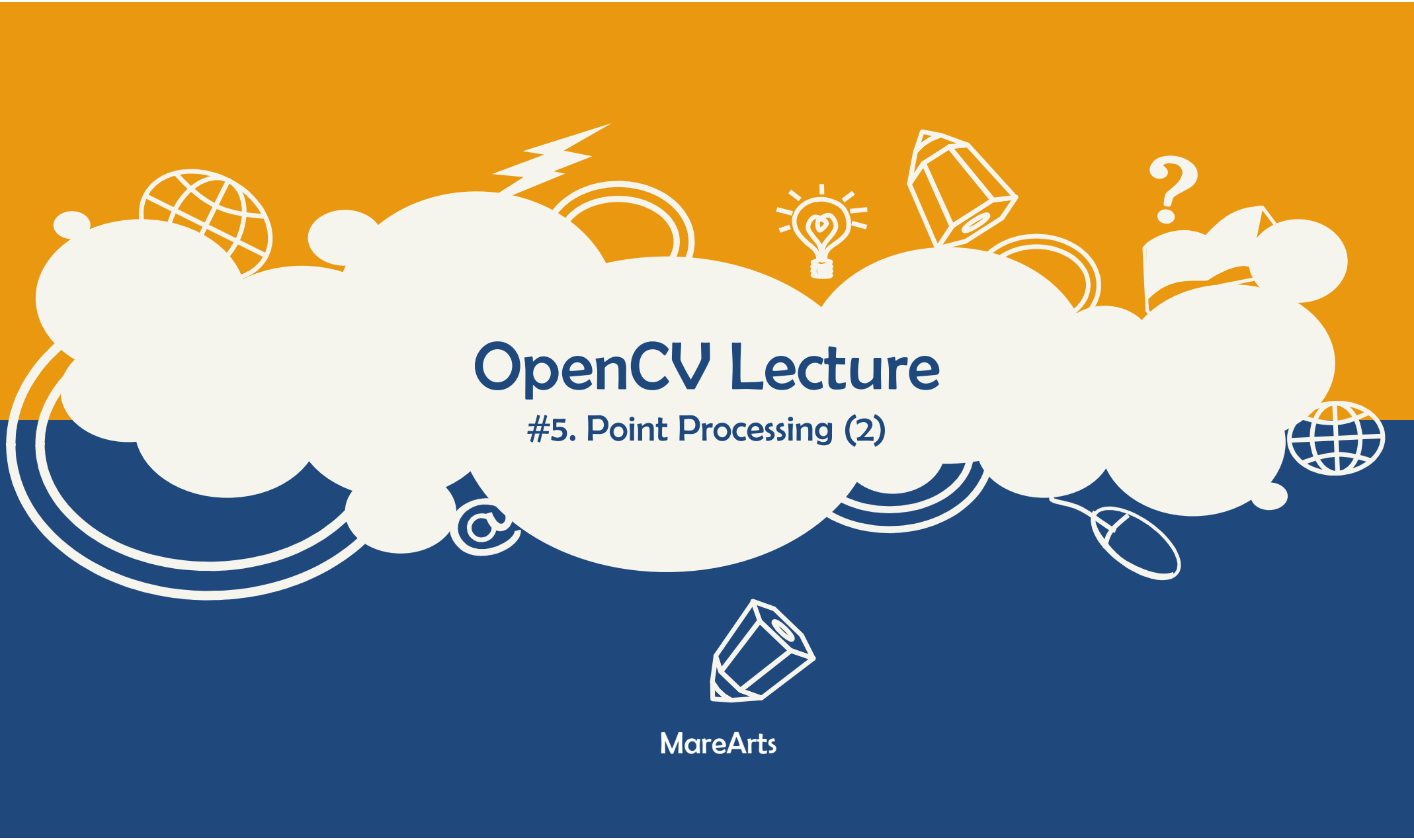


OpenCV Lecture

#5. Point Processing (2)

MareArts

Contents

- Binary processing
  - what is the binary?
  - about threshold function
  - about adpativeThreshold function
- other application related with point processing
  - using floodfill function
  - what is the integral image and using

What is the binary image?

- What is the binary image?
  - not gray image.
  - Binary image refer to 2 colors, black and white.
- How to make binary image?
  - This is very simple, but also very difficult?. ^^

How to make binary image?

- Based on the threshold value,
  - If pixel value lager than threshold, then 255
  - less than threshold, then 0

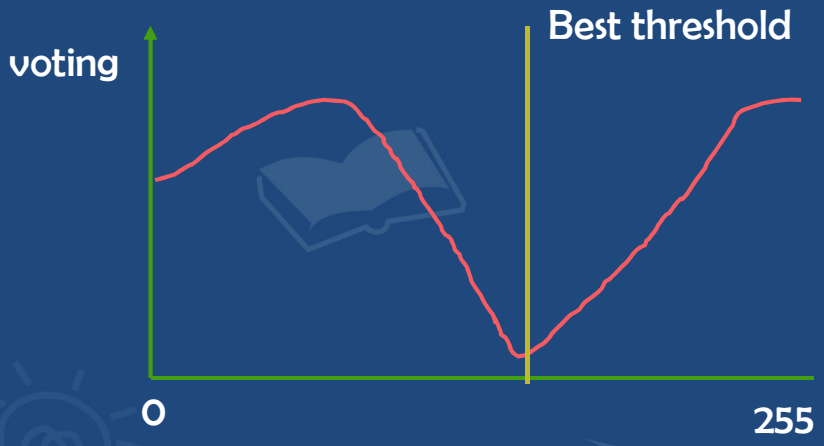
12	22	200	67
156	143	12	199
233	77	234	74
255	230	33	87

threshold = 128

0	0	255	0
255	255	0	255
255	0	255	0
255	255	0	0

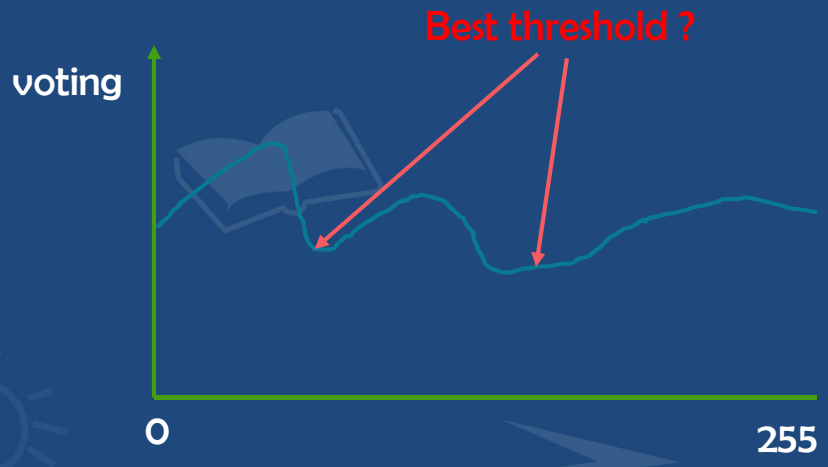
How to make binary image?

