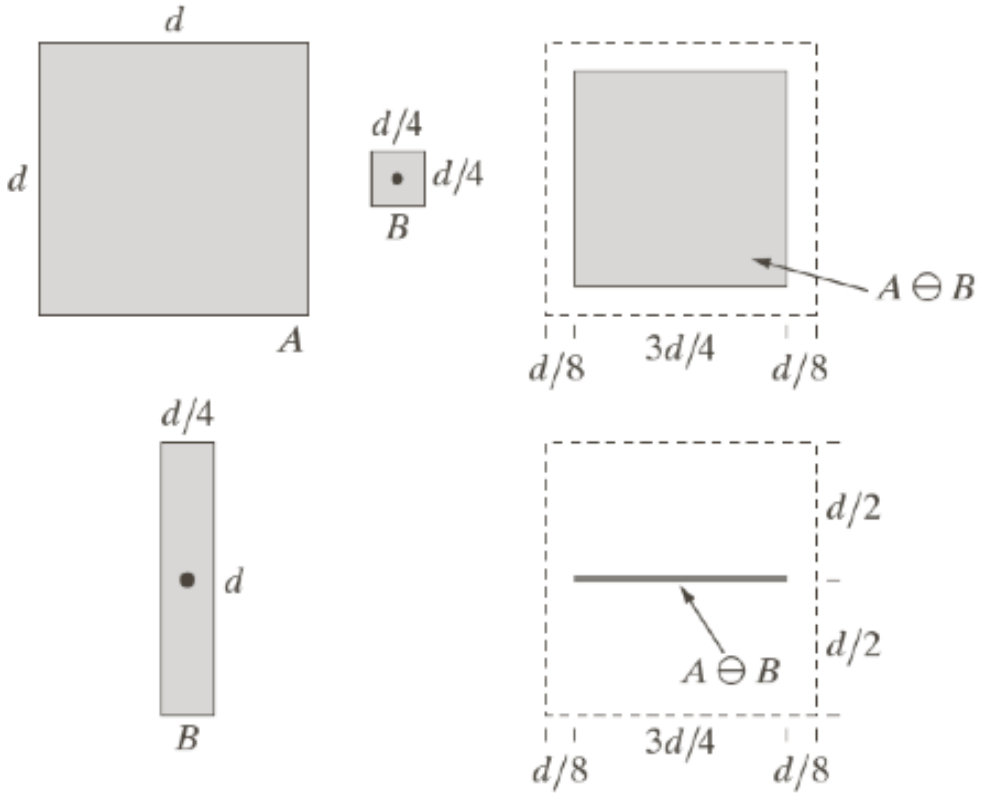
- Size가 클수록 세밀한 표현을 못함
- MEAN보다 GAUSSIAN이 더 세밀한 표현

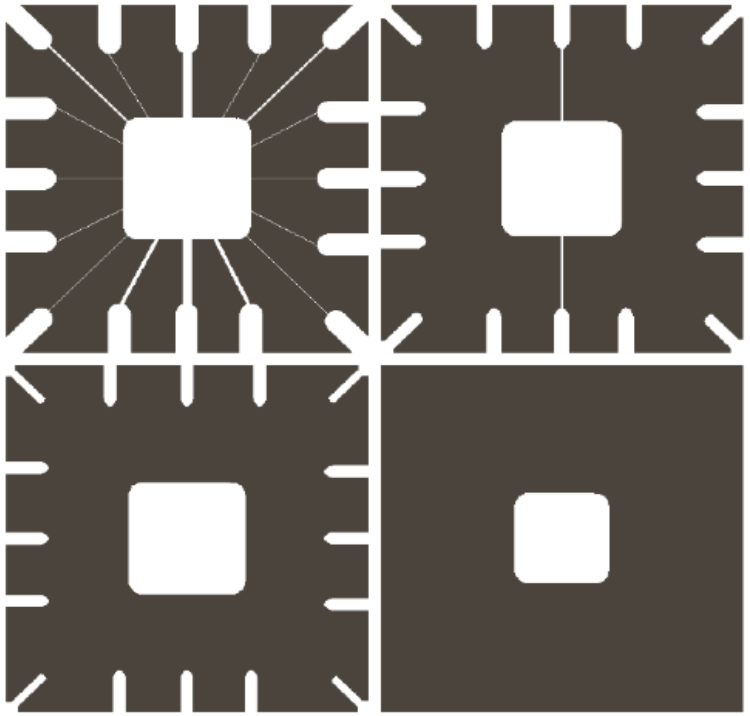
MORPHOLOGY 연산

Morphological Operation

- 침식(Erosion)
  - 커널 B에 의한 A에 대한 침식 (A ⊖ B)
  - A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).
  - 이진 영상의 객체를 축소 또는 가늘게 만들 수 있음

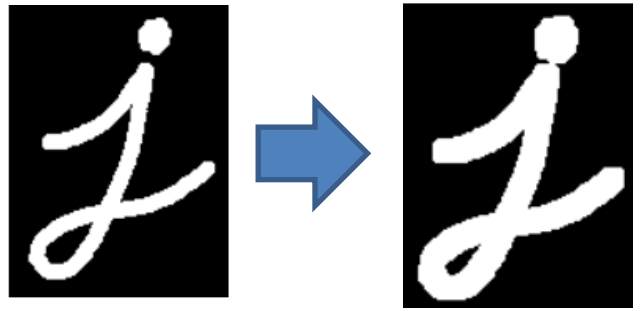


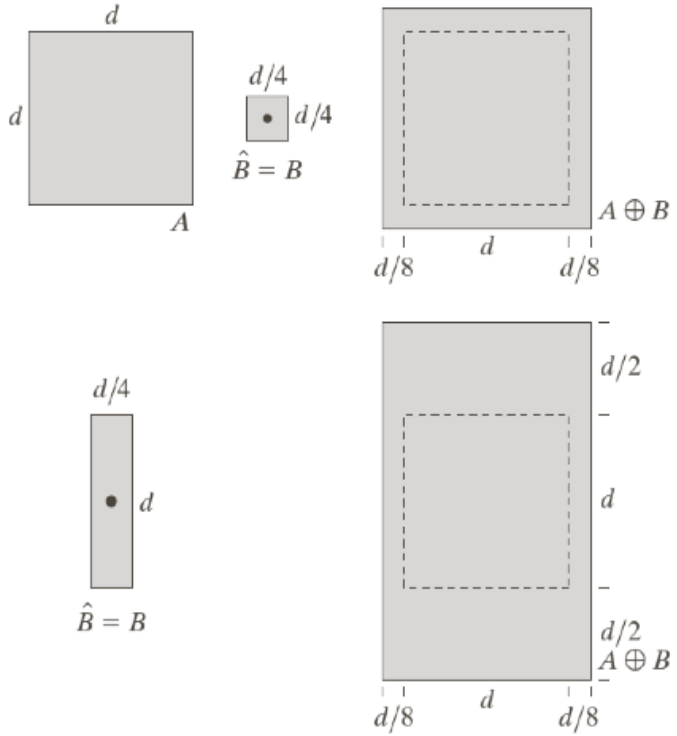


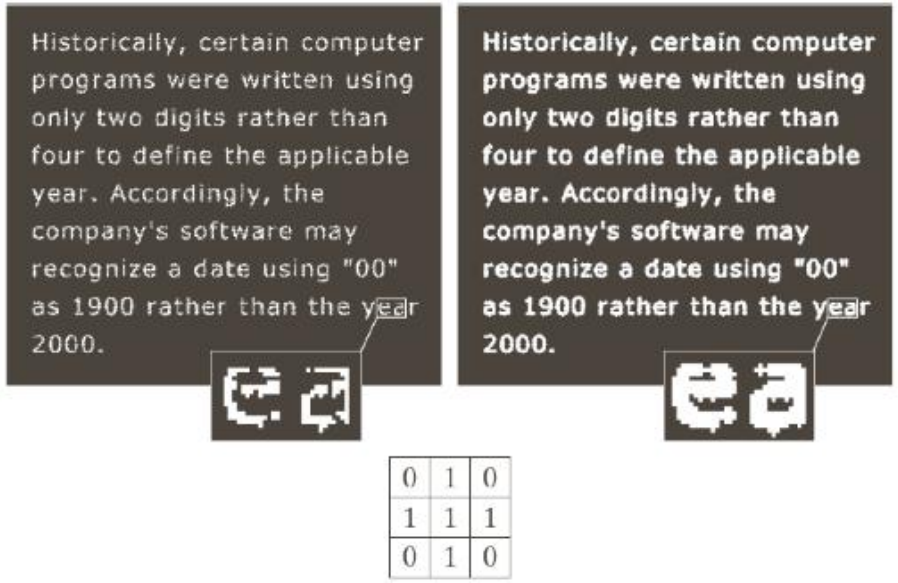


Morphological Operation

- 팽창(Dilation)
  - 커널 B에 의한 A에 대한 팽창 ($A \oplus B$)
  - just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'.
  - 이진 영상에서 객체들을 커지게 혹은 두꺼워지게 만듬







OpenCV에서 침식(Erosion)과 팽창(Dilation) 함수

- dilate (src, dst, kernel, anchor, iterations, borderType, borderValue)
- erode(src, dst, kernel, anchor, iterations, borderType, borderValue)
  - kernel: structuring element
  - anchor: position of the anchor within the element
  - iteration: number of times dilation/erosion is applied

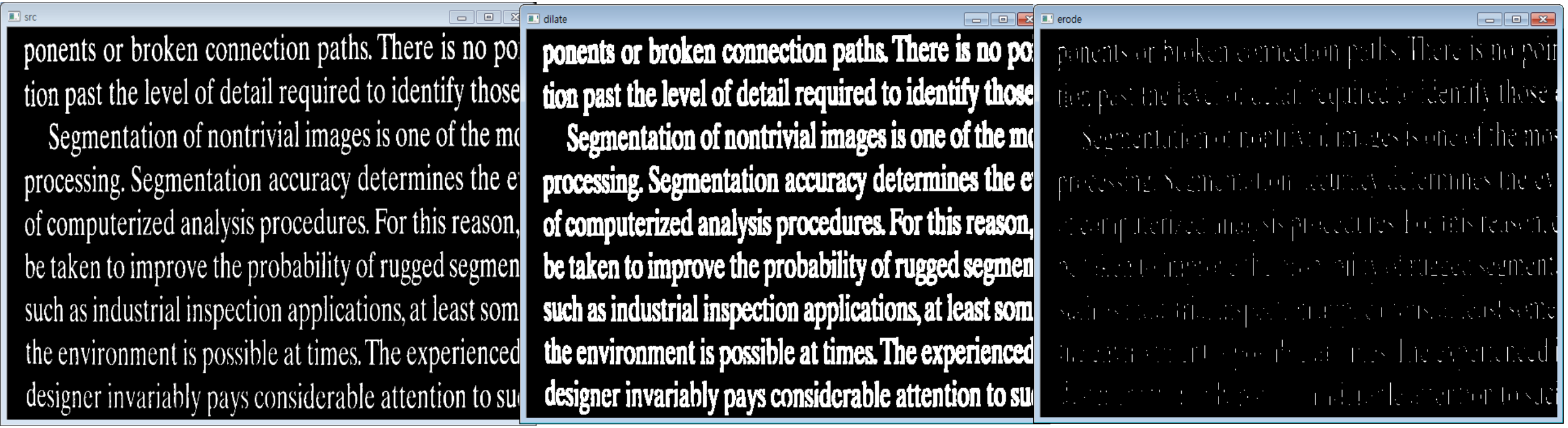
```
Mat src = imread("morph1.tif");
Mat dst;

int n = 3;
int element_shape = MORPH_RECT;
Mat element = getStructuringElement(element_shape, Size(n, n));

double thresh = 100, maxval = 255;
int threshType = THRESH_BINARY;
threshold(src, dst, thresh, maxval, threshType);

Mat resDilate, resErode;
dilate(dst, resDilate, element, Point(-1, -1), 3);
erode(dst, resErode, element, Point(-1, -1), 3);
```

Morphological Operation

원본

팽창(Dilation)

침식(Erosion)

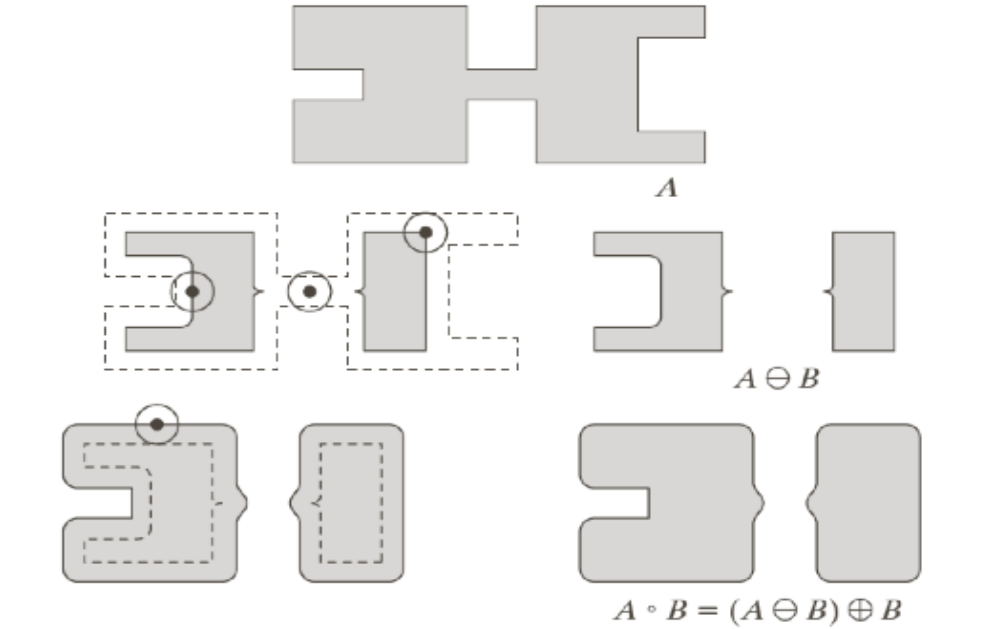


Morphological Operation

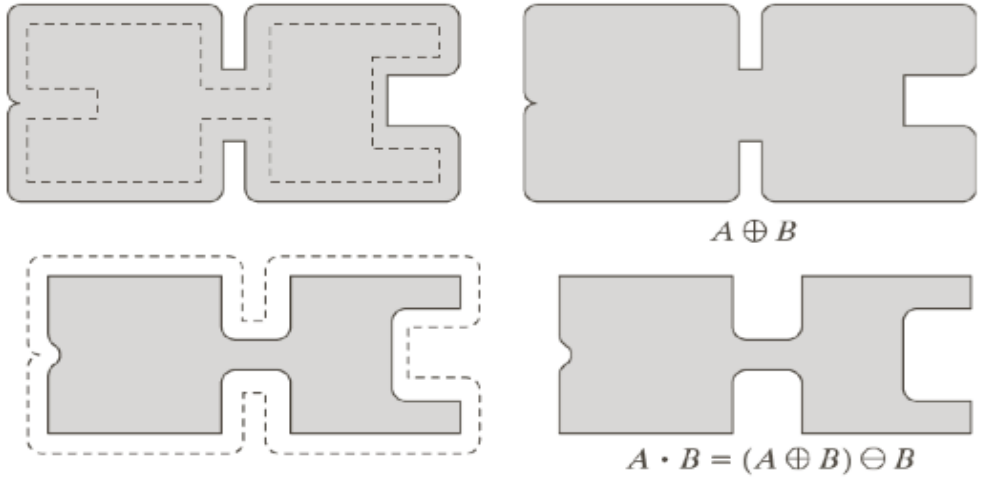
- Opening
  - B에 의한 A의 열기는 B에 의한 A의 침식 후 그 결과를 B에 의해 팽창
  - 일반적으로 객체의 윤곽을 부드럽게 만들고, 좁은 지협을 끊고, 가느다란 돌출부를 제거
  - useful for removing noise of white dots
- Closing
  - B에 의한 A의 닫기는 B에 의해 A를 팽창시킨 후, 그 결과를 B로 침식
  - 닫기도 윤곽을 부드럽게 만드는 경향이 있으나, 열기와는 반대로 일반적으로 윤곽의 좁은 끊김과 길고 가는 깊은 틈을 붙이고, 작은 홀을 제거하고, 간극을 채운다.
  - useful for removing noise of black dots

Morphological Operation

Opening
=
erosion+
dilation



Closing
=
dilation +
erosion



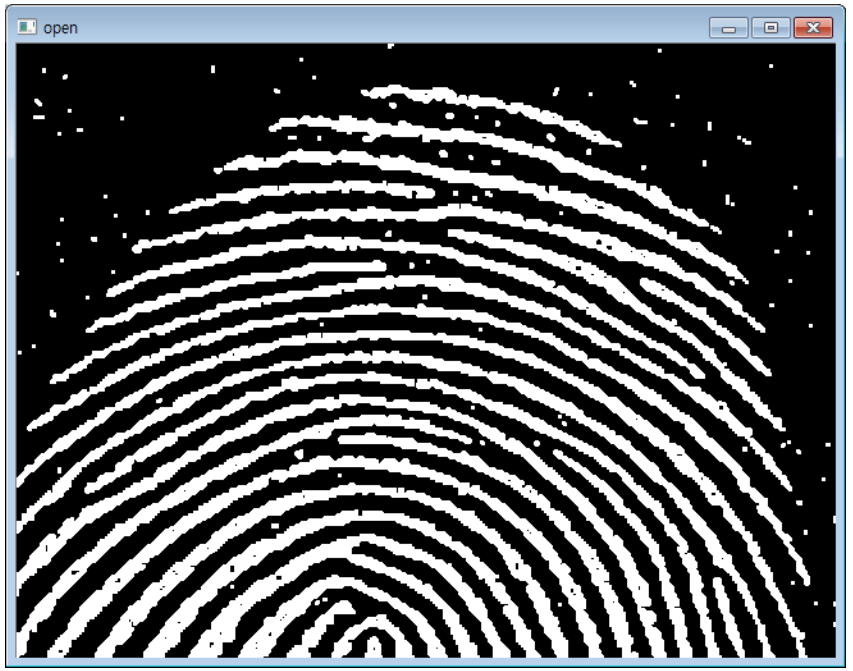
Morphological Operation

- Opening and Closing
  - morphologyEx(src, dst, op, kernel, anchor, iterations, borderType, borderValue)
    - op: type of a morphological operation

```
Mat src = imread("morph3.tif");
Mat dst;

int n = 3;
int element_shape = MORPH_RECT;
Mat element = getStructuringElement(element_shape, Size(n, n));

double thresh = 100, maxval = 255;
int threshType = THRESH_BINARY;

threshold(src, dst, thresh, maxval, threshType);

Mat resOpen, resClose;
morphologyEx(dst, resOpen, CV_MOP_OPEN, element);
morphologyEx(dst, resClose, CV_MOP_CLOSE, element);
```

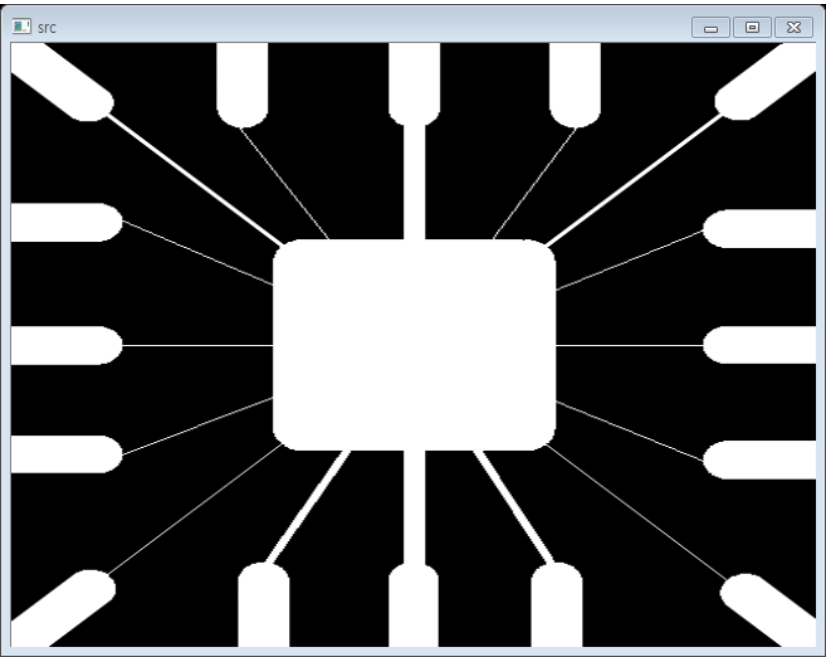
Morphological Operation

원본 Opening Closing

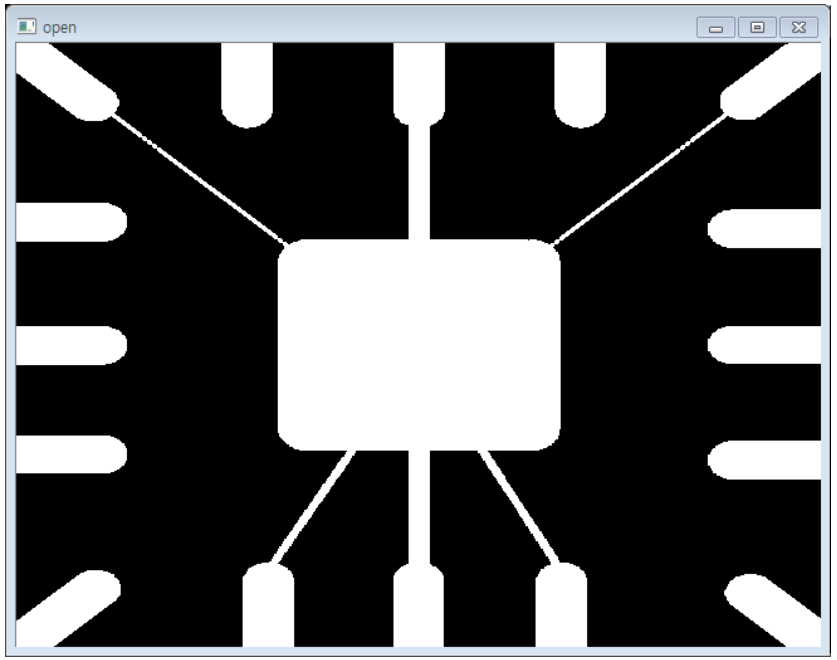


Morphological Operation

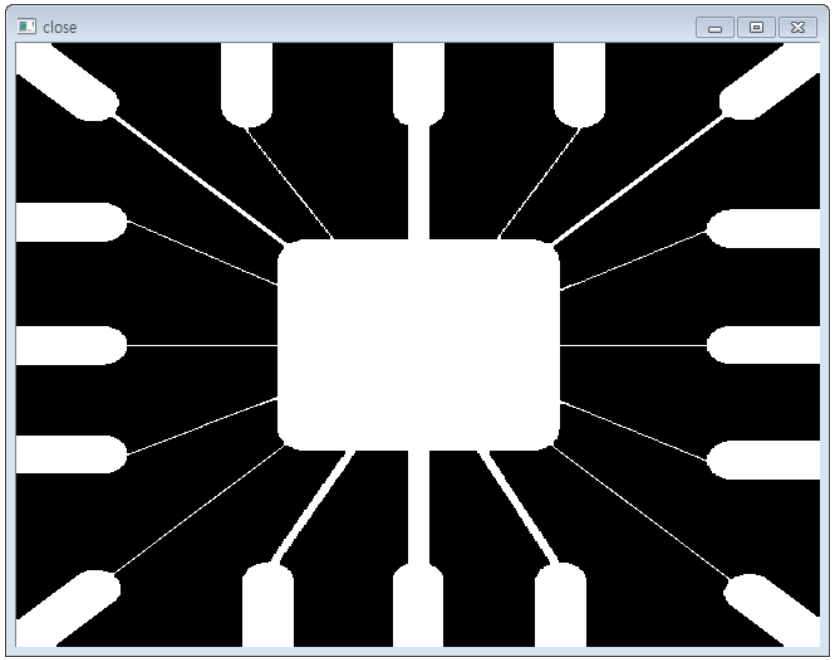
원본



Opening



Closing

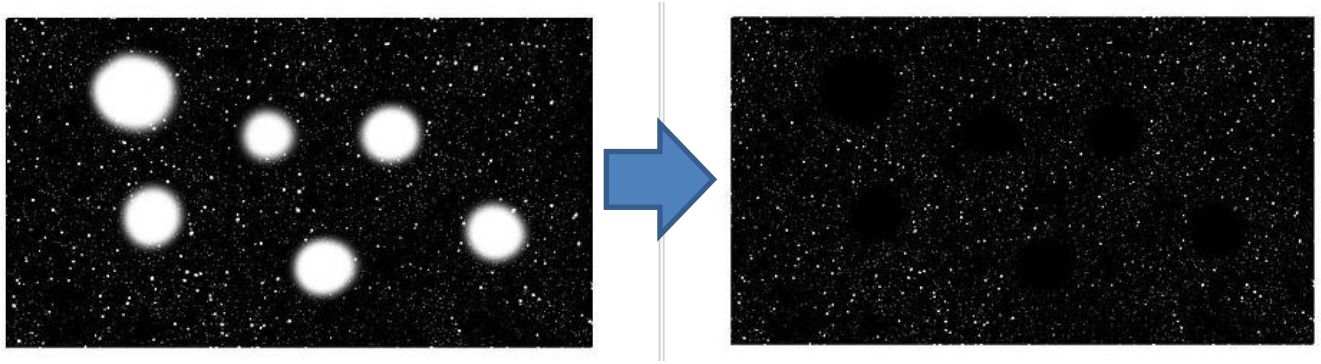


More Morphological Operations

- Morphological Gradient
  - Gradient = Dilation - Erosion
  - 윤곽선 추출에 사용

- White Top Hat
  - TopHat = SRC – OPENING
  - returns object that are
    - "**smaller**" than the structuring element, and
    - are **brighter** than their surroundings.

- Black Top Hat
  - CLOSING - SRC
  - returns objects that are
    - "smaller" than the structuring element, and and
    - are **darker** than their surroundings.



Background and Foreground Extraction

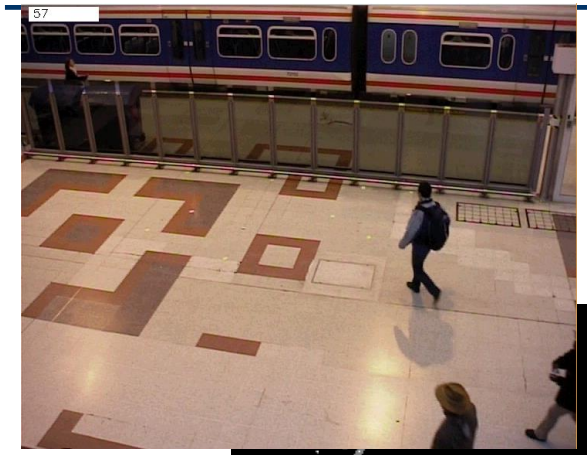
전경과 배경 분리하기

- 배경(Background)이란?
  - 영상에서 움직이지 않는 부분
  - 전경(foreground)이란 ? 입력 – 배경
- 전경 구하기
  - 배경을 알면 가능함
- 전경 구하기의 어려움
  - 배경이 시간에 따라 변함
  - 조명의 변화에 의해 배경의 색이나 밝기가 변함
  - 노이즈로 인해서 단순 배경과 입력 영상의 차이로는 전경을 구할 수 없음
  - 끊임 없이 움직이는 배경 (바람에 의한 나뭇잎의 움직임, 물결, 깃발의 흔들임)
- 해결방안
  - 배경을 하나로 정하지 않고 인접한 프레임간의 차이를 전경으로 인식
  - 여러 프레임의 차이를 누적시킴 (누적시킬 때 최근 프레임의 가중치를 더 높임)
  - Mixture of Gaussian (MOG) 모델 사용

전경과 배경 분리하기: MOG2 클래스

- createBackgroundSubtractorMOG2 (int history, float VarThreshold, bool ShadowDetection)
  - VarThreshold: Squared Mahalanobis distance to decide whether it is well described by the background model (보통 16)
  - Shadowdetection : 그림자 별도 인식 여부 (true or false)

```
VideoCapture cap("surveillance.avi");
Mat frame, fgMOG2;
Ptr<BackgroundSubtractorMOG2> pMOG2;
pMOG2 = createBackgroundSubtractorMOG2(100, 20.0, true);

while (1) {
 cap >> frame;
 if (frame.empty())
    break;
 pMOG2->apply(frame, fgMOG2, 0.1);
 //get the frame number and write it on the current frame
 imshow("Frame", frame);
 imshow("FG MOG2", fgMOG2);
}
```



pMOG2->setVarThreshold(30);

pMOG2->setVarThreshold(100);

EDGE and LINE DETECTION

What is Edges?

