Computer Vision

07: Image Thresholding

Dr Ram Prasad Krishnamoorthy

Associate Professor
School of Computing and Data Science

ram.krish@saiuniversity.edu.in

SAI UNIVERSITY

Image Thresholding

Single point operations:

Generates new image whose pixel depends only the corresponding pixel in original image.

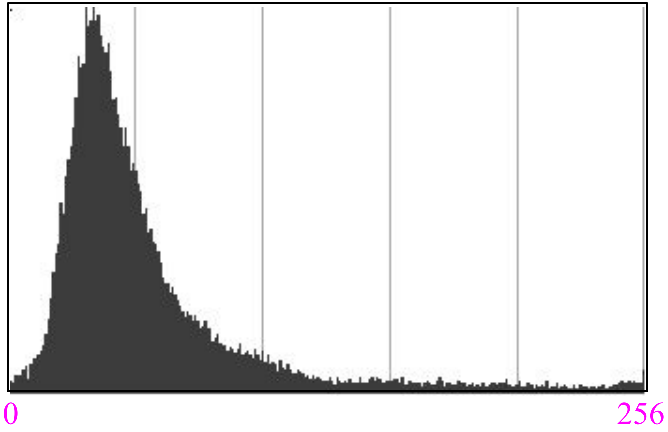
- Gamma corrections
- Thresholding
- Histogram equalization

Image Histograms

Image Thresholding

Image Histograms:

- Represents the pixel intensity histogram.
- Counts the number of pixels found at each intensity level.
- Graphically represents 256 numbers showing the pixel distribution.



Thresholding

Image Thresholding

Thresholding:

In many applications, it is useful to separate out foreground which contains object from the background. Threshold is a convenient way to perform this segmentation.

- Typically thresholding is performed on **grayscale image**, though it can be done on color images.
- The output is a binary image - ***black*** for **background** and ***white*** for **foreground** (reverse is also okay)
- **Segmentation** is defined by a single parameter called *intensity-threshold*.

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

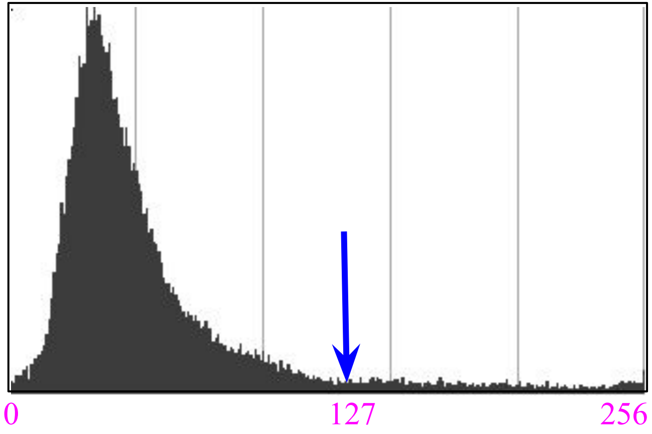
Image Thresholding

Thresholding

Single thresholding is not appropriate for all images.

It will be helpful to look at *intensity-histograms* to determine about feasibility of thresholding.

- If there is a bi-modal distribution of intensities, then binary thresholding can be achieved easily.



