Mat operator

Sung Soo Hwang

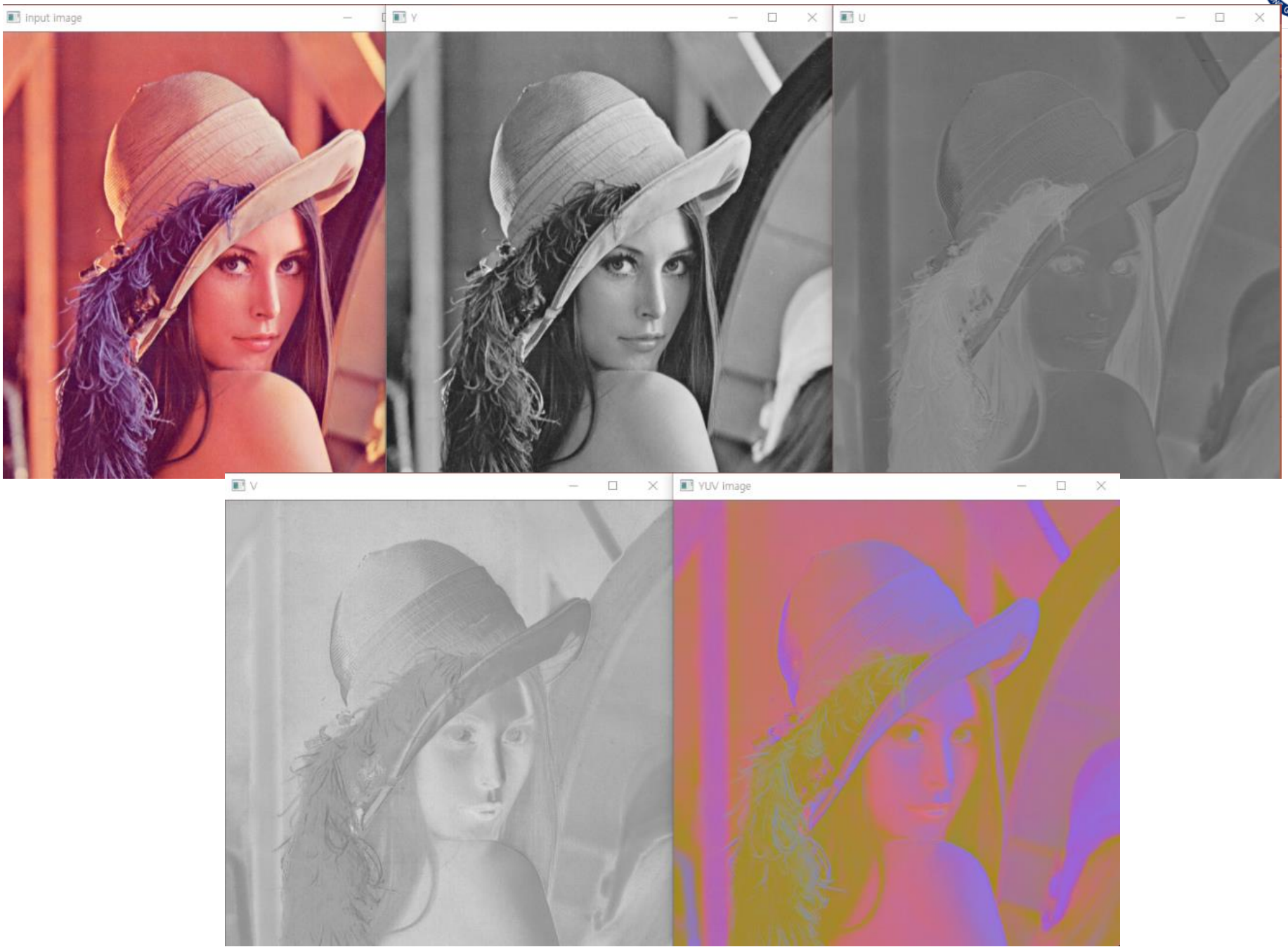
Mat Operator

- Color space conversion
  - void cvtColor(Mat src, Mat dst, int code, int dstCn =0)
    - Convert an image frame one color space to another
    - Code: CV_BGR2GRAY, CV_BGR2HSV, CV_BGR2YCrCb, CV_BGR2Lab, ……)
    - dstcn: destination channel number. If 0, automatically determined by src and dst
  - 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]* the number of arrays must match src.channels()
  - merge(InputArrayOfArray mv, OutputArray dst): reverse of split
    - mv: vector of matrices all of the matrices in mv must have same size and depth
    - dst : output array of the same size and depth as mv[0]