- 에지(edge)의 정의
  - 에지 화소: 영상 함수의 밝기가 급격하게 변하는 화소
  - 에지: 연결된 에지 화소의 집합
- 에지 타입
  - 계단 에지
    - 1화소 거리에서 이상적으로 일어나는 에지
    - 컴퓨터에서 생성된 영상에서 발생
  - 비탈 에지
    - 일반적인 디지털 영상은 무뎌지고 노이즈가 낀 에지를 가짐
    - 비탈의 경사는 에지의 무딘 정도에 반 비례
  - 지붕 에지
    - 영역을 지나는 선에 대한 모델
    - 선의 굵기와 날카로움에 의해 폭이 결정

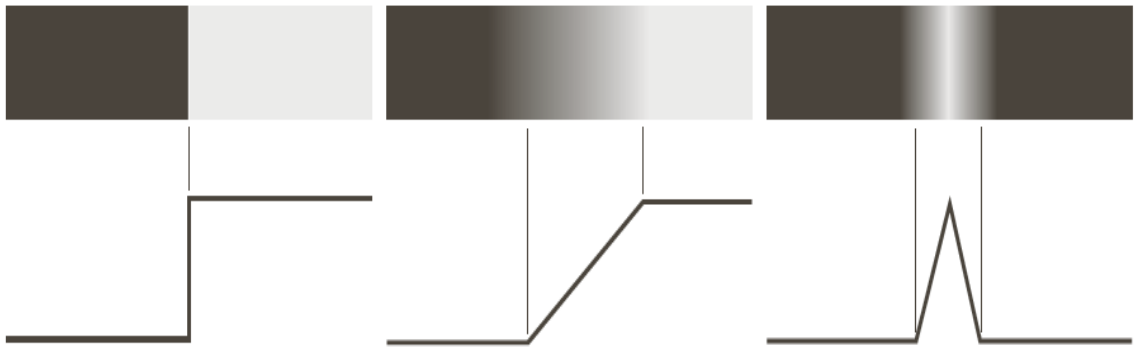
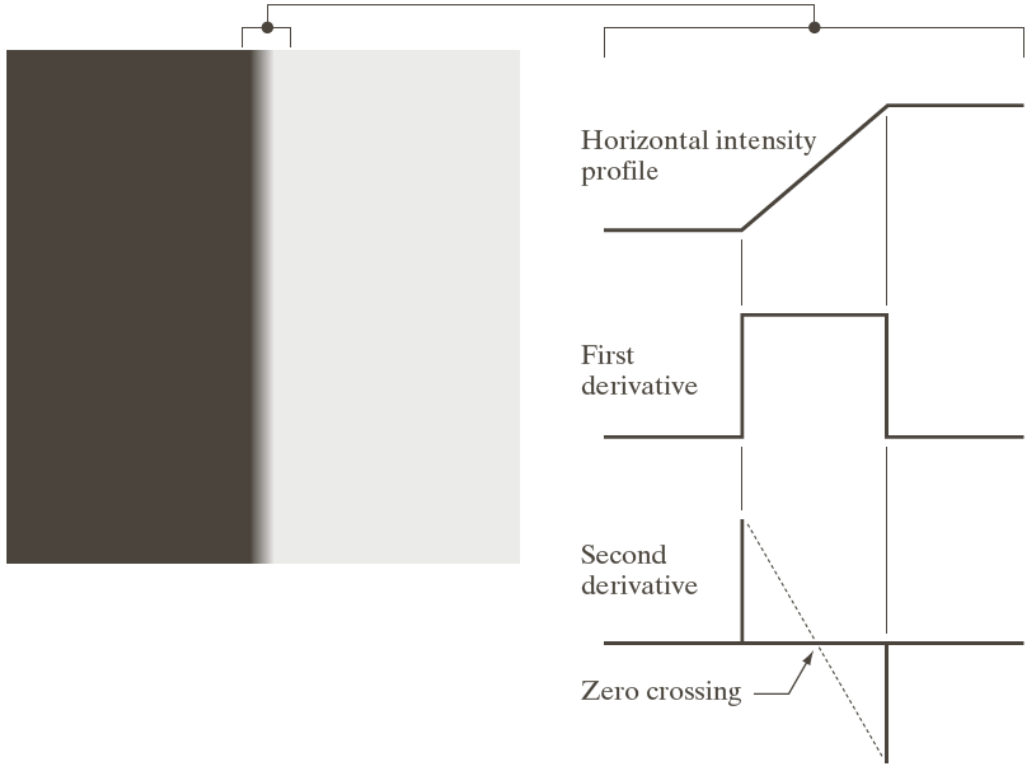


Image Derivatives

- Dirivatives in Continuous Domain
  - $\frac{dy}{dx} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- Derivatives in 1-D in Discrete Domain
  - Forward difference : $f'(x') = f(x') - f(x'+1)$
    → Filter [1, -1]
  - Backward difference: $f'(x') = f(x') - f(x'-1)$
    → Filter [-1,1]
  - Central differences: $f'(x') = \frac{1}{2}[f(x'+1) - f(x'-1)]$
    → Filter [-1/2,0,1/2]



- 1차 및 2차 미분에 의한 에지 검출
  - 1차 미분의 크기를 통해 에지의 존재 검출 가능
  - 2차 미분의 부호로 에지 화소 기준으로 밝은 쪽의 위치 파악

Image Derivatives in 2-D

- 자기 자신을 포함한 3x3 픽셀들 (8개의 이웃한 픽셀)을 사용
  - computing differences of 3 piexles is a good approximation of actual derivatives
  - averaging is a better estimate of the variations in the central pixel
- Horizontal Direction

$$[\mathbf{h}_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

  - X 방향의 difference만을 독립적으로 구함
  - 가운데 3개 pixel의 1-D Central difference의 평균을 구한 것
- Vertical Direction

$$[\mathbf{h}_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

  - Y 방향의 difference만을 독립적으로 구함
  - 가운데 3개 pixel의 1-D Central difference의 평균을 구한 것

Image derivatives Kernal with OpenCV

```
Mat get_HorizDerivKernel() {
  Mat kernel = (Mat_<float>(3, 3) << -1, 0, 1, -1, 0, 1, -1, 0, 1);
  kernel = kernel / 3.0;
  return kernel;
}

Mat get_VertDerivKernel() {
  Mat kernel = (Mat_<float>(3, 3) << -1, -1, -1, 0, 0, 0, 1, 1, 1);
  kernel = kernel / 3.0;
  return kernel;
}
```

$$[\mathbf{h}_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix};$$

$$[\mathbf{h}_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Visualizing image derivatives with OpenCV

```
Mat img = imread( “images/Fig04_house.tif” , 0);

Mat horiz_kernel = get_HorizDerivKernel();
Mat vert_kernel = get_VertDerivKernel();
Mat deriv_X, deriv_Y;

std::cout << horiz_kernel << std::endl;
std::cout << vert_kernel << std::endl;

filter2D(img, deriv_X, CV_16S, horiz_kernel);
filter2D(img, deriv_Y, CV_16S, vert_kernel);

convertScaleAbs(deriv_X, deriv_X); // 16S -> 8U
convertScaleAbs(deriv_Y, deriv_Y); // 16S -> 8U

imshow("Horiz_Derivative", deriv_X);
imshow("Vertical_Derivative", deriv_Y);
waitKey(0);
```

Image Derivatives Results

$$[\mathbf{h}_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad [\mathbf{h}_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

input original

horizontal

vertical

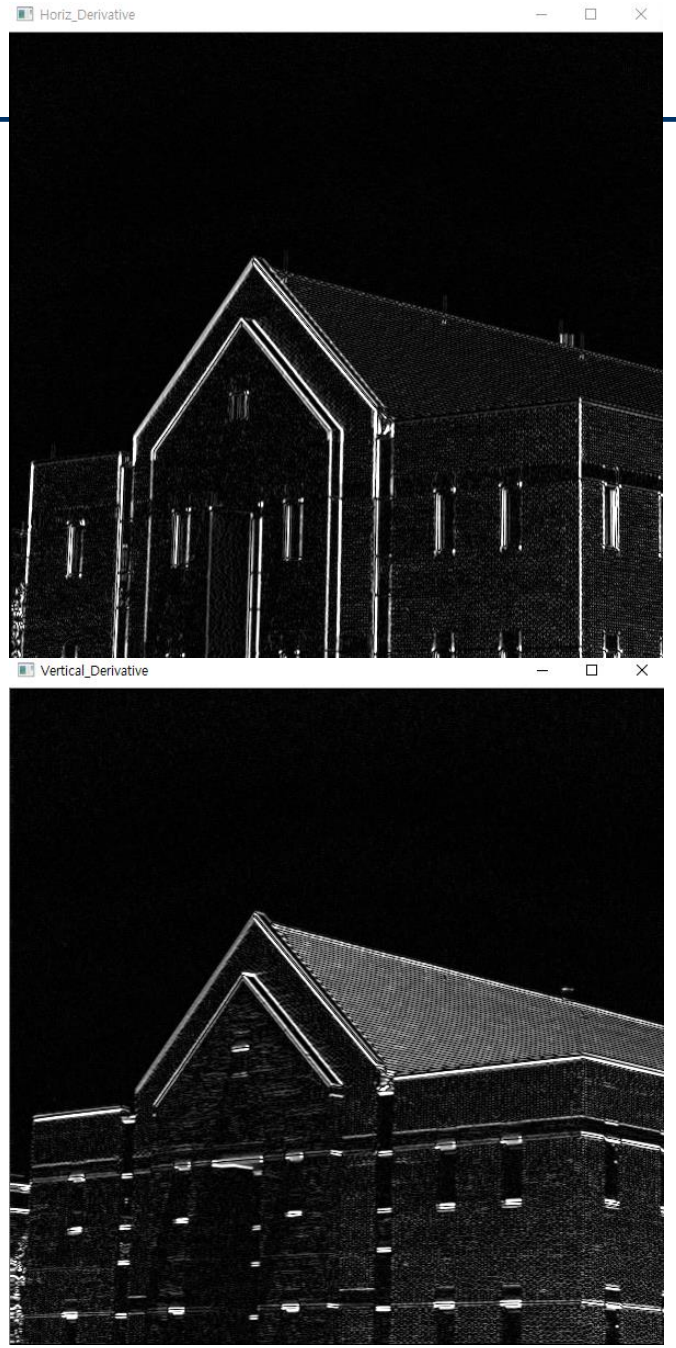
Sobel Derivative Filter

- Sobel Filter
  - Double the weight of nearer pixels to central pixel
  - 멀리 있는 pixel 값보다 가까이 있는 pixel 값이 더 중요하다는 논리
- Sobel (input, output, ddepth, dx, dy, ksize, scale, delta, border)
  - ddepth : output Mat의 type,
  - dx: order of derivatives in x direction 0,1,2
  - dy: order of derivatives in y direction: 0,1,2
  - ksize: Sobel kernel size (default=3)
  - scale: default=1.0
  - delta : default=0
  - border: default = BORDER_DEFAULT

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

```
Sobel(img, deriv_X, CV_16S, 1,0);
Sobel(img, deriv_Y, CV_16S, 0,1);
convertScaleAbs(deriv_X, deriv_X); // 16S -> 8U
convertScaleAbs(deriv_Y, deriv_Y); // 16S -> 8U
imshow( “detivatives X” , deriv_X);
imshow( “derivatives Y” , deriv_Y);
```

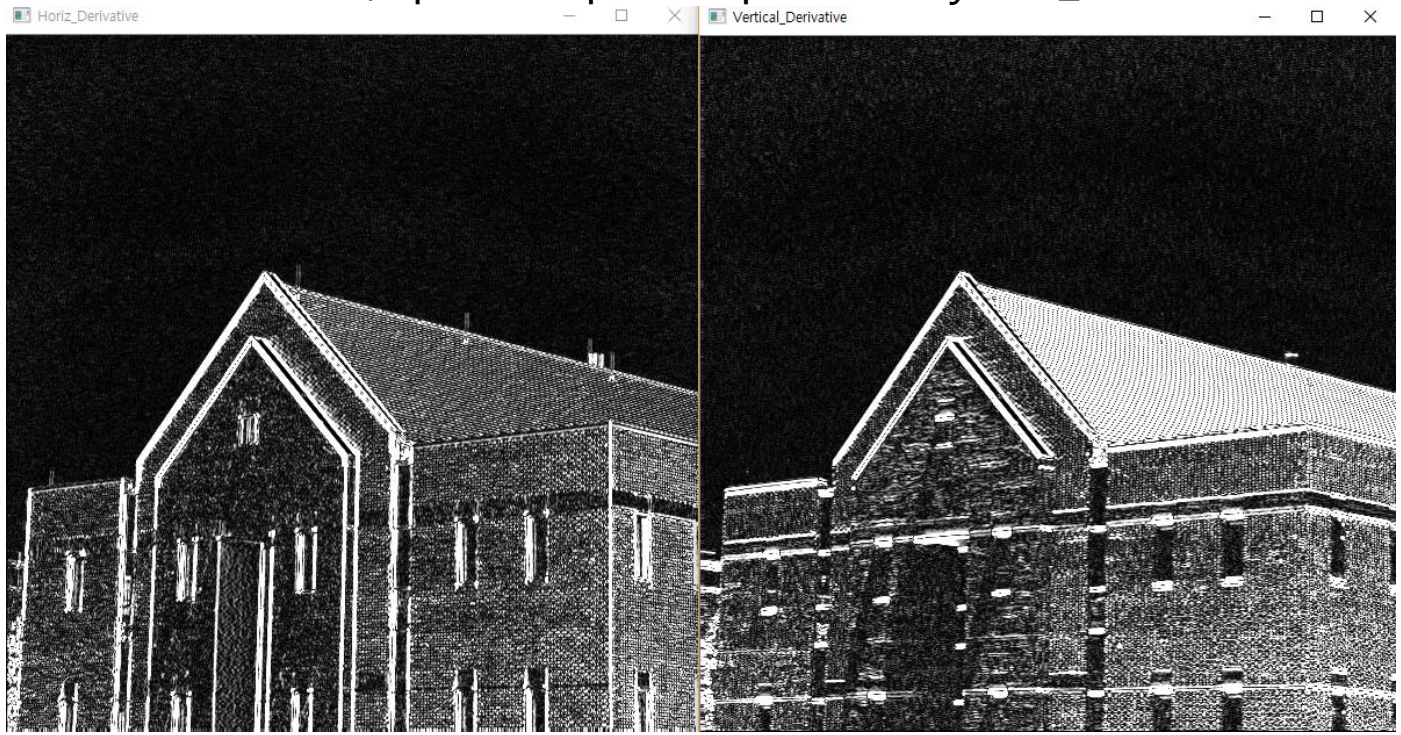


Scharr: Another Variation of the Derivative Kernel

- Scharr Kernel has different weights but, basic concept is the same as Sobel

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} \qquad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

  - Scharr(input, output, depth, dx dy scale, delta, border)
    - Same of Soble(input, output, depth, dx, dy, CV_SCHARR, scale, delta, border)



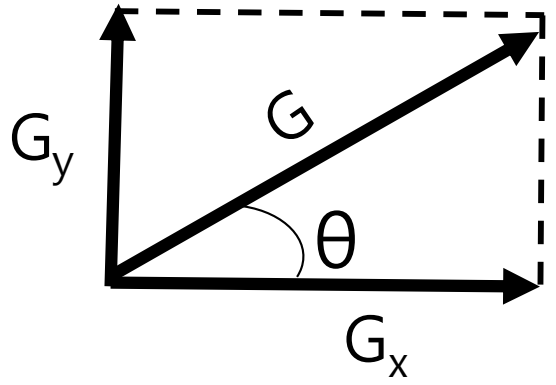
Sobel Edge Detector-1

- Edge: Region where the Changes in pixel intensities are high

➔ Derivatives can be utilized to detect abrupt changes of pixel intensity

- Derivatives along x and y direction are computed as G_x and G_y by Sobel filter
- Gradient: inverse tangent function

$$\theta = tan^{-1}(\frac{G_y}{G_x})$$



- **Gradient Magnitude**

$$|G| = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

```
Mat getGradientMag(Mat &dev_X, Mat &dev_Y) {
  Mat mag(dev_X.rows, dev_X.cols, CV_16S);
  for(int i =0; i < dev_X.rows; i++)
    for (int j = 0; j < dev_X.cols; j++)
       mag.at<short>(i, j) = abs(dev_X.at<short>(i, j)) + abs(dev_Y.at<short>(i, j));
  return mag;
}
```

Sobel edge Detector-2

- Edges are obtained by thresholding the Gradient Magnitude

```
void thresholdGradientMag(Mat &mag, Mat &edges, int th){
  edges = Mat::zeros(mag.size(), CV_8U);
  for (int i = 0; i < mag.rows; i++)
    for (int j = 0; j < mag.cols; j++)
        if (mag.at<short>(i, j) > th)
            edges.at<uchar>(i, j) = 255;
}
```

```
void Sobel_Magnitudes(char *name) {
  Mat img = imread(name, 0);
  Mat dev_X, dev_Y, edges;

  Sobel(img, dev_X, CV_16S, 1, 0);
  Sobel(img, dev_Y, CV_16S, 0, 1);
  Mat gradient_mag = getGradientMag(dev_X, dev_Y);
  thresholdGradientMag(gradient_mag, edges, 100);
  imshow("edge", edges);
  waitKey(0);
}
```

Canny Edge Detector

- Invented by John E. Canny in1986
- Canny is superior to Sobel
- Key Idea
  1. Non-Maximum Suppression: thining the edges
     - 각 픽셀의 gradient가 얻어지면 그 방향과 같은 gradient를 가진 인접 픽셀들의 gradient magnitude들과 비교하여 가장 크기가 큰 픽셀만 에지로 남긴다.
     - blur한 영역의 에지들을 제거하는 효과를 가진다.
  2. Double Threshold
     - 이미지의 노이지로 인해서 나타난 false positive들 (에지로 인식되었으나 사실은 에지가 아닌 픽셀들)을 제거함
     - 각 픽셀의 gradient magnitude가 high threshold보다 크면 strong edge로 분류한다.
     - 각 픽셀의 gradient magnitud가 low threshold 보다 작으면 제거한다.
     - 각 픽셀의 gradient magnitud가 high와 low threshold 사이에 있으면, weak edge로 분류한다. weak edge는 인접한 8개의 pixel이 strong edge일때만 최종적으로 strong edge로 분류된다.

Canny in OpenCV

- Canny (input, output, threshold1, threshold2, ksize, L2flag)
  - threshold1: low threshold
  - threshold2 : high threshold (recommended as 2* threshold1 ~ 3* threshold1)
  - ksize: size of edge detection kernel
  - L2flag: L2 norm must be used to compute the gradient magnitude, instead of the sum of absolutes

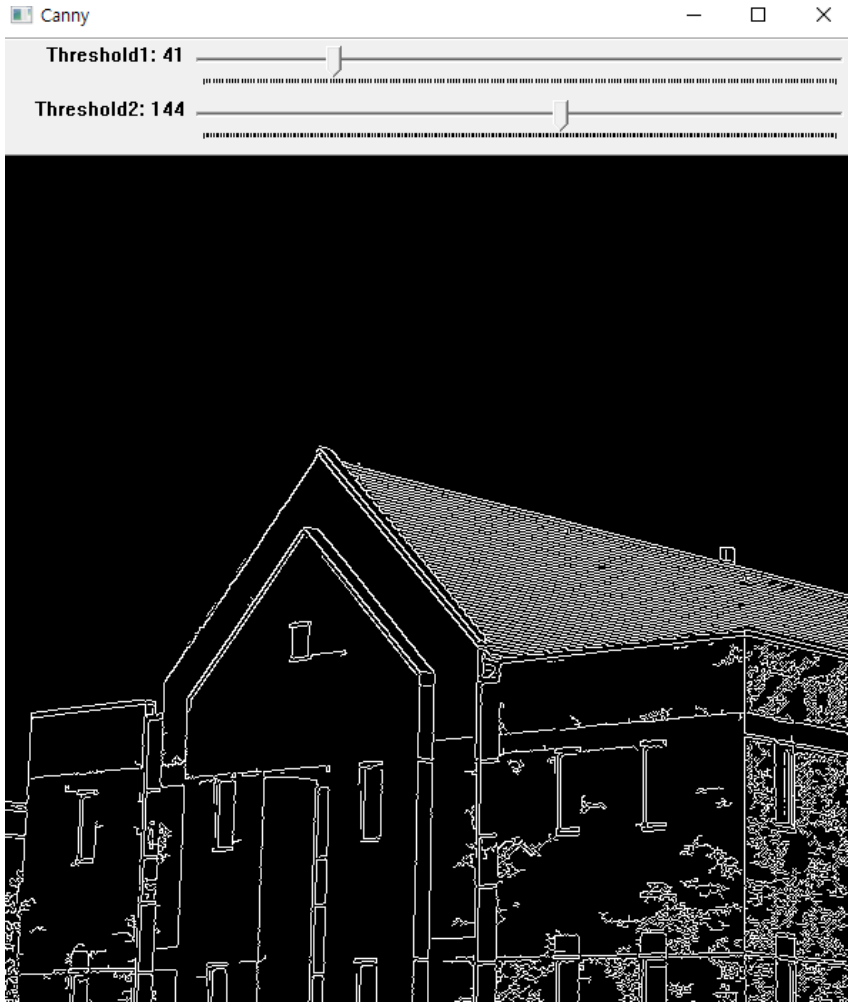
- Example)

```
Mat img = imread(name, 0);
Mat dst;
Canny(img, dst, 80, 200, 3, false);
imshow("Canny", dst);
waitKey(0);
```

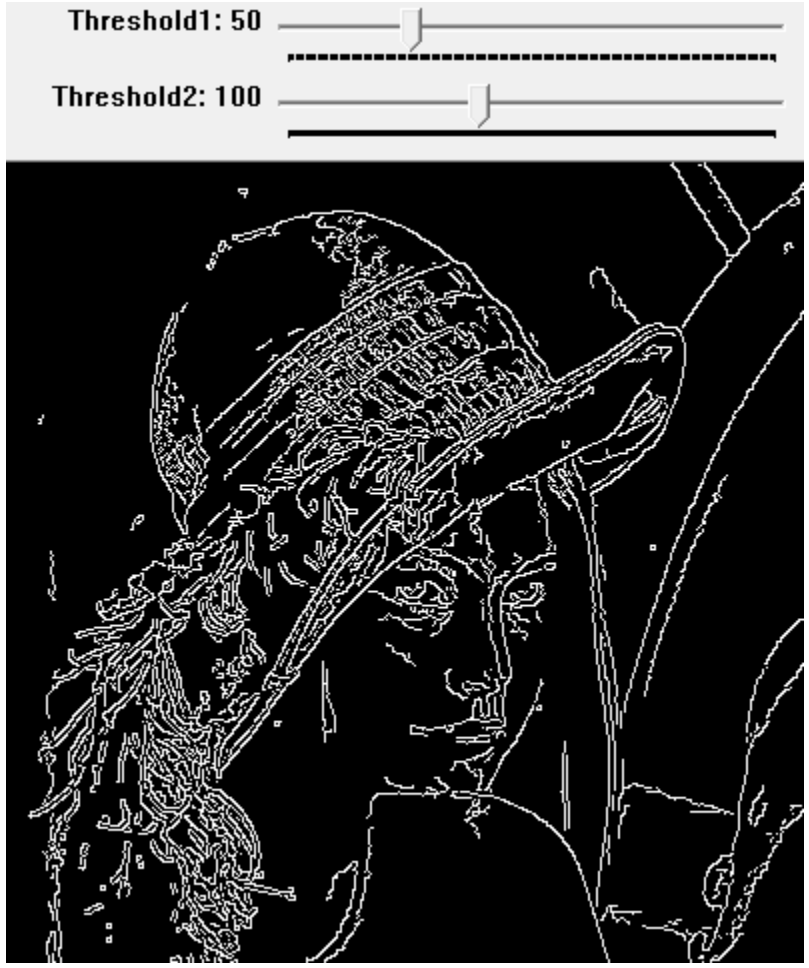
Canny edge 실험: Changing thrshold1 and threshold2

- low threshold과 high threshold를 동적으로 바꾸기
  - *createTrackbar* automatically calls *callback function* (CannyThr) everytime the slider is moved

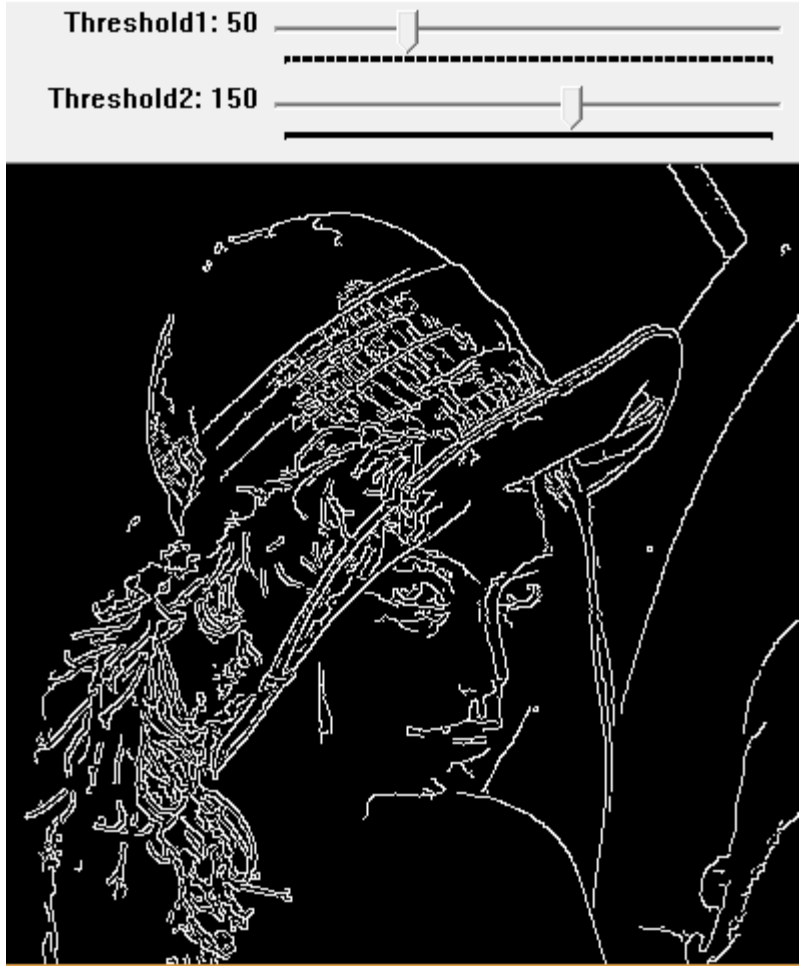
```
static Mat input_img, edge_img;
static char *win_name = "Canny";
static int lowTh, highTh;

void CannyThr(int, void*){ // Callback function
  Canny(input_img, edge_img, lowThr, highThr, 3);
  imshow(win_name, edge_img);
}

void Canny_test_thresholds(char *name) {
  input_img = imread(name, 0);
  edge_img.create(input_img.size(), input_img.type());
  namedWindow(win_name, CV_WINDOW_AUTOSIZE);
  createTrackbar("Threshold1", win_name, &lowThr,200, CannyThr);
  createTrackbar("Threshold2", win_name, &highThr,255,CannyThr);
  waitKey(0);
}
```



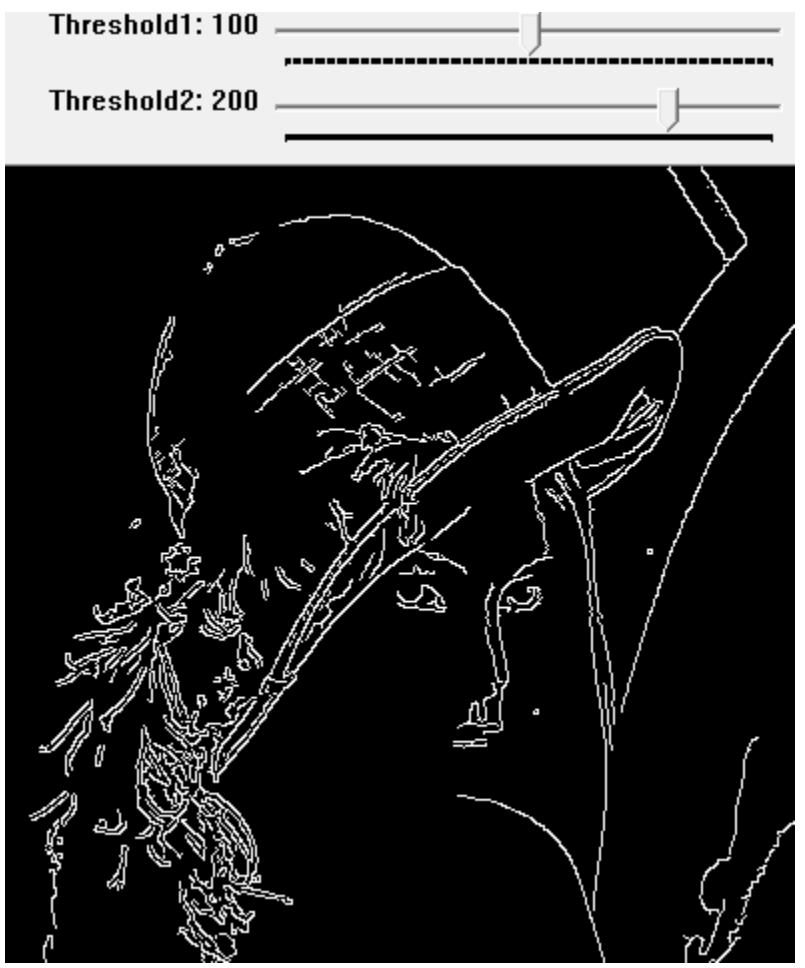
Canny Edge 실험 결과



th1=50, th2=100



th1=50, th2=150

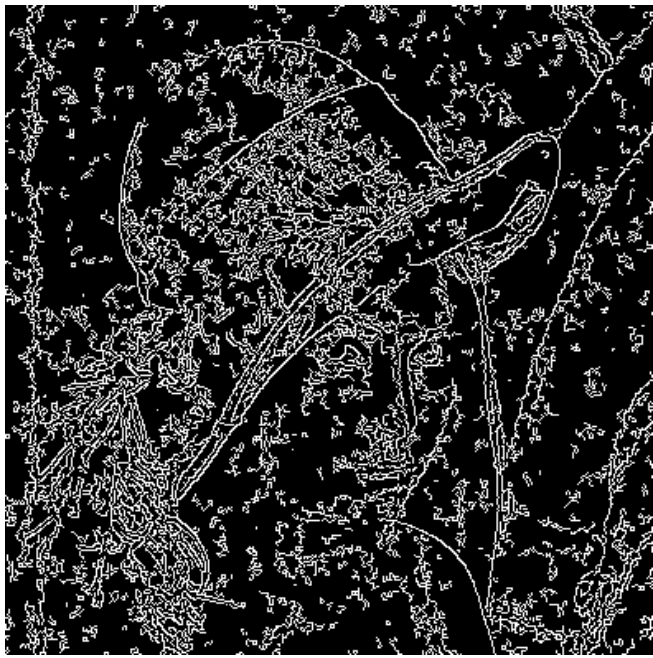


th1=100, th2=200

Edge Detection in Noisy Images

- Noise가 있는 이미지의 Edge 인식이 어려운 이유
  - 에지가 아닌 픽셀값이 바뀌어 edge로 오인식
  - 에지인 픽셀의 값이 바뀌어 edge로 미인식

```
Mat img = imread(name, IMREAD_GRAYSCALE);
imshow("input", img);

Mat gaussian_noise = img.clone(); //alloc space
randn(gaussian_noise, 0,20); // makes noise
dst = img + gaussian_noise; // inserts noise
imshow("Noisy", dst);

Mat dst;
Canny(dst, dst, 70, 180); // detect edges
imshow("Noisy Canny", dst);
waitKey(0);
```



Laplacian Filter

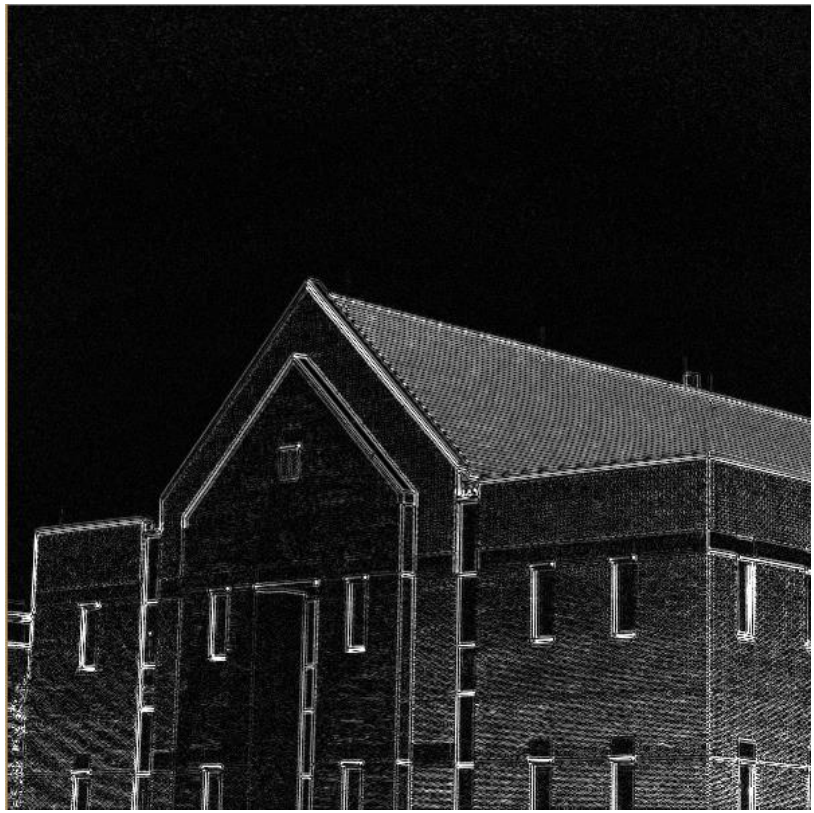
- Second Derivative (derivative of derivative)
- In OpenCV Laplacian operaor API:

$$g = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$$

Laplacian (input, output, ddepth, ksize, scale, delta, border)

- ksize : kernel size (default = 1)
- scale = 1, delta = 0 by default

```
Mat img = imread(name, IMREAD_GRAYSCALE);

Laplacian(img, dst, CV_16S, 3);
convertScaleAbs(dst, dst);

imshow("Laplacian", dst);
```



