Threshold_InRange_GrabCut

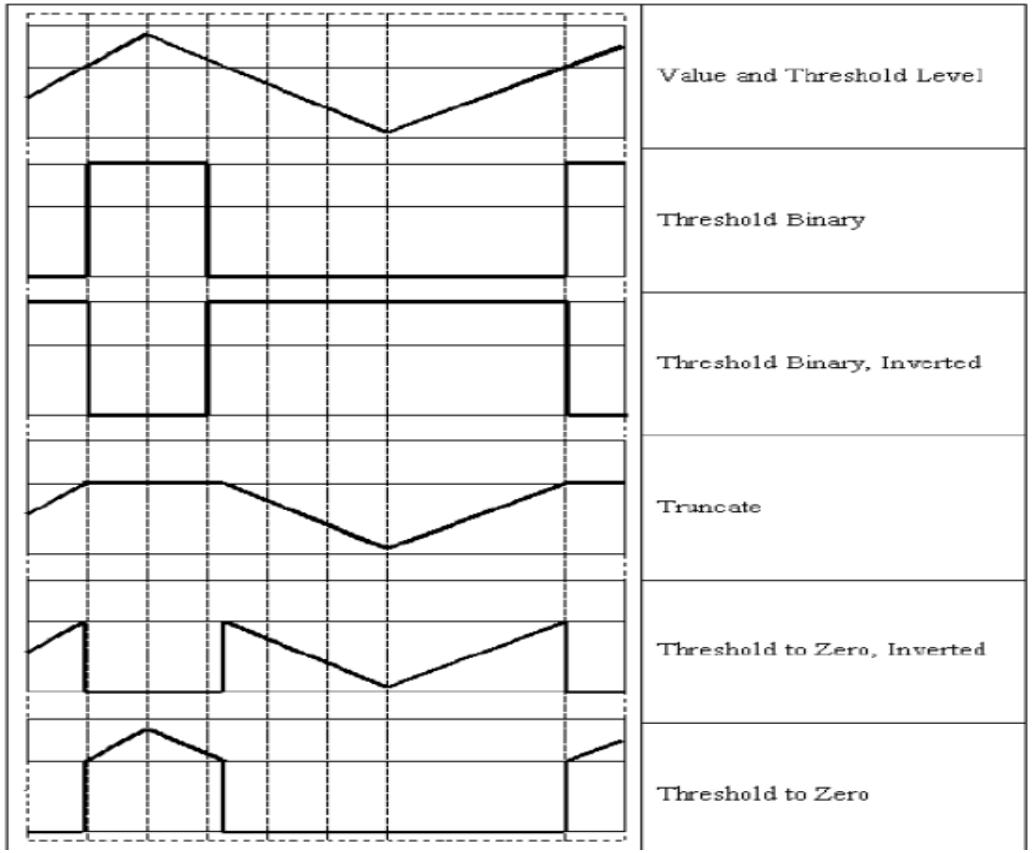
Sung Soo Hwang

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

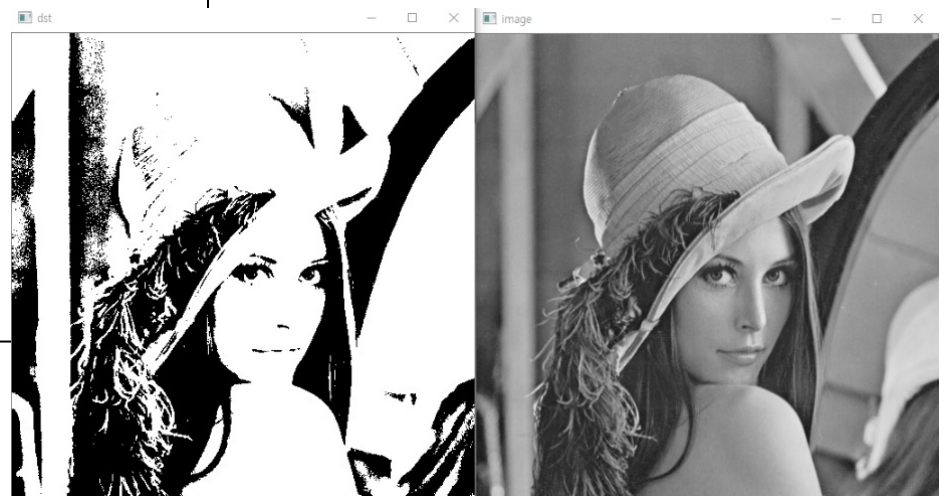
- Threshold operation



Mat Operator

- Threshold operation
  - Example code

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

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



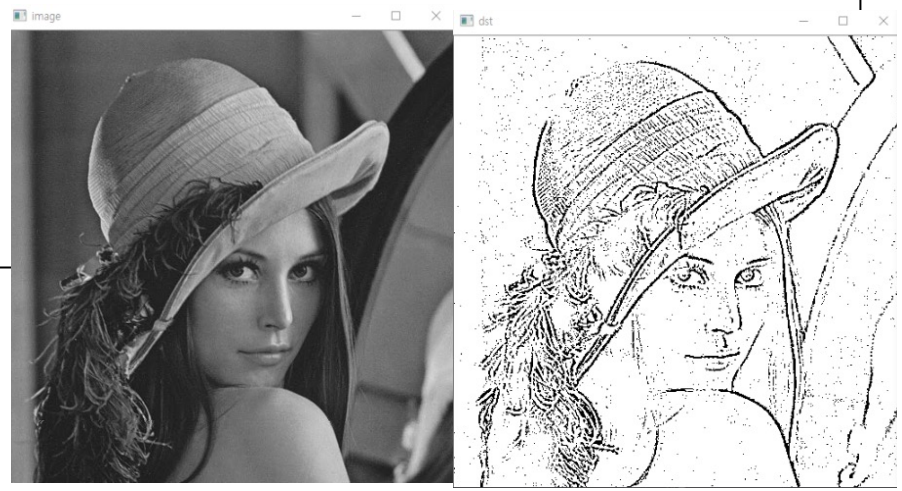
Mat Operator

- 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.png");
	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.png");

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

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