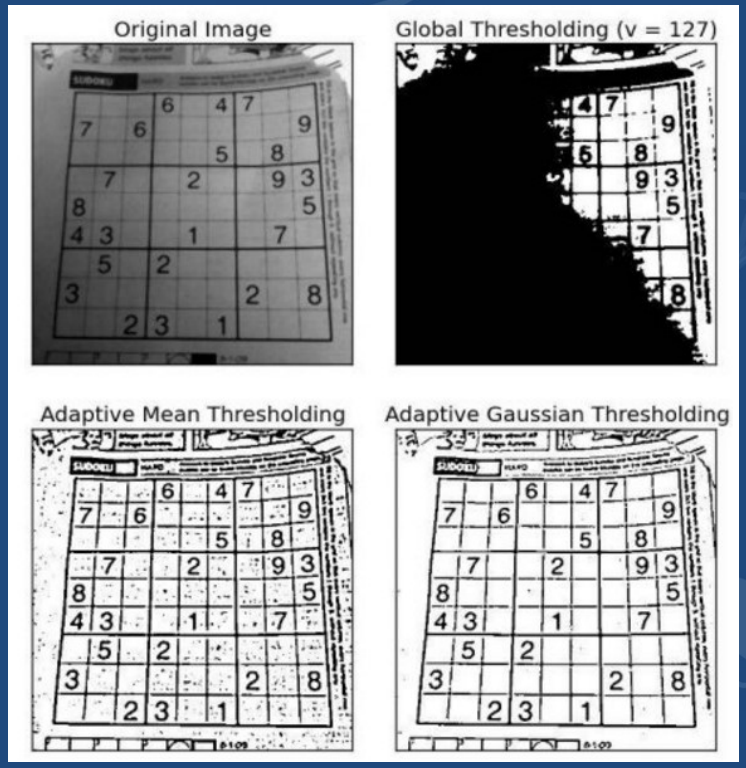
- How to determine threshold?
  - using Histogram



How to make binary image?

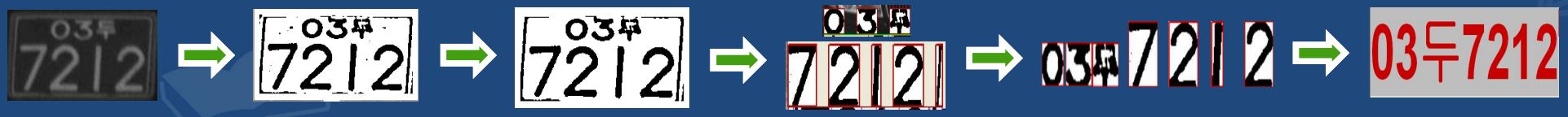
- How to determine threshold?
  - can not determine only one value.





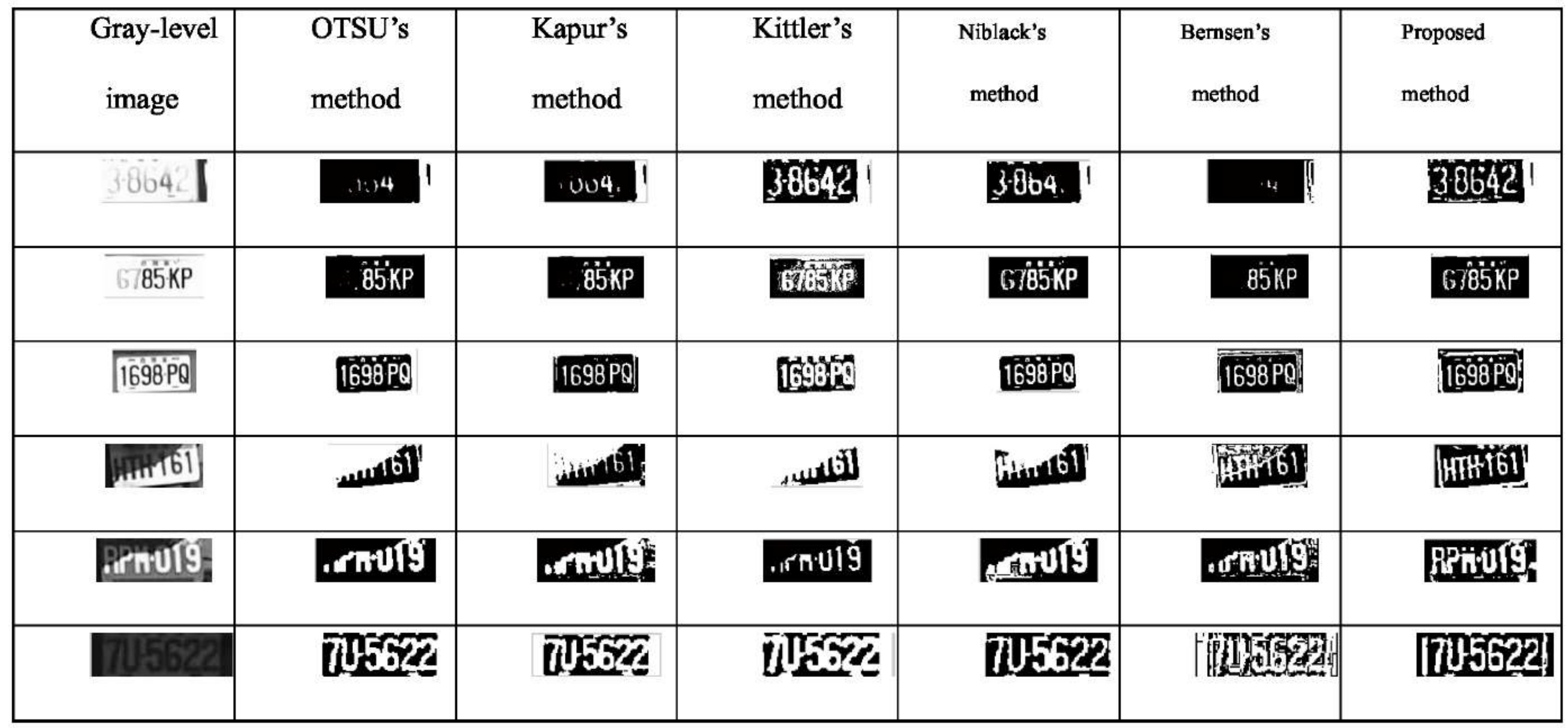
How to make binary image?

