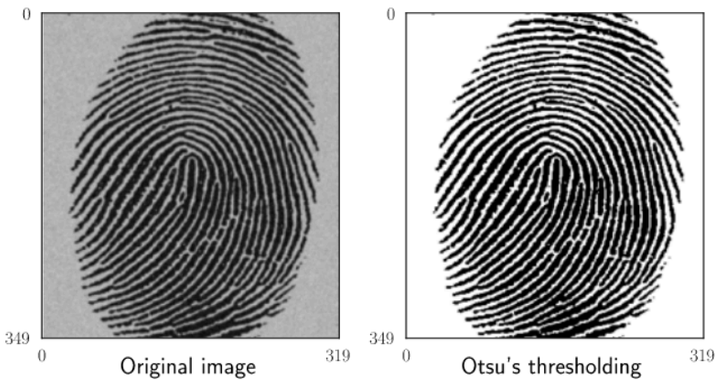
Image Processing - Thresholding



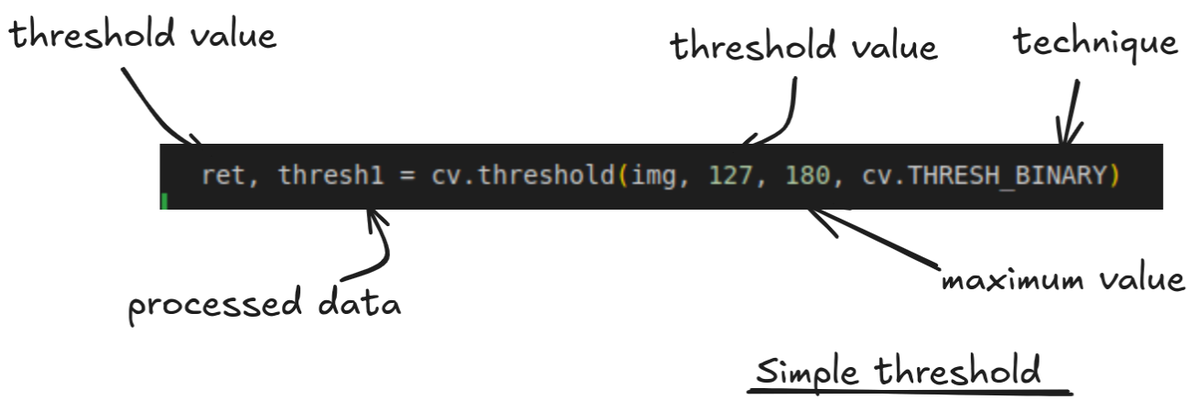
There are multiple type of Thresholding techniques available in OpenCV

- Simple Thresholding
- Adaptive Thresholding
- Otsu's Thresholding

Above techniques are differentiated based on how they calculate the threshold value for the process.

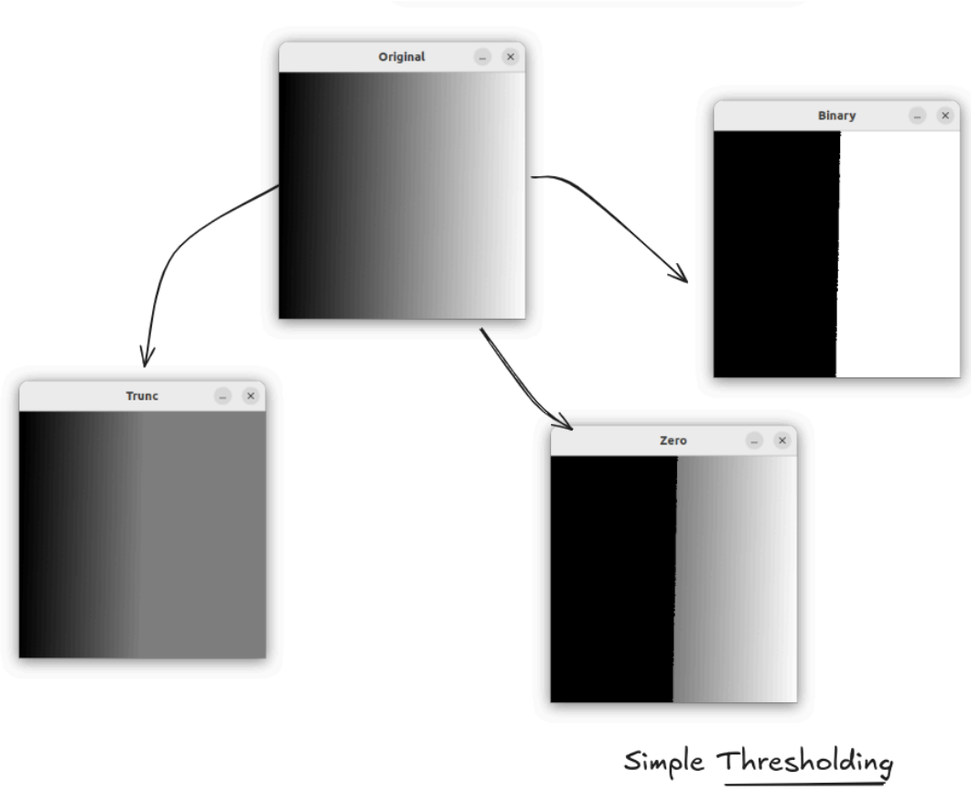
In Thresholding process, if the pixel value is smaller than the threshold, it will be assigned a value of 0, otherwise it is set to a maximum value(usually specified in the function itself.)