Blur Detection

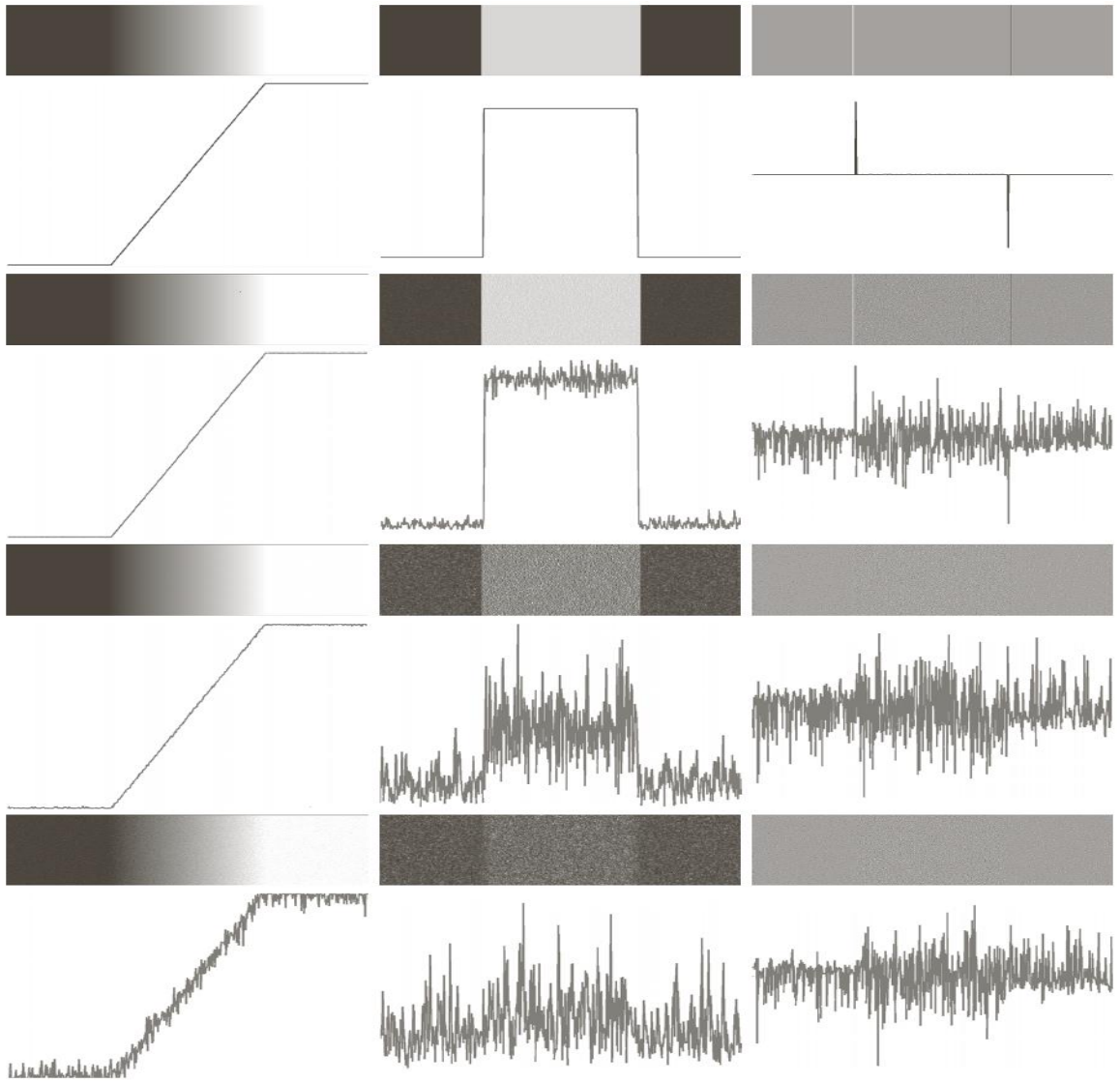
- Laplacian can be used to **detect the amount of blur** in image
- If the Laplacian() ksize = 1 (default) Then the kernel is applied
  - The measure of sharpness (opposite of blurness)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Why ?
  - Blurry images have patches that are uniform with respect to distribution of intensity values
  - There is no well-defined edge-like boundaries where the intensity change abruptly
  - ➔ Smaller variance of the difference made by the kernel

에지 검출에 미치는 노이즈의 영향

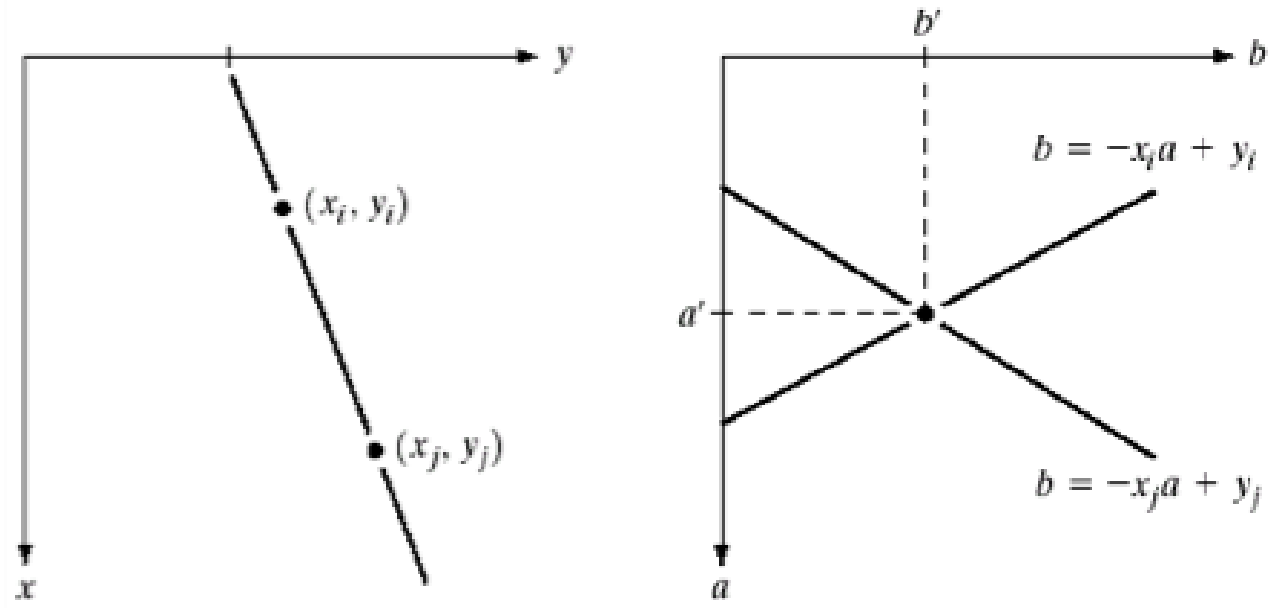
- 노이즈 영향
  - 1차 미분과 2차 미분은 노이즈에 매우 민감함
  - 2차 미분이 보다 민감
  - 날카로움에 의해 폭이 결정



직선검출

- Hough Line 변환의 필요성
  - 영상에서 에지 추출 후 에지 정보를 연결하여 관심 정보(직선) 추출
  - 관심 객체에 대한 정보 없이 추출이 어려움

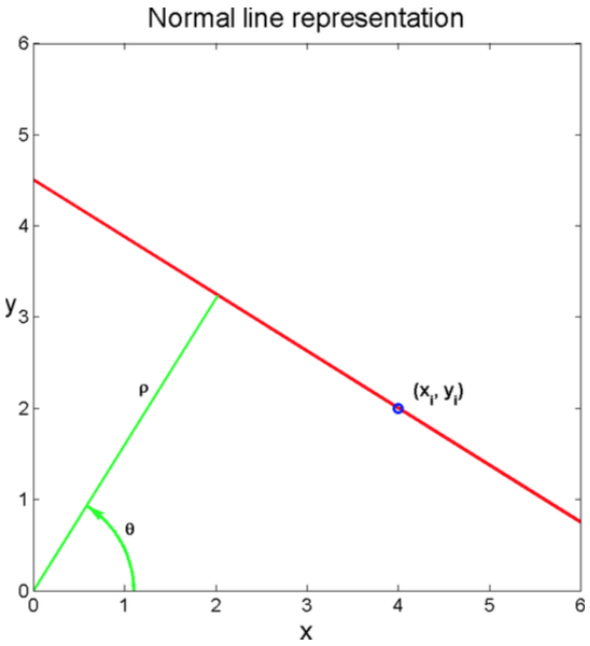
- Hough Line 변환의 개념
  - x,y 평면상의 한점 (x1,y1)을 지나는 모든 직선(y1 = a x1 + b)을 a, b 평면에서 점으로 매핑

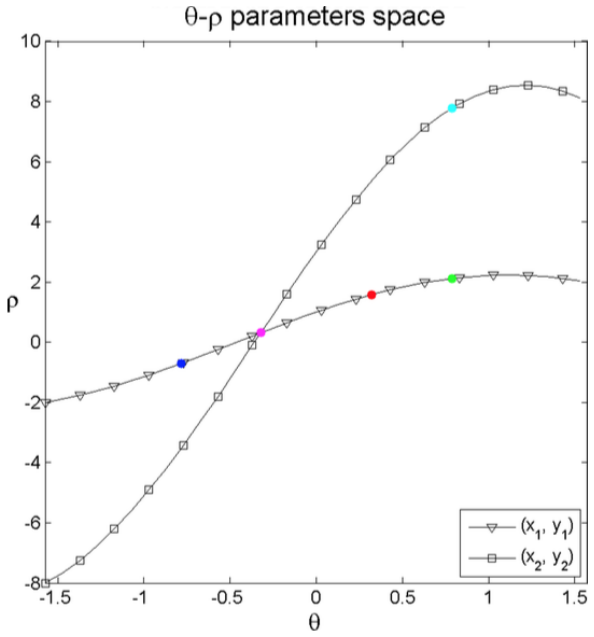


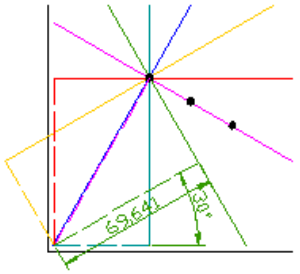
직선검출

- $\rho\theta$평면의 활용
  - 선이 수직선에 접근함에 따라 기울기가 무한대로 접근 ➔ a,b 평면에서 표현 불가
  - 직선의 정보를 $\rho\theta$ 평면에서 다음과 같이 표현가능

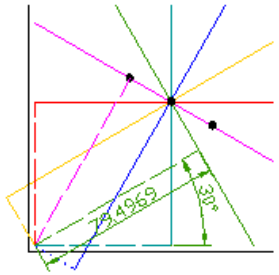
$$xcos\theta + ysin\theta = \rho$$



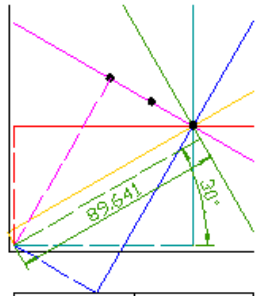




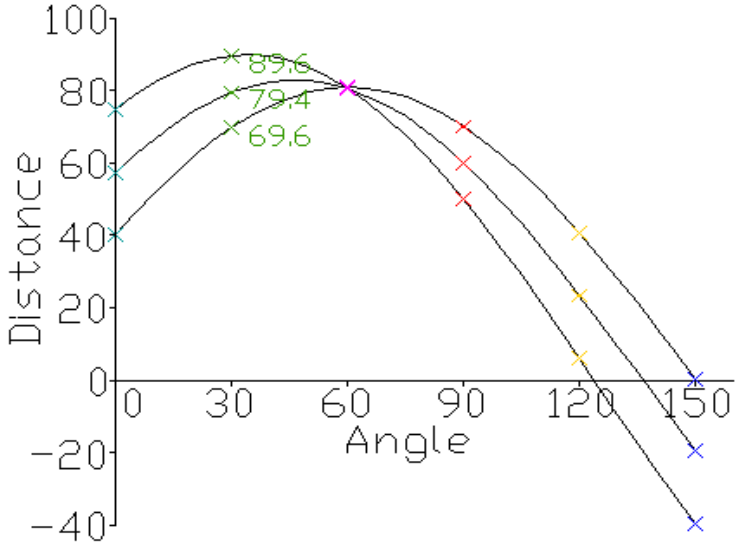
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

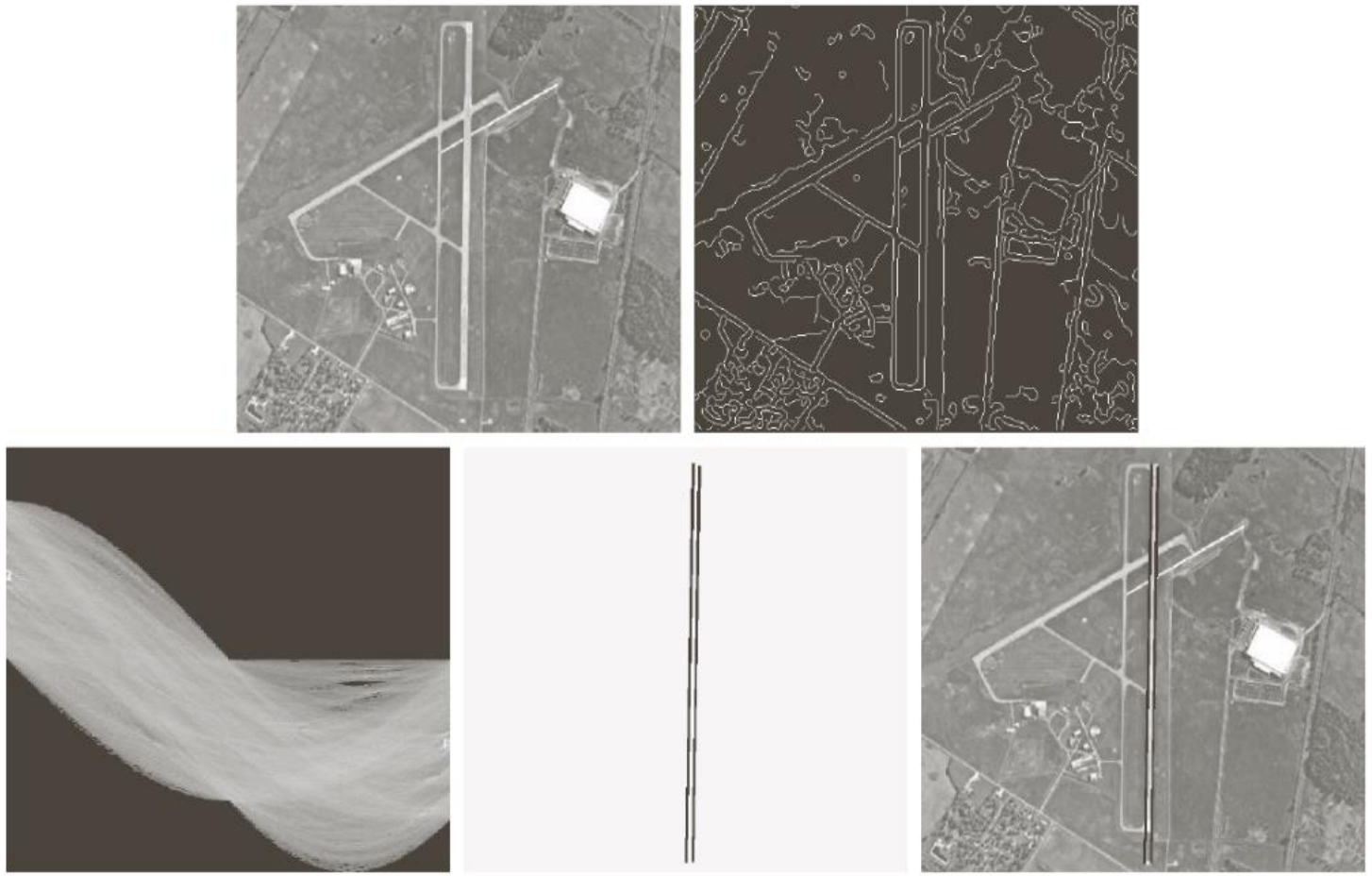


Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6



직선검출

- 직선의 검출 방식
  1. Grayscale 영상 변환
  2. 에지 영상 획득
  3. $\rho\theta$평면에서 구획 지정
  4. 에지를 이루는 점들에 대해서 각각 ρ, θ 계산하여 각 구획에 vote
  5. 문턱값이상의 vote를 받은 구간에서의 ρ, θ값을 직선으로 간주



직선 검출

HoughLinesP(input, output, rho, theta, threshold, minLineLength = 0, maxLineGap = 0)

rho: Distance resolution of the accumulator in pixels

theta: Angle resolution of the accumulator in radians

threshold: Accumulator threshold parameter

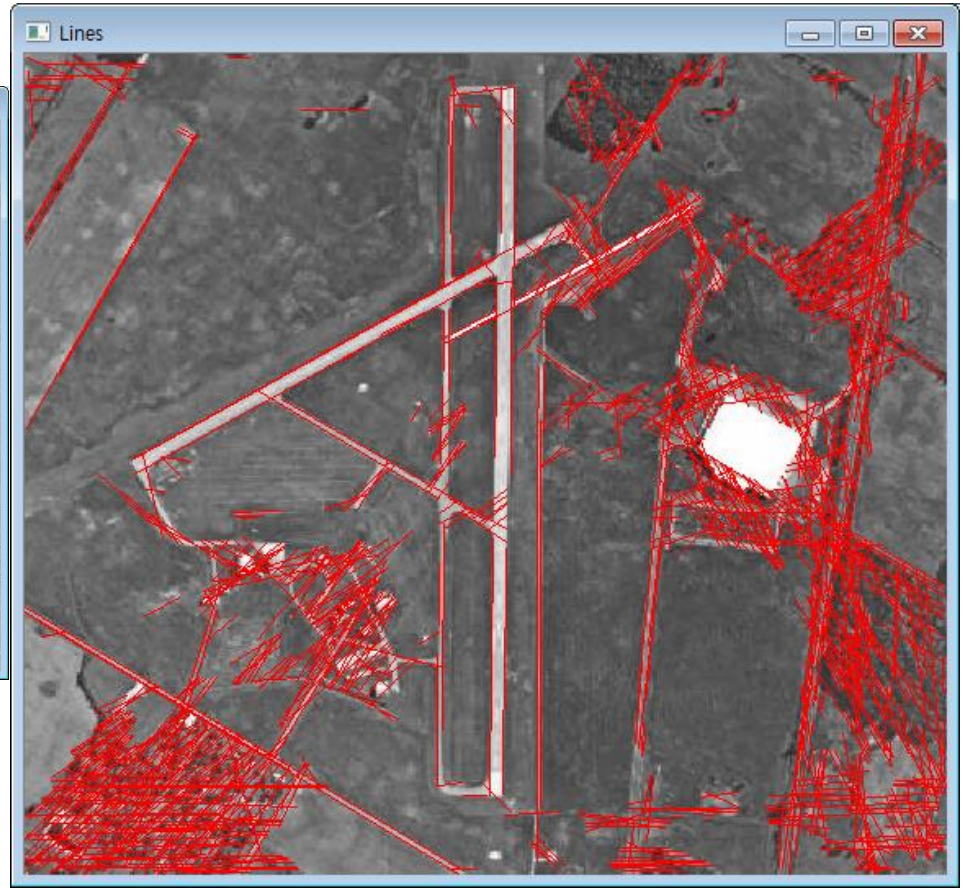
minLinelength: Minimum line length

maxLineGap: Maximum allowed gap between points

```
//edge extraction
Mat edge;
Canny(src, edge, 80, 200,3);

//Hough Transform
vector<Vec4i> lines;
HoughLinesP(edge, lines, 1, CV_PI / 180, 30, 50, 10);

//draw lines
for (size_t i = 0; i < lines.size(); i++)
  line(src, Point(lines[i][0], lines[i][1]),Point(lines[i][2], lines[i][3]), Scalar(0, 0, 255), 1);

imshow("Lines", src);
```

직선 검출

HoughLinesP(edge, lines, 1, CV_PI / 180, **50, 10, 10**) // minlength = 10



HoughLinesP(edge, lines, 1, CV_PI / 180, **50, 80, 10**)// minlength=80

POLYGON DETECTION

윤곽선(Contour) 검출

- findContours(image, contours, hierarchy, mode, method, offset): 점을 모아서 다각형 윤곽선 정보로 변환
  - image: 8-bit single-channel
  - contours: detected contours – vector of points (vector<vector<Point>>)
  - hierarchy: optional output vector, containing information about the image topology
    - hierarchy[i] is 0-based index in contours : [i][0]=next, [i][1]=prev, [i][2]=first child, [i][3]=parent
  - mode: contour retrieval mode
    - **CV_RETR_EXTERNAL** retrieves only the extreme outer contours.
    - **CV_RETR_LIST** retrieves all of the contours without establishing any hierarchical relationship
    - **CV_RETR_CCOMP** retrieves all of the contours and organizes them into a two-level hierarchy.
    - **CV_RETR_TREE** retrieves all of the contours and reconstructs a full hierarchy of nested contours.
  - method: contour approximation method
    - **CV_CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. 4각형의 경우, 4개의 Point만을 포함한다.
    - **CV_CHAIN_APPROX_NONE** stores absolutely all the contour points. 하나의 점도 생략하지 않음

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(edge_mat, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_NONE);
```

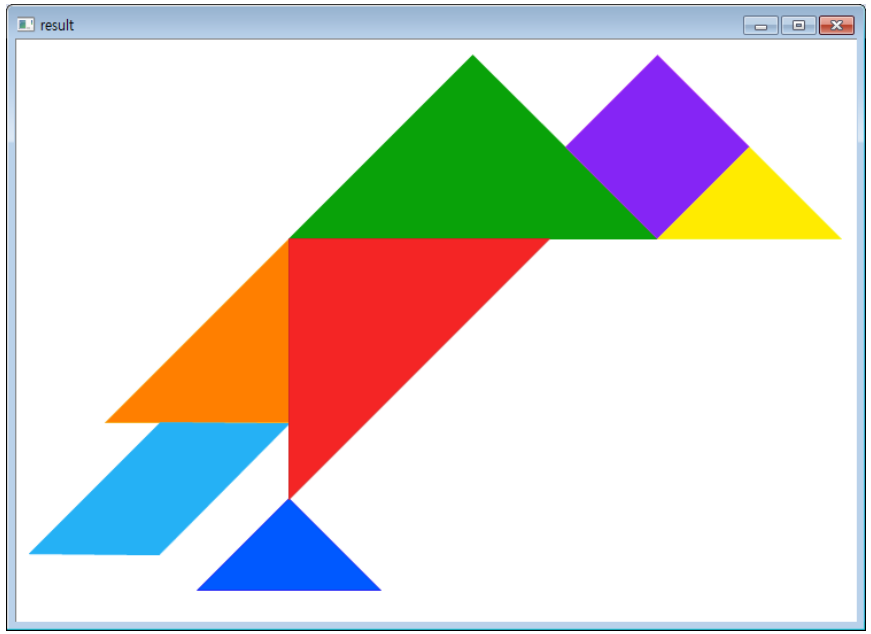
윤곽선(Contour) 그리기

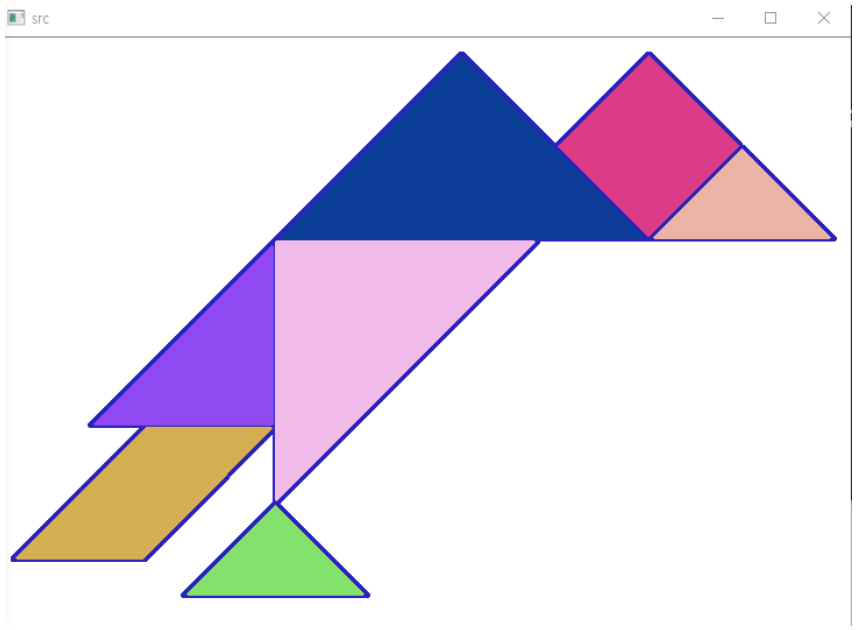
- drawContours(image, contours, contourIdx, color, thickness, hierarchy, maxLevel, offset)
  : 윤곽선을 이미지에 그려줌
  - contours: All the input contours
  - contourIdx: Parameter indication a contour to draw
  - color: color of contours
  - thickness: thickness of lines the contours are drawn with

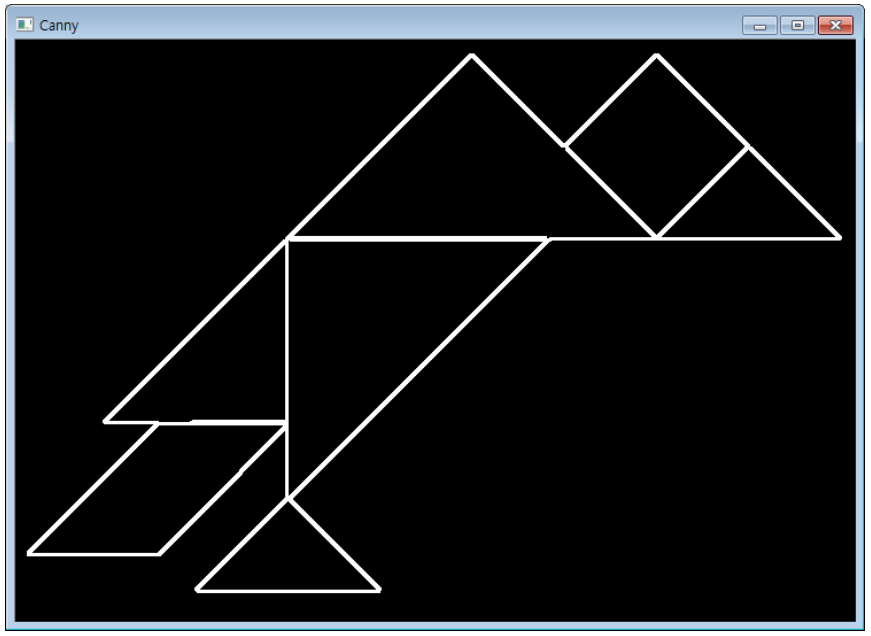
윤곽선 검출 및 그리기

```
Mat src = imread("bird.png" );

//edge extraction
Mat canny;
Canny(src, canny, 30, 60);
dilate(canny, canny, Mat(), Point(-1, -1), 1);

//find contours
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(canny, contours, hierarchy,  CV_RETR_TREE, CV_CHAIN_APPROX_NONE);

Mat result = src.clone();
for (size_t i = 0; i < contours.size(); i++){
        Scalar color(rand()%256, rand()%256, rand()%256);
        drawContours(resilt, contours, i, color, -1);
}
imshow("src", src);
imshow("Canny", canny);
imshow("result", result);
waitKey(0);
```

findContours and 및 drawContours 예제



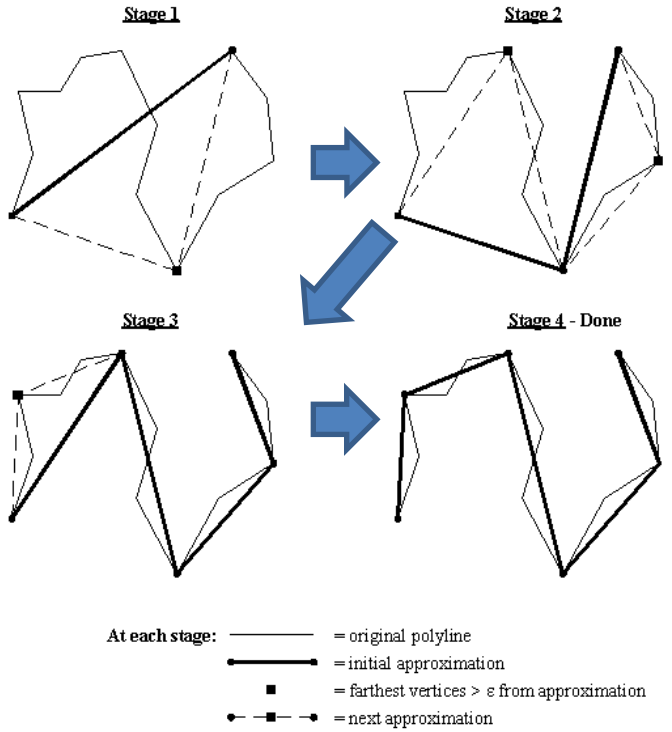




다각형으로 근사화

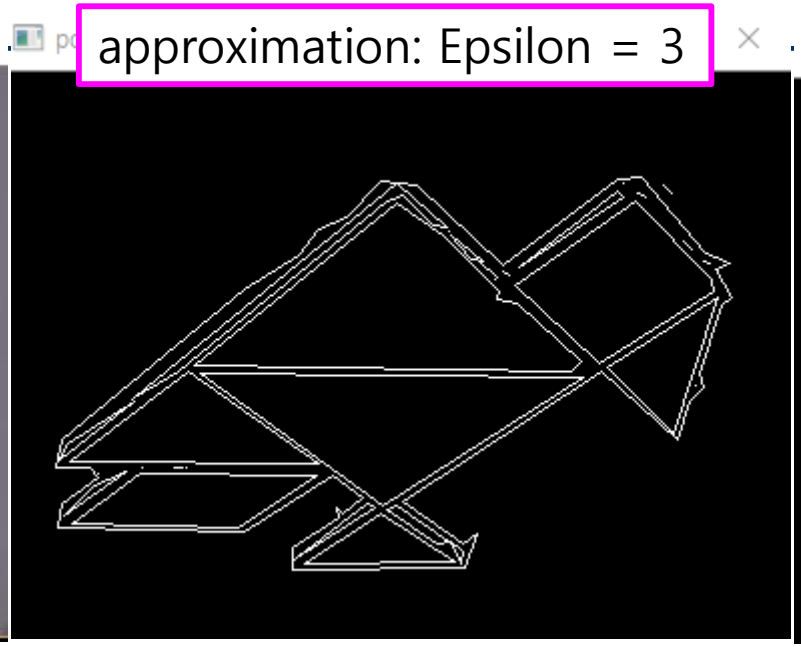
- approxPolyDP(curve, approxCurve, epsilon, closed)
  : 윤곽선을 다각형으로 근사화
  findContours의 결과를 입력으로 사용
  - curve: input vector of a 2D point – vector<Point>
  - approxCurve: Result of the approximation – vector<Point>
  - epsilon: specifying the approximation accuracy
  - closed: if true, the approximated curve is closed

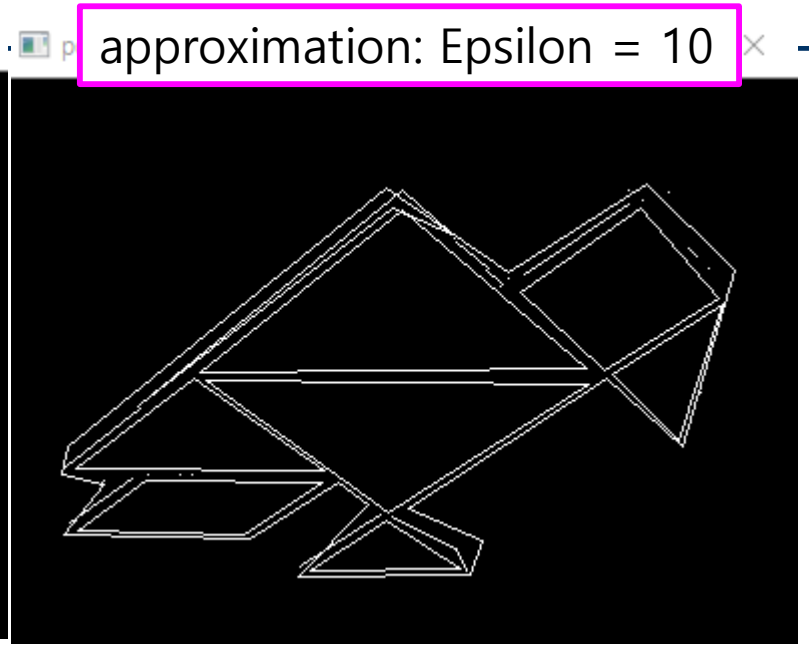
```
//find contours
Mat src = imread ( “bird.png” , 0);