- Binarization is also very important in LPR(License Plate Recognition)



How to make binary image?

- Binarization is also very important in LPR(License Plate Recognition)



Adaptive Binarization: A New Approach to Licence Plate Characters Segmentation [IEEE 2012]

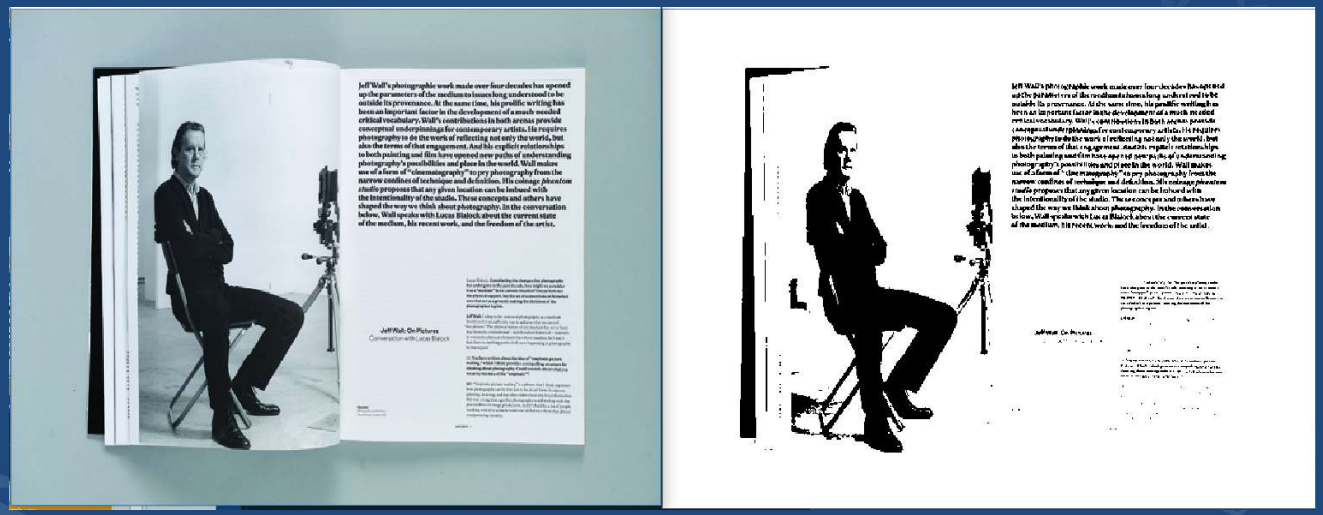
example code for basic threshold

- Let's look basic code for using threshold function

C++: double threshold(InputArray **src**, OutputArray **dst**, double **thresh**, double **maxval**, int **type**)

Python: cv2.threshold(src, thresh, maxval, type[, dst]) → retval, dst

```
int main(int, char)
{
    namedWindow("img", 0);
    namedWindow("binary", 0);

    Mat img = imread("book.jpg");

    Mat gray, binary;
    cvtColor(img, gray, CV_RGB2GRAY);
    threshold(gray, binary, 128, 255, CV_THRESH_BINARY);


    imshow("img", img);
    imshow("binary", binary);

    waitKey(0);


    return 0;
}
```



http://study.marearts.com/2018/05/image-binary-simple-example-for-image.html

let's use threshold function in OpenCV

- more detail..

C++: double threshold(InputArray **src**, OutputArray **dst**, double **thresh**, double **maxval**, int **type**)

Python: cv2.threshold(src, thresh, maxval, type[, dst]) → retval, dst

- **THRESH_BINARY**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_BINARY_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

- **THRESH_TRUNC**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO**

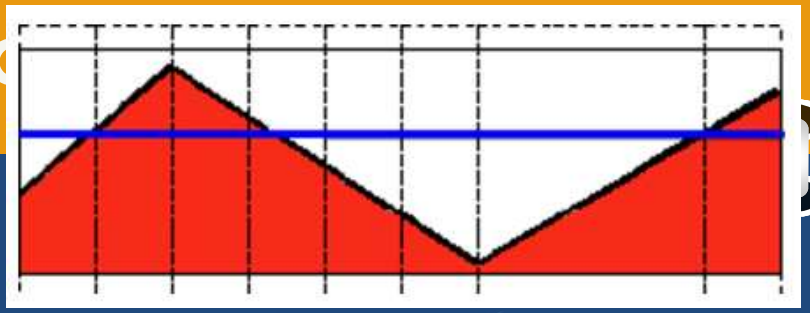
$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

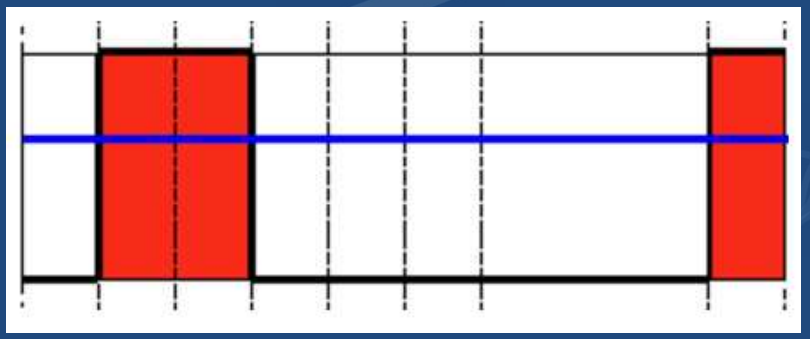
there are 5 options

let's use threshold func



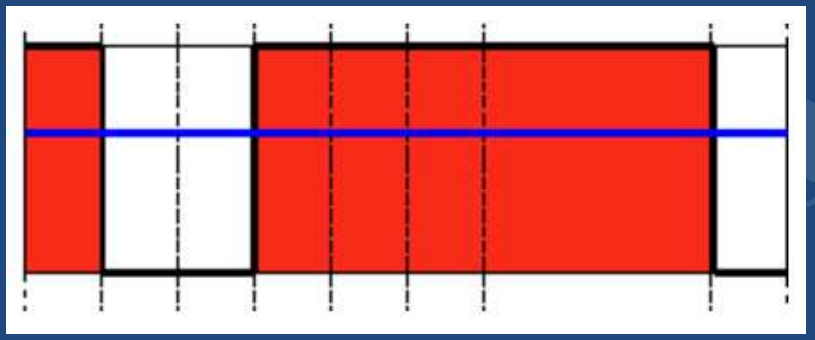
- 1. THRESH_BINARY

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxVal} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