1. Simple Thresholding



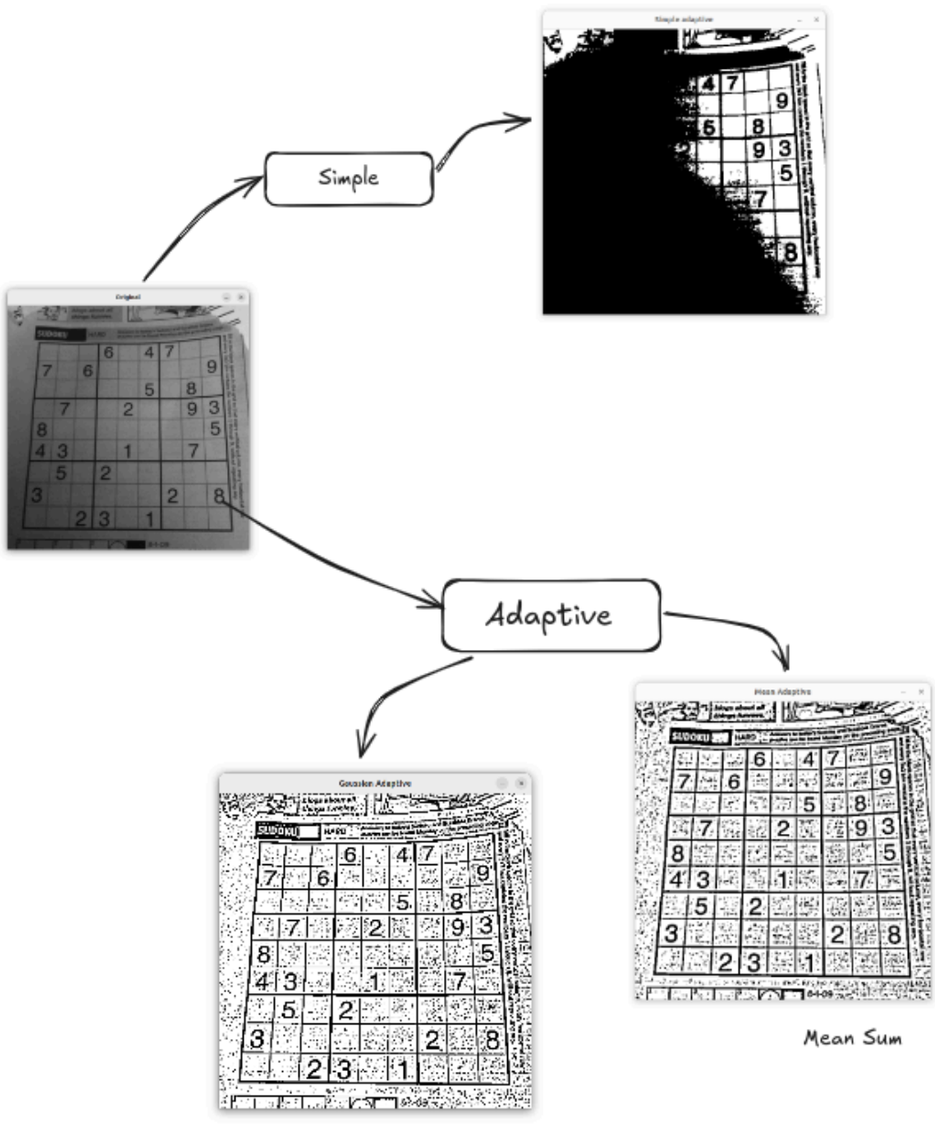
- Function `cv.threshold` uses 4 parameters.
- it returns two outputs. The first is the threshold that was use and second output is the process image.
- First parameter is the image source(Grayscale is preferred)
- Second parameter is the threshold value.
- Third parameter is the maximum value(given to the pixels having intensity greater than threshold value)
- Fourth parameter is type of thresholding technique. Techniques available to use are :-
  - `THRESH_BINARY`
  - `THRESH_BINARY_INV`
  - `THRESH_TRUNC`
  - `THRESH_TOZERO`

- `THRESH_TOZERO_INV`



Simple Thresholding

2. Adaptive Thresholding



Mean Sum

Gaussian Weighted Sum

Unlike simple thresholding, where we used one global value as threshold, in this technique we use different threshold values for different regions. This makes it easier to process image under different lighting conditions. Adaptive algorithms can analyze the image for differing light conditions and process the image accordingly.