Mat Operator

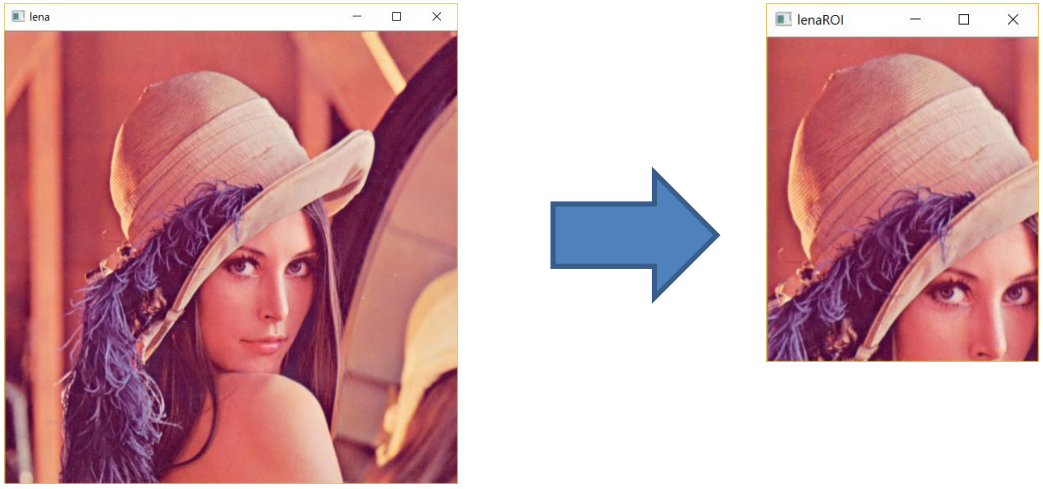
- Color space conversion
  - Example code

```
int main() {
	Mat image, image_YUV, dst;
	vector<Mat> yuv_channels(3);

	image = imread("lena.png");

	cvtColor(image, image_YUV, CV_BGR2YUV);

	split(image_YUV, yuv_channels);

	merge(yuv_channels, dst);

	imshow("input image", image);
	imshow("Y", yuv_channels[0]);
	imshow("U", yuv_channels[1]);
	imshow("V", yuv_channels[2]);
	imshow("YUV image", dst);

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


input image
Y
U
V
YUV image

Mat Operator

- ROI(Region of Interest)
  - A sub-region in an image that we are interested in



- Try to change value in ROI and see what happens in the original image

Mat Operator

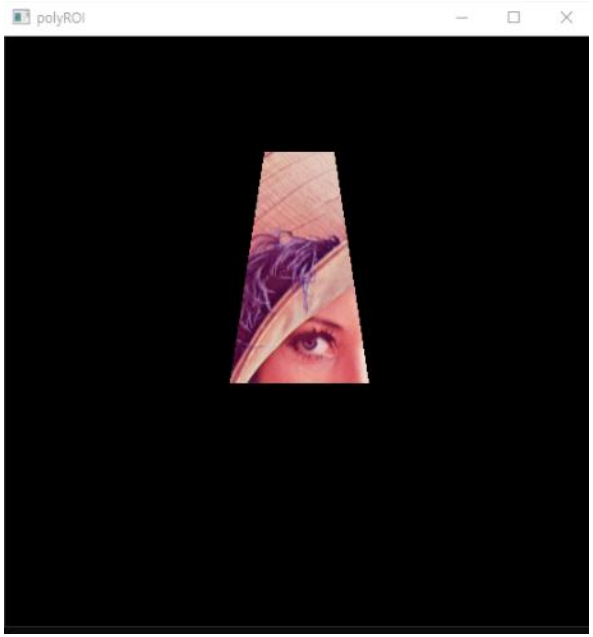
- ROI(Region of Interest)
  - Example code:

```
int main() {
    Mat image = imread("lena.png");
    Rect rect(100, 30, 250, 300);
    Mat rect_roi = image(rect);
    imshow("rectROI", rect_roi);