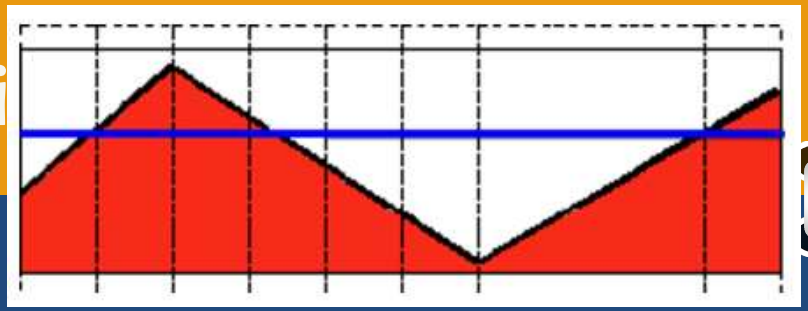


- 2. THRESH_BINARY_INV

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxVal} & \text{otherwise} \end{cases}$$

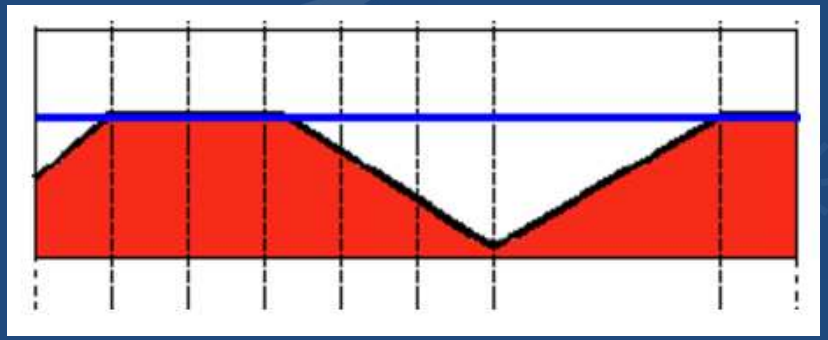


let's use threshold functi



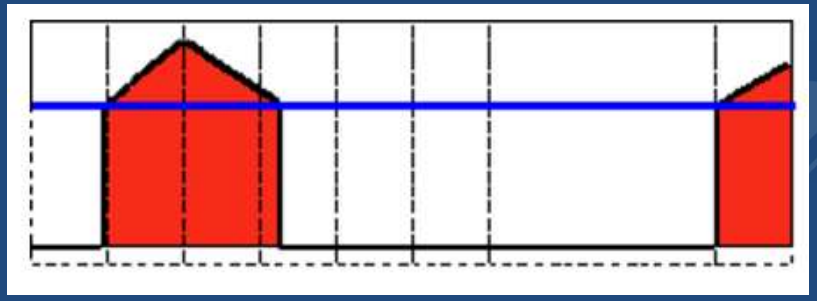
- 3. THRESH_TRUNC

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

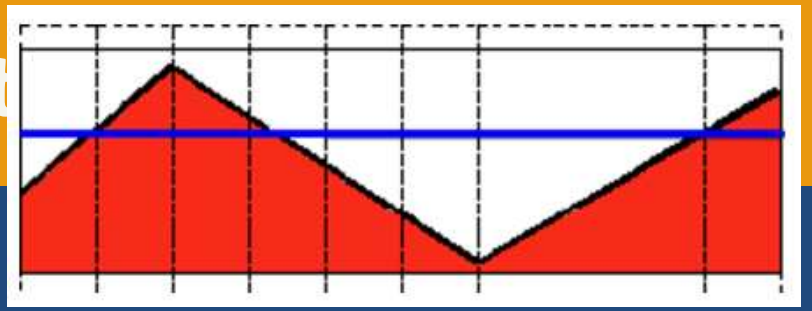


- 4. THRESH_TOZERO

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

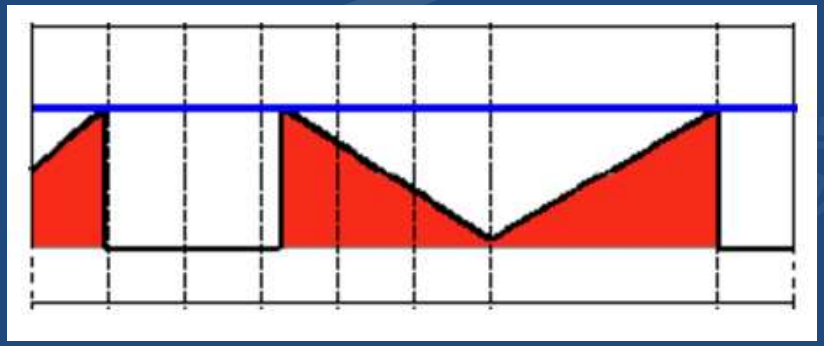


let's use threshold funct



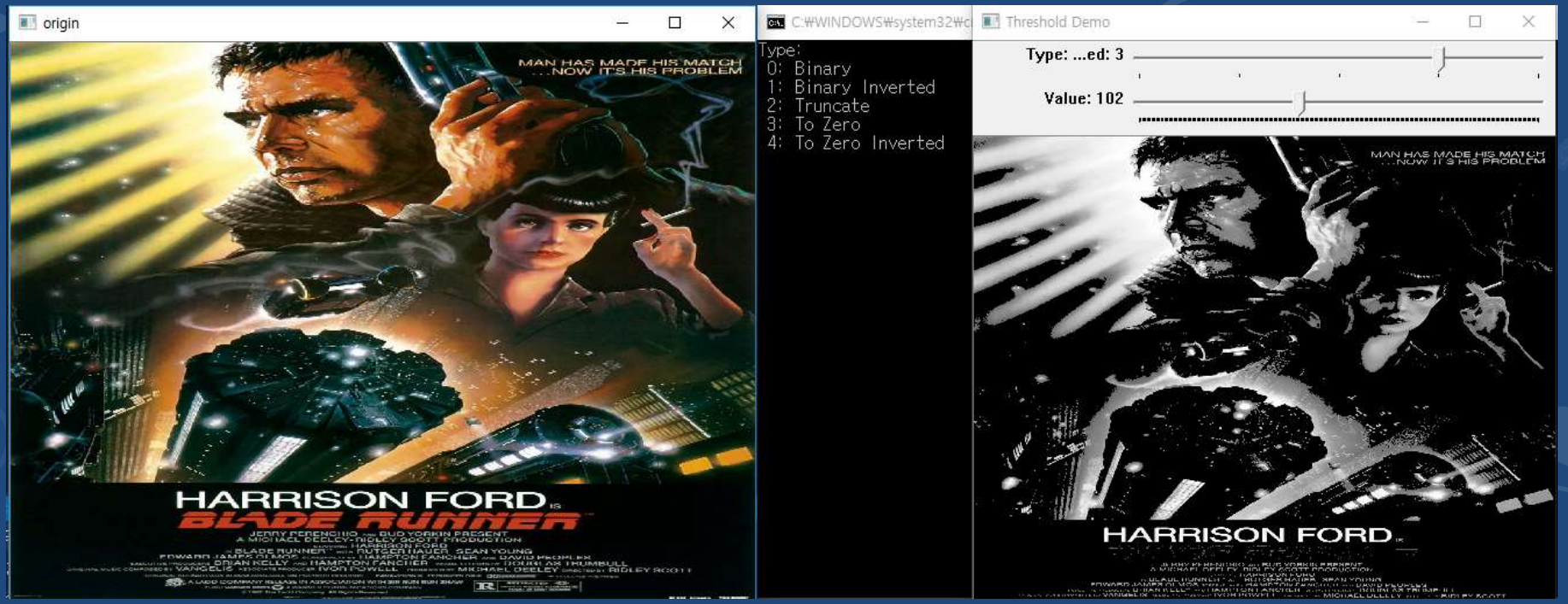
- 5. THRESH_TOZERO_INV

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$



example code for applying various options

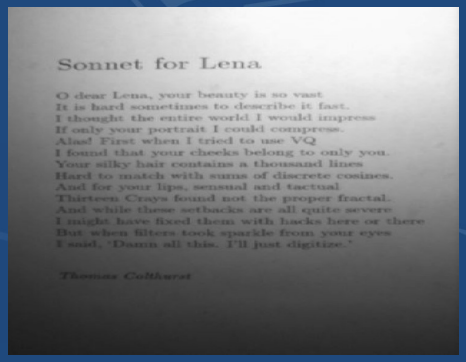
- Let's look threshold function in detail