Global Thresholding

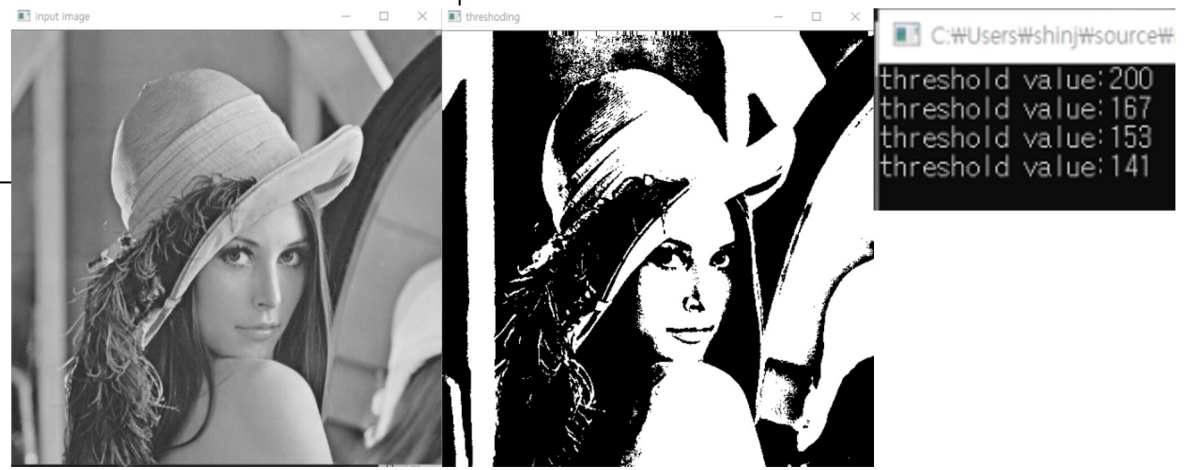
- Basic method
  - Example code

```
int main() {
	Mat image, thresh;
	int thresh_T, low_cnt, high_cnt, low_sum, high_sum, i, j, th;

	thresh_T = 200;
	th = 10;
	low_cnt = high_cnt = low_sum = high_sum = 0;

	image = imread("lena.png", 0);
	cout << "threshold value:" << thresh_T << endl;

	while (1) {
		for (j = 0; j < image.rows; j++) {
			for (i = 0; i < image.cols; i++) {
				if (image.at<uchar>(j, i) < thresh_T) {
					low_sum += image.at<uchar>(j, i);
					low_cnt++;
				}
				else {
					high_sum += image.at<uchar>(j, i);
					high_cnt++;
				}
			}
		}
```