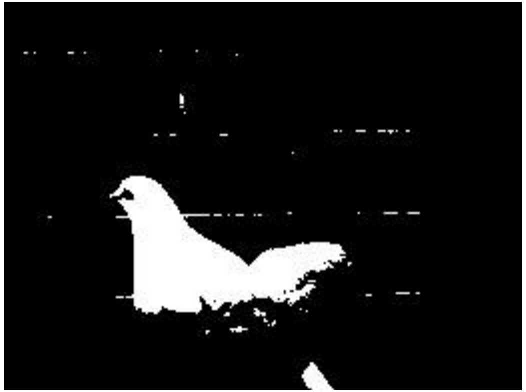
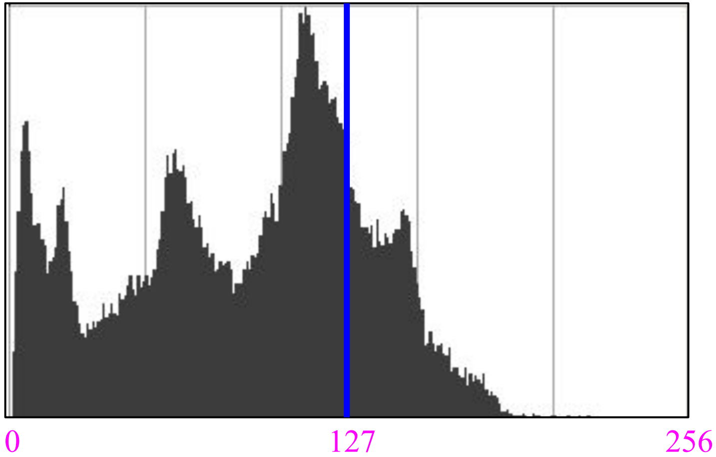


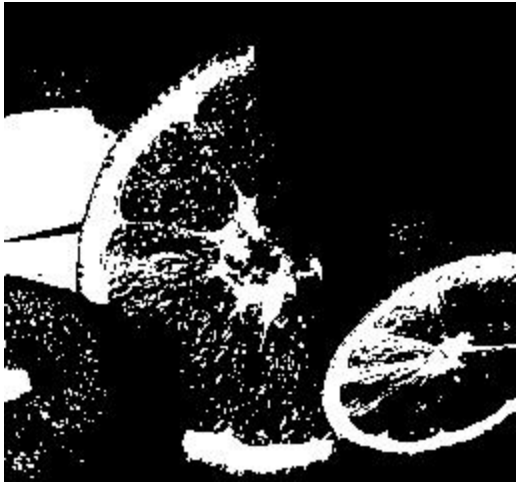
Image Thresholding

Thresholding

Single thresholding is not appropriate for all images.

- When there is no bimodal distribution, then it becomes difficult to threshold.
- Due to lot of pixel value variations, we might need adaptive thresholding.





Various Threshold Methods

Purpose: Isolate objects from the background

Image Thresholding

Thresholding

Various thresholds available in openCV:

THRESH_BINARY Python: cv.THRESH_BINARY	$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxval} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\mathtt{dst}(x,y) = \begin{cases} \mathtt{threshold} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\mathtt{dst}(x,y) = \begin{cases} \mathtt{src}(x,y) & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$

Used when object is darker than the background.

Part of the image remains unchanged.

Keeps dark regions as they are, cuts off anything too bright.

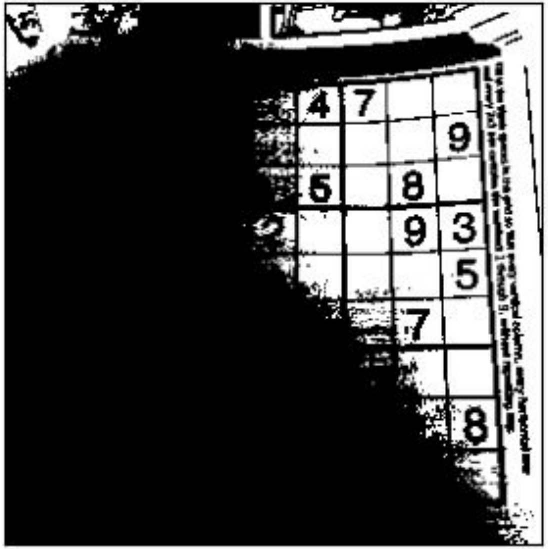
Image Thresholding

Thresholding

Various thresholds available in openCV:

Original Image

BINARY



BINARY_INV

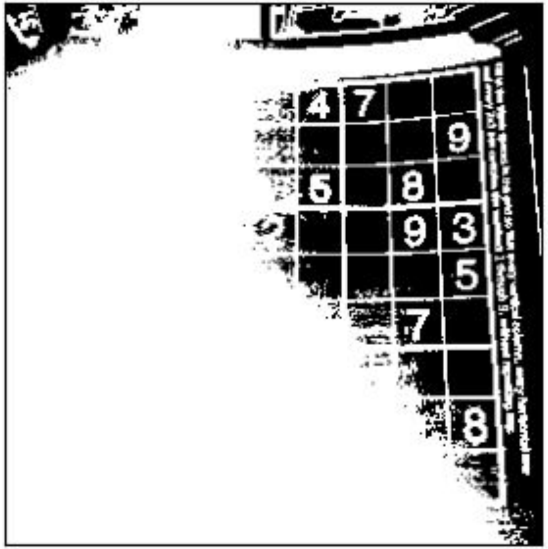


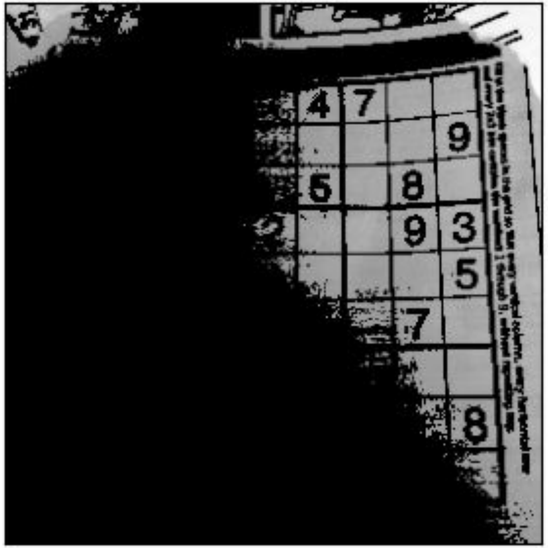
Image Thresholding

Thresholding

Various thresholds available in openCV:

TRUNC

TOZERO



TOZERO_INV

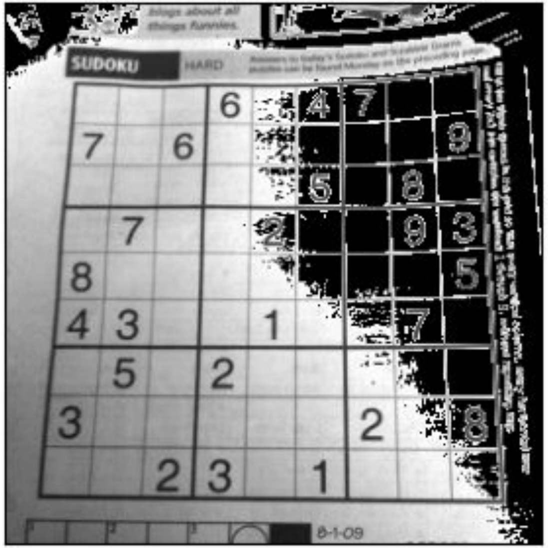


Image Thresholding

Thresholding

Adaptive Thresholding

- *Threshold is calculated differently for each small region using local image statistics (mean or weighted mean)*
- *Effective for images with varying lighting—threshold adapts across regions.*
- *Ideal for uneven illumination and complex backgrounds.*