  - by applying various options



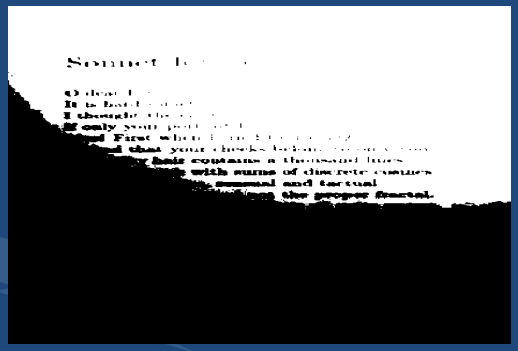
http://study.marearts.com/2018/05/threshold-demo-in-opencv.html

adaptive threshold

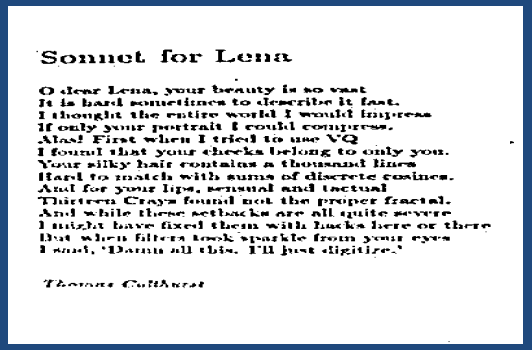
- One threshold can not binarize in different brightness.
- So we get different thresholds for different regions of the same image.
  - adaptive thresholding
  - the algorithm calculate the threshold for a small regions of the image



origin



threshold



adaptiveThreshold

adaptive threshold

adaptiveThreshold

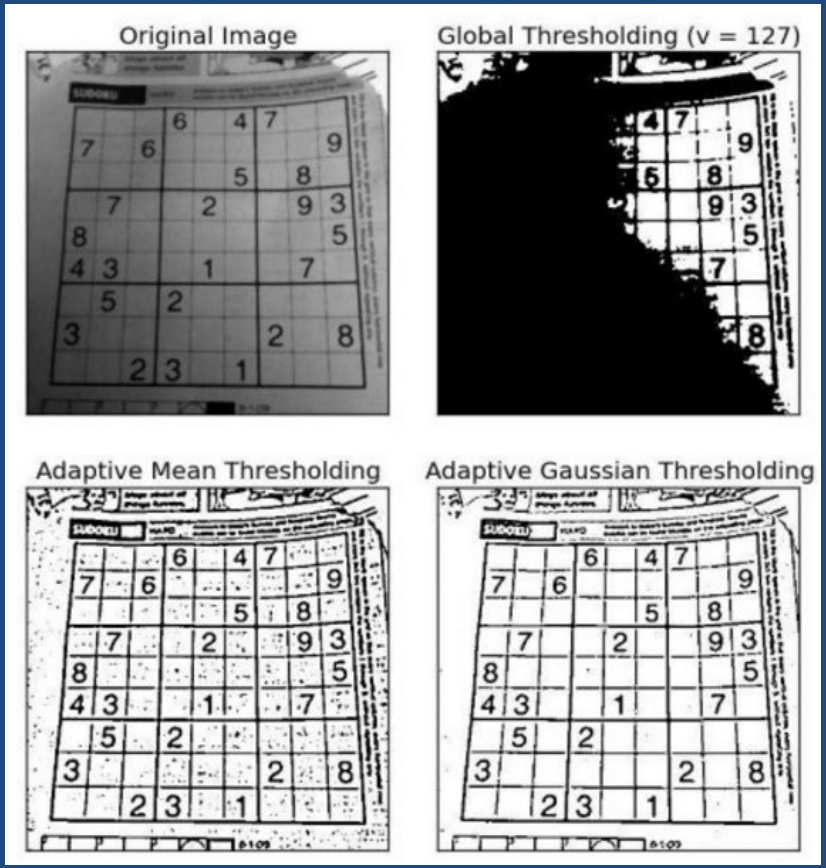
C++: void adaptiveThreshold(InputArray **src**, OutputArray **dst**, double **maxValue**, int **adaptiveMethod**, int **thresholdType**, int **blockSize**, double **C**)

Python: cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) → dst

Parameters:

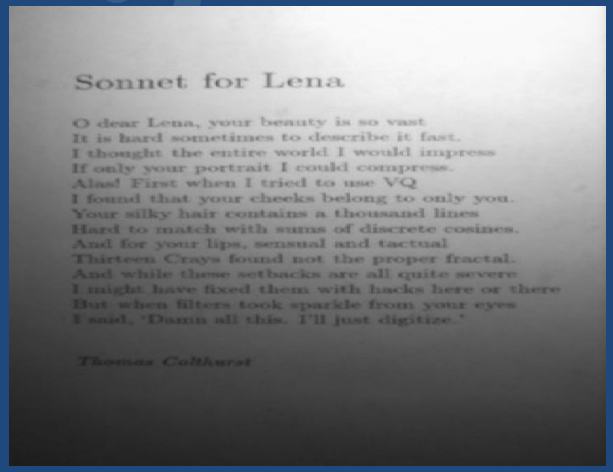
- **src** – Source 8-bit single-channel image.
- **dst** – Destination image of the same size and the same type as src .
- **maxValue** – Non-zero value assigned to the pixels for which the condition is satisfied. See the details below.
- **adaptiveMethod** – Adaptive thresholding algorithm to use, ADAPTIVE_THRESH_MEAN_C or ADAPTIVE_THRESH_GAUSSIAN_C . See the details below.
- **thresholdType** – Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV .
- **blockSize** – Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.
- **C** – Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well.

- ADAPTIVE_THRESH_MEAN_C
- ADAPTIVE_THRESH_GAUSSIAN_C

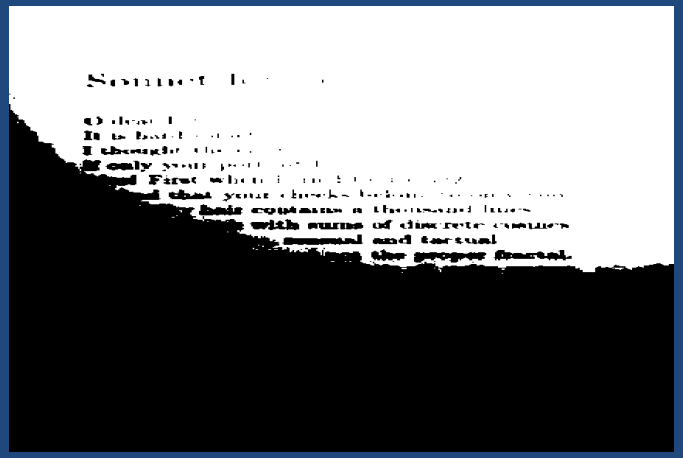


example code for adaptiveThreshold

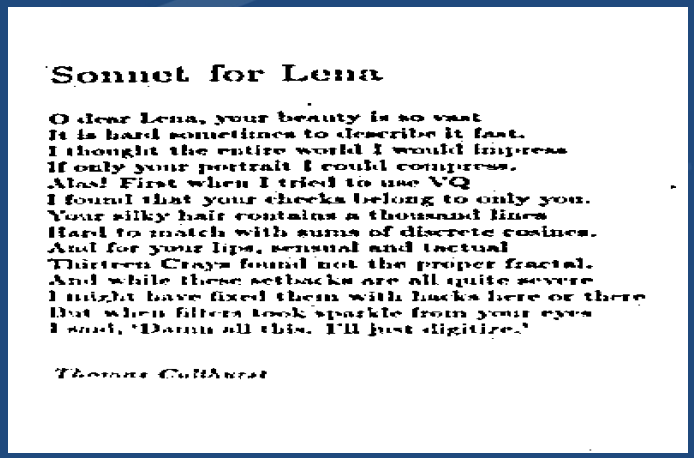
- adaptiveThreshold vs threshold



origin



threshold

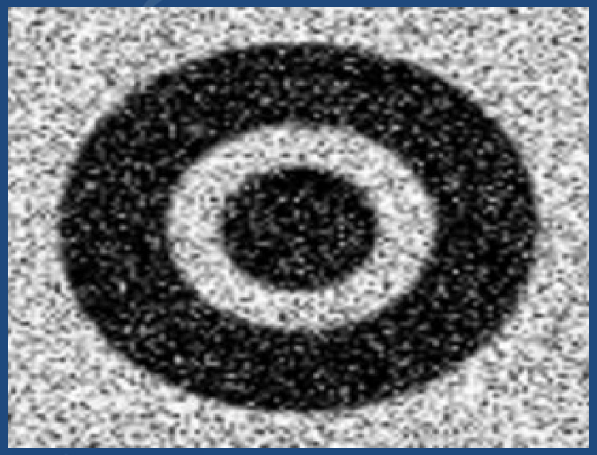


adaptiveThreshold

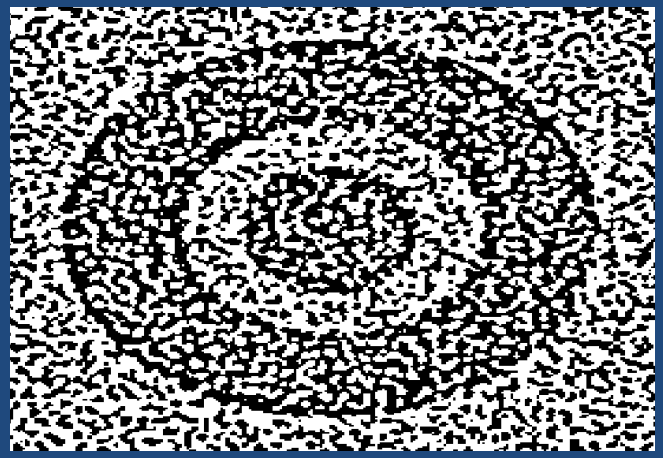
http://study.marearts.com/2018/05/adaptivethreshold-test-code-in-opencv.html

