

Task 1-A program that binarizes an image.

Matriculation Number: 10004430

Date Of submission: 26-06-2025

Introduction:

In this report we are doing image binarization. In this particular process we are converting a grey scale image to two different thresholding i.e.,

1. Global Adaptive Thresholding
2. Local Adaptive Thresholding.

Methodology:

1.Global Adaptive Thresholding-

Global Adaptive Thresholding is a technique used to convert a grayscale image into a binary image (black and white) based on a *single threshold value* computed from the image itself.

Let's break it down intuitively:

What It Does

Instead of using a fixed threshold (like 127 out of 255), global adaptive thresholding **calculates the best threshold** based on the image's content. It adapts to the image's pixel intensities by refining the threshold over several iterations.

2. Local Adaptive Thresholding-

Local Adaptive Thresholding is a technique used to binarize an image based on **local statistics**—meaning the threshold value is calculated for each pixel using its surrounding neighbourhood rather than using a global value.

Why Use Local Adaptive Thresholding?

It's especially useful when the image has:

- **Uneven lighting**
- **Shadows or gradients**
- **Variable background intensity**

Unlike global thresholding, this method adapts to brightness changes over small regions, preserving fine details better in diverse lighting conditions.

Algorithm:

1.Global Adaptive Thresholding-

- Initial threshold = mean image intensity
- Iterative refinement using class means μ_1, μ_2
- Converges when $\Delta T < \epsilon$

2. Local Adaptive Thresholding

- Window size $W_w \times H_w$
- Each pixel compared to the local mean
- Binarization is locally adaptive to lighting variations

Implementation:

- Platform: Python
- IDE: Spyder
- Input: Original image
- Output: Two binarized images
- Adjustable parameters: ϵ , window size $W_w \times H_w$

Code:

```

1  import cv2
2  import numpy as np
3
4  def global_adaptive_threshold(image, epsilon=1.0):
5      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
6      T = np.mean(gray)
7      delta = float('inf')
8
9      while delta > epsilon:
10         lower = gray[gray <= T]
11         upper = gray[gray > T]
12
13         if lower.size == 0 or upper.size == 0:
14             break
15
16         mu1 = np.mean(lower)
17         mu2 = np.mean(upper)
18         new_T = (mu1 + mu2) / 2
19         delta = abs(T - new_T)
20         T = new_T
21
22     _, binary = cv2.threshold(gray, T, 255, cv2.THRESH_BINARY)
23     return binary
24
25 def local_adaptive_threshold(image, window_size=(15, 15)):
26     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
27     Ww, Hw = window_size
28     pad_w, pad_h = Ww // 2, Hw // 2
29
30     # Pad the image
31     padded = cv2.copyMakeBorder(gray, pad_h, pad_h, pad_w, pad_w, cv2.BORDER_REFLECT)
32     result = np.zeros_like(gray)
33
34     for y in range(gray.shape[0]):
35         for x in range(gray.shape[1]):
36             window = padded[y+pad_h:x+pad_h+Hw, x+pad_w:x+pad_w+Ww]
37             local_mean = np.mean(window)
38             result[y, x] = 255 if gray[y, x] > local_mean else 0
39
40     return result
41
42 if __name__ == "__main__":
43     image = cv2.imread("C://Users/HP/Downloads/avengers.jpeg")
44     global_result = global_adaptive_threshold(image, epsilon=1.0)
45     local_result = local_adaptive_threshold(image, window_size=(15, 15))
46
47     cv2.imshow("Global Adaptive Thresholding", global_result)
48     cv2.imshow("Local Adaptive Thresholding", local_result)
49     cv2.waitKey(0)
50     cv2.destroyAllWindows()
51
52     cv2.imwrite("global_result.jpg", global_result)
53     cv2.imwrite("local_result.jpg", local_result)
54
55

```

packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27) [64 v.10.9 64 bit (AMD64)] Type "copyright", "credits" or "license" for more information.

IPython 8.27.0 -- An enhanced Interactive Python.

In [1]: runfile('C://Users/HP/Desktop/thresholding.py', wdir='C://Users/HP/Desktop')

Code file:



Actual Image Used:



Global Adaptive Threshold:



Local Adaptive Threshold:



Conclusion:

- Both techniques are useful.
- Local thresholding is more robust in real-world cases.

Reference Used:

Youtube : <https://www.youtube.com/watch?v=l1dhyw-EjSw>

AI Platform: Copilot

Lecture notes