    waitKey(0);
}
```

Mat Operator

- ROI(Region of Interest)
  - Example code:

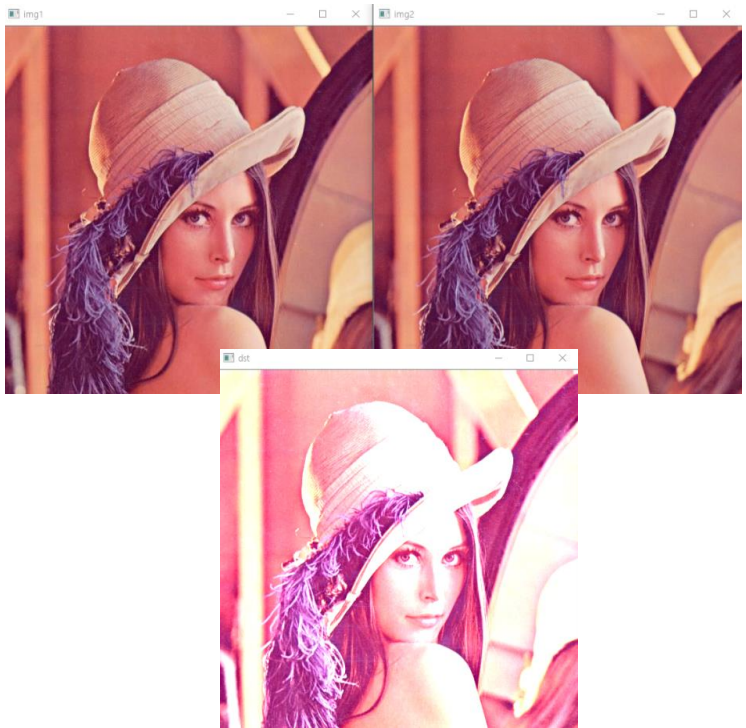
```
int main() {
    Mat image = imread("lena.png");
    Mat poly_roi;
    Mat poly_mask = Mat::zeros(image.size(), image.type());
    Point poly[1][4];
    poly[0][0] = Point(226, 100);
    poly[0][1] = Point(286, 100);
    poly[0][2] = Point(316, 300);
    poly[0][3] = Point(196, 300);
    const Point* ppt[1] = { poly[0] };
    int npt[] = { 4 };
    // function that draws polygon with given points
    fillPoly(poly_mask, ppt, npt, 1, Scalar(255, 255, 255), 8);
    image.copyTo(poly_roi, poly_mask);
    imshow("polyROI", poly_roi);

    waitKey(0);
}
```



Mat Operator

- Addition/Subtraction operation
  - void add (Mat src1, Mat src2, Mat dst, Mat mask= noArray(), int dtype = -1)
    - Save the result of src1 + src2 to dst
    - 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
  - Example code