Otzu Binarization

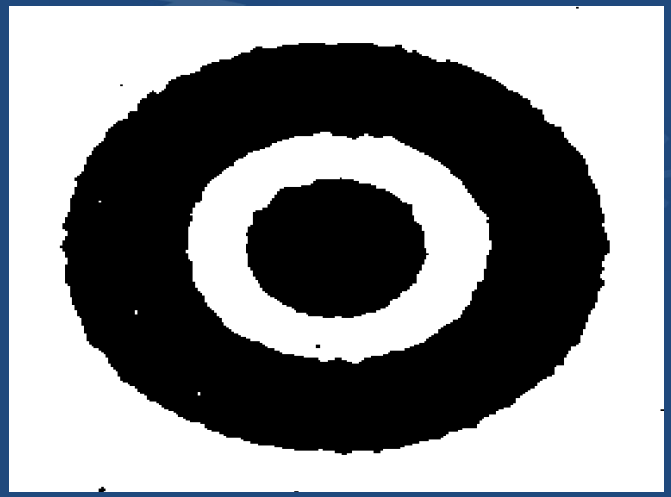
- Otzu method



origin



adaptiveThreshold



Otzu method

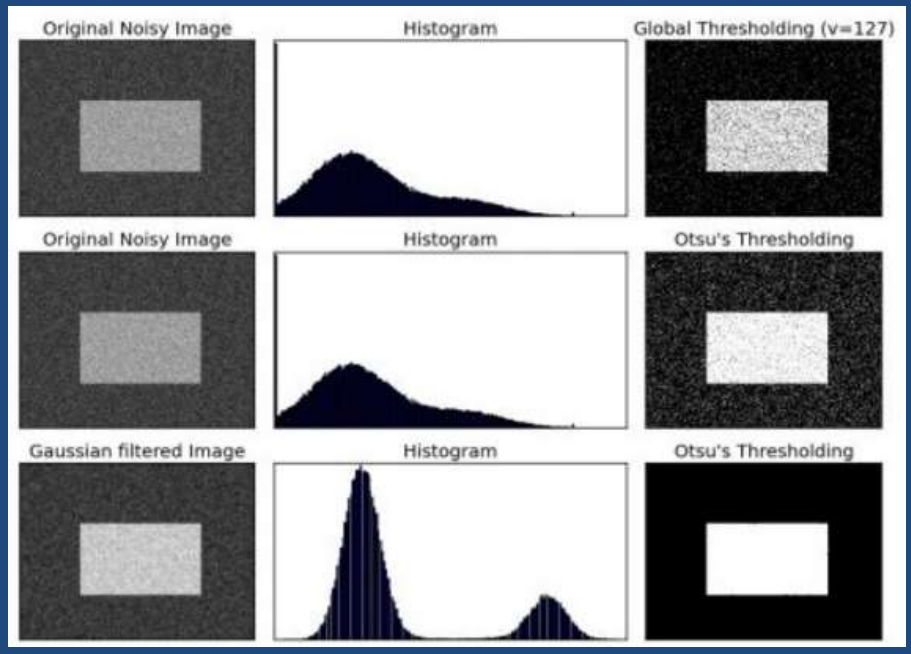
http://study.marearts.com/2017/04/otzu-binary-simple-example.html

Otzu Binarization

- Otzu method

N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst., Man, Cybern., vol. SMC-9, pp. 62-66, 1979

  - threshold(Mat, 0, 255, THRESH_BINARY + THRESH_OTSU);



Otzu Binarization

- Otsu's algorithm tries to find a threshold value (t) which minimizes the **weighted within-class variance** given by the relation :

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_1(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_1(t)]^2 \frac{P(i)}{q_2(t)}$$

good reference site:

http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

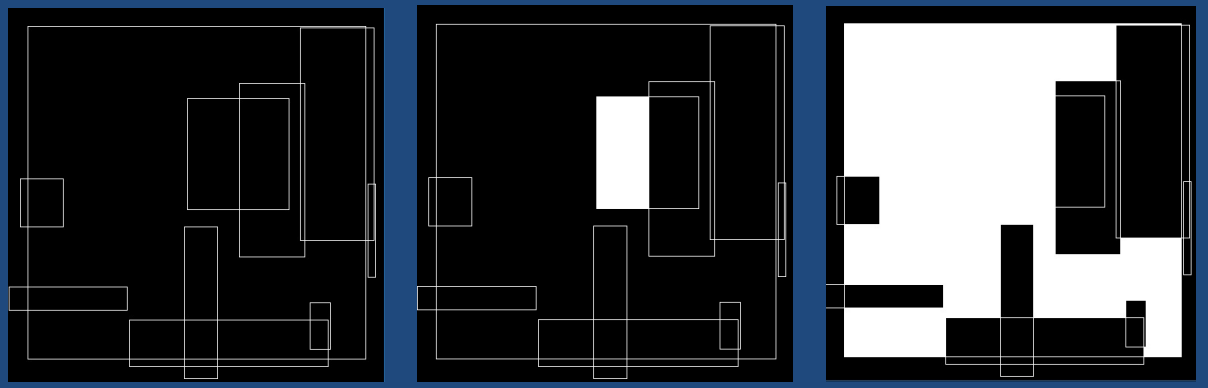
http://darkpgmr.tistory.com/115

other functions for apply point processing

- foodFill function

C++: int floodFill(InputOutputArray **image**, Point **seedPoint**, Scalar **newVal**, Rect* **rect**=0, Scalar **loDiff**=Scalar(), Scalar **upDiff**=Scalar(), int **flags**=4)

```
for (int i = 0; i < count; ++i)
{
    x = rand() % width;
    y = rand() % height;
    w = rand() % (x - width);
    h = rand() % (y - height);

    printf("(%d): left:%d top:%d - right:%d bottom:%d (width:%d, h
    rectangle(img, Rect(x, y, w, h), CV_RGB(255, 255, 255), 1);
}



while (1){
    if (click_x != 0 && click_y != 0)
    {
        floodFill(img, Point(click_x, click_y), Scalar(255));
    }

    imshow("img", img);

    if (waitKey(10) > 0)
```



http://study.marearts.com/2017/04/floodfill-opencv-function-example.html

other functions for apply point processing

- foodFill function

https://www.learnopencv.com/filling-holes-in-an-image-using-opencv-python-c/

Figure 2.

Steps for implementing imfill in OpenCV

Please refer to Figure 2. while reading the steps below.

1. Read in the image.
2. Threshold the input image to obtain a binary image.
3. Flood fill from pixel (0, 0). Notice the difference between the outputs of step 2 and step 3 is that the background in step 3 is now white.
4. Invert the flood filled image (i.e. black becomes white and white becomes black).
5. Combine the thresholded image with the inverted flood filled image using bitwise OR operation to obtain the final foreground mask with holes filled in. The image in Step 4 has some black areas inside the boundary. By design the image in Step 2 has those holes filled in. So we combine the two to get the mask.

SATYA MALLICK

I am an entrepreneur who loves Computer Vision and Machine Learning. I have a dozen years of experience (and a Ph.D.) in the field.

I am a co-founder of TAAZ Inc where the scalability, and robustness of our computer vision and machine learning algorithms have been put to rigorous test by more than 100M users who have tried our products. Read More...