Global Thresholding

- Basic method
  - Example code

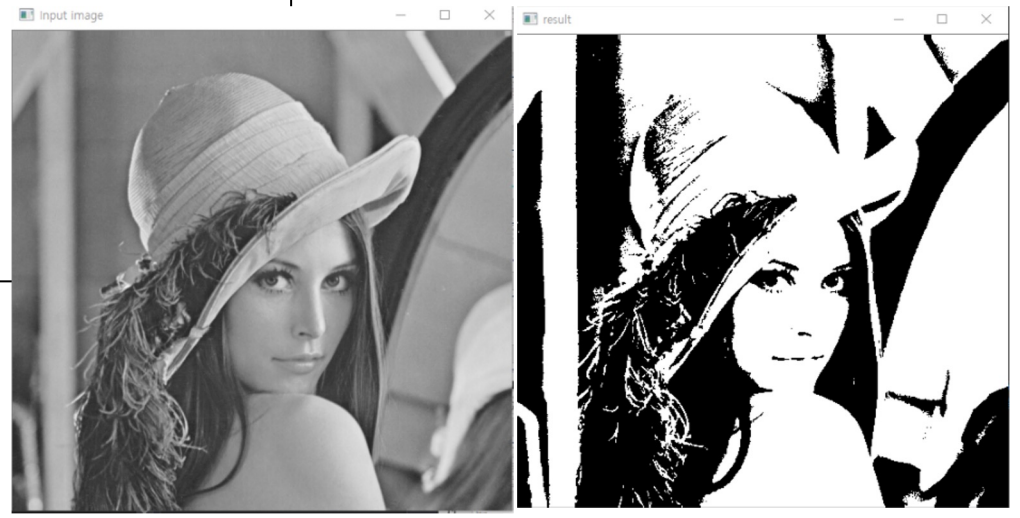
```
                if (abs(thresh_T - (low_sum / low_cnt + high_sum / high_cnt) / 2.0f) < th) {
                        break;
                }
                else {
                        thresh_T = (low_sum / low_cnt + high_sum / high_cnt) / 2.0f;
                        cout << "threshold value:" << thresh_T << endl;
                        low_cnt = high_cnt = low_sum = high_sum = 0;
                }
        }
        threshold(image, thresh, thresh_T, 255, THRESH_BINARY);

        imshow("Input image", image);
        imshow("thresholding", thresh);
        waitKey(0);
}
```