```
int main() {
	Mat img1 = imread("lena.jpg");
	Mat img2 = imread("lena.png");
	Mat dst;
	add(img1, img2, dst);
	imshow("dst", dst);
	waitKey(0);
}
```



Mat Operator

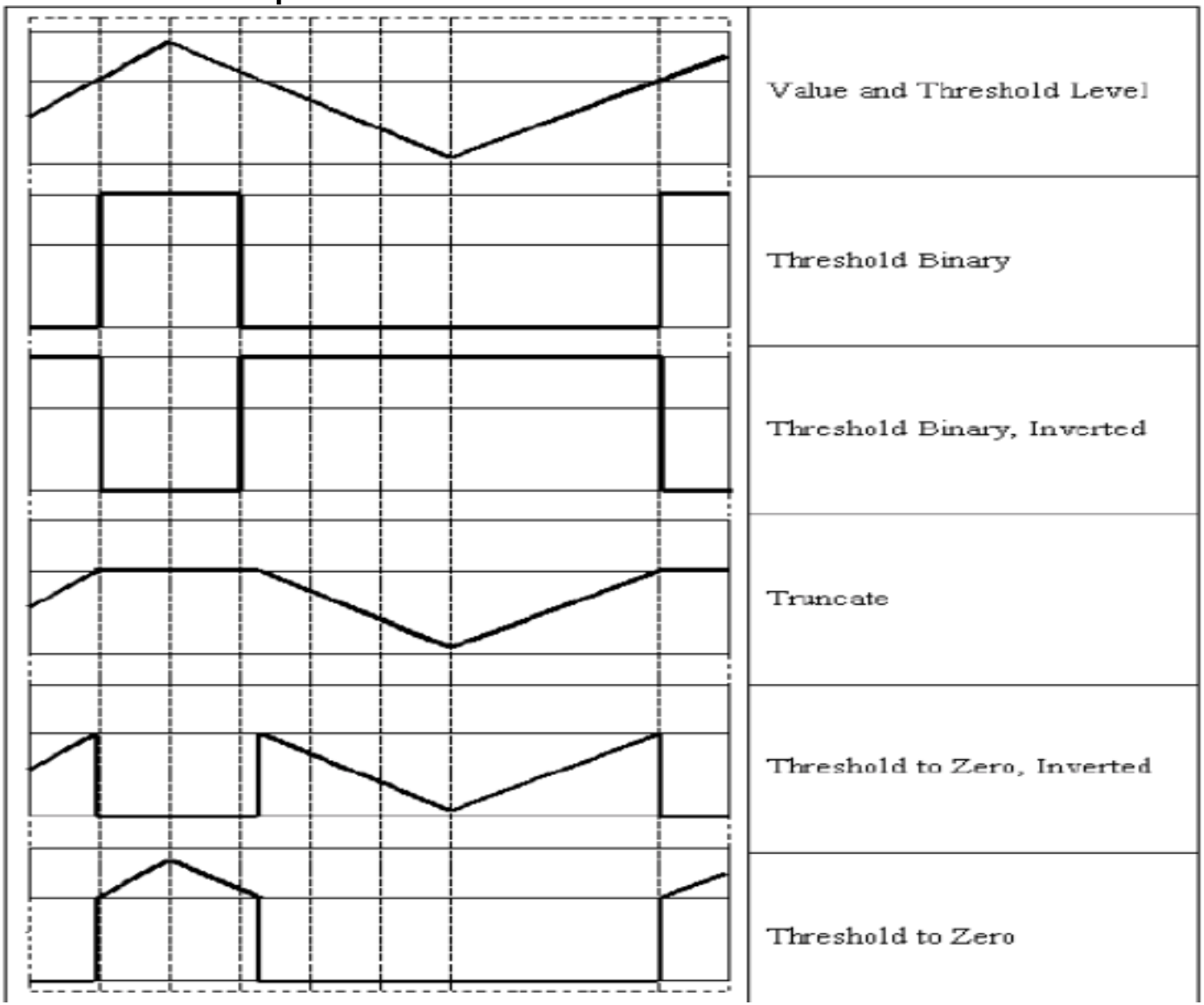
- Addition/Subtraction operation
  - void scaleAdd(Mat src1, double scale, Mat src2, Mat dst)
    - dst(I) = scale * src1(I) + src2(I)
  - void absdiff(Mat src1, Mat src2, Mat dst)
    - dst(I) = saturate(| src1(I)-src2(I) |)
  - void subtract(Mat src1, Mat src2, Mat dst, Mat mask=noArray(), int dtype = -1)
    - dst(I) = saturate(src1(I) – src2(I)) if mask(I) != 0

Mat Operator

- Threshold operation
  - 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
    - 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

Mat Operator

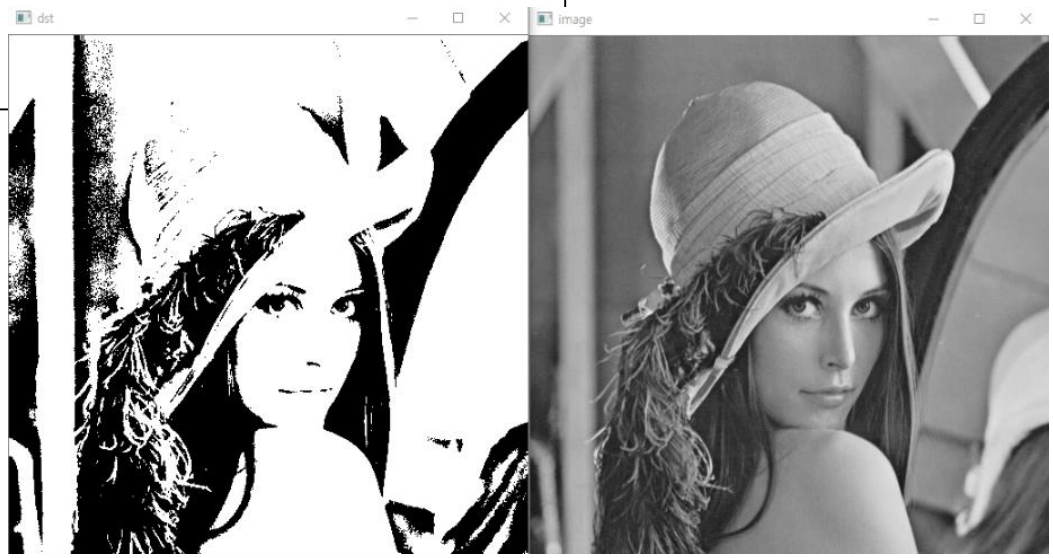
- Threshold operation



Mat Operator

- Threshold operation
  - Example code