Mat canny;
Canny(src, canny, 80, 100);

vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(canny, contours, CV_RETR_TREE, CV_CHAIN_APPROX_NONE);

vector<Point> poly;
approxPolyDP(contours[0], poly, 15, true);
```

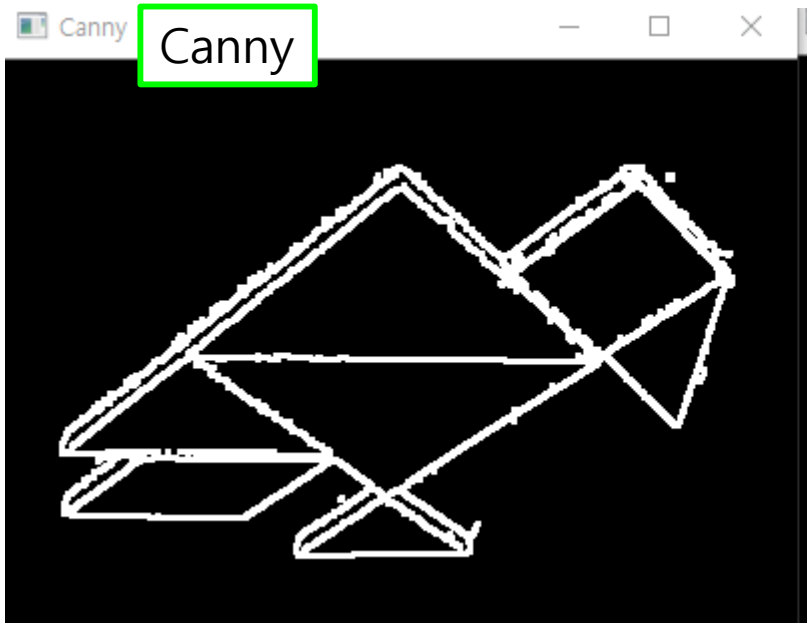


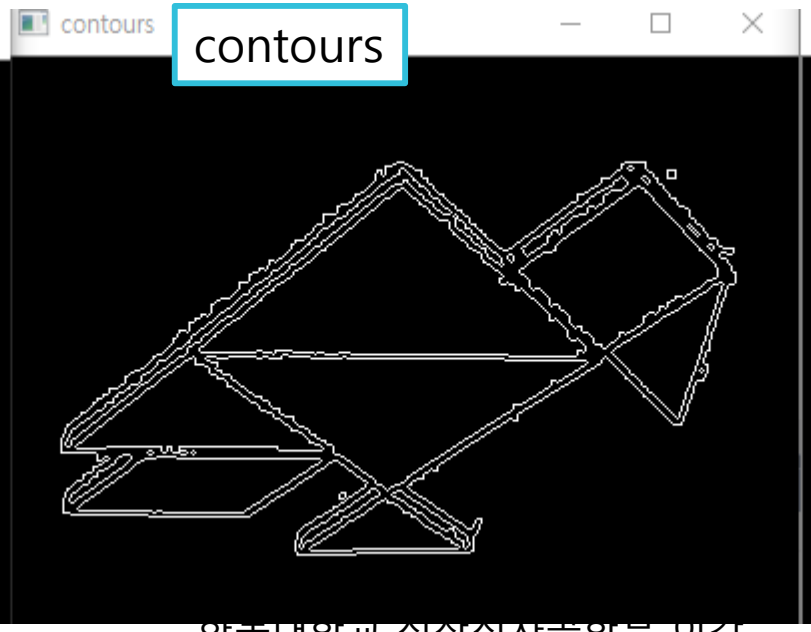
다각형으로 근사화(예)

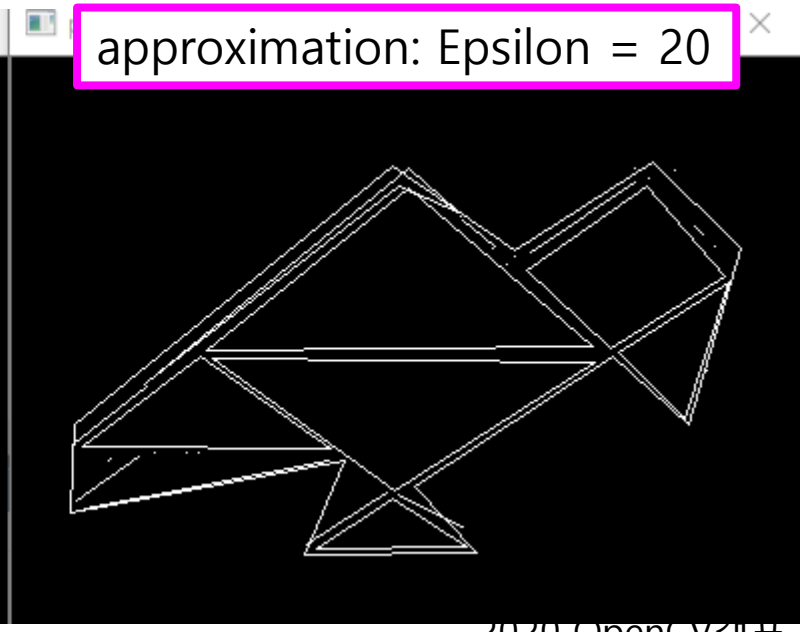

src


approximation: Epsilon = 3


approximation: Epsilon = 10

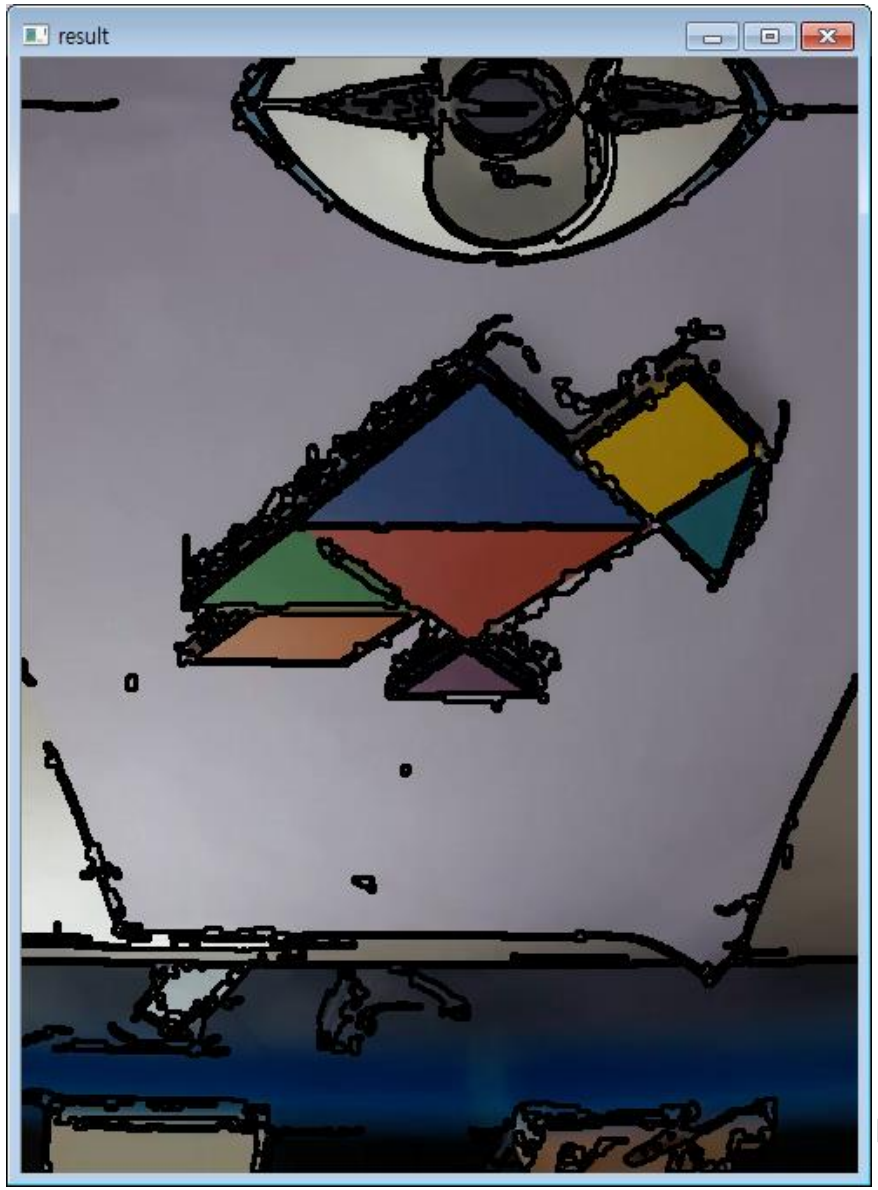

Canny
Canny


contours
contours


approximation: Epsilon = 20

다각형으로 근사화

findContours -> drawContours

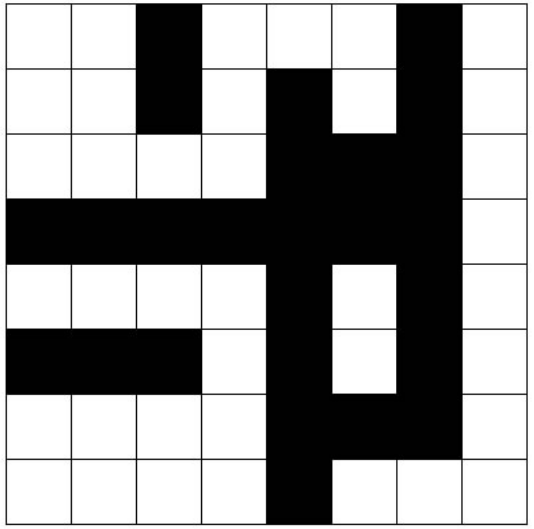


approxPolyDP -> polylines

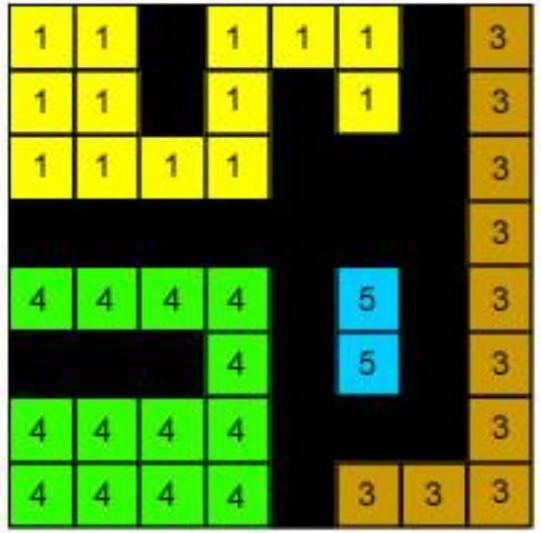
CLUSTERING 기반의 SEGMENTATION

Connected Component (Labeling)

- 입력된 이진 영상에서 서로 연결된 전경 (white,255)픽셀들을 찾아서 서로 연결된 픽셀들에게 동일한 label을 부여하기



1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	1
1	1	1	1	0	1	1	1



Connected Components OpenCV function

- ConnectedComponents (InputArray image,OutputArray lables, int connectivity,=8 int itype=CV_32S)
- ConnectedComponentsWithStats (InputArray image,OutputArray lables, OutputArray stats, OutputArray centroid, int connectivity,=8 int itype=CV_32S)

```
int cv::connectedComponents ( InputArray  image,
                              OutputArray labels,
                              int         connectivity = 8,
                              int         ltype = CV_32S
                            )
```

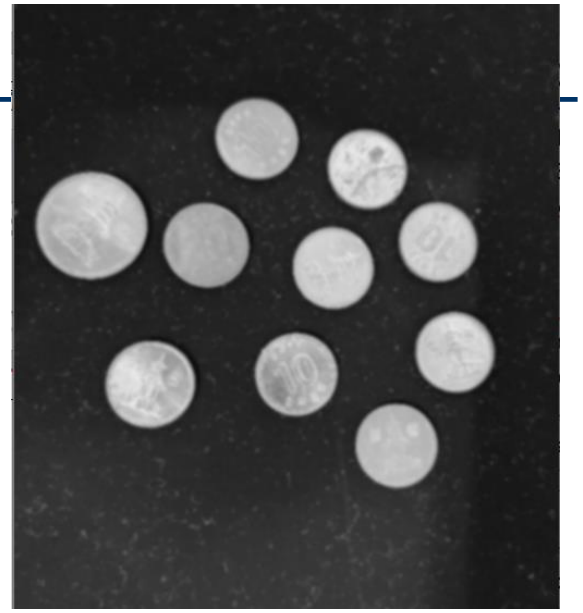
computes the connected components labeled image of boolean image
image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image.

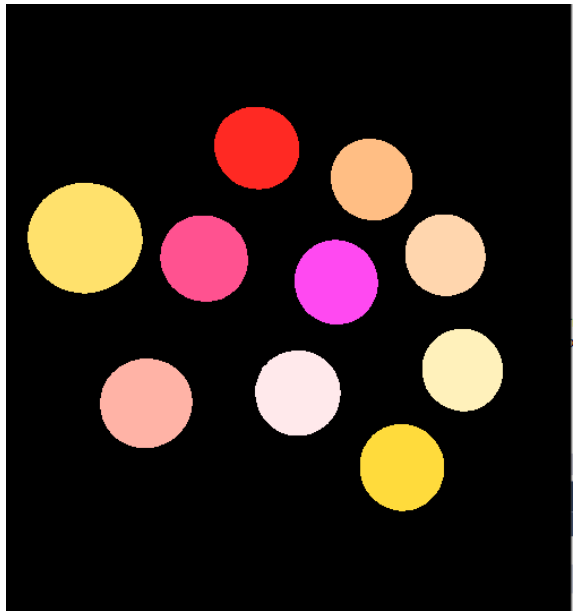
Parameters

image	the image to be labeled
labels	destination labeled image
connectivity	8 or 4 for 8-way or 4-way connectivity respectively
ltype	output image label type. Currently CV_32S and CV_16U are supported.

Connected Component Example

```
img = imread(name, IMREAD_GRAYSCALE)
GaussianBlur(img, img, Size(7, 7), 1.5, 1.5);
threshold(img, img_edge, 100, 255, THRESH_BINARY| THRESH_OTSU);
imshow("Image after threshold and open", img_edge);

int i, nccomps = cv::connectedComponents( img_edge, labels);
cout << "Total Connected Components Detected: " << nccomps << endl;
vector<cv::Vec3b> colors(nccomps + 1);
colors[0] = cv::Vec3b(0, 0, 0); // background pixels remain black.
for (i = 1; i <= nccomps; i++)
   colors[i] = cv::Vec3b(rand() % 256, rand() % 256, 255);

img_color = cv::Mat::zeros(img.size(), CV_8UC3);
for (int y = 0; y < img_color.rows; y++)
       for (int x = 0; x < img_color.cols; x++){
            int label = labels.at<int>(y, x);
             img_color.at<cv::Vec3b>(y, x) = colors[label];
}
cv::imshow("Labeled map", img_color);
```





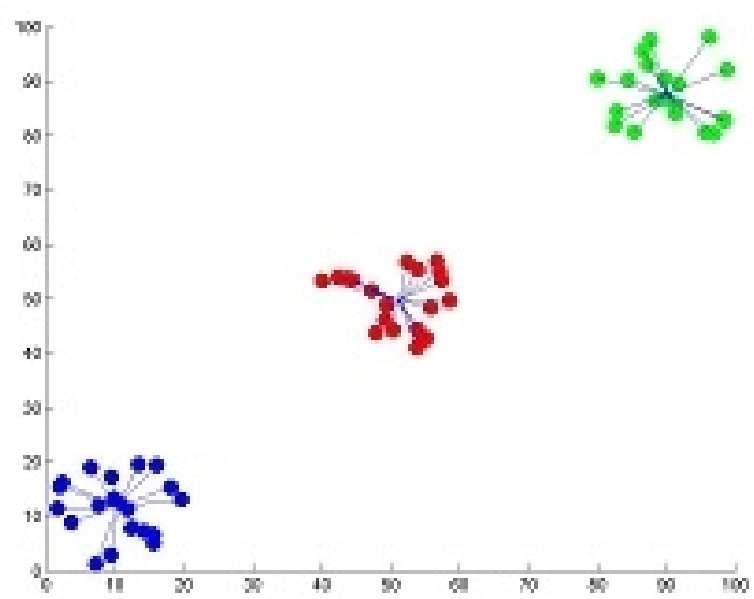
K-means Clustering

- Algorithm
  - in/out
    - input: data points X= {a1, a2, … , an }, K=number of clusters
    - output labels for each data points (0-based cluster index
  - procedure

1. Initialize K centroid u_i , i = 1…k from X
2. Assign each data point to the class c_i of its nearest centroid trying to minimize the objective function:

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

3. Update the centroids as the average of all data points assigned to that class
4. Repeat 2 and 3 until convergence

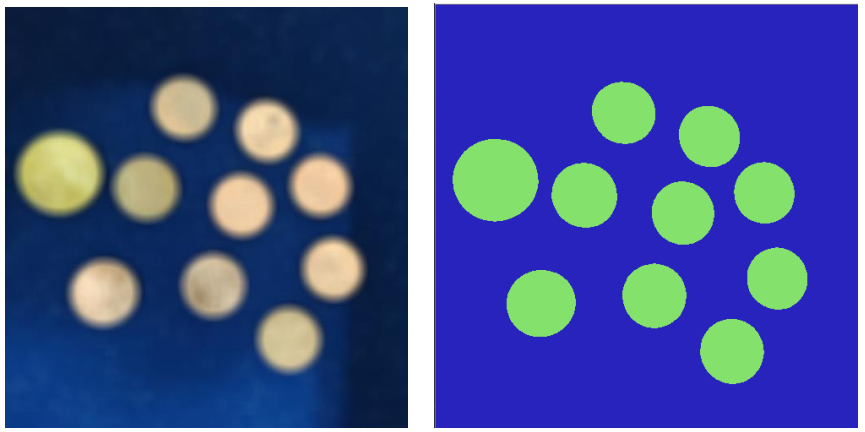


K-means Clustering OpenCV function

- double kmeans (inputArray data, int K, inputOutputArray bestLabels, TermCriteria criteria, int attempts, int flags, OutputArray centers = noArray())
  - 여러변 (attempts) 시도해서 가장 좋은 결과를 선택
  - data : An array of N-Dimentional points whith floating coordnates
    - Mat points (count 2, CV_32F);
    - Mat points (count, 1, CV_32FC2);
    - Mat points (1, count, CV_32FC2);
    - std::vector<cv::Point2f> points (sampleCount);
  - K : number of clusters
  - bestLabels : in/out integer array that stires the cluster indices for every sample
  - criteria: algorithm termination criteria (max number of iteration, desired accuracy)
  - attempts : different initial labelings
  - flags:
    - KMEANS_RANDOM_CENTERS (select random intial centers in each attempt),
    - KMEANS_PP_CENTERS: kmeans++ center initialization
    - KMEANS_USE_INITIAL_LABELS : during first attempt, user-supported labels instead of computing them

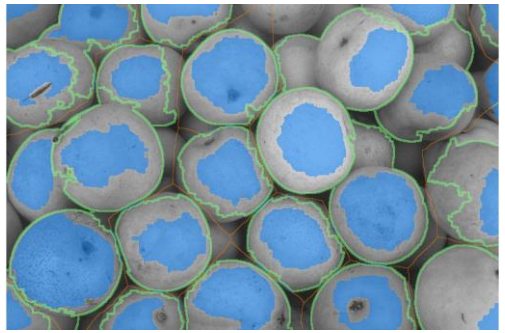
K-means Clustering Example Code

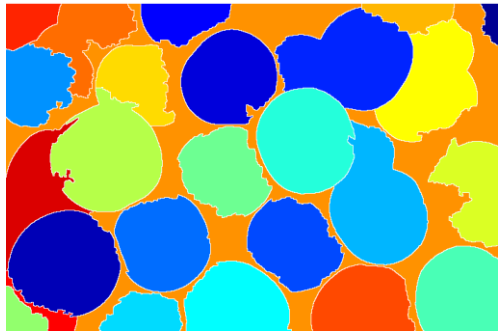
```
Mat src = imread(path);
Mat p = Mat::zeros(src.cols*src.rows, 5, CV_32F);
Mat bestLabels, centers, clustered;
vector<Mat> bgr; cv::split(src, bgr); // i think there is a better way to split pixel bgr color
for(int i=0; i<src.cols*src.rows; i++) {
  p.at<float>(i,0) = (i/src.cols) / src.rows;
  p.at<float>(i,1) = (i%src.cols) / src.cols;
  p.at<float>(i,2) = bgr[0].data[i] / 255.0;
  p.at<float>(i,3) = bgr[1].data[i] / 255.0;
  p.at<float>(i,4) = bgr[2].data[i] / 255.0;
}
int K = 2;
cv::kmeans(p, K, bestLabels, TermCriteria( CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, 10, 1.0), 3, KMEANS_PP_CENTERS, centers);

vector<Vec3b> colors(K);
for(int i=0; i<K; i++)
    colors[i] = Vec3b( rand()%256, rand()%256, rand()%256);

clustered = Mat(src.rows, src.cols, CV_32F);
for(int i=0; i<src.cols*src.rows; i++) {
  clustered.at<Vec3b>(i/src.cols, i%src.cols) = color;
```

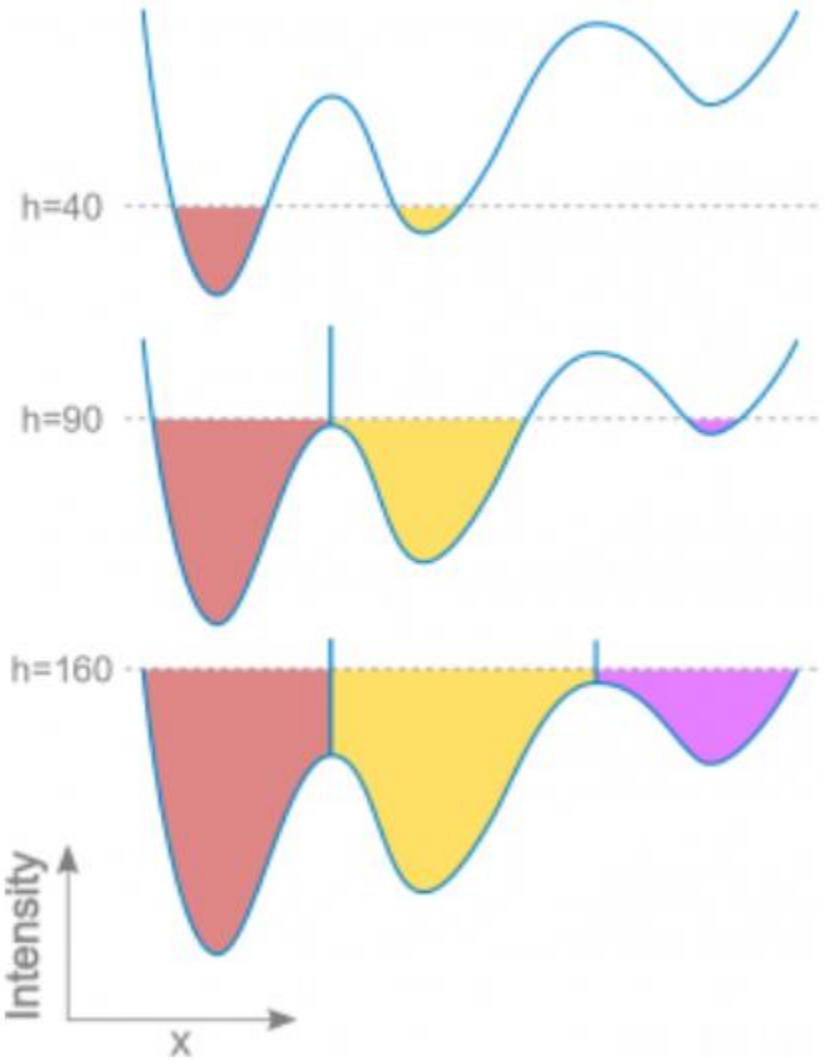


watershed 알고리즘

- 영상 분할 (segmentation) 방법의 한가지
- 큰픽셀값 가지는 local maxima를 언덕, 작은 픽셀 값을 가지는 local minima 를 골짜기로 간주하고 물을 한방울씩 채워가는 과정을 상상
- 너무 많은 분할이 만들어지는 것을 막도록 seed를 분할별로 지정해주는 방안도 있음





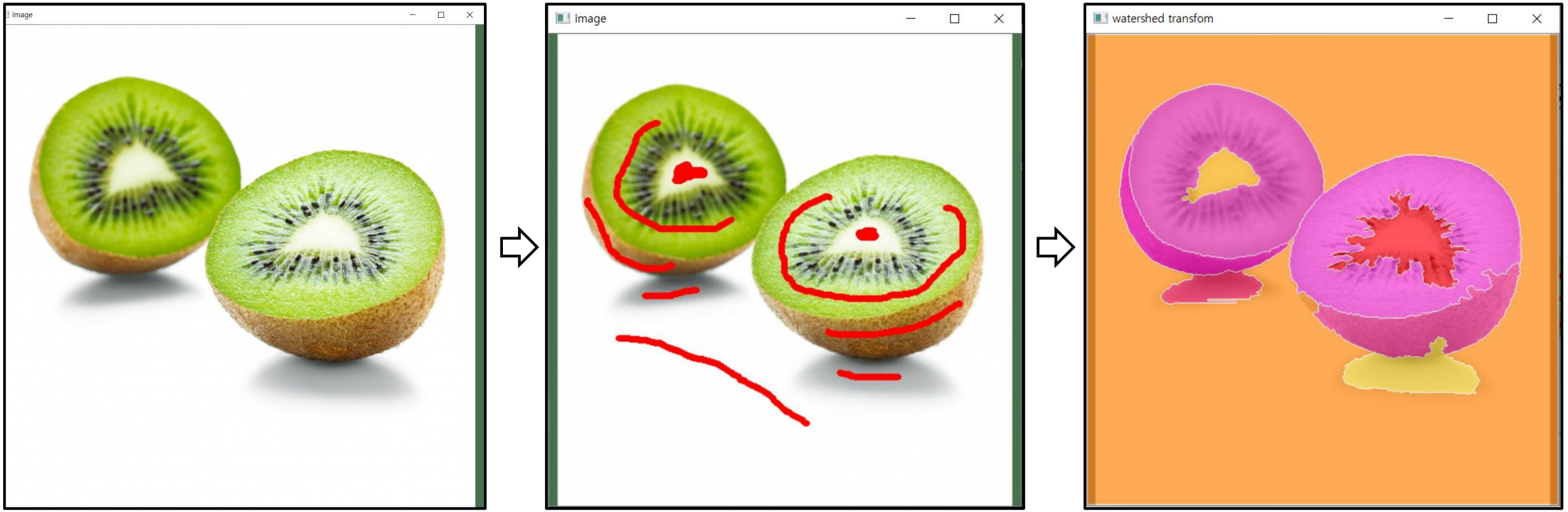


watershed in OpenCV function

- void watershed(inputArray **image**, inputOutputArray **markers**)
  - image: input 8-bit 3-channel image
  - markers: Input/output 32-bit single-channel image (map) of markers. It should have the same size as image ➔ 마커 값과 위치 입력 및 결과값 저장

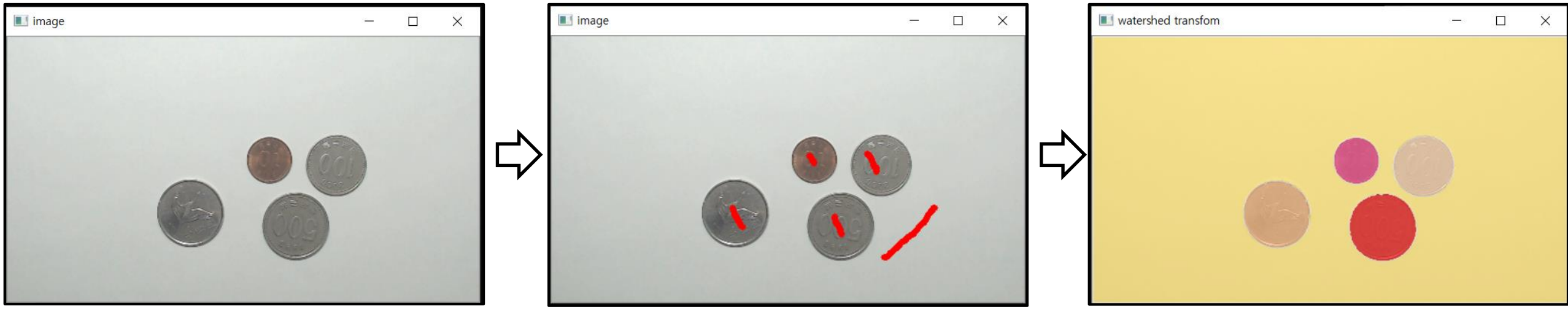
  - image processing 하기 전에 makers의 픽셀들의 값을 준비해야 함
    - 확실한 foreground 영역의 일부: 양의 정수 (서로다른 foreground region은 다른 정수값)
    - undefined 영역: 0의 값 (watershed에 의해서 어느 영역인지 값이 정해질 부분)
  - watershed 함수 알고리즘의 기능: markers의 양의 정수값들이 “seed“가 되어 undefined 영역의 각 픽셀 값을 seed 값 중 하나로 결정함. 두개의 region을 나누는 경계선 픽셀들은 -1의 값으로 정해짐

watershed 예 – 마우스로 marking하여 image segmentation



WATERSHED 예제

watershed 예제 – 마우스로 marking하여 image segmentation



watershed 예제

- Mouse marking 동작 함수

```
#include <opencv2/core/utility.hpp>
#include "opencv2/opencv.hpp"
#include <cstdio>
#include <iostream>

using namespace cv;
using namespace std;

Mat markerMask, img;
Point prevPt(-1, -1);
static void onMouse(int event, int x, int y, int flags, void*)
{
    if (x < 0 || x >= img.cols || y < 0 || y >= img.rows)
        return;
    if (event == EVENT_LBUTTONUP || !(flags & EVENT_FLAG_LBUTTON))
        prevPt = Point(-1, -1);
    else if (event == EVENT_LBUTTONDOWN)
        prevPt = Point(x, y);
    else if (event == EVENT_MOUSEMOVE && (flags & EVENT_FLAG_LBUTTON))
    {
        Point pt(x, y);
        if (prevPt.x < 0)
            prevPt = pt;
        line(markerMask, prevPt, pt, Scalar::all(255), 5, 8, 0);
        line(img, prevPt, pt, Scalar::all(255), 5, 8, 0);
        prevPt = pt;
        imshow("image", img);
    }
}
```

watershed 예제 – watershed 동작 함수 (1)

```
31  void watershed_test(char *filename)
32  {
33      Mat img0 = imread(filename, 1), imgGray;
34      if (img0.empty()) {
35          cout << "Couldn't open image ";   return;
36      }
37
38      resize(img0, img0, Size(500, 500 * (double)img0.rows / (double)img0.cols));
39      img0.copyTo(img);
40      cvtColor(img, markerMask, COLOR_BGR2GRAY);
41      cvtColor(markerMask, imgGray, COLOR_GRAY2BGR);
42      markerMask = Scalar::all(0);
43      imshow("image", img);
44      setMouseCallback("image", onMouse, 0);
45
46      for (;;)
47      {
48          char c = (char)waitKey(0);
49          if (c == 27)  break;
50          if (c == 'r'){
51              markerMask = Scalar::all(0);
52              img0.copyTo(img);
53              imshow("image", img);
54          }
```

이어서

```
56          if (c == 'w' || c == ' '){
57              int i, j, compCount = 0;
58              vector<vector<Point> > contours;
59              vector<Vec4i> hierarchy;
60
61              findContours(markerMask, contours, hierarchy, RETR_CCOMP, CHAIN_APPROX_SIMPLE);
62              if (contours.empty())  continue;
63              Mat markers(markerMask.size(), CV_32S);
64              markers = Scalar::all(0);
65              int idx = 0;
66              for (; idx >= 0; idx = hierarchy[idx][0], compCount++)
67                  drawContours(markers, contours, idx, Scalar::all(compCount + 1), -1, 8, hierarchy, INT_MAX);
68              if (compCount == 0)  continue;
69              imshow("markers", markers*10000);
```

다음장으로 이어집니다.

watershed 예제 – watershed 동작 함수 (2)

```
56      if (c == 'w' || c == ' '){
57          int i, j, compCount = 0;
58          vector<vector<Point> > contours;
59          vector<Vec4i> hierarchy;
60
61          findContours(markerMask, contours, hierarchy, RETR_CCOMP, CHAIN_APPROX_SIMPLE);
62          if (contours.empty())  continue;
63          Mat markers(markerMask.size(), CV_32S);
64          markers = Scalar::all(0);
65          int idx = 0;
66          for (; idx >= 0; idx = hierarchy[idx][0], compCount++)
67              drawContours(markers, contours, idx, Scalar::all(compCount + 1), -1, 8, hierarchy, INT_MAX);
68          if (compCount == 0)  continue;
69          imshow("markers", markers*10000);
```

이어서