Global Thresholding

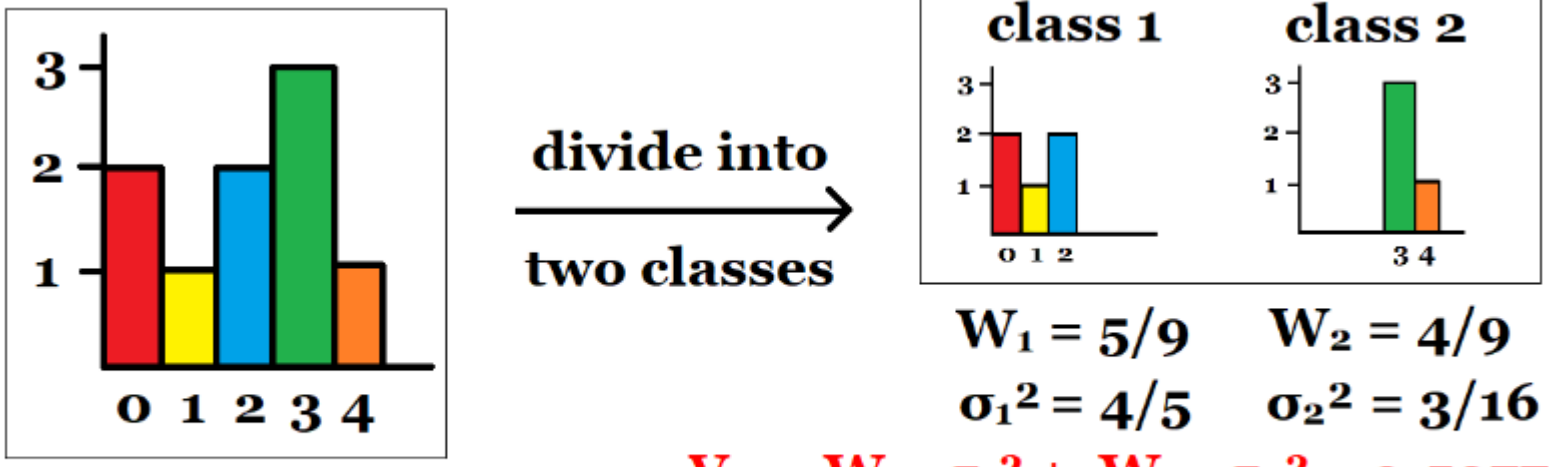
- Otsu's algorithm
  - Example code

```
int main() {
	Mat image, result;
	image = imread("lena.png", 0);
	threshold(image, result, 0, 255, THRESH_BINARY |
	THRESH_OTSU);
	imshow("Input image", image);
	imshow("result", result);

	waitKey(0);
}
```



Global Thresholding

- Otsu's algorithm
  - Within-class variance
    - If pixels are classified into N classes, then the within class variance is $(V_W) = \sum_{i=0}^{N}(W_i \times \sigma_i^2)$ where W_i is $^{\#\ of\ pixels\ in\ class\ i}/_{total\ pixel}$



$W_1 = 5/9$ $W_2 = 4/9$

$\sigma_1^2 = 4/5$ $\sigma_2^2 = 3/16$

$V_w = W_1 * \sigma_1^2 + W_2 * \sigma_2^2 = 0.52777$

Image by HBY coding academic | An example to compete Vw separated by a threshold T

Local(Adaptive) Thresholding

- Set a threshold for each point depending on the intensity distributions of adjacent pixels
  - Example code

```
int main() {
	Mat image, binary, adaptive_binary;
	image = imread("opencv.jpg", 0);

	threshold(image, binary, 150, 255, THRESH_BINARY);
	adaptiveThreshold(image, adaptive_binary, 255, ADAPTIVE_THRESH_MEAN_C,
	THRESH_BINARY, 85, 15);

	imshow("Input image", image);
	imshow("binary", binary);
	imshow("adaptive binary", adaptive_binary);
	waitKey(0);
}
```

GrabCut

- GrabCut operation
  - void grabCut(cv::InputArray img, cv::InputOutputArray mask, cv::Rect rect, cv::InputOutputArray bdgModel, cv::InputOutputArray fgdModel, int iterCount, int mode)
    - img - Input image
    - mask - mask image specifying background, foreground
    - rect - Coordinates of squares with foreground
    - bdgModel, fgdModel - Array used internally by algorithm
    - iterCount - Number of iterations the algorithm must run
    - mode - There are two types, GC_INIT_WITH_RECT and GC_INIT_WITH_MASK, each using a rectangle or mask to perform this algorithm

GrabCut

- GrabCut operation
  - Example code:

```
int main() {
        Mat result, bgdModel, fgdModel, image, foreground;
        image = imread("dog.png");
        //inner rectangle which includes foreground
        Rect rectangle(15, 0, 155, 240);
        grabCut(image, result, rectangle, bgdModel, fgdModel, 10, GC_INIT_WITH_RECT);
        compare(result, GC_PR_FGD, result, CMP_EQ);
        foreground = Mat(image.size(), CV_8UC3, Scalar(255, 255, 255));
        image.copyTo(foreground, result);
        imshow("original", image);
        imshow("Result", result);
        imshow("Foreground", foreground);
        waitKey(0);
}
```

GrabCut

- GrabCut operation
  - Example result: