Assignment Project Report

Adaptive Thresholding: Edge detection in images

Name: Arun Govind
Course: AI and ML
(Batch 4)

- **Problem Statement**

  Perform three thresholding algorithms on an image.

- **Prerequisites**

  - <u>Software:</u>
    - Python 3 (Use anaconda as your python distributor as well)
  - <u>Tools:</u>
    - Numpy
    - OpenCV
  - <u>Dataset used:</u> A photo of Indian Institute of Technology, Guwahati

- **Method Used**

  Thresholding is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e., below the threshold or above the threshold.

  In Computer Vision, this technique of thresholding is done on grayscale images. So initially, the image has to be converted in grayscale color space.

- **Implementation:**

  Here, we will be implementing three different types of thresholding, they are:
  - Simple thresholding
  - Adaptive thresholding
  - OTSU method thresholding

  1. Code Simple thresholding

     The basic Thresholding technique is Binary Thresholding. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value.

```
image1 = cv2.imread('iitg.jpg')
```

```
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
```

```
# applying different thresholding techniques on the input image
# all pixels value above 120 will be set to 255
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)
```

```
# the window showing output images
# with the corresponding thresholding
# techniques applied to the input images
cv2.imshow('Binary Threshold', thresh1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4)
cv2.imshow('Set to 0 Inverted', thresh5)
```

```
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

2. Adaptive thresholding:

Another Thresholding technique is Adaptive Thresholding. In Simple Thresholding, a global value of threshold was used which remained constant throughout.

So, a constant threshold value won't help in the case of variable lighting conditions in different areas.

Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting.

```
# image is loaded with imread command
image1 = cv2.imread('iitg.jpg')
```

```
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
```

```
# applying different thresholding
# techniques on the input image
thresh1 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY, 199, 5)

thresh2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY, 199, 5)
```

```
# techniques applied to the input image
cv2.imshow('Adaptive Mean', thresh1)
cv2.imshow('Adaptive Gaussian', thresh2)
```

```
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

3. OTSU thresholding

```
# 3. OTSU thresholding of the image
```

```
# image is loaded with imread command
image1 = cv2.imread('iitg.jpg')
```

```
# to convert the image in grayscale
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
```

```
# thresholding
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