```
int main() {
	Mat image = imread("lena.jpg");
	cvtColor(image, image, CV_BGR2GRAY);
	Mat dst;
	threshold(image, dst, 100, 255, THRESH_BINARY);

	imshow("dst", dst);
	imshow("image", image);
	waitKey(0);
	return 0;
}
```



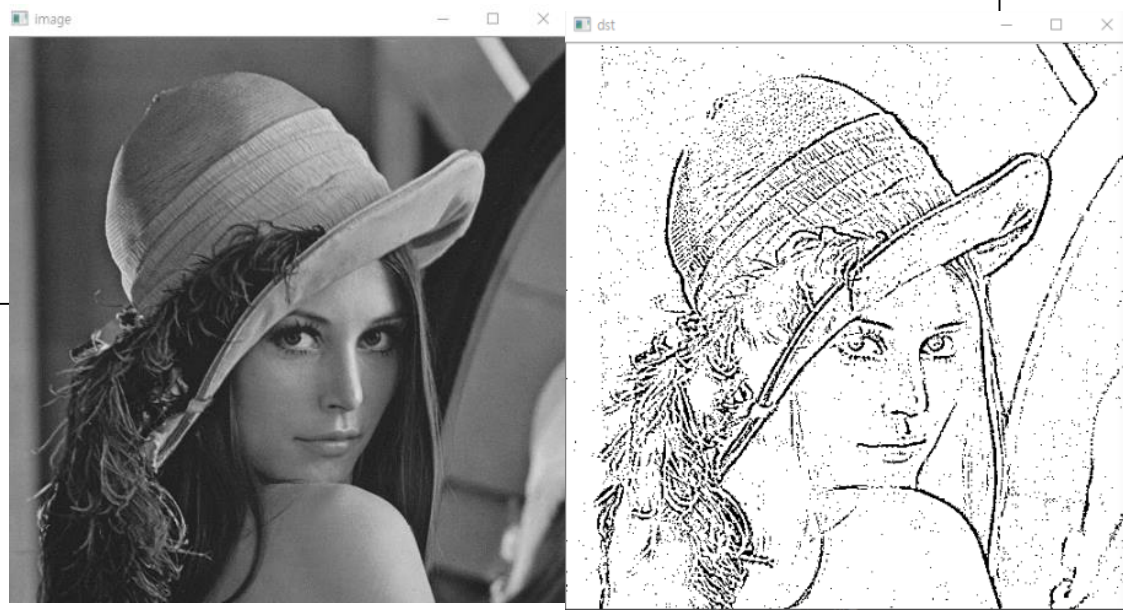
Mat Operator

- Threshold operation
  - 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
    - blockSIze : size of neighborhood used to calculate threshold (3,5,7)
    - C : constant subtracted from mean or weighted mean
    - dst(x, y) is computed as MEAN(blockSize x blockSize)-C or GAUSSIAN(blockSize x blockSize) –C around (x,y)

Mat Operator

- Threshold operation
  - Example code

```
int main() {
	Mat image = imread("lena.jpg");
	cvtColor(image, image, CV_BGR2GRAY);
	Mat dst;
	adaptiveThreshold(image, dst, 255, ADAPTIVE_THRESH_MEAN_C,
	THRESH_BINARY, 7, 10);
	imshow("dst", dst);
	imshow("image", image);
	waitKey(0);
	return 0;
}
```



Mat Operator

- Threshold operation
  - Void inRange(cv::InputArray src, cv::InputArray lowerb, cv::InputArray upperb, cv::OutputArray dst)
    - Src – first input array.
    - Lowerb – inclusive lower boundary array or a scalar
    - Upperb – inclusive upper boundary array or a scalar
    - Dst – output array of the same size as src and CV_8U type

Mat Operator

- Threshold operation
  - Example code:

```
int main() {
	Mat image = imread("hand.jpg");

	cvtColor(image, image, CV_BGR2YCrCb);
	inRange(image, Scalar(0, 133, 77), Scalar(255, 173, 127), image);

	imshow("inRange", image);
	waitKey(0);
	return 0;
}
```

Mat Operator

- Others
  - 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)
  - 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)
  - Void convertScaleAbs(InputArray src, OutputArray dst, double alpha=1, double beta=0)
    - Src – input array
    - Dst – output array
    - Alpha – optional scale factor
    - Beta – optional delta added to the scaled values

    Dst(I) = saturate_cast<uchar>(|src(I)| * alpha + beta|)

Mat Operator

- Others
  - Example code

```
int main() {
    Mat image = imread("lena.png");
    Mat after_convertTo, after_convertScaleAbs;
    imshow("original", image);
    image.convertTo(after_convertTo, CV_16SC1);
    imshow("after convertTo", after_convertTo);
    convertScaleAbs(image, after_convertScaleAbs, 2, 3);
    imshow("after convertScaleAbs", after_convertScaleAbs);
    image.setTo(Scalar(0));
    imshow("after setTo", image);
    waitKey(0);
}
```