```
71          vector<Vec3b> colorTab;
72          for (i = 0; i < compCount; i++)
73              colorTab.push_back(Vec3b(rand()%256, rand()%256, rand()%256));
74          watershed(img0, markers);
75           Mat wshed(markers.size(), CV_8UC3);
76          // paint the watershed image
77          for (i = 0; i < markers.rows; i++)
78              for (j = 0; j < markers.cols; j++){
79                  int index = markers.at<int>(i, j);
80                  if (index == -1)
81                      wshed.at<Vec3b>(i, j) = Vec3b(255, 255, 255);
82                  else if (index <= 0 || index > compCount)
83                      wshed.at<Vec3b>(i, j) = Vec3b(0, 0, 0);
84                  else
85                      wshed.at<Vec3b>(i, j) = colorTab[index - 1];
86              }
87          wshed = wshed * 0.7 + imgGray * 0.3;
88          imshow("watershed transfom", wshed);
89      }
90  }
91  }
```

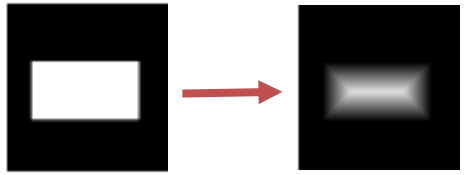
Distance Transform 알고리즘

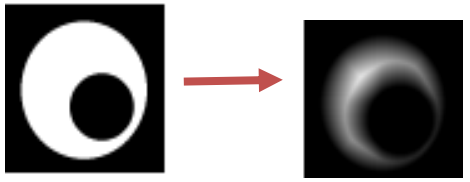
- input : binary image (background = 0, foreground : non-zero)
- output : grayscale image (Each foreground pixel values are changed to show the **distance to the closest background border** pixel (pixel value = 0)

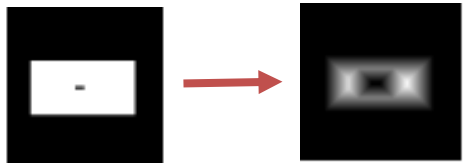
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

⇒

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







Distance metrics

Euclid distance: $D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

City block distance: $D_{City} = |x_2 - x_1| + |y_2 - y_1|$

Chessboard distance: $D_{Chess} = \max(|x_2 - x_1|, |y_2 - y_1|)$

입력이미지의 특징이 잘 나타나는 이진화가 선행되어야 함

Distance Transform in OpenCV API

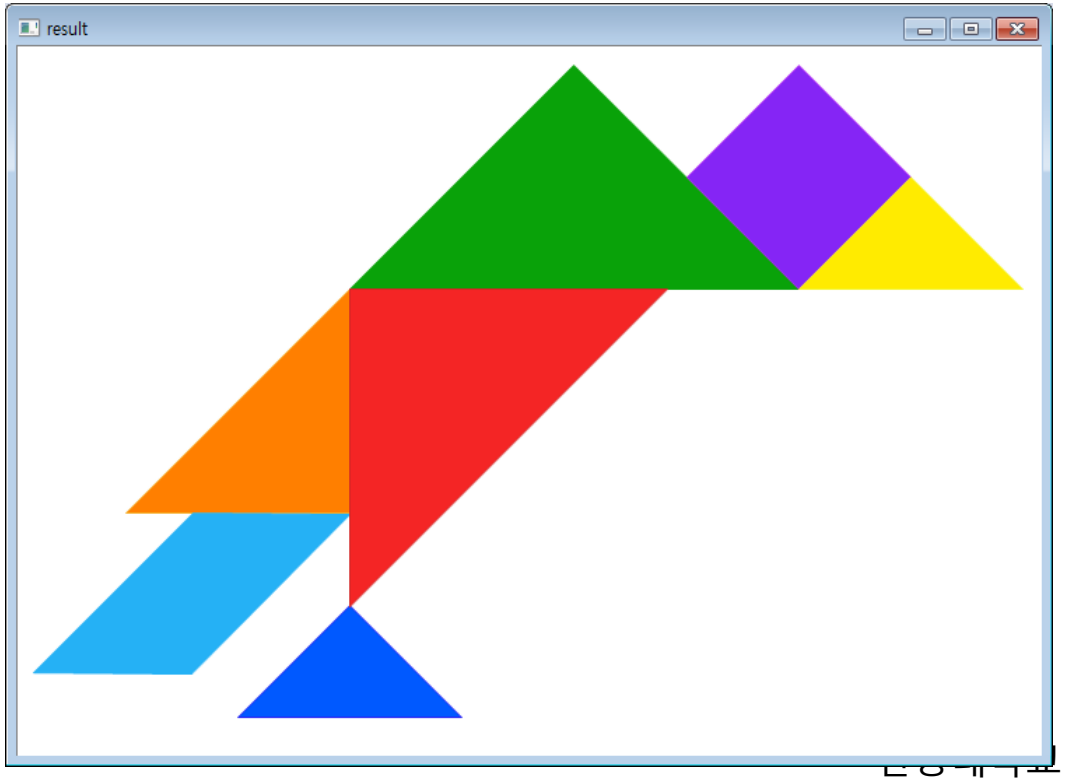
- void distanceTransform (inputArray src, outputArray dst,
  int distanceType, int maskSize, int dstType CV_32F)

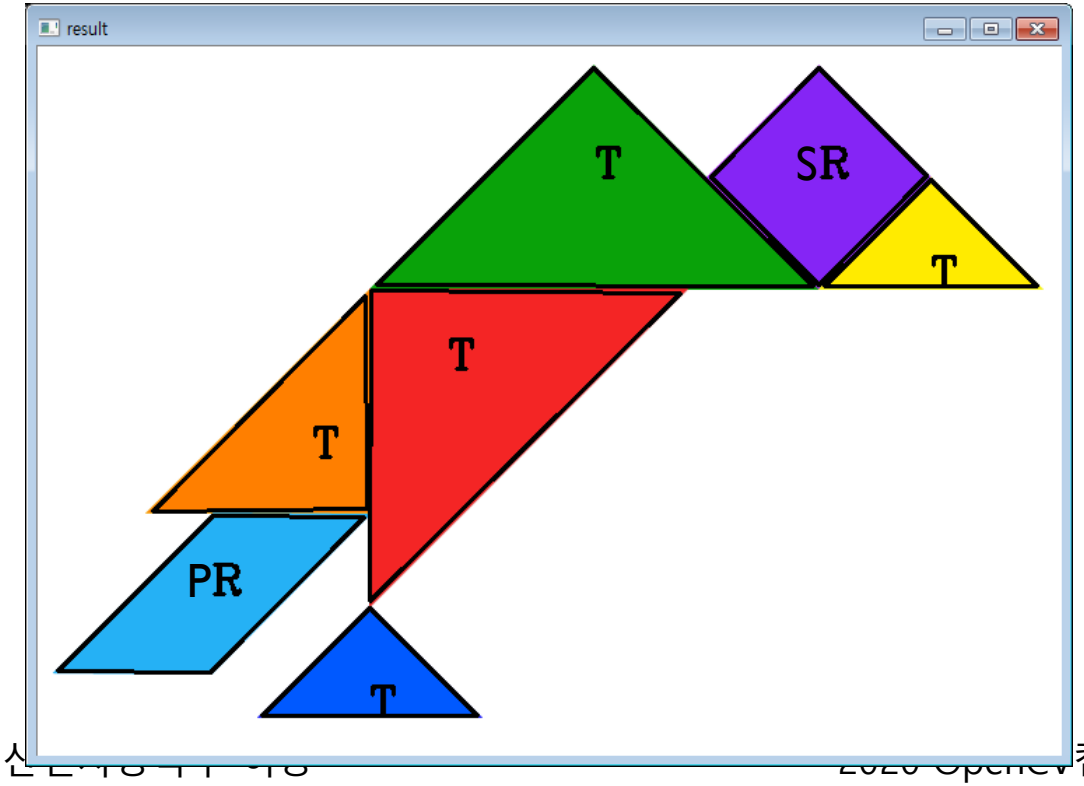
  - src: 8-bit single-channel source imag
  - dst: output image with calculated distance (8-bit or 32-bit single-channel image)
  - distanceType: DIST_L1 (city block distance),
    DIST_L2 (simple euclidiean distance),
    DIST_C (chessboard distance)
  - maskSize: distance 계산을 위한 mask DIST_L1 또는 DIST_C일 때는 무조건 3x3임
  - distType :dst의 image 픽셀 타입, CV_8U (distanceTYpe이 DIST_L1 일때만)또는 CV32F

영상처리 연습문제

[과제 I] 색으로 도형 분류하기

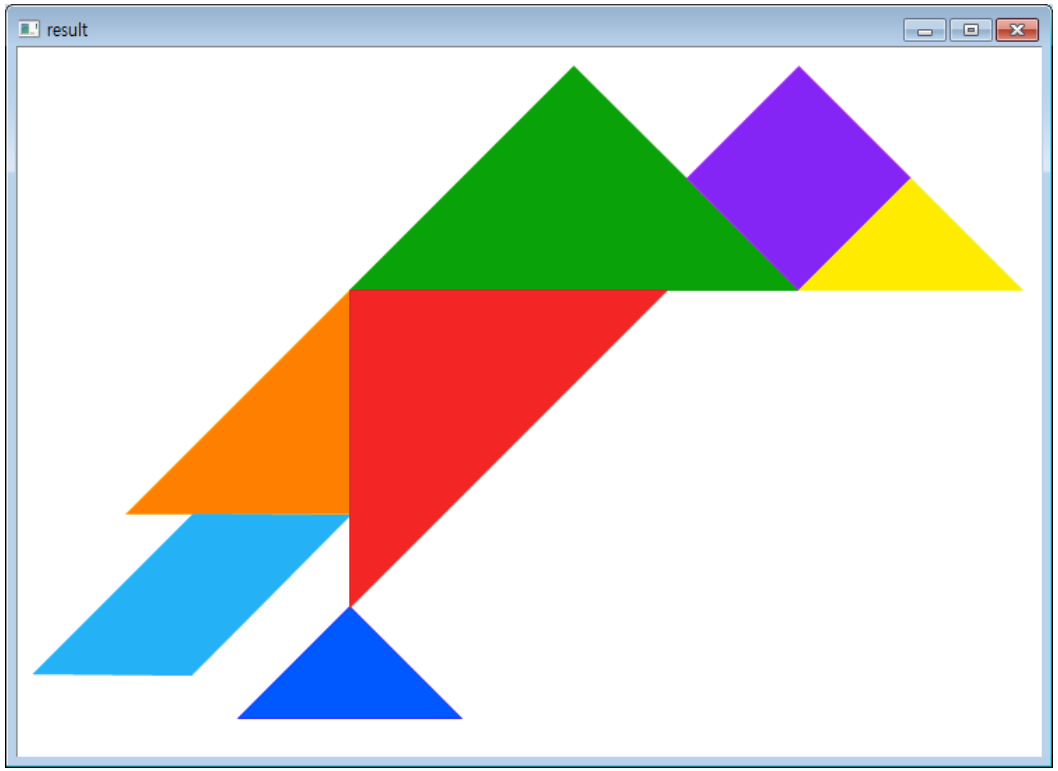
- 입력된 영상에서 색깔에 따라 도형의 갯수 세기 (HSV로 색을 변환후 H 채널을 이용하기)
  - ※ inRange(source, from , to , dest) 함수는 source의 화소가 from<= source <= to 에 해당하면 dest의 해당 화소를 255로, 아니면 0으로 지정하는 함수이다.
  - ※ split() 함수는 3채널 영상을 1채널 영상 3개로 나누는 함수이다. Merge() 함수는 split된 영상 백터를 하나의 영상으로 통합하는 함수이다.



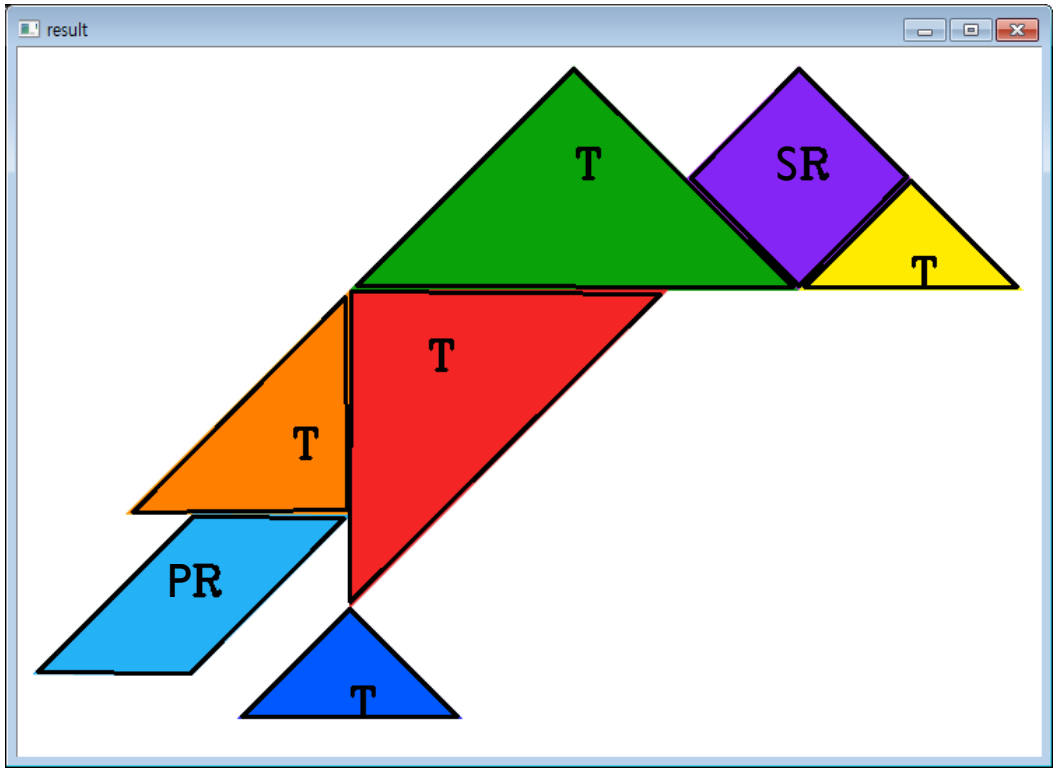


[과제2] 윤곽선으로 도형찾기

노이즈가 없는 이미지에서 삼각형, 정사각형, 평행사변형을 구분하여 그림에 표시
(힌트: 도형의 꼭지점 수와 변의 길이의 비를 이용)



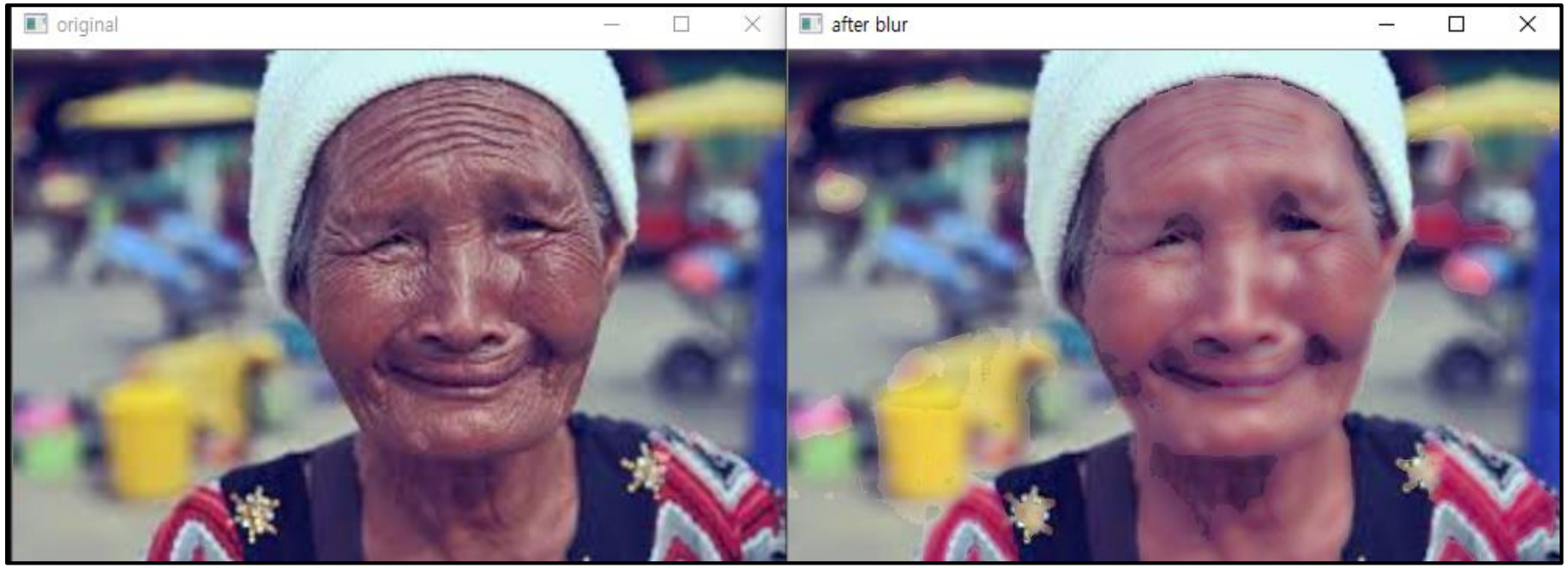
입력이미지



출력이미지

[과제3] 사진에서 얼굴 찾아 뽀샵하기

- 컬러공간을 BGR → HSV로 변환
- 얼굴이 포함된 사진에서 얼굴 영역만 H값으로 찾아서 보여주기 (0~20, 160~180)
- 얼굴 영역의 V 채널만 blurring하고
- 얼굴 영역의 S 채널값을 20만큼 증가하고
- 다시 3개의 채널을 merge 하기



[과제 4] 동영상 플레이어

컬러 동영상 입력을 받아 grayscale/color 로 선택하여 출력하기
키보드의 space bar를 입력할때마다 가다 서다 토글, ESC 입력시 재생 종료 ,
숫자 1을 입력하면 color로 , 0을 입력하면 grayscale로 출력

```
#include “opencv2/opencv.hpp”
using namespace cv;

int main() {
VideoCapture capture(/*"영상 이름 혹은 경로"*/);
Mat frame, grayImage;
 bool stop = falser;
for ( ; ; ){
    if (!stop){
                    capture >> frame;
                    if (!capture.read(frame))
                        break;
                    //cvtColor를 사용하여 흑백으로 변경
                    //imshow를 사용해 원본과 grayImage 출력
                    key=waitKey(30);
                    // if (key == 27)
                            break;

     }
 }
```

[과제5] 블랙박스 영상에서 차선 검출하기

- 블랙박스에 나타난 차선의 특징
  - 색상
    - 차선의 색상은 노란색이거나 흰색
  - 명암 대비
    - 차선은 도로 위에 존재하는데, 도로의 색상은 검정색 혹은 회색인 경우가 대부분이다. 차선의 색상이 노란색 혹은 흰색이므로, 차선 영역의 근처에서는 강한 색의 대비가 발생한다. 이러한 색의 대비를 에지 검출기를 활용하여 검출함으로써 차선 영역을 추출
  - 직선
    - 차선은 실선일 수도 있고 점선일 수도 있으나, 직선 도로에서의 차선은 직선이다. 또한 차량의 움직임을 고려하였을 때 도로의 곡률이 극단적일 수 없으므로, 곡면 도로에 그려진 차선의 경우에도 구간적으로 선형이다(piecewise linear). 따라서 Hough Line 변환을 통해 영상 내의 직선을 검출
  - 그 외 추가적 특성 : ???

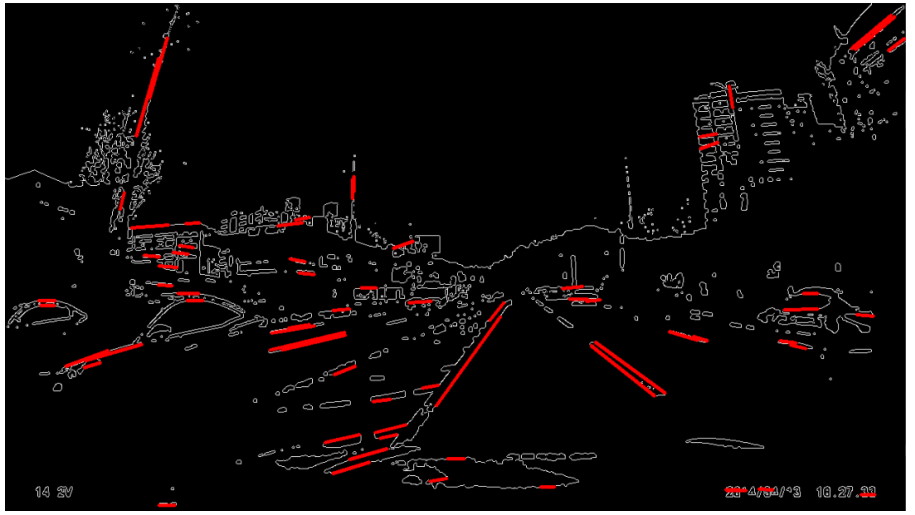
과제5 힌트: 차선검출 방식 예시

1. 입력 영상 스무딩 (Gaussian Filter)

2. 색상정보에 의한 차선 영역 검출(흰색+노랑색)

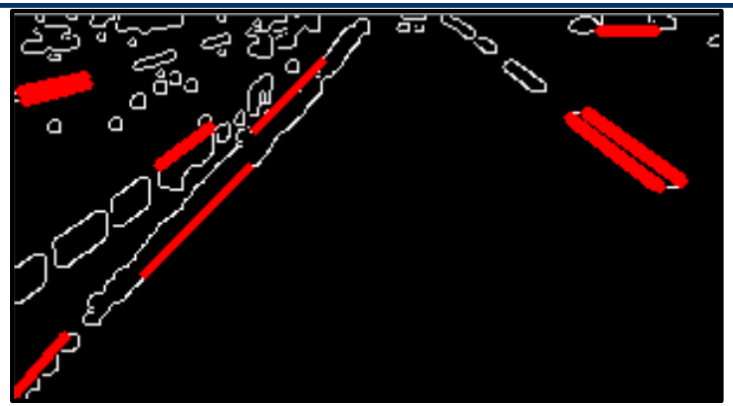
3. 에지 추출 (Canny)

4. 직선 추출 (Hough 변환)



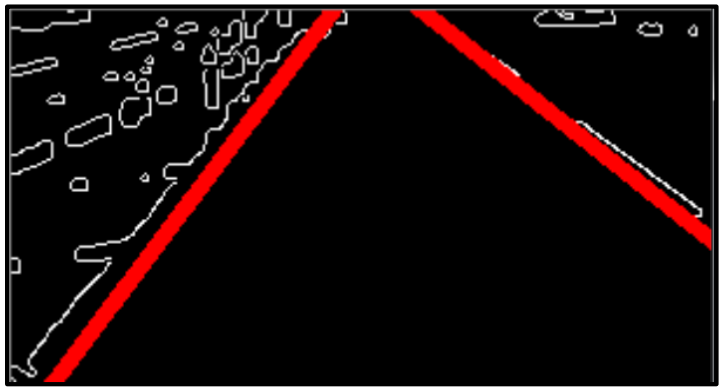
5. 추가적 필터

- ROI 설정

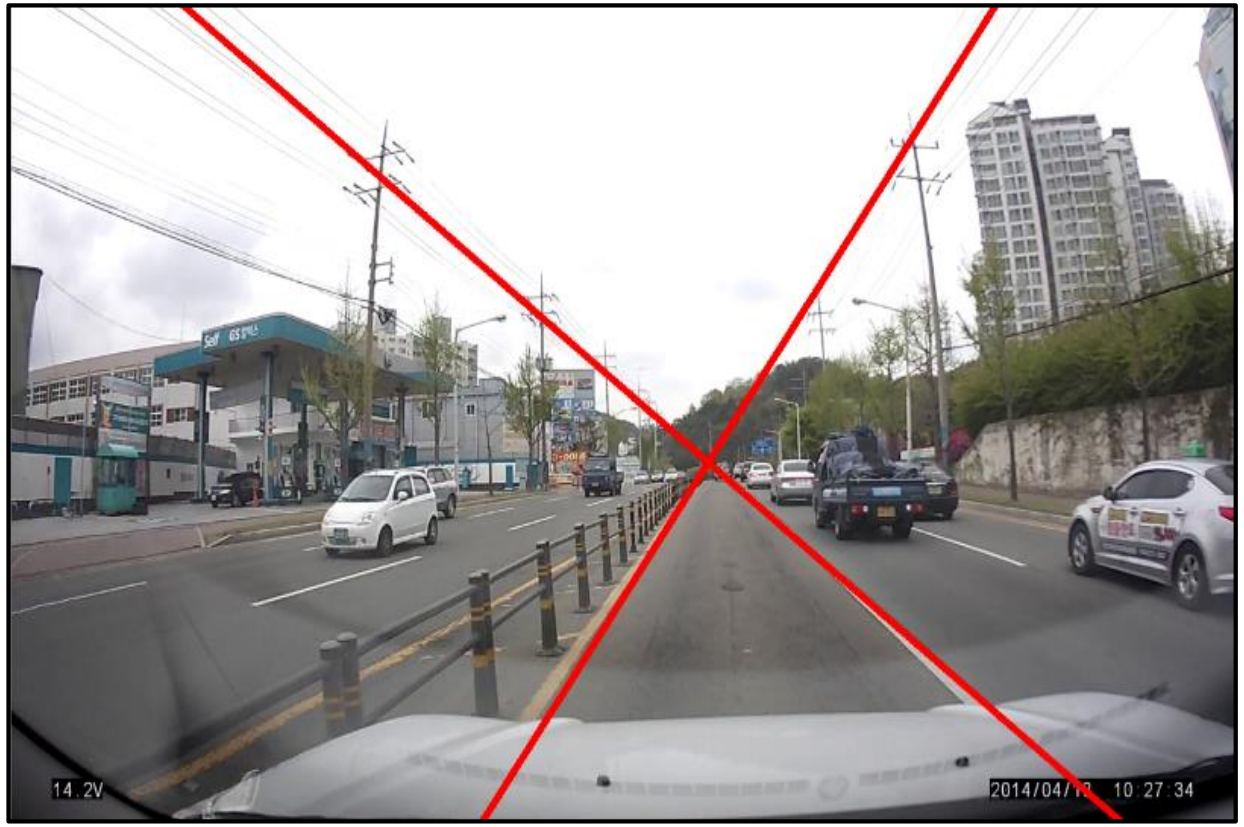


- 직선의 각도

- 직선의 통합



과제5 힌트: 최종 결과

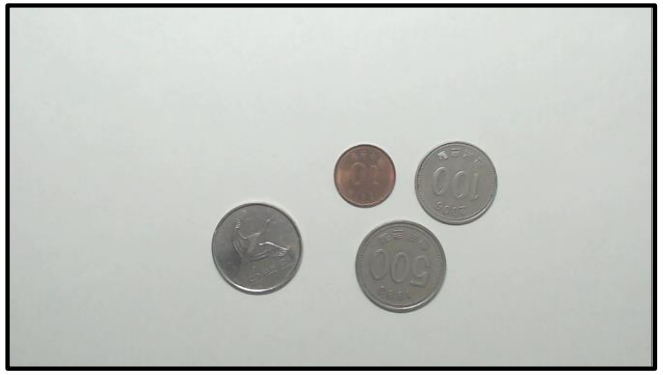


주간

야간

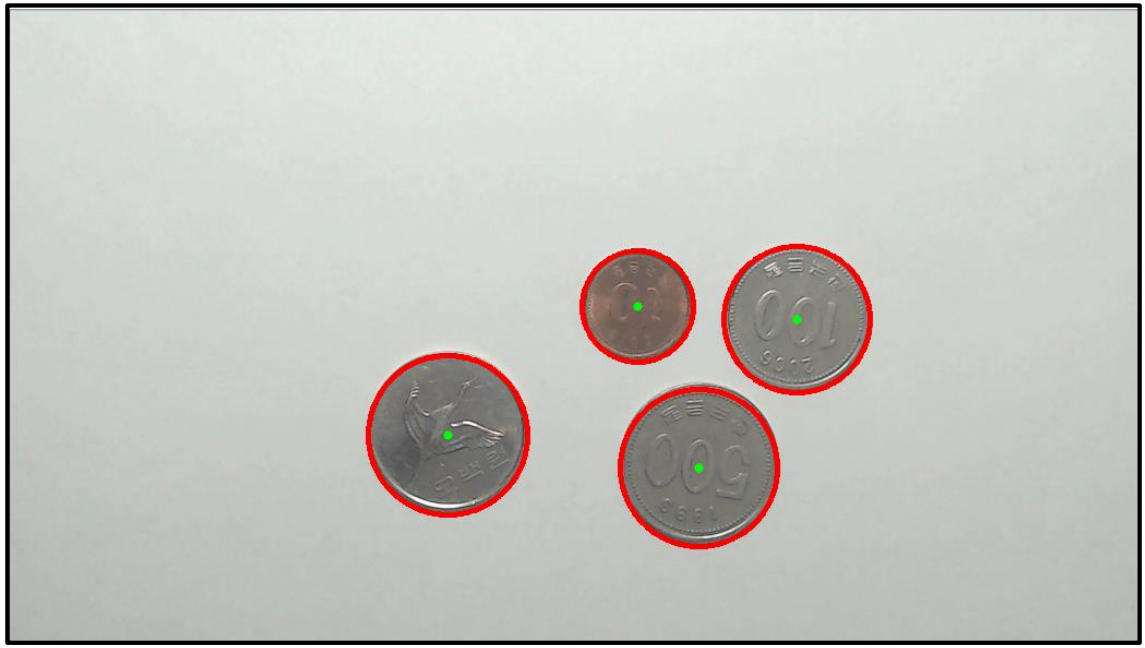
[과제6] 동전갯수 세기 (1): HoughCircles을 이용

문제: 입력 받은 영상을 그레이스케일로 만들고, 화면에 출력하되
(1) 입력을 Grayscale로 변환하고 영상 가로변의 길이를 500으로 하고 원 영상의 가로 세로 비를 유지하도록 세로변의 길이를 정하라.
(2) HoughCircles 변환함수를 이용하여 원을 검출하라 (동전 갯수가 5개 이하인 영상)
(3) 동전 갯수가 10개 이상인 영상에서도 테스트를 해보라.



입력이미지

중간 이미지
(grayscale 이미지)

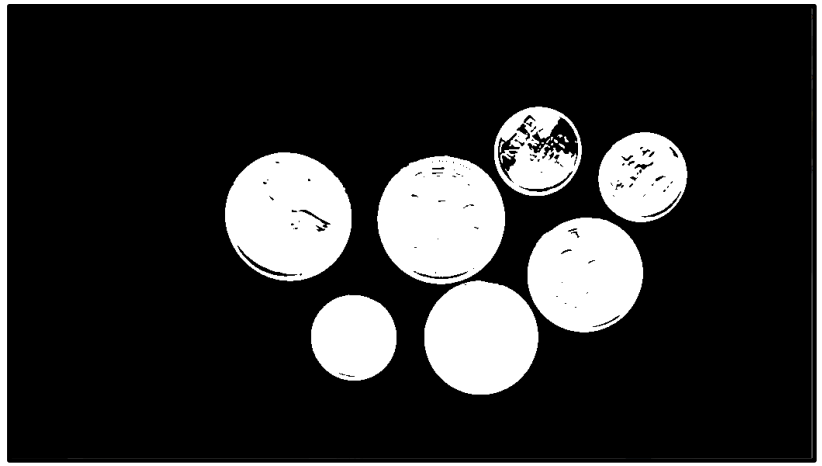


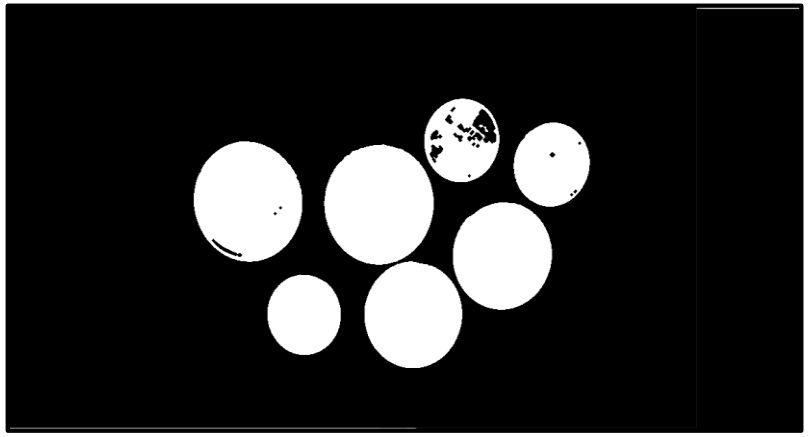
출력이미지
(붉은 테두리 원과 녹색 중점을 원마다 그려 동전 표시)

[과제7] 동전 갯수 세기 (2): 윤곽선 추출 이용

- 문제: 배경이 단색의 밝은 색이고 동전들이 서로 떨어져있는 경우 동전의 갯수를 세어라

(컬러변환) ➔ (노이즈제거) ➔ 이진화 ➔(에지추출) ➔ 모폴로지 ➔ 윤곽선 추출 ➔ 필터링

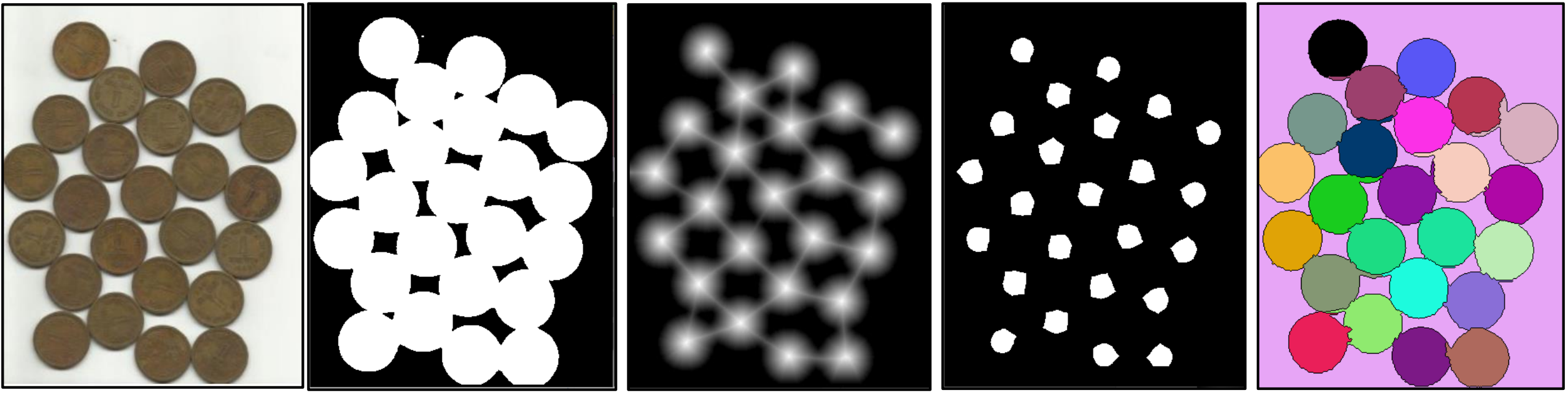