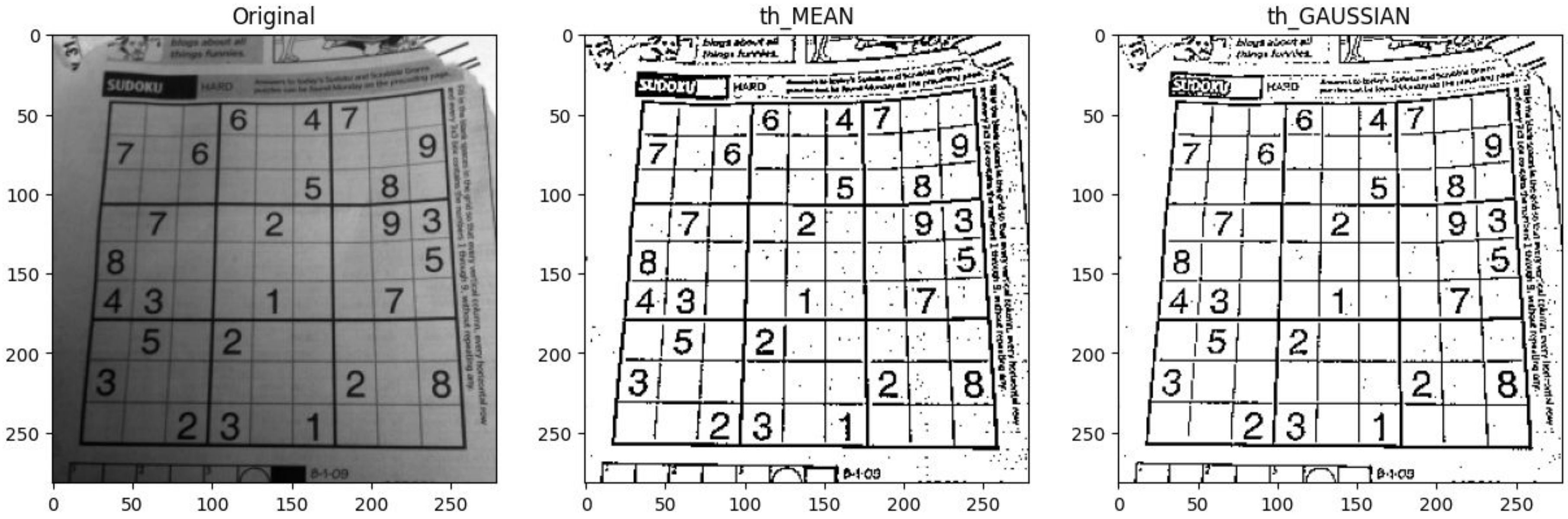
Various thresholds available in openCV:

ADAPTIVE_THRESH_MEAN_C Python: cv.ADAPTIVE_THRESH_MEAN_C	the threshold value $T(x, y)$ is a mean of the $\texttt{blockSize} \times \texttt{blockSize}$ neighborhood of (x, y) minus C
ADAPTIVE_THRESH_GAUSSIAN_C Python: cv.ADAPTIVE_THRESH_GAUSSIAN_C	the threshold value $T(x, y)$ is a weighted sum (cross-correlation with a Gaussian window) of the $\texttt{blockSize} \times \texttt{blockSize}$ neighborhood of (x, y) minus C . The default sigma (standard deviation) is used for the specified blockSize .

Image Thresholding

Thresholding

Various thresholds available in openCV:



Otsu Method

Image Thresholding

Thresholding

Various thresholds available in openCV:

THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value

Otsu's method is a popular technique in image processing for automatic image thresholding.

- It's particularly useful for images with bimodal histograms, where two distinct peaks represent foreground and background pixels
- Proposes a way to mathematically measure this "separability" using a single number.
- The method then systematically tests every possible threshold value (from 0 to 255) and picks the one that maximizes this measure.

Image Thresholding

Key Steps

- **Normalized histogram:** For each intensity level i, calculate the probability $p_i = \frac{n_i}{N}$ where n_i is the number of pixels at intensity i, and N is the total number of pixels.
- **Class probabilities:** For threshold t:
  - $\omega_0(t) = \sum_{i=0}^{t-1} p(i)$ (background)
  - $\omega_1(t) = \sum_{i=t}^{L-1} p(i)$ (foreground), where L is the number of possible intensities.
- **Class means:**
  - $\mu_0(t) = \frac{\sum_{i=0}^{t-1} i \cdot p(i)}{\omega_0(t)}$
  - $\mu_1(t) = \frac{\sum_{i=t}^{L-1} i \cdot p(i)}{\omega_1(t)}$

Image Thresholding

- **Between-class variance:** For every threshold:

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

.

- **Find the optimal threshold t^*:** Iterate through all possible t, selecting the one where $\sigma_b^2(t)$ is maximized (best separation).
  - maximizing $\sigma_b^2(t)$ means finding the threshold where the two classes are as far apart as possible in terms of intensity, weighted by their probabilities, ensuring good separation between foreground and background.
- **Segment:** Assign all pixels with intensity > t^* to the foreground, and ≤ t^* to the background.

Image Thresholding

Thresholding