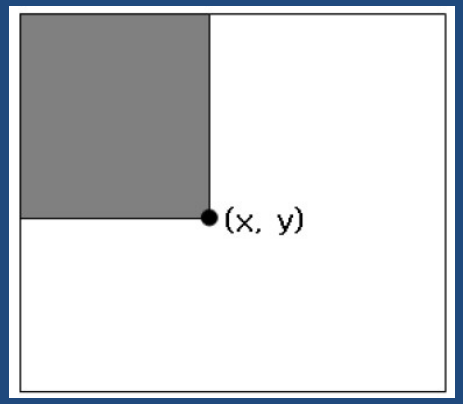
other functions for apply point processing

- integral function
  - Quickly method to add all pixel values for a particular region.

```
C++: void integral(InputArray src, OutputArray sum, int sdepth=-1 )
```

  - Conventional method to add for a region

$$\mathrm{sum}(X,Y) = \sum_{x<X,y<Y} \mathrm{image}(x,y)$$

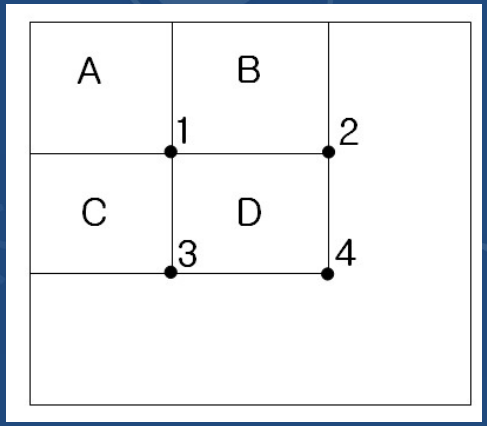


other functions for apply point processing

- integral function

C++: void integral(InputArray **src**, OutputArray **sum**, int **sdepth**=-1)

$$\sum_{x_1 \le x < x_2,\, y_1 \le y < y_2} \text{image}(x,y) = \text{sum}(x_2,y_2) - \text{sum}(x_1,y_2) - \text{sum}(x_2,y_1) + \text{sum}(x_1,y_1)$$



D Area = (Point 4 + Point 1) – (Point 2 + Point 3)

other functions for apply point processing

- integral function

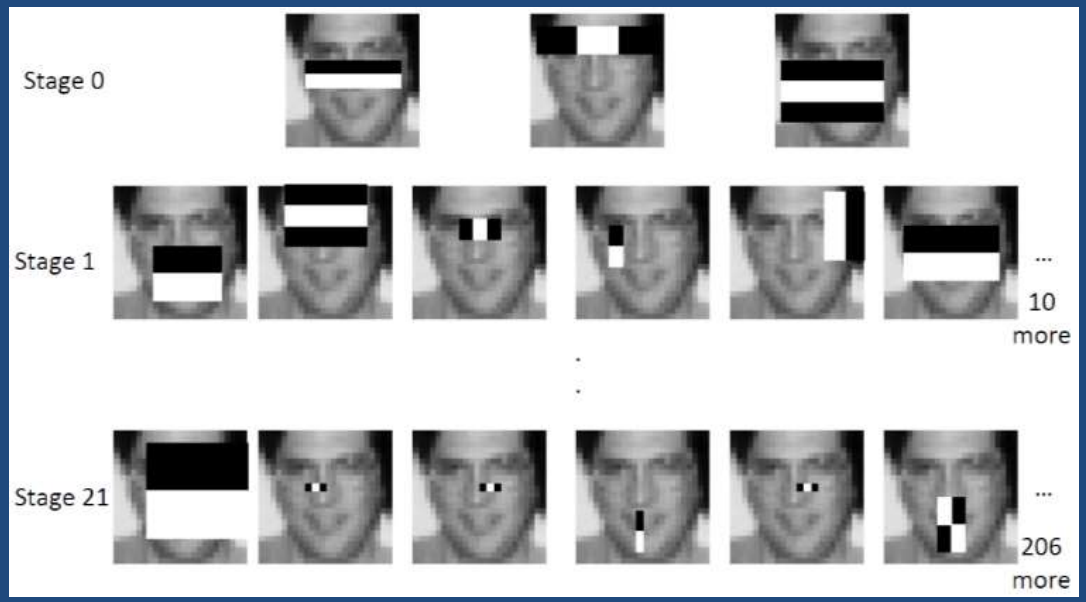
```
origin
[  6,   7,   9,   9,   7,   0,   6,   3,   6,   9;
   1,   8,   7,   8,   5,   3,   8,   1,   7,   3;
   3,   3,   5,   4,   8,   2,   6,   1,   2,   2;
   6,   1,   0,   7,   3,   5,   0,   6,   3,   3;
   7,   5,   0,   5,   3,   0,   2,   7,   1,   7;
   9,   8,   8,   3,   9,   5,   4,   1,   8,   3;
   8,   1,   8,   7,   7,   0,   3,   8,   8,   3;
   8,   9,   5,   1,   1,   3,   3,   3,   4,   7;
   2,   7,   6,   8,   2,   4,   9,   5,   6,   1;
   5,   0,   5,   7,   8,   4,   1,   0,   4,   8]
for loop : sum = 23

integral
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
 0, 6, 13, 22, 31, 38, 38, 44, 47, 53, 62;
 0, 7, 22, 38, 55, 67, 70, 84, 88, 101, 113;
 0, 10, 28, 49, 70, 90, 95, 115, 120, 135, 149;
 0, 16, 35, 56, 84, 107, 117, 137, 148, 166, 183;
 0, 23, 47, 68, 101, 127, 137, 159, 177, 196, 220;
 0, 32, 64, 93, 129, 164, 179, 205, 224, 251, 278;
 0, 40, 73, 110, 153, 195, 210, 239, 266, 301, 331;
 0, 48, 90, 132, 176, 219, 237, 269, 299, 338, 375;
 0, 50, 99, 147, 199, 244, 266, 307, 342, 387, 425;
 0, 55, 104, 157, 216, 269, 295, 337, 372, 421, 467]

 p1:6.000000, p2:22.000000, p3:10.000000, p4:49.000000
integral : sum = 23.000000
```

```
////////////////////////////////////////////////////////

Mat integralImg;
integral(img, integralImg, CV_64F);

cout << integralImg << endl;

double p1 = integralImg.at<double>((y), (x));
double p2 = integralImg.at<double>((y), (x + w));
double p3 = integralImg.at<double>((y + h), (x));
double p4 = integralImg.at<double>((y + h), (x + w));

printf("%lf %lf %lf %lfWn", p1, p2, p3, p4);
printf("integral : sum = %lf Wn", (p1+p4) - (p2+p3));
```

http://study.marearts.com/2018/05/opencv-integral-test-source-code.html

other functions for apply point processing

- integral function
  - It is very useful to calculate a large number of haar-like features in the AdaBoost algorithm.



Thank you.

- See you later ~

In from of ChangWon University dormitory
(picture from http://blog.naver.com/PostList.nhn?blogId=maple3419)