```
Coin 1 bb=[110 x 111 from (389, 370)]
Coin 2 bb=[147 x 147 from (535, 352)]
Coin 3 bb=[149 x 148 from (668, 271)]
Coin 4 bb=[165 x 166 from (474, 191)]
Coin 5 bb=[163 x 166 from (278, 187)]
Coin 6 bb=[115 x 116 from (759, 161)]
Coin 7 bb=[113 x 115 from (625, 128)]
coint#=7
```

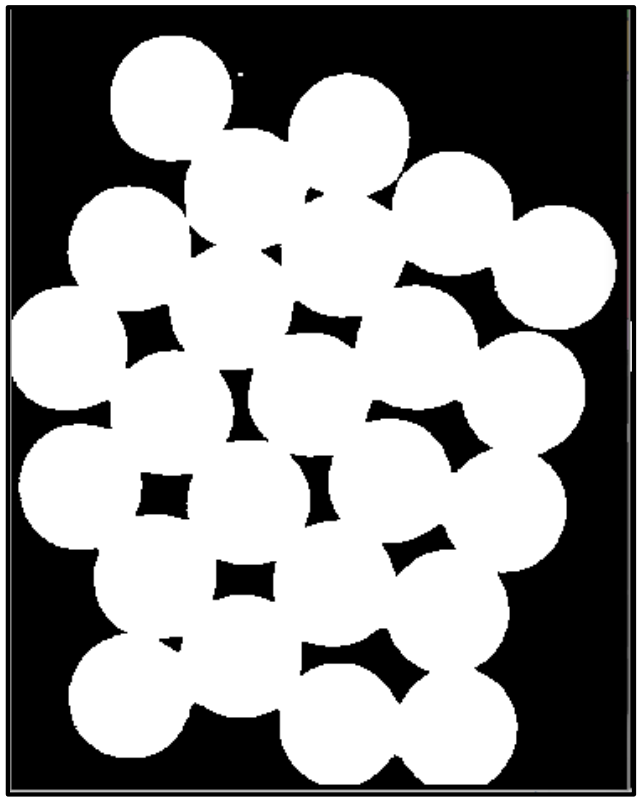
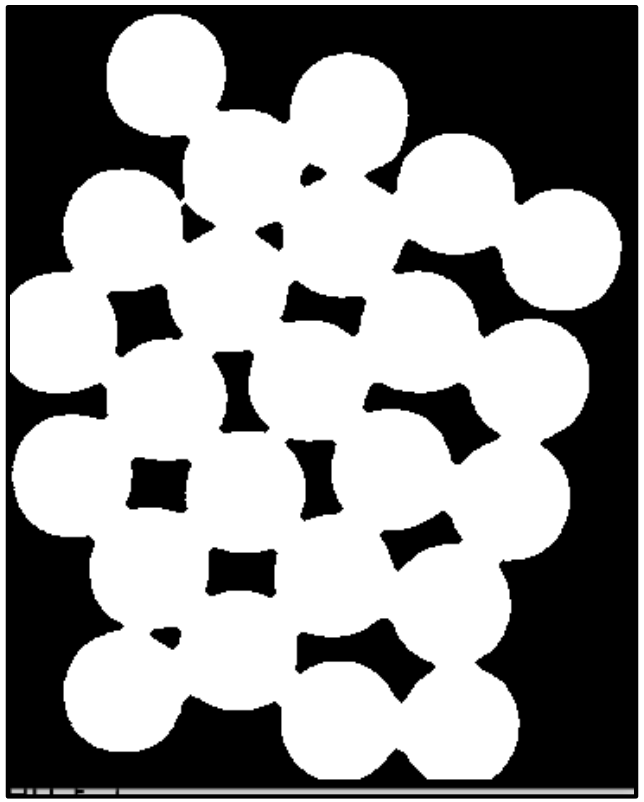
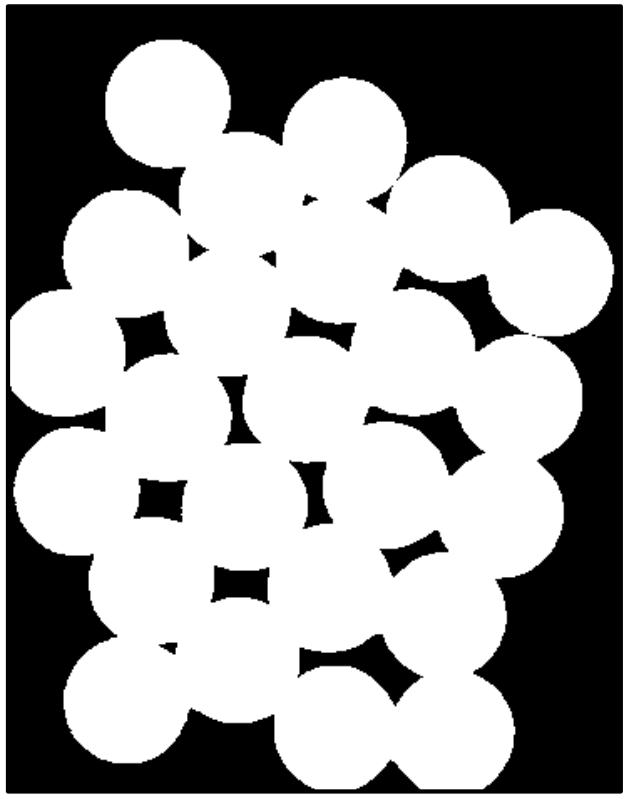
[과제8] 동전갯수 세기 (3): distanceTransform & watershed 이용

- 이진화 -> 모폴로지 -> distance transform -> watershed



동전세기 (3): 이진화

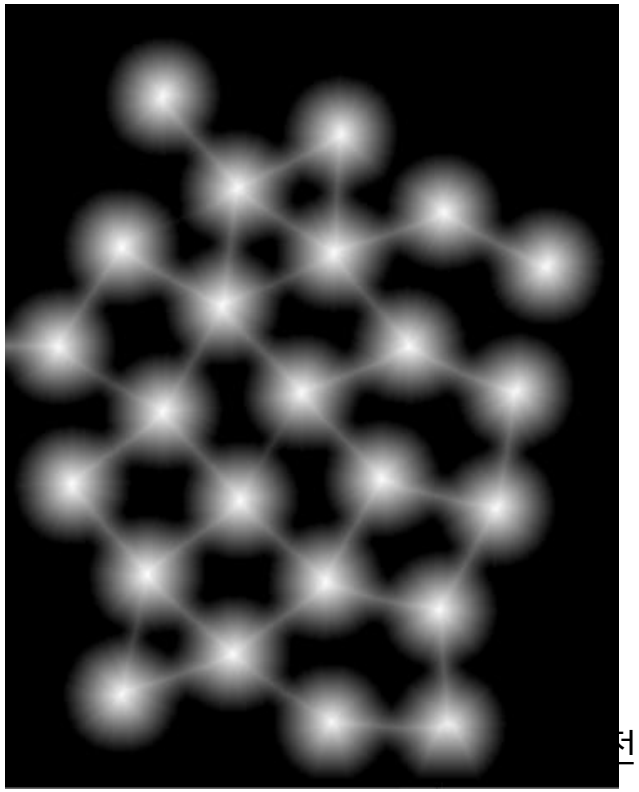
- Grayscale 로 변환 → 이진화 (OTSU) → BW반전
- → 모폴로지 연산 (침식 2번, 팽창 3번)

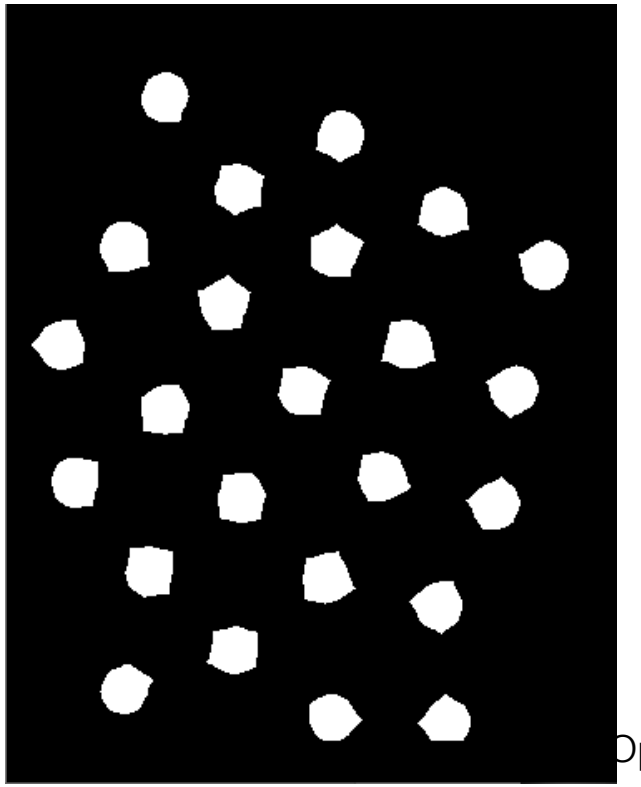
동전세기(3) : distance transform

1. foreground opening에 distance transform 적용 ➔ grayscale 이미지
2. normalize ➔ 0.0 ~ 1.0 픽셀
3. threshold : normalize된 이미지에 대해 max 값의 60% 이상을 기준으로 이진화
4. 8-bit single-channel 이미지로 type 변환

distance transform 결과

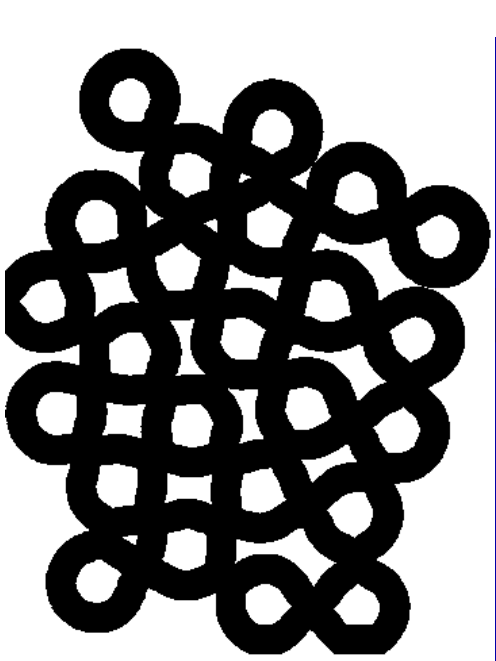


threshold 결과

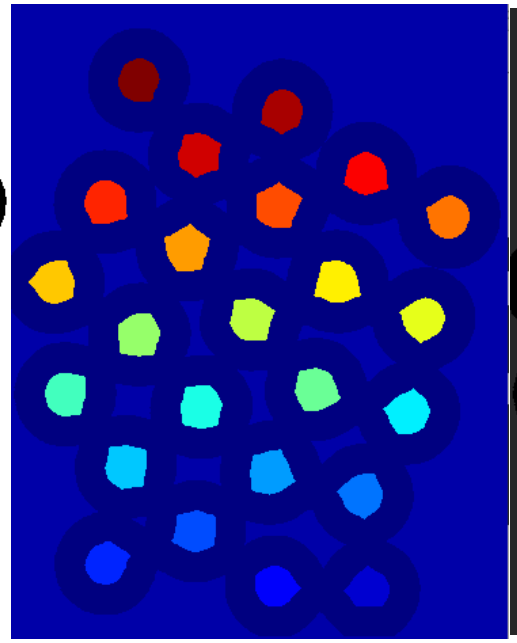


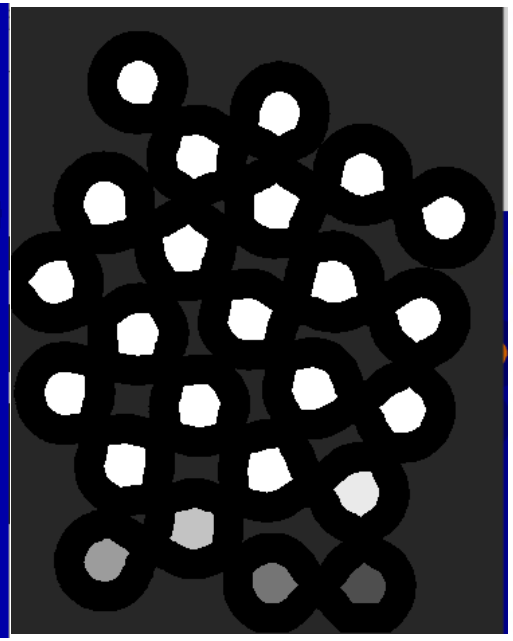
동전세기: watershed 적용

1. undefined 영역 정의 = sure_background – sure_foreground : 향후 watershed 알고리즘에 의해 어느 영역인지 정의될 부분
2. 일부 픽셀에 seed 값 할당
   - 각각의 sure_foreground 영역 픽셀에 각기 다른 seed값 (양의 정수) 배정
   - sure_background 영역 픽셀에도 (foreground 영역과 다른) seed값 배정
   - undefined 영역 픽셀에 0 배정
3. watershed 함수 적용
   - 첫 인자는 입력 컬러 이미지, 둘째 인자는 markers (in/out)
   - 픽셀값이 0인 영역들이 seed 값 중의 하나로 변경
   - 경계선 픽셀은 -1 값이 됨

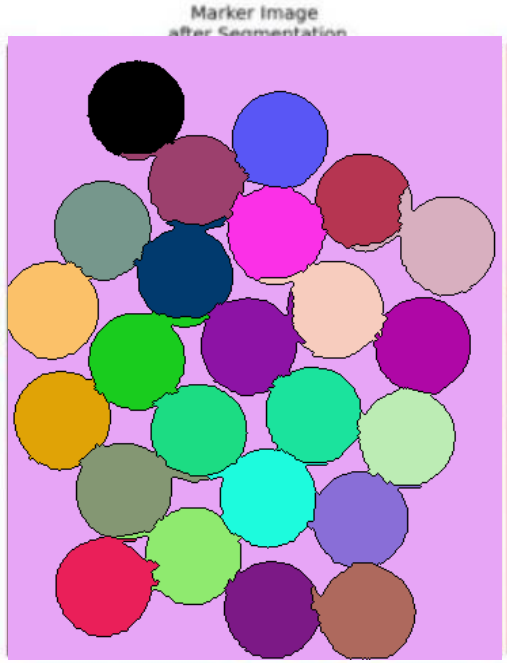


undefined 영역 (inverse)

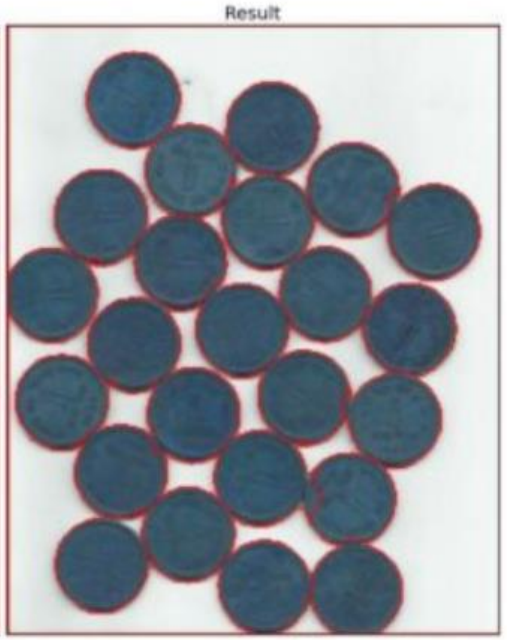




seed값 배정 (검은색은 0= undefined)



watershed 적용 결과



동전둘레 표시

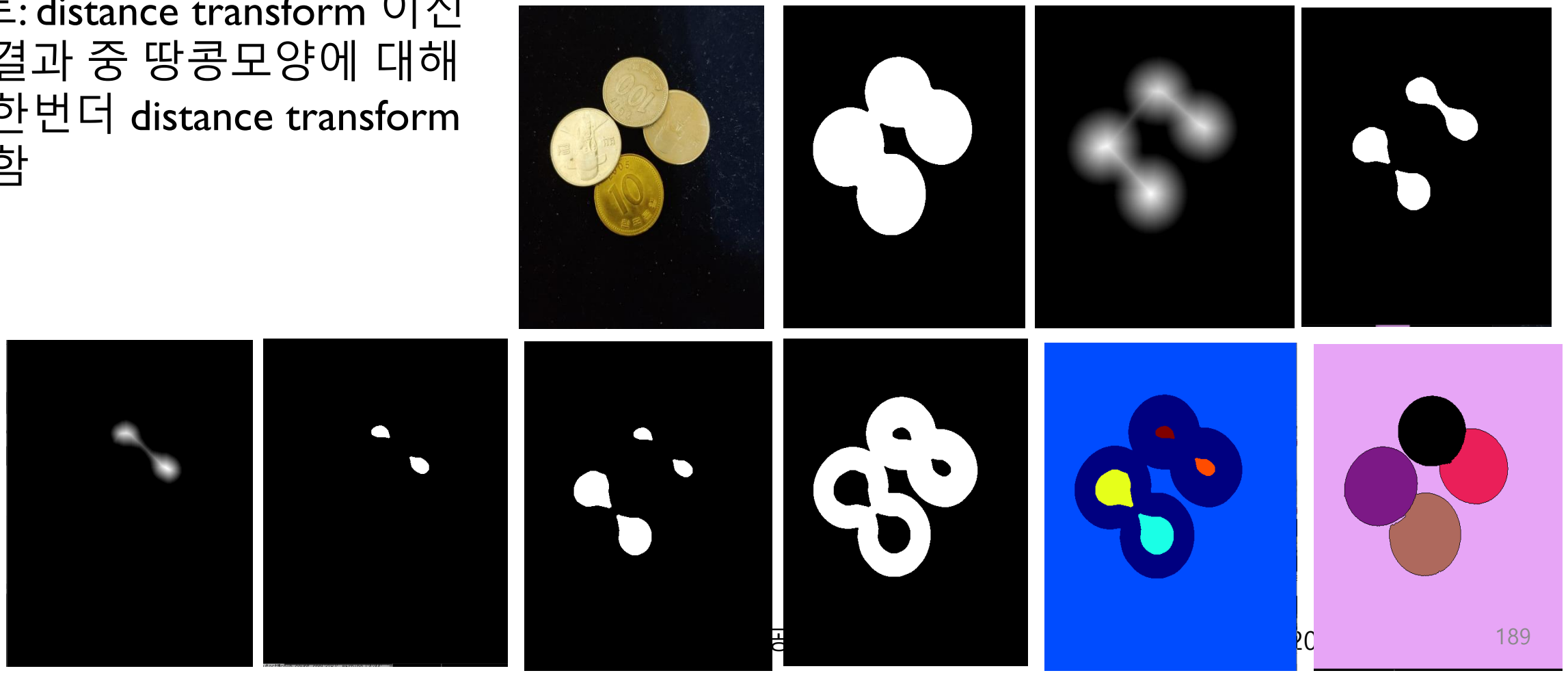
[과제 동전8] 세기 코드 테스트하기

- 개별 동전이 서로 분리된 경우
- 동전끼리 맞닿은 경우
- 여러 크기/색깔의 동전이 있는 경우
- 동전 면에 빛 반사가 심한 경우
- 배경에 그림자 등 노이즈가 많은 경우
- 동전끼리 겹쳐진 경우
- 복합적 문제 경우

[과제9] 겹쳐진 동전 세기

- [과제 8]에서 개발한 방법을 개선하여 동전이 겹쳐진 경우에도 카운트할 수 있도록 distance transform 결과에 postptocessing하여 seed를 정확히 생성하기

- 힌트: distance transform 이진화 결과 중 땅콩모양에 대해서 한번더 distance transform을 함



[과제10] 동영상에서 동전의 갯수 세기

- [과제 8-9]에서 개발한 방법을 사용하여 웹캠으로 입력된 영상에서 실시간으로 동전 갯수 실시간으로 세기

Mat 클래스

Mat

- n-dimensional dense numerical single or multi-channel array
- 주요 멤버 변수

  int flag ; // bit-flags for number of channels, depth, continuity flag

  int dims; // dimenstionality >=2

  int rows, cols; // number of columns , rows

  uchar *data; // pointer to data

  int *refcount ; // pointer to reference counter

  Msize size, step;
- N-차원의 경우, M(i_0,i_1,...i_{dim-1}) element 의 주소 계산식
  - addr(M(i_0,i_1,...i_{dim-1}) = M.data + M.step[0] * i_0 + M.step[1] * i_1 + ... + M.step[$_{dim}$] * i_{dim}
- 2-dimensio인 경우, M(i,j) element 의 주소 계산식
  - addr(Mi,j) = M.data + M.step[0] * i + M.step[1] * j
  - data 부분은 cvMat, IplImage와 fully compatible

Pixel type

```
Mat mtx(3, 3, CV_32F);
        // make a 3x3 floating-point matrix
Mat cmtx(10, 1, CV_64FC2);
        // make a 10x1 2-channel floating-point matrix (10-element complex vector)
Mat img(Size(1920, 1080), CV_8UC3);
        // make a 3-channel (color) image of 1920 columns and 1080 rows.
Mat grayscale(image.size(), CV_MAKETYPE(image.depth(), 1)); // make a 1-channel image
```

- Face detection은 8-bit grayscale or color image에 대해서만 작동
- Linear algebra 관련 함수나 Machine learning은 대부분 32-bit floating point에 대해서 작동
- Color space conversion은 8-bit unsigned, 16-bit unsigned, 32-bit floating point에 대해서만 작동

Pixel type (계속)

```
Mat mtx(3, 3, CV_32F);
        // make a 3x3 floating-point matrix
Mat cmtx(10, 1, CV_64FC2);
        // make a 10x1 2-channel floating-point matrix (10-element complex vector)
Mat img(Size(1920, 1080), CV_8UC3);
        // make a 3-channel (color) image of 1920 columns and 1080 rows.
Mat grayscale(image.size(), CV_MAKETYPE(image.depth(), 1)); // make a 1-channel image
```

- Face detection은 8-bit grayscale or color image에 대해서만 작동
- Linear algebra 관련 함수나 Machine learning은 대부분 32-bit floating point에 대해서 작동
- Color space conversion은 8-bit unsigned, 16-bit unsigned, 32-bit floating point에 대해서만 작동

Mat 생성자 종류

- Mat( )
- Mat (int rows, int cols, int type); // type is CV_8UC1, CV_64FC3, CV_32SC etc.
- Mat (Size size, int type);
- Mat (int rows, int cols, int type, const Scalar&s);
- Mat (Size size, int type, const Scalar& s);
- Mat (const Mat & m);
- Mat (int rows, int cols, int type, void *data, size_t step=AUTO_STEP);
- Mat (Size size, int type, void *data, size_t step=AUTO_STEP);
- Mat (const Mat &m, const Rect & roi); // copy part of m
- Mat (int ndim, const int* sizes, int type);
- Mat (int ndim, const int *sizes, int type, const Scalar & s);
- Mat (const Mat &m, Range *ranges); // copy part of m
- Mat (const CvMat*m, bool copyData=false); // old CvMat to Mat
- Mat (const IplImage *img, bool copyData=false); // old IplImage to Mat

Mat=header+data

- OpenCV의 이미지를 담는 data structure (container)
  - 초창기에는 C의 struct인 IplImage를 사용, 모든 메모리 할당과 관리를 프로그래머가 전적으로 책임짐
  - Mat : C++의 class, 프로그래머의 메모리 관리에 대한 부담을 줄임. OpenCV 2.0부터
- Mat은 헤더와 matrix 데이터 자체를 저장하는 data 구분됨
  - 헤더의 크기는 고정적임. Matrix 데이터 크기 및 저장소에 대한 포인터, 저장방법 등
  - Matrix 저장부는 동적으로 할당됨.
- Matrix data 복사 최소화
  - 함수 호출시 matrix 자체보다 reference를 pass함
  - Reference Counter를 사용해서 data를 여러 header가 공유함
  - 아래 예에서 하나의 matrix 저장소만 생성되고 나머지 Mat은 헤더만 생성 (copy operator는 header만 copy하고 data는 포인터로 공유하고 ref. counter 증가시킴)

```
Mat A, C; // creates just the header parts
A = imread(argv[1], CV_LOAD_IMAGE_COLOR); // here we'll know the method used (allocate matrix)
Mat B(A); // Use the copy constructor
C = A; // Assignment operator
```

Mat (이어서)

- 여러 Mat이 공통된 data matrix를 공유할 때, 한 Mat의 수정은 다른 Mat에도 동일한 영향을 미친다.
- 모든 Mat의 헤더가 clear되면 Referece counting mechanism에 의해서 자동을 데이터 메모리 반납됨
- Mat 자료구조에서 Data matrix의 일부만 사용하는 헤더도 가능함
  - ROI 지정 가능

```
Mat D (A, Rect(10, 10, 100, 100) ); // using a rectangle
Mat E = A(Range::all(), Range(1,3)); // using row and column boundaries
```

- Matrix의 header와 data부분을 모두 copy하는 것도 가능
  - clone()
  - copyTo()

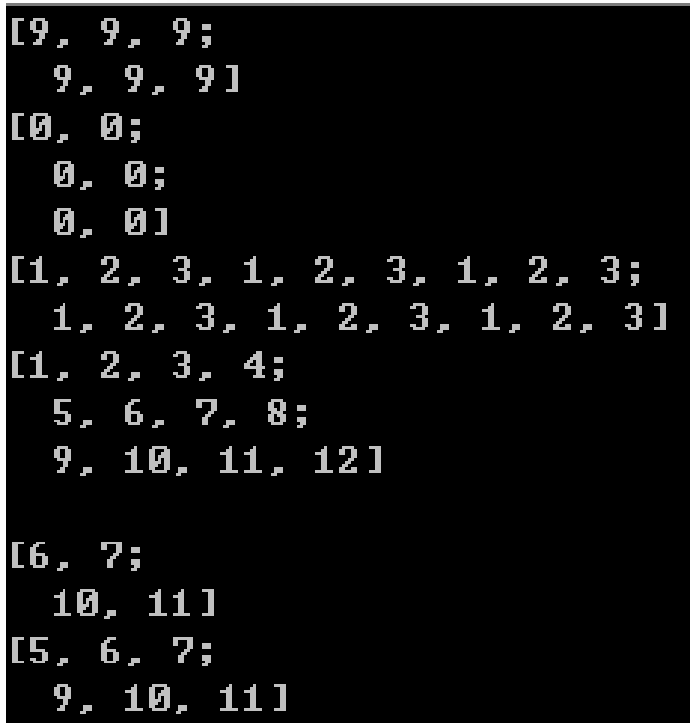
```
Mat F = A.clone();
Mat G; A.copyTo(G);
```

create 멤버함수에 의한 Mat 생성

- Mat::create는 ndims, rows, cols, type, size 등에 의해서 새로운 행렬을 만들되, 기존에 가지고 있던 행렬과 크기와 type이 다르면, Mat::release를 자동으로 호출하여 data 기억장소를 반환하고 새로운 데이터를 생성함.
- 대부분 openCV C++ 함수들은 처리결과를 저장하기 위해서 이 creat를 호출한다.
- 생성자
  - Mat::create(int rows, int cols, int type)
  - Mat::create(Size sz, int type)
  - Mat::create(int ndims, int *sizes, int type)

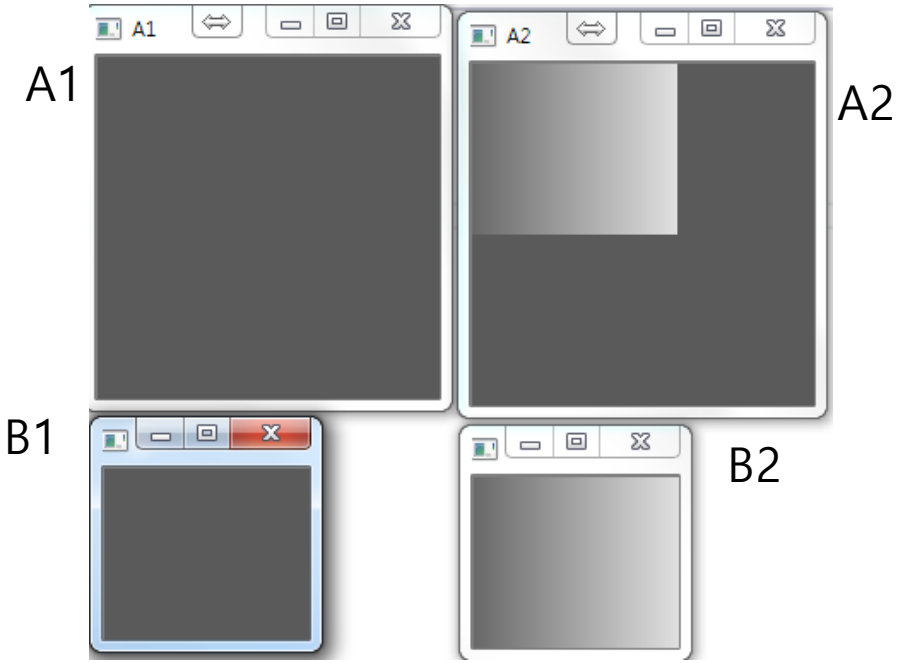
Mat 클래스 데이터 생성 예제 (1)

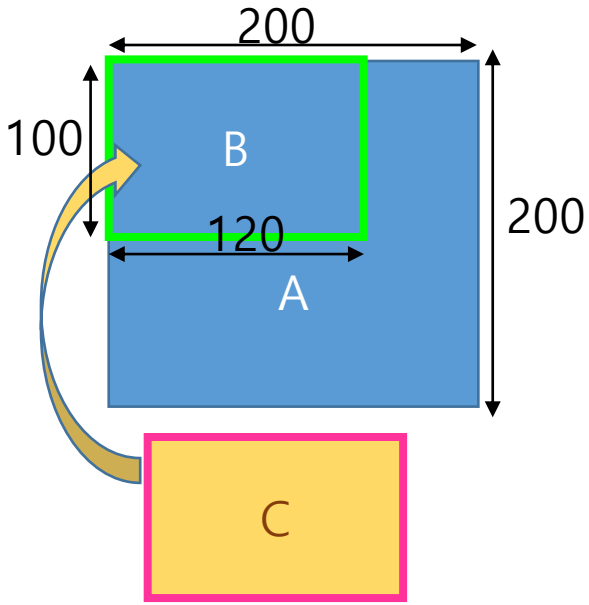
```
Mat A(2, 3, CV_8UC1, 9);
Mat B(Size(2,3), CV_8UC1, Scalar(0));
Mat C(2,3, CV_8UC3, Scalar(1, 2, 3));
float data[] = { 1, 2, 3, 4, 5, 6,7,8,9,10,11,12 };