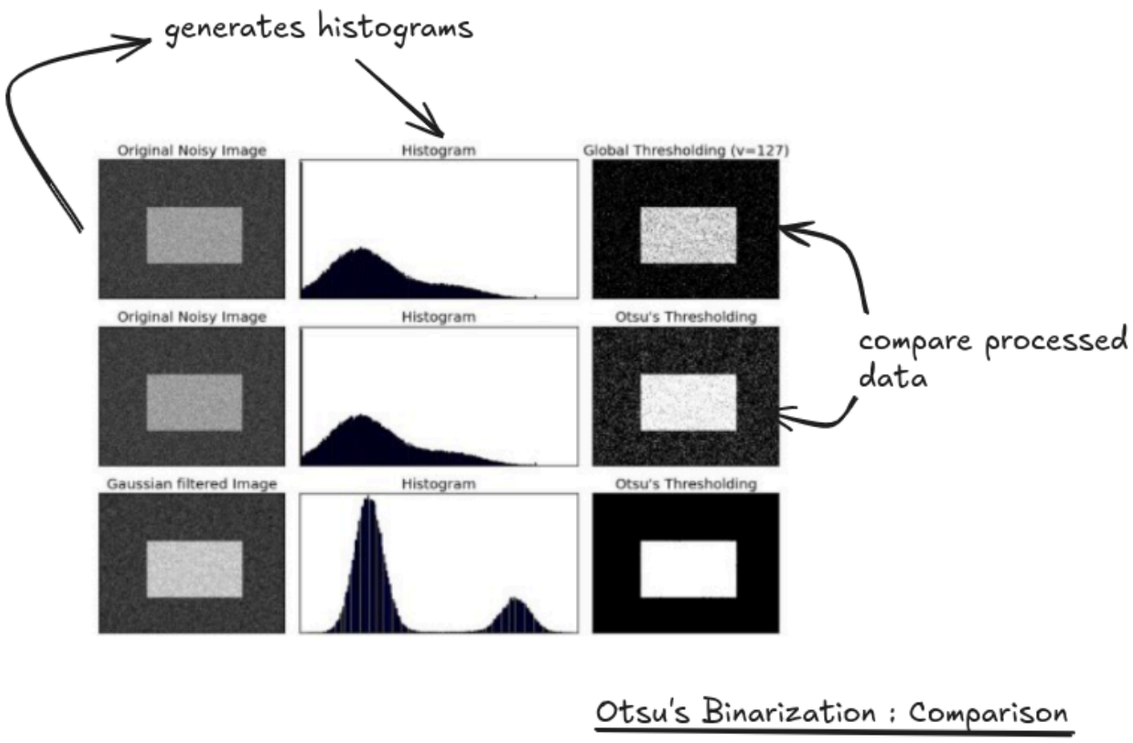
`cv.adaptiveThreshold` : Function is used to calculate adaptive threshold value.
This function accepts three input parameters:

- Algorithm determines the threshold for a pixel based on a small region.
- There are two different techniques available to calculate threshold for a small region.
- `ADAPTIVE_THRESH_MEAN_C` : Mean of the neighbourhood region minus the constant **C** (we pass it)
- `ADAPTIVE_THRESH_GAUSSIAN_C` : Gaussian-weighted sum of the neighbourhood values minus the constant **C.**
- **blockSize**: determines the size of the neighbourhood area
- **C**: constant that is subtracted from the mean or weighted sum of the pixels, generated by adaptive algorithms.

3. Otsu's Binarization

In otsu's binarization algorithm, unlike simple and adaptive thresholding, the cutoff threshold value is instead calculated automatically.

`cv.threshold()` : method is used (same as simple thresholding)



Otsu's Binarization : Comparison

Algorithm :-
First of all, the source image's histogram is computed. Depending on the histogram, the threshold value is calculated such as the threshold value is equally distant from two peaks.

A good threshold values would be in the middle of those two peak values in the generated histogram of the source image.

`cv.threshold` is also used here, though we pass `THRESH_OTSU` as an extra flag.

Source Code :-

```
import cv2 as cv
import numpy as np

img = cv.imread("samples/otsu-raw.jpeg", cv.IMREAD_GRAYSCALE)

# global simple thresholding
ret, th1 = cv.threshold(img, 127, 255, cv.THRESH_BINARY)


#Otsu's thresholding
ret, th2 = cv.threshold(img, 0, 25, cv.THRESH_BINARY+cv.THRESH_OTSU)

cv.imshow("Original Image", img)
cv.imshow("Simple threshold", th1)
cv.imshow("Otsu's processed", th2)

cv.waitKey(0)
cv.destroyAllWindows()
```