Mat D(3, 4, CV_32FC1, data);
cout << A << endl << B << endl << C << endl << D << endl<<endl;

Mat D1(D, Rect(1,1, 2, 2));
Mat D2( D,Rect(0, 1, 3, 2));
cout << D1 << endl;
cout << D2 << endl;
```



Mat 클래스 데이터 생성 예제 (2)

```
Mat A = Mat(200, 200, CV_8UC1, 90);
Mat B = Mat(A, Rect(0, 0, 120,100));
imshow("B1", B); imshow("A1", A);

uchar dataB[100 * 120];
for (int i = 0; i < 100 * 120 ; i += 1)
    dataB[i]  = i % 120 + 105;
C = Mat(100, 120, CV_8UC1, dataB);
C.copyTo(B);

imshow("B2", B); imshow("A2", A);
waitKey(0);
```





Mat을 초기화하는 기타 방법들

- Identity 행렬 (eye)
- 1인 행렬
- 영행렬 (zeros)

```
Mat E = Mat::eye(4, 4, CV_64F);
cout << "E = " << endl << " " << E << endl;
Mat O = Mat::ones(2, 2, CV_32F);
cout << "O = " << endl << " " << O << endl;
Mat Z = Mat::zeros(3,3, CV_8UC1);
cout << "Z = " << endl << " " << Z << endl;
```

```
E=
 [1, 0, 0, 0;
  0, 1, 0, 0;
  0, 0, 1, 0;
  0, 0, 0, 1]
O=
 [1, 1;
  1, 1]
Z=
 [0, 0, 0;
  0, 0, 0;
  0, 0, 0]
```

- 작은 행렬의 간편한 초기화

```
Mat C = (Mat_<char>(3,3) << 0, -1, 0, -1, 5, -1, 0, -1, 0);
cout << "C = " << endl << " " << C << endl;
```

```
C =
 [0, -1, 0;
  -1, 5, -1;
  0, -1, 0]
```

- 존재 행렬의 열을 복사해서 생성

```
Mat RowClone = C.row(1).clone();
cout << "RowClone = " << endl << " " << RowClone << endl;
```

```
RowClone =
 [-1, 5, -1]
```

Mat::row, col rowRange, colRange 멤버 함수에 의한 Mat 생성 (부분행렬)

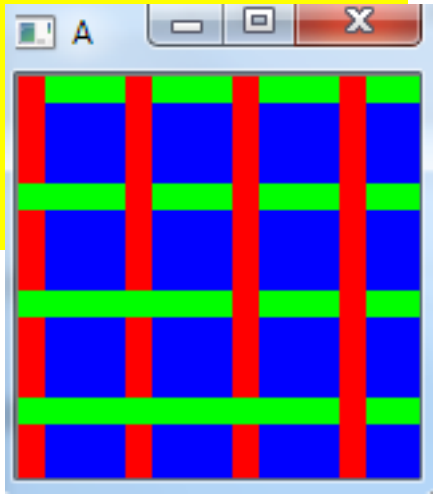
- row(), col(), rowRange(), colRange()함수들은, 기존 행렬 데이터의 일부를 가리키는 행렬을 생성한다. 즉, 데이터는 기존의 행렬 것을 공유하고 **헤더만 생성하여** retunrn한다.
- Mat Mat::row(int y) const
- Mat Mat::col(int x) const
- Mat Mat::rowRange(int start, int end) const
- Mat Mat::colRange(int start, int end) const

Row,Col,RowRange 예제

```
Mat M(3, 3, CV_32FC1);
for (int i = 0; i < M.rows; i++)
for (int j = 0; j < M.cols; j++)
M.at<float>(i, j) = i*M.cols + j;

Mat Q = M.row(0);
Mat P = M.col(0);
Mat R = M.colRange(0, 2);

cout << M << endl << Q << endl;
cout << P << endl << R << endl;
cout << endl;

M.row(0) = M.row(0) * 10;
M.row(1) = M.row(1) * 100;
M.row(1).copyTo(M.row(2));

cout << M << endl << Q << endl;
cout << P << endl << R << endl;
cout << endl;
```

```
[0, 1, 2;
  3, 4, 5;
  6, 7, 8]
[0, 1, 2]
[0;
  3;
  6]
[0, 1;
  3, 4;
  6, 7]

[0, 10, 20;
  300, 400, 500;
  300, 400, 500]
[0, 10, 20]
[0;
  300;
  300]
[0, 10;
  300, 400;
  300, 400]
```

```
Mat A(150, 150, CV_32FC3, Scalar(100, 0, 0));
Mat B;
for (int i = 0; i < A.rows; i += 20*2){
    B = A.rowRange(i, i + 10);
    B = Scalar(0, 100, 0);
    B = A.colRange(i, i + 10);
    B = Scalar(0, 0, 100);
}
imshow("A",A);
waitKey(0);
```

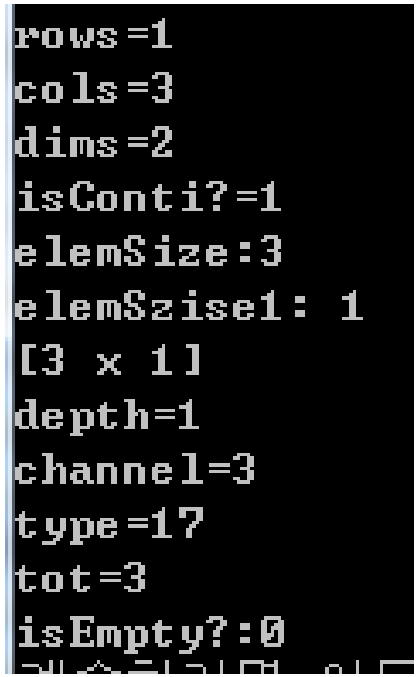


그밖의 Mat 멤버들

- 멤버변수
  - int rows, cols, dims: row, column, dimension,
  - int * refcount: pointer to referene counter
- 멤버함수
  - size_t total (): 행렬 요소의 갯수
  - bool isContinuous() : 각 행의 마지막에 공백이 없이 값이 저장되었는가? 즉, m.step == m.cols*m.elemSize() 인지를 확인 (row가 1개이거나 create로 생성된 경우는 true, 이미 만들어진 Mat의 일부를 col(), diag() 등으로 추출할 경우 false 가 될 수도 있음)
  - int type() : CV_8UC1, CV_8UC2, CV_8UC3, ... CV_64FC1, CV_64FC2, CV_64FC3 중
  - int depth () : CV_8U, CV_8S, CV_16U, CV_16S, CV_32_S, CV_32F, CV_64F 중
  - int channels() : 행렬의 채널의 갯수
  - Size size() : Size(cols,rows)
  - size_t elemSize() : 행렬 요소의 byte 단위 크기
  - size_t elemSize1() : 행렬 요소에서 채널 하나 당 요소의 byte 단위 크기
  - bool empty() : 공백 행렬인가?

Mat의 기타 멤버 함수 테스트

```
Mat A(1, 3, CV_8SC3);
cout << "rows=" << A.rows << endl;
cout << "cols=" << A.cols << endl;
cout << "dims=" << A.dims << endl;

cout << "isConti?=" << A.isContinuous() << endl;



cout << "elemSize:" << A.elemSize() << endl;
cout << "elemSzise1: " << A.elemSize1() << endl;
cout << A.size() << endl;
cout << "depth=" << A.depth() << endl;
cout << "channel=" << A.channels() << endl;
cout << "type=" << A.type() << endl;

cout << "tot="<< A.total() << endl;
cout << "isEmpty?:" <<A.empty() << endl;
```



주의)
1) Mat 생성시 인자는 rows, cols 순서 vs. Size는 width, height 순으로 다름
2) type = (channel-1)*8+depth=16+1
3) tot = rows * cols

Type conversion old to new Mat class

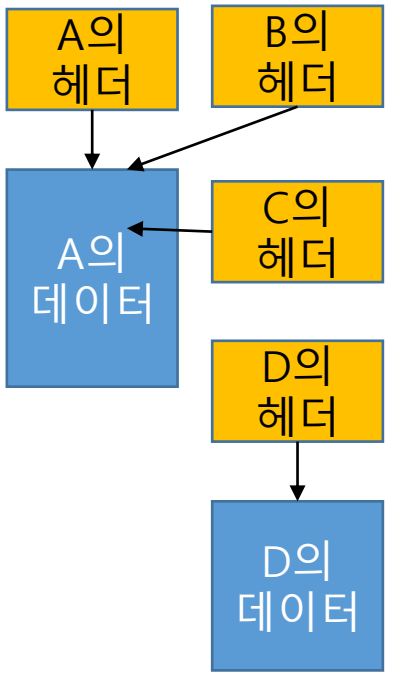
- Mat cvarrToMat(const CvArr * arr, bool copyData=false, bool allowND =true, int coiMode = 0)
  - Convert CVMat, IplImage, or CvMatND header to Mat header
  - copyData : optionally copy data if copyData == true

```
CvMat* A = cvCreateMat(10, 10, CV_32F);
cvSetIdentity(A);
IplImage A1;
cvGetImage(A, &A1);

Mat B = cvarrToMat(A);
Mat B1 = cvarrToMat(&A1);

IplImage C = B;
CvMat C1 = B1;
```

Automatic Memory management in openCV

- Destructor는 reference counte가 0일 때만 deallocation 수행
- Assignment, copyTo()는 데이터를 공유하고 reference counter만 증가시킴
- Clone() 은 실제로 모든 메모리값을 복사함

```
Mat A(1000, 1000, CV_64F); // 실제 메모리(8Mb)할당
Mat B = A; // 메모리 할당하지 않음.
Mat C = B.row(3); // 메모리 할당하지 않음. A의 3번째 row만 C가 포인팅함.
Mat D = B.clone(); // 실제 메모리 D에 할당하고 A의 3번째 row값을 복사함.


B.row(5).copyTo(C); // A의 5번째 row를 A의 3번째 row에 복사하는 효과
A = D; // A가 D와 데이터를 공유함. A가 가지고 있던 데이터는 C,B에 의해 유지됨
B.release(); // B를 empty로 만듬. 그러나, C가 여전히 데이터를 유지함
C = C.clone();// 원래 A의 배열은 반납되고 새로운 메모리할당
```



Automatic output memory allocation

```
VideoCapture cap(0);
Mat frame, edges;
namedWindow("edges",1);
for(;;) {
    cap >> frame;  // >>연산자에 의해서 frame에 메모리 자동 할당
    cvtColor(frame, edges, CV_BGR2GRAY); //cvtColor 함수가 edge에 메모리 자동 할당
    GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5); // 크기나 타입의 변화가 없기에 메모리할당이 없음.
    Canny(edges, edges, 0, 30, 3);  //
    imshow("edges", edges);
    if(waitKey(30) >= 0) break;
}
```

- Mat::create() method 에 의해서 Input과 output의 해상도나 type이 달라지면 자동으로 이전 메모리는 deallocate시키고 새 메모리를 alloc함.
- mixChannel과 CV::RNG::fill은 예외적으로 자동으로 메모리할당을 못함.
- Saturation arithmatics : 8(16)비트 사이의 범위로 축약 (결과가 32비트에는 적용불가)

$$I(x, y) = \min(\max(\text{round}(r), 0), 255)$$

```
I.at<uchar>(y, x) = saturate_cast<uchar>(r);
```

Mat 행렬 메모리 해제 및 확보

- Mat::release()
  - reference counter 값을 1 감소
  - 1감소 결과 counte가 0이 되면 data 메모리를 시스템에 반환하고 data = NULL로
- Mat::reserve(int row)
  - 예상되는 필요한 만큼의 data 메모리를 미리 할당받아서 메모리 할당이 일어나는 횟수를 줄이고자 함.
  - 인자는 row의 갯수
  - 이미 rows만큼의 row 데이터가 있는 경우는, 그 값은 보존되고 추가로 할당이 됨.

실습과제: iteration 방법들의 실행시간 비교하기

- 문제 : 읽어들인 이미지를 grayscale 800x600으로 resize하고, 행렬의 원소(픽셀) 값에 50을 더해서 출력하기

- 실행 시간 측정 방법

```
#include <time.h>
double t1,t2=0
t1= getTickCount();
// code to measure and observe
t2= (getTickCount()-t) /getTickFrequency()
```

OpenCV 기타 자료구조

Point_<_Tp> template class

- 2D Point Template Class
- 생성자
  - Point_()
  - Point_(_Tp x, _Tp y)
  - Point_(const Point_ &pt)
  - Point_(const Size_<_Tp> & sz)
  - Point_(const Vec<_Tp, 2> & v)
- 멤버 변수 (public)
  - _Tp x, y
- 멤버 연산자
  - Point_ & operator= (const Point_ pt);
  - operator Vec<_Tp, 2> ( )
  - template<typename _Tp> operator Point_ <Tp, 2>( )

```
typedef Point_<int> Point2i;
typedef Point2i Point;
typedef Point_<float> Point2f;
typedef Point_<double> Point2d;
```

```
Point2f  pt1(0.1f, 0.2f), pt2 (0.3f, 0.4f);
Point pt3 = (pt1 + pt2)*10.0f;
Point2f pt4 = (pt1 – pt2) * 10.0f;
Point pt5 = Point (10,10);
cout << pt1 <<endl << pt2 <<endl;
cout << pt3 << endl << pt4 << endl;
cout << pt5 << endl;
```

```
[0.1, 0.2]
[0.3, 0.4]
[4, 6]
[-2, -2]
[10, 10]
```

Size_ <_Tp> template class

- Template class for specifying size of an image or rectangle (width, height)
- 생성자
  - Size_ ( )
  - Size_ (_Tp _width, _Tp _height);
  - Size_ (const Size_ & sz);
  - Size_ (const Point_ <_Tp> & pt)
- 멤버변수(public)
  - _Tp **width, height**

```
typedef Size_<int> Size2i;
typedef Size2i Size;
typedef Size_<float> Size2f;
```

```
Size sz1 (320, 240), sz2 (640, 480);
Size sz3 = sz1 * 2;
Size sz4 = sz1 + sz2;
Size sz5 = Size(800, 600);
cout << sz1 << endl << sz2 << endl;
if(sz2 == sz3)
    cout << sz3 << endl;
```

```
[320 x 240]
[640 x 480]
[640 x 480]
```

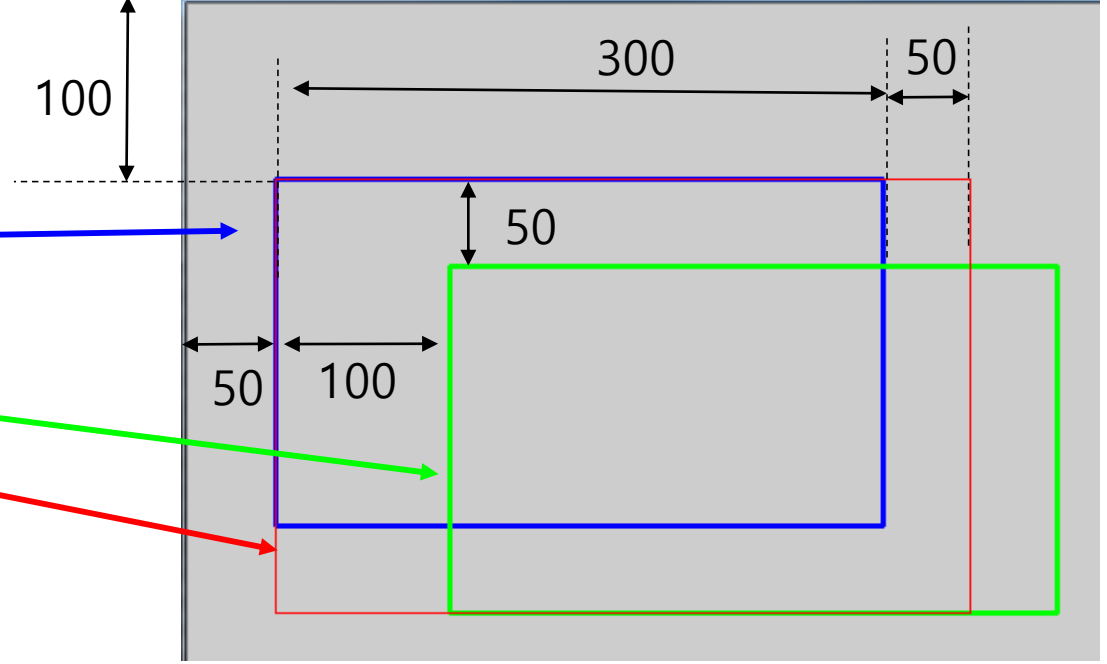
Rect_ <Tp_> template class

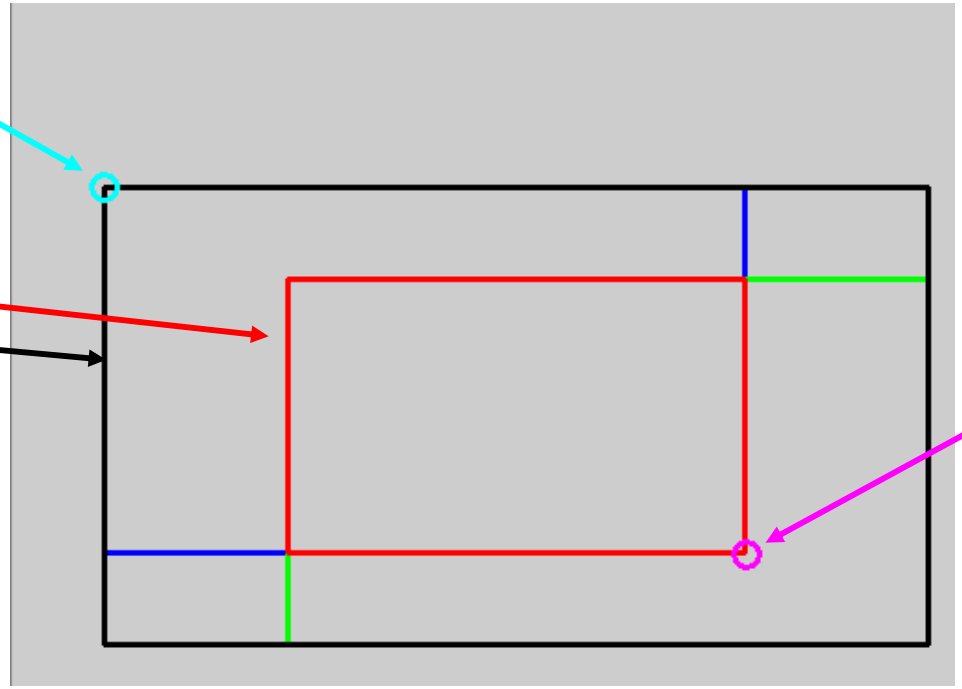
- Template class for 2D rectangle : coordinate of top-left corner + (width, height)
- 멤버 변수 (public)
  - _Tp **x, y, width, height**
- 생성자
  - Rect_ ( )
  - Rect_ (_Tp _x, _Tp _y, _Tp _width, _Tp _height)
  - Rect_ (const Rect_ & r)
  - Rect_ (const Point_<_Tp> &org, const Size_<_Tp> & sz)
  - Rect_ (const Point_<_Tp> &pt1, const Point_<_Tp> &pt2)
- 멤버함수
  - Point_<_Tp> tl( ) const; // top-left corner
  - Point_<_Tp> br( ) const; // bottom-right corner
  - _Tp area( ) const // area (width * height)
  - Rect_ & operator= (const Rect_& r);
  - bool contains (const Point_<_Tp> & pt) const; // check whether the rectangle contains the pt
- 연산
  - rect = rect + point (shift a rectangle by point)
  - rect = rect + size (expand or shrink by size)
  - rect = rect1 & rect2 (rectangle intersection)
  - rect = rect1 | rect2 (minimum rectangle containing rec2 and rec1)

typedef Rect_<int> **Rect**

Examples

```
Ret rt1 (50, 100, 350, 200);
Point pt1 (100,50);
Size sz (50,50);
Rect rt3 = rt1 + pt1; // Rect offset 변화
Rect rt4 = rt1 + sz; // Rect 크기 변화

Point ptTL = rt1.tl();
Point ptBR = rt1.br();
Rect rt5 = rt1 & rt2;
Rect rt6 = rt1 | rt2;

Point ptTL = rt1.tl();
Point ptBR = rt1.br();
Rect rt5 = rt1 & rt3;
Rect rt6 = rt1 | rt3;

Mat img(600, 800, CV_8UC3);
rectangle(img, rt1, Scalar(255, 0, 0), 2);
rectangle(img, rt3, Scalar(0, 255, 0), 2);
rectangle(img, rt5, Scalar(0, 0, 255), 2);
rectangle(img, rt6, Scalar(0, 0, 0), 2);
circle(img, ptTL, 7, Scalar(255, 255, 0), 2);
circle(img, ptBR, 7, Scalar(255, 0, 255), 2);
waitKey(0);
```

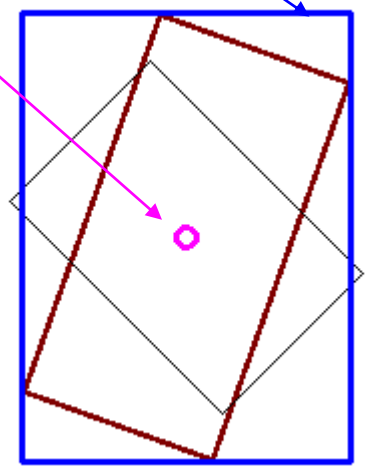




RotatedRect class

- 회전된 사각형을 표현하는 클래스
- member 변수
  - Point2f center; // 회전 중심 좌표
  - Size2f size; // 사각형 크기
  - float angle ; // 회전각 (시계방향(
- member 함수
  - Rect boundingRect() ; // 회전한 사각형을 둘러싸는 사각형
  - void points(Point2f pt[]) // 4개의 Points return
- 생성자
  - RotatedRect();
  - RotatedRect(const Point2f & center, const Size2f & size, float angle);

```
Point2f center(200.0f, 200.0f);
Size siz1(100, 200), siz2(100, 150);
RotatedRect rt1(center, siz1, 20.0f);
RotatedRect rt2(center, siz2, -45.0f);
Point2f points[4], points2[4];

rt1.points(points);
rt2.points(points2);
Rect rt3 = rt1.boundingRect();
Mat img(400, 400, CV_8UC3, Scalar(255, 255, 255));
for (int i = 0; i < 4; i++) {
  line(img, points[i], points[(i + 1) % 4], Scalar(0, 0, 128),2);
  line(img, points2[i], points2[(i + 1) % 4], Scalar(0, 0, 0),1);
}
rectangle(img, rt3, Scalar(255, 0, 0), 2);
circle(img, center, 5, Scalar(255, 0, 255), 2);
imshow("img", img);
waitKey(0);
```



boundingRect, minAreaRect

```
vector<RotatedRect> boundRect, candidates;
for (int i = 0; i < contours.size(); i++){
    Rect fullrect = boundingRect(contours[i]);
    if(fullrect.size().width > 100 && fullrect.size().heigh > 100)
            boundRect[i] = minAreaRect(contours[i]);
}
for (int i = 0; i< contours.size(); i++)
{
    float ratio = 0.0f;
    if (boundRect[i].size.height > boundRect[i].size.width)
        ratio = boundRect[i].size.height / boundRect[i].size.width;
    else
        ratio = boundRect[i].size.width / boundRect[i].size.height;

    if (ratio < 1.2 && boundRect[i].boundingRect().area()>500 )//실제 물체 판별 과정
        candidates.push_back(boundRect[i]);
}
```

Vec class

- Template class for short numerical vectors (special case of Matx)

```
template<typename _Tp, int n> class Vec : public Matx<_Tp, n, 1> {...};
```

```
typedef Vec<uchar, 2> Vec2b;
typedef Vec<uchar, 3> Vec3b;
typedef Vec<uchar, 4> Vec4b;
typedef Vec<short, 2> Vec2s;
typedef Vec<short, 3> Vec3s;
typedef Vec<short, 4> Vec4s;
typedef Vec<int, 2> Vec2i;
typedef Vec<int, 3> Vec3i;
typedef Vec<int, 4> Vec4i;
typedef Vec<float, 2> Vec2f;
typedef Vec<float, 3> Vec3f;
typedef Vec<float, 4> Vec4f;
typedef Vec<double, 2> Vec2d;
typedef Vec<double, 3> Vec3d;
typedef Vec<double, 4> Vec4d;
```

use [ ] operator to access elements of Vec
following operations are all implemented

- v1 = v2 + v3
- v1 = v2 – v3
- v1 = v2 * scale
- v1 = scale * v1
- v1 != v2
- v1 == v2
- v1 = v2.cross(v3)
- v1 = v2.mul(v3)
- scalar = v2.dot(v3)

```
Vec<float, 3> X(10, 0, 0), Y(0, 20, 0);
Vec3f Z = X.cross(Y); // cross product
cout << X << endl << Y << endl;
cout << Z << endl;
X = Vec3f(10, 20, 30);
Y = Vec3f(1, 2, 3);
Z = X.mul(Y);
cout << X<<" mul " << Y<<  "=" << Z << endl;
cout << "dot " << X.dot(Y) << endl;
Z = X + Y - Vec3f(0, 20, 3);
Z = 10 * Z;
Z = Z * 10;
cout << Z[0] << ',' << Z[1] << ',' << Z[2] << endl;
```

```
[10, 0, 0]
[0, 20, 0]
[0, 0, 200]
[10, 20, 30] mul [1, 2, 3]=[10, 40, 90]
dot 140
1100,200,3000
```

Scalar_<_Tp> class

- template class for a 4-element vector derived from Vec

template<typename _Tp> class Scalar_ : public Vec<_Tp,4> { … }

- 생성자
  - Scalar_ ( );
  - Scalar_ (_tp v0, _Tp v1, _Tp v2 = 0, _Tp v3 = 0)
  - Scalar_ (_Tp v0);
- typedef Scalar_<double> **Scalar**

```
[1, 2, 3, 4]
[10, 20, 30, 0]
[100, 200, 300, 0]
[459.83, 459.83, 459.83, 459.83]
[255, 0, 0, 0]
[255, 255, 255, 255]
```

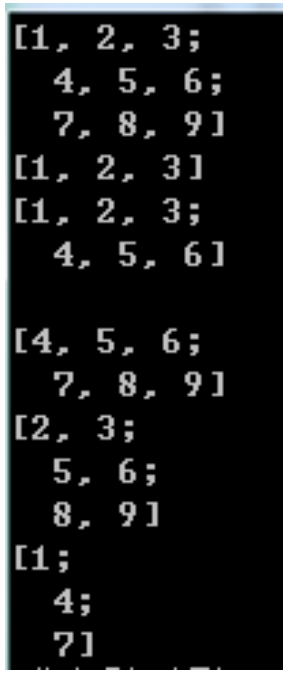
```
Scalar X = Vec4f(1, 2, 3, 4);
Scalar Y = Scalar(10, 20, 30);
Scalar Z = Scalar(100, 200, 300);
cout << X << endl << Y << endl
        << Z << endl;
Scalar X1 = Scalar::all(459.83);
cout << X1 << endl;
Scalar_<uchar> S1;
S1 = Scalar_<uchar>(255, 0, 0);
cout << S1 << endl;

S1 = Scalar_<uchar>::all(255);
cout << S1 << endl;
```

Range class

- Template class specifying a continuous subsequence of sequence
- used for specify row or column span in a matrix(Mat)
- Range (a,b) is the same as a:b in Matlab or a..b in Python

```
class Range {
 public:
  Range();
  Range(int _start, int _end);
  int size( ) const;
  bool empty( ) const;
  static Range all( );
  int start, end;
}
```

```
Matx33f A(1, 2, 3,
          4, 5, 6,
          7, 8, 9);
Mat B(A);
cout << B << endl;
cout << B(Range(0, 1), Range(0, 3)) << endl;
cout << B(Range(0, 2), Range(0, 3)) << endl <<endl;

Mat C = B(Range(1, 3), Range::all());
cout << C << endl;
C = B(Range::all(), Range(1, 3));
cout << C << endl;
C = B(Range(0, 3), Range(0, 1));
cout << C << endl;
```



Matx

- **작은 Matrix** 를 나타내는 template class
  - 주로 1x1에서 6x6까지 나타냄
- template <typename _Tp, int m, int n> class Matx
  - typedef Matx<float, 1,2> Matx12f;
  - typedef Matx<double, 1,2> Matx12d;

  .....
  - typedef Matx<float, 6, 6> Matx66f;
  - typedef Matx<double, 6,6> Matx66d;
- 없는 연산자는 Mat 타입으로 변형하여 사용

```
Matx23f A (1,2,3,
           4,5,6);
cout << "A=" << (Mat)A << endl;
Matx13f A0 = A.row(0);
Matx21f A1= A.col(1);
cout << "A0= " << (Mat)A0 <<endl;
cout << "A1= " << (Mat)A1 << endl;

Matx23f B = Matx23f::all(10.0f);
Matx23f C,D,E;
C = A + B;
D = A – B;
E = A * 2;
cout << "C= " << (Mat)C <<endl;
cout << "D= " << (Mat)D << endl;
cout << "E= " << (Mat)E << endl;

Matx33f F = Matx33f::zeros();
Matx33f G = Matx33f::ones();
Matx33f H = Matx33f::eye();
cout << "F= " << Mat(F) <<endl;
cout << "G= " << Mat(G) <<endl;
cout << "H= " << Mat(H) <<endl;
```

```
A = [1, 2, 3;
  4, 5, 6]
A0 = [1, 2, 3]
A1 = [2;
  5]
B= [10, 10, 10;
  10, 10, 10]
C = [11, 12, 13;
  14, 15, 16]
D = [-9, -8, -7;
  -6, -5, -4]
E = [2, 4, 6;
  8, 10, 12]
F = [0, 0, 0;
  0, 0, 0;
  0, 0, 0]
G = [1, 1, 1;
  1, 1, 1;
  1, 1, 1]
H = [1, 0, 0;
  0, 1, 0;
  0, 0, 1]
```

OpenCV 화일 입출력 API

외부 영상 화일 읽어오기

- 영상저장 데이터 구조: Mat
- 정지영상 화일 읽어들이기

```
Mat img;

img = imread ("test.jpg",1);
// jpg, bmp,
```

외부 영상 화일 읽어오기

- 실시간으로 영상 화일 프레임 읽어들이기

```
Mat frame;
VideoCapture cap("test.avi");

while (1) {
   cap >> frame;
// process on frame
}
```

```
Mat frame;
Video cap;
if(cap.open("test.avi") ==0)
        return 0;
while(1) {
    if (cap.grab() == 0)
         break;
     cap.retrieve(frame);
    // process in frame
}
```

외부 웹캠으로 실시간 영상 읽어오기

```
Mat frame;
// capture from webcam
int deviceID = 0;
VideoCapture cap(deviceID);

while(1) {
    cap >> frame;
    // process on img
}
```

```
Mat frame;
// capture from webcam
int deviceID = 0;
VideoCapture cap(deviceID);

while(1) {
    if (cap.grab() == 0)
        break;
    cap.retrieve(frame);
    // process in frame
}
```

image read : 정지영상 화일 읽기

Mat **imread**(const string& **filename**, int **flags**=1)

- The function determine the type of image by content
- supported type: BMP, JPEG, JEPG2000, PNG, TIF, PGM, PPM
- filename: Name of file to be loaded
- flag :
  - >0 : return a 3-channel color image
  - = 0 : return a grayscale
  - < 0 : return loaded image sa is (with alpha channel)
- returns NULL if improper file name

```
Mat img = imread("test.bmp",);
Mat gimg = imread ("test.bmp",0);
```

image write : Mat 값을 이미지화일로 저장하기

- bool **imwrite**(const string& filename, InputArray img, const vector<int>& params=vector<int>())
- image format is chosen based on the filename extension
- param: format specific
  - for JPEG, 0 ~ 100 (highest quality), default = 95
  - for PNG: 0 ~ 9 (highest compression level) , default = 3

화면에 Mat값 이미지로 출력

- void **imshow** (const string &winname, InputArray mat)
  - winname: Name of window
  - if the window was with created CV_WINDOW_AUTOSIZE flag, the image is shown with original size.
  - Otherwise, image is scaled to fit the window
  - if the window was not created before the function call, it automatically create window with CV_WINDOW_AUTOSIZE
  - the function scales the image depending on its depth
    - if image is 8-bit unsigned : it displays as is
    - If image is 16-bit or 32-bit unsigned : it is divided by 256
    - If the image is 32-bit float, the pixel values are multiplied by 255
  - the function should be followed by waitKey (m) that displays for m milisec
    - wiatKey(0) displays indefinitely until any key is pressed

Window 생성, 소멸

- void **namedWindow** (**const** string &winname, int flag = WINDO_AUTOSIZE)
  - winname: Name of window
  - flag : WINDOW_NORMAL (user can resize window)
  - WINDOW_AUTOSIZE (window size is automatically adjusted to fit image size)
- void **destroyWindow(cosnt** string &winname)
  - destroys the window with the given name
- void **destroyAllWindows**()
  - destroy all opened HighGUI windows
- void **moveWindow**(**const** string & winname, int x, int y)
  - Moves window with given name to the specified position (x,y)
- int **waitKey** (int delay = 0)
  - waits for a keypress event for a given delay ms (delay >0) or infinitely (delay<=0)
  - it works if there is at least one HighGUI window created and active.

영상 데이터 처리 기본 API

Grayscale영상 얻기

- **cvtColor 함수로 변환하기**

```
Mat img, gray;
img = imread("color_img.jpg);
cvtColor(img, gray, CV_BGR2GRAY);
imshow("gray image", gray);
```

- 읽어들일때 처음부터 그레이스케일로 변환하여 읽기
  - Mat img = imread("filename", **0**);

컬러 공간 변환하기

- cvtColor 함수

  void **cvtColor**(Mat src, Mat dst, int code, int dstCn =0)
  - Convert an image frame one color space to another
  - Code:
    CV_BRG2GRAY,
    CV_BRG2HSV,
    CV_BGR2YCrCb,
    CV_BGR2Lab,
    CV_GRAY2BGR
  - dstcn : destination channel number. If 0, automatically determined by src and dst

영상 변환

- flip(Input Array src, OutputArray dst, int flipCode)
  - flipCode : flag to specify how to flip array
    - 0: flipping around x-axis (상하 대칭)
    - 1 : flipping around y-axis (좌우 대칭)
- resize (InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER_LINEAR)
  - dst : it has the size of dsize (when non-zero) or computed from src.size(), fx, fy
  - dsize: output image size, if 0 it is computed as **Size(round(fx*src.cols), round(fy*scr.rows))**
  - fx : scale factor of x, if 0 it is computed as **dsize.width/src.cols**
  - fy: scal factor of y, if 0 it is computed as **dsize.height/src.rows**
  - **interpolation : INTER_NEAREST, INTER_LINEAR(faster, OK), INTER_AREA(best for shrink), INTER_CUBIC (slow, best for enlarge)**

```
// resize src so that it fits the precreated dst
resize(src, dst, dst.size( ), 0, 0);
```

- void split(Mat src, Mat* mv)
  - Splits multi-channel array into separate single-channel arrays
  - mv: output array (vector of arrays) mv[c][I] = $src[I]_c$
    * the number of arrays must match src.channels()
- Void mixChannels (Mat * src, size_t nsrcs, Mat * dst, size_t ndst, const int *fromTo, size_t npairs)
  - nsrc: number of matrices in src
  - ndst : number of matrices in dst
  - fromTo : mapping table of channels fromTo[2*k] of src → fromTo[2*k+1] of dst
- merge

```
Mat rgba( 100, 100, CV_8UC4, Scalar(1,2,3,4) );
Mat bgr( rgba.rows, rgba.cols, CV_8UC3 );
Mat alpha( rgba.rows, rgba.cols, CV_8UC1 );
Mat out[] = { bgr, alpha }; // rgba[0] -> bgr[2], rgba[1] -> bgr[1], rgba[2] -> bgr[0], rgba[3] -> alpha[0]
int from_to[] = { 0,2, 1,1, 2,0, 3,3 };
mixChannels( &rgba, 1, out, 2, from_to, 4 );
```

Drawing Functions(1)

- void **rectangle**(Mat& **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)
- void **rectangle**(Mat& **img**, Rect **rec**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)
  - Thickness : negative value (CV_FILLED) means a filled rectangle

- void **line**(Mat& **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)

- void **circle**(Mat& **img**, Point **center**, int **radius**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)

Drawing Functions(2)

- void **fillConvexPoly**(Mat& **img**, const Point* **pts**, int **npts**, const Scalar& **color**, int **lineType**=8, int **shift**=0)
  - much faster than fillPoly
  - 가로방향 수평선에 polygon이 두번 만나기만 하면 모두 처리 가능함.

- void **fillPoly**(Mat& **img**, const Point** **pts**, const int* **npts**, int **ncontours**, const Scalar& **color**, int **lineType**=8, int **shift**=0, Point **offset**=Point())

- void **putText**(Mat& **img**, const string& **text**, Point **org**, int **fontFace**, double **fontScale**, Scalar **color**, int **thickness**=1, int **lineType**=8, bool **bottomLeftOrigin**=false
  - fontFace :

    FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN,
    FONT_HERSHEY_DUPLEX, FONT_HERSHEY_COMPLEX,
    FONT_HERSHEY_TRIPLEX, … can be combined with FONT_HERSHEY_ITALIC

Mat의 ROI 지정과 수정

- ROI (Region of Interest)는 전체 Mat 행렬에서 일부 지정된 부분 행렬을 말한다.
- 영상에서 ROI 지정은 Range 클래스와 Rect 클래스를 사용하는 방법이 있다.

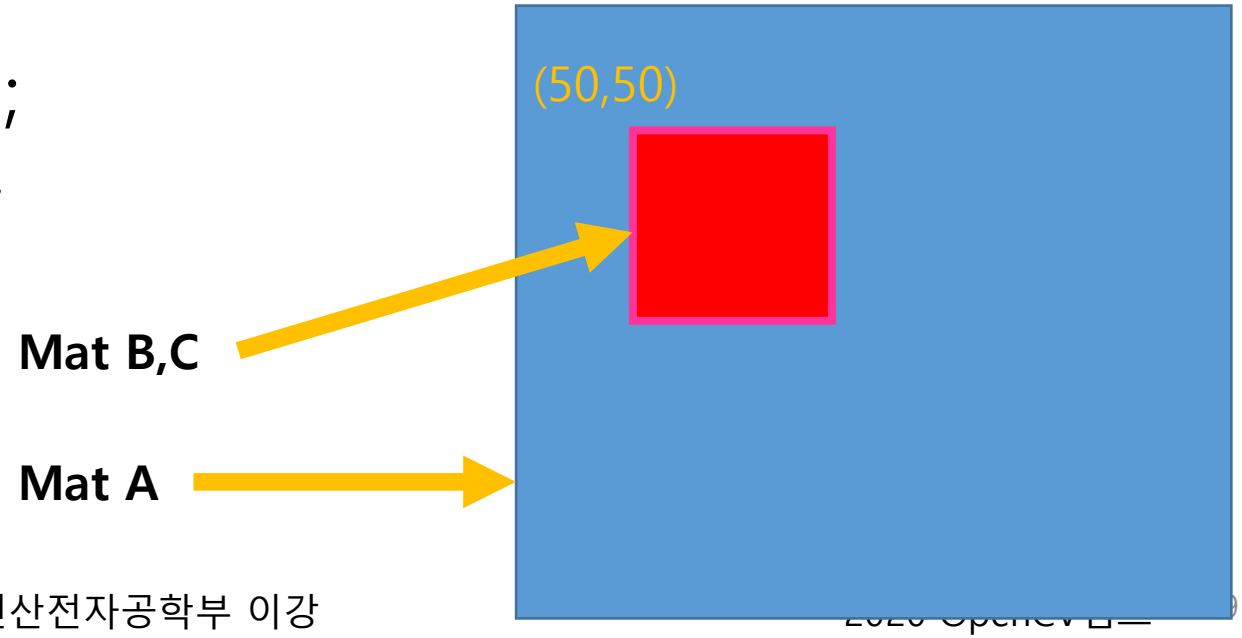
범위 지정 클래스

- Range : 하나의 colum or row
- Size : (가로 , 세로) 길이 순서쌍
- Rect : top-left corner 좌표 (x,y)와 (height, width) 순서쌍

ROI 지정 방법(1): Rect Class

- Rect Class를 생성해 ROI 영역 지정
- Rect(int x, int y, int width, int height)로 Rect 영역 지정
  -x,y는 rect의 offset이고 width, height는 rect의 크기

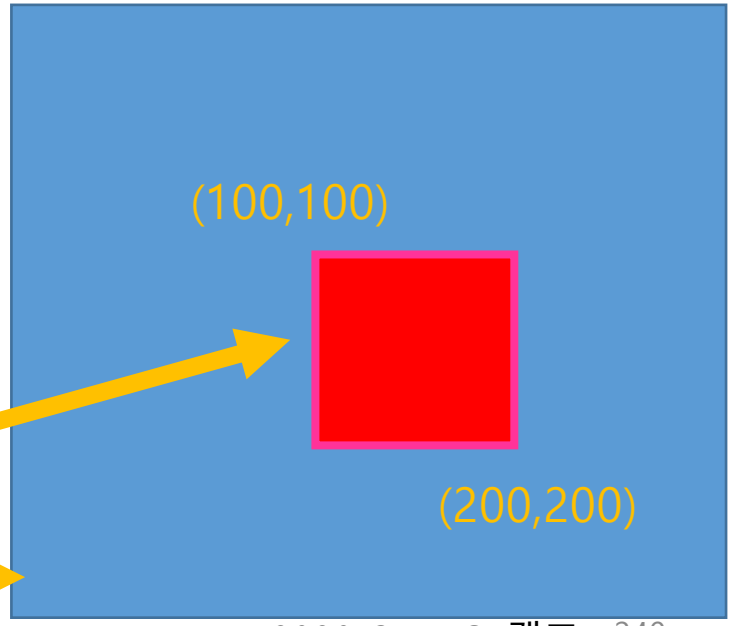
Ex) Mat A(300, 300, CV_8UC1);
Rect roi(50, 50, 100, 100);
Mat B = A(roi);
Mat C(A, roi);



ROI 지정 방법(2): Range Class

- Range Class를 생성해 ROI 영역 지정
- Range(int from, int end) 로 생성
- From <= roi < end의 범위를 지정

Ex) Mat A(300, 300, CV_8UC1);
Range x(100, 200);// 열의 범위 지정
Range y(100, 200);// 행의 범위 지정
Mat B = A(x,y);



ROI 설정이후 영역 수정 : adjustROI()

- Roi 영역을 수정하는 함수
- Mat& Mat::adjustROI(int dtop, int dbottom, int dleft, int drightt)의 형태
- dtop은 위로, dbttom은 아래로, dleft는 좌로, dright는 우로 해당 값만큼 roi 영역을 확장

Ex) B.adjustROI(1,1,1,1);// ROI B 를 상,하,좌,우로 1만큼씩 증가

Matrix Arithmetic Operations (1)

void **add** (Mat src1, Mat src2, Mat dst, Mat mask=noArray(),
int dtype = -1)

- mask : optional operation mask(8-bit single channel array)
- dtype : optional depth of output array
- dst(I) = saturate(src1(I)+src2(I) if mask(I) != 0
- Same as dst = src1 + src2

void **scaleAdd**(Mat src1, double alpha, Mat src2, Mat dst)

- dst(I) = scale * src1(I) + src2(I)

void **addWeighted**(InputArray **src1**, double **alpha**, InputArray **src2**, double **beta**, double **gamma**, OutputArray **dst**, int**dtype**=-1)

$$\texttt{dst}(I) = \texttt{saturate}(\texttt{src1}(I) * \texttt{alpha} + \texttt{src2}(I) * \texttt{beta} + \texttt{gamma})$$

Matrix Arithmetic Operations (2)

void **absdiff**(Mat src1, Mat src2, Mat dst)

- Dst(I) = saturate(| src1(I)-src2(I) |)

void **compare**(Mat src1, Mat src2, Mat dst, int cmpop)

- dst ← src1 cmpop src2 (src1 or src2 may be a scalar)
- cmpop : CMP_EQ(==), CMP_GE(>=), CMP_GT(>), CMP_LT(<), CMP_LE(<=), CMP_NE(not equal)

void **subtract**(Mat src1, Mat src2, Mat dst, Mat mask=noArray(), int dtype = -1)

- mask : optional operation mask (8-bit single channel array)
- dst(I) = saturate(src1(I) – src2(I)) if mask(I) != 0

Mat function

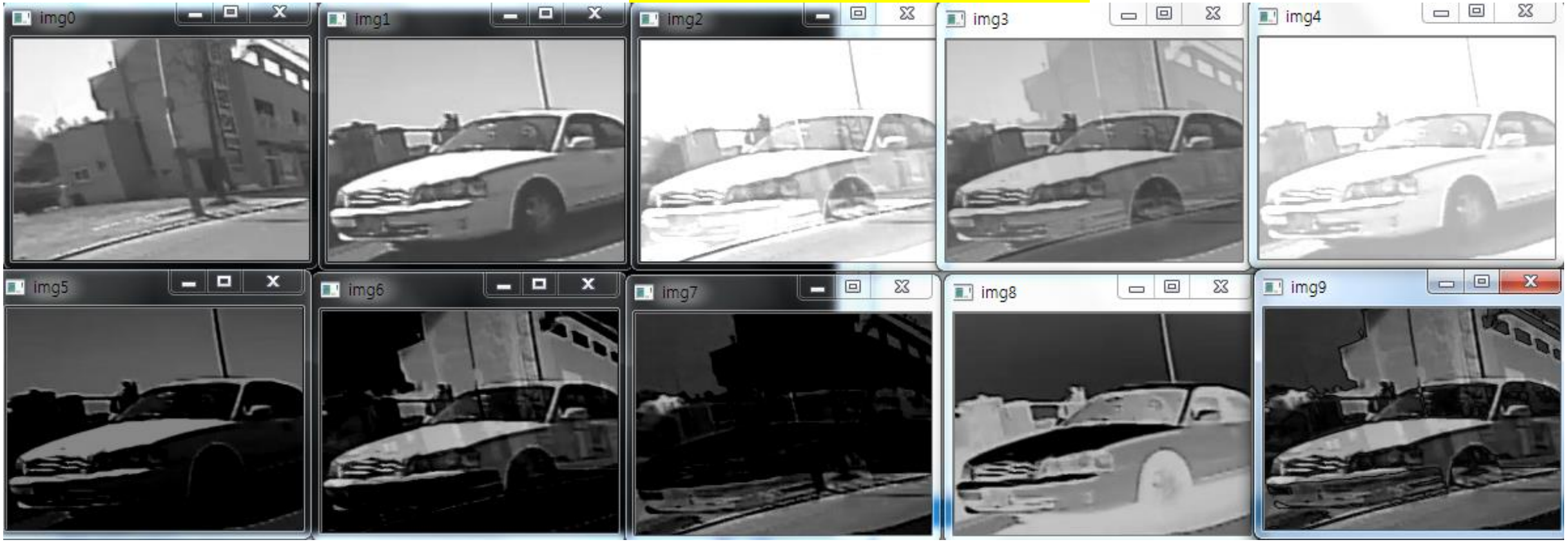
- reduce (InputArray src, OutputArray dst, int dim, int rtype, int dtype)
  - 입력 이미지(src)를 row방향 또는 column 방향으로 projection해서 1차원 배열(dst)을 얻는다.
  - dim : 입력 이미지를 reduce하는 축, 1 (single coilumn으로) 또는 0 (single row로) reduce
  - rtype: reduce하는 방법
    - CV_REDUCE_SUM : 입력 이미지의 모든 컬럼/row를 더한다.
    - CV_REDUCE_AVG , CV_REDUCE_MAX, CV_REDUCE_MIN
  - dtype: -1이면 입력과 동일한 타입

```
Mat sum_col(1, rect.width, CV_32FC1);
reduce(binary_img, sum_col,0, CV_REDUCE_SUM, CV_32FC1); // column sum  reduce to a single row
sum_col = sum_col * 1.0 / 255.0;   // 이진영상의 각 컬럼별로 0이 아닌 픽셀의 갯수를 셈
```

Mat bitwise operation

- **bitwise_or** (InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray())
  - src1: input array or scalar
  - src2: input array or scalar
  - dst : output array that has same size and type as input arrays
  - mask : optional operation mask, 8-bit single channel array
  - dst(I) = src1(I) V src2(I) if mask(I) ==1
- **bitwise_and** (InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray())
- **bitwise_xor** (InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray())
- **bitwise_not**(InputArray src, OutputArray dst, InputArray mask=noArray())

```
#include <vector>
void main() {
  vector<Mat> img(10);
  img[0] = imread("building.jpg",0);
  img[1] = imread("car.jpg", 0);
  img[2] = img[1] + img[0];
  addWeighted(img[0], 0.3, img[1], 0.7, 0, img[3]);
  img[4] = img[1] + 120;
  img[5] = img[1] - 120;
img[6] = img[1] - img[0];
img[7] = img[0] - img[1];
subtract(255, img[1],img[8]);
absdiff(img[0], img[1], img[9]);
  for (int i = 0; i < img.size(); i++)
  {  char name[10];
     sprintf(name, "img%d", i);
     imshow(string(name), img[i]);
  }
  waitKey(0);
}
```



Mat member function

void **Mat::convertTo**(OutputArray **m**, int **rtype**, double **alpha**=1, double **beta**=0)

- m : output matrix; if the size or type is not proper, it is reallocated
- rtype: desired output matrix
- m(x,y) = saturate_cast<rType> (alpha * (*this)(x,y) + beta)

void Mat& **Mat::setTo**(InputArray **value**, InputArray **mask**=noArray()

- Sets all or some of the array elements to the specified value
- Mask : Operation mask of the same size as *this
- Same as Operator = (const Scalar &s)

Mat **Mat::reshape**(int **cn**, int **rows**=0) const

- Change the shape and/or the number of channels of 2D matrix without copying the data (rows*cols*channels() must be the same after transform)
- cn: new number of channels. If 0, the number of channel remains the same.
- rows : number of rows. If 0 the number of rows remains the same.

Mat 단순 통계 함수 (1)

- scalar mean(InputArray src, InputArray mask=noArray())
  - 입력 이미지의 각 채널별 평균을 구한다. mask가 있을 경우 mask가 0이 아닌 부분에 한해서 통계를 낸다.
- void meanStdDev(InputArray src, OutputArray mean, OutputArray dtddev, InputArray mask=noArray())
  - 입력이미지(src)의 평균과 표준편차를 채널별로 구한다. 단, mask가 있을 경우, mask 값이 0이 아닌 부분에 대한 통계만 구한다.

```
cvtColor(color_img, hsv_img, COLOR_BGR2HSV);
Scalar mean_hsv, stddev_hsv;
cv::meanStdDev(hsv_img, mean_hsv, stddev_hsv, bin_final_mask);
```

- double Mahalanobis (InputArray v1, InputArray v2, InputArray icovar)
  - 두 입력 행렬의 마할라노비스 거리를 구한다.
  - icovar: inverse covariance matrix (calcCovarMatrix 함수+invert() 함수로 구함)

Mat 단순 통계함수(2)

- Scalar sum(Mat src)
  - Calculate the sum of array elements, independently of channels
- int countNonZero(Mat src)
  - Calculate the number of non-zero elememts

영상 분할을 위한 기본 API

Thresholding (1)

- double threshold (Mat src, Mat dst, double thresh, double maxval, int type)
  - Apply fixed level thresh to each array element
  - Typically used to get binary image from grayscale input image (compare() can be used for this, too.)
  - maxval : dst(I) = maxval if src(I) > thresh, 0 otherwise, when type is THRESH_BINARY
  - Type :

    THRESH_BINARY,

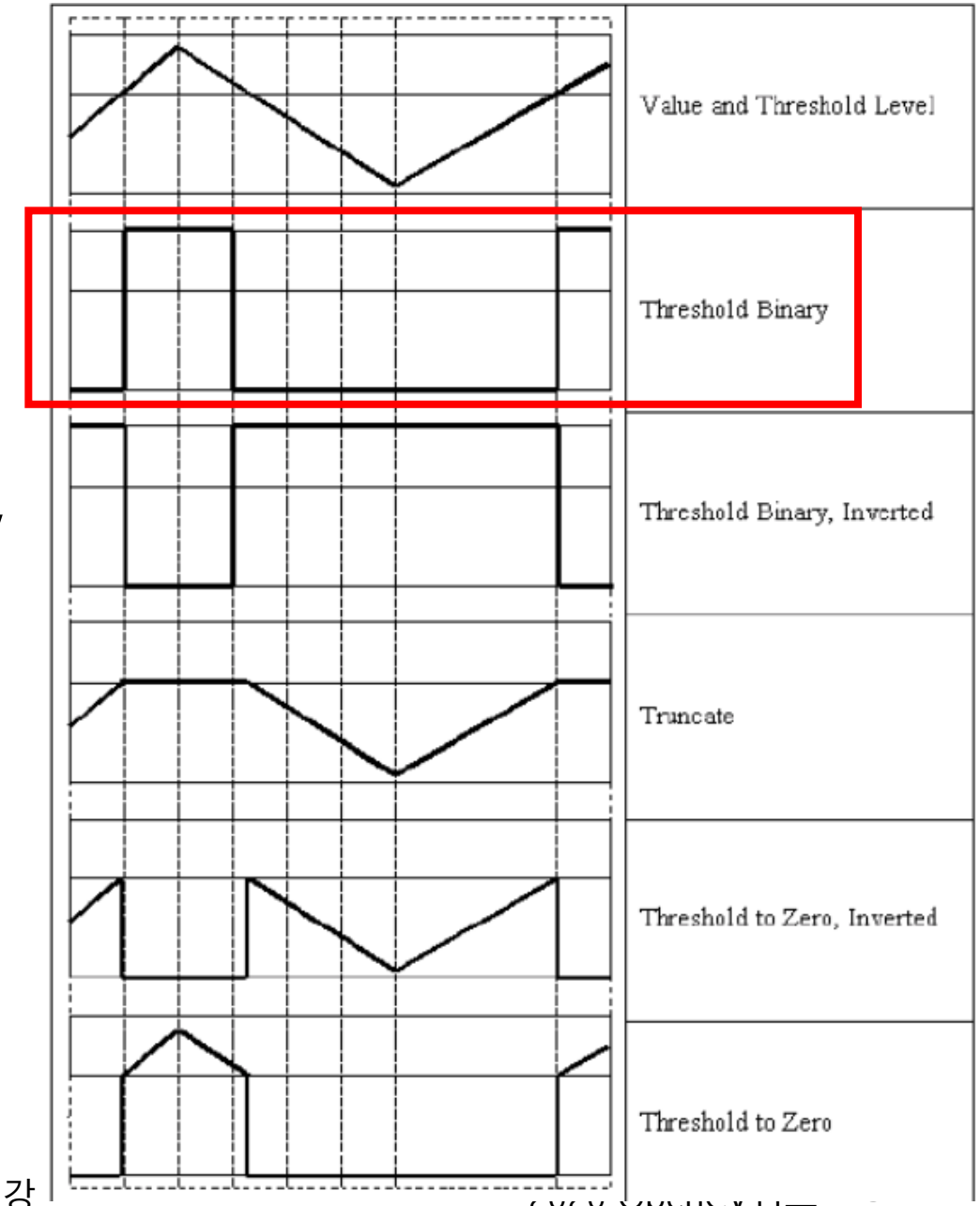
    THRESH_BINARY_INV,

    THRESH_TRUNC,

    THRESH_TOZERO,

    THRESH_TOZERO_INV,

    THRESH_OTSU
  - THRESH_OTSU 와 위의 방법들을 결합(+)할 수 있음.



Thresholding (2)

- Void adaptiveThreshold(Mat src, Mat dst, double maxval, int adaptiveMethod, int thresholdType, int blockSize, double C)
  - adaptiveMethod:
    - ADAPTIVE_THRESH_MEAN_C
    - ADAPTIVE_THRESH_GAUSSIAN_C
  - thresholdType:
    - THRESH_BINARY
    - THRESH_BINARY_INV
  - blokSIze : size of neighborhood used to calculate threshold (3,5,7)
  - C : constant subtracted from mean or weighted mean
  - T(x,y) is computed as MEAN(blockSize x blockSize)-C or GAUSSIAN(blockSize x blockSize) –C around (x,y)

Filters

- blur (InputArray src, OutputArray dst, Size ksize, Point anchor=Point(-1,-1), int borderType=BORDER_DEFAULT)
  - ksize : 커널 크기

- Medianblur (InputArray src, OutputArray dst, Size ksize)
  - ksize: 커널 크기

- GaussianBlur (InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT)
  - ksize : 커널 크기
  - sigmaX ≠ sigmaY 이면, sigmaY = sigmaX이다.
  - sigmaX = 0, sigmaY = 0 이면, ksize의 width와 height를 이용하여 계산

미분연산

- void Sobel (InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, doubl;e scale=1, double delta=0, int borderType=BORDER_DEFAULT)
  - ddepth = -1 이면, dst.depth() = src.depth()
  - dx: x방향 미분 차수
  - dy: y방향 미분 차수
  - ksize : kernel size
    - 1이 아니면 ksize x ksize 크기의 커널 적용
    - -1이면, 3x3 kerne이 되고 Scharr 3x3 커널이 적용됨

-1	0	1
-2	0	2
-1	0	1

dx=1, dy=0, ksize=3

-1	-2	-1
0	0	0
1	2	1

dx=0, dy=1, ksize=3

1	-10	-1
0	0	0
-1	10	1

dx=1, dy=1, ksize=3

Edge Extraction

- void **Canny** (InputArray image, OutputArray edges, double thresh1, double thresh2, int apertureSize=3, bool L2gradient=false)

- 절차
  1. 가우시안 필터링 수행
  2. Sobel 연산자로 gradient vector 크기 계산
  3. 가는 에지를 얻기 위해 3 x 3 창을 이용하여 벡터방향의 최대 화소만 남김
  4. 두개의 문턱값을 이용하여 에지들을 벡터방향으로 연결한다.

Hough 변환(직선 검출)

- void HoughLines (InputArray img, OutputArray lines, double rho, double theta, int threshold, double srn=0, double stn=0)

- void HoughLinesP(InputArray img, OutputArray lines, double rho, double theta, int threshold, double minLineLength=0, double maxLineGap=0)
  - 직선의 양끝점 좌표 (x1,y1, x2,y2)를 lines (CV_32SC4)에 저장
  - rho : 원점에서의 거리 해상도
  - theta: x축과의 각도 라디안 각도 해상도
  - minLength: 검출할 직선의 최소 길이
  - maxLineGap : 직선 위에서 에지점들 간의 최대 허용 간격

Hough 변환 (원검출)

- HoughCircles (inputArray image, OutputArray circles, int method, double dp, double minDist,

double param1 = 100, param2 = 100, int minRadius =0, int maxRadius = 0)
  - image: 8-bit single-channel grayscale input image
  - circles: output vector of found circles. each vector has 3 or 4 elements (x,y,radius, votes)
  - method: **HOUGH_GRADIENT**
  - dp: inverse ratio of the accumulator resolution to the image resolution. if dp = 2, the accumulator has half as big width and height
  - minDist: Minimum distance between the centers of the detected circles (if too small multiple neighbor circles may be falsely detected. if too large, some circles may be missed)
  - param1: first method-specific parameter (in case HOUGH_GRADIENT, the higher threshold for Canny edge detector)
  - param2: second method-specific parameter (in case HOUGH_GRADIENT, accumulator threshold for the circle centers at the detection stage. The smaller, more false circles may be detected
  - minRadius: minimum circle radius
  - maxRadius: maximum circle radius if <=0 use the maximum image dimention. if <0, return center without finding the radius)

Morphological (1)

- void erode **(InputArray src, OutputArray dst, InputArray kernel**, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDR_CONSTANT, const Scalar & borderValue= morphologyDefaultBorderValue())
  - 침식연산
- void dilate **(InputArray src, OutputArray dst, InputArray kernel**, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDR_CONSTANT, const Scalar & borderValue= morphologyDefaultBorderValue())
  - 팽창연산

Morphological (2)

- void morphologyEx **(InputArray src, OutputArray dst, int op**, **InputArray kernel**, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDR_CONSTANT, const Scalar & borderValue= morphologyDefaultBorderValue())
  - op=MORPH_OPEN : dst = dilate (erode(src, kernel), kernel)
  - op=MORPH_CLOSE : dst = erode (dilate(src, kernel), kernel)
  - op=MORPH_GRADIENT : dst = dilate(src, kernel) – erode (src, kernel)
  - op=MORPH_TOPHAT : dst = src – open (src, kernel)
  - op=MORPH_BLACKHAT: dst = close(src,kerbel) - src

Contour: findContours

void **findContours** (InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())

- image : Source 8-bit single channel image. 0이 아닌 픽셀은 1로 간주되어 이진 영상으로 간주됨, inRange, Canny, threshold, adaptiveThreshold, compare 등을 호출하여 이진 영상 만듬.
- countours : 출력 vector, vectors of points 형태로 저장됨
- hierarchy : optional output vector, contour 갯수와 동일한 원소 가짐. 각 i번째 원소마다, hierarchy[i][0], hierarchy[i][1], hierarchy[i][2], hierarchy[i][3] 의 원소를 가짐 (동일 레빌의 이전, 이후, child, parent countor 의 index 값. 해당값이 없으면 음수값을 가짐)
- mode :
  - CV_RETR_EXTERNAL: extreme countour 만 추출
  - CV_RETR_LIST: 계층 구조없이 모든 countour 추출
  - CV_RETR_CCOMP : two-level 계층구조로 countour 추출함
  - CV_RETR_TREE: 모든 계층구조를 추출해서 저장함.
- method:
  - CV_CHAIN_APPROX_NONE: 모든 점을 다 포함.
  - CV_CHAIN_APPROX_SIMPLE:: 수평, 수직, 대각선은 end-point만 포함
  - CV_CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS: 알고리즘 적용하여 approximation
- offset: countour 좌표에 일정 값을 더해서 출력. ROI 에서 추출한 countour를 원 이미지에 그릴때 유용함.

Countour: 모양 그려 보이기

- drawCountour : countour를 그림.

void drawContours(InputOutputArray **image**, InputArrayOfArrays **contours**, int **contourIdx**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, InputArray **hierarchy**=noArray(), int **maxLevel**=INT_MAX, Point **offset**=Point())

Countour: 면적, 둘레 길이

- acrLength: contour로 이루어진 둘레 길이 게산
- contourArea: contour로 둘러싸인 면적 계산

Contour: contour의 단순화(1)

- void **approxPolyDP** (InputArray **curve**, OutputArray **approxCurve**, double **epsilon**, bool **closed**):

- Rect **boundingRect** (InputArray *points*)

- RotatedRect **minAreaRect** (InputArray **points**) :

Contour: contour의 단순화(2)

- void **minEnclosingCircle** (InputArray **points**, Point2f& **center**, float& **radius**)

- RotatedRect **fitEllipse** (InputArray **points**)

- void **fitLine** (InputArray **points**, OutputArray **line**, int **distType**, double **param**, double **reps**, double **aeps**)

ConvexHull (볼록 다각형)

- convexHull : convexHull을 계산하여 좌표 반환

void convexHull(InputArray **points**, OutputArray **hull**, bool **clockwise**=false, bool **returnPoints**=true)

  - points: input point들의 집합 (std:vector 또는 Mat 타입)
  - hiull: output convex hull , set of points or integer vector of indices of original points
  - clockwise :
  - returnPoints : true이면 point vector를 return하고 false이면 index vector를 return함 (hull이 vector이면 무시됨)

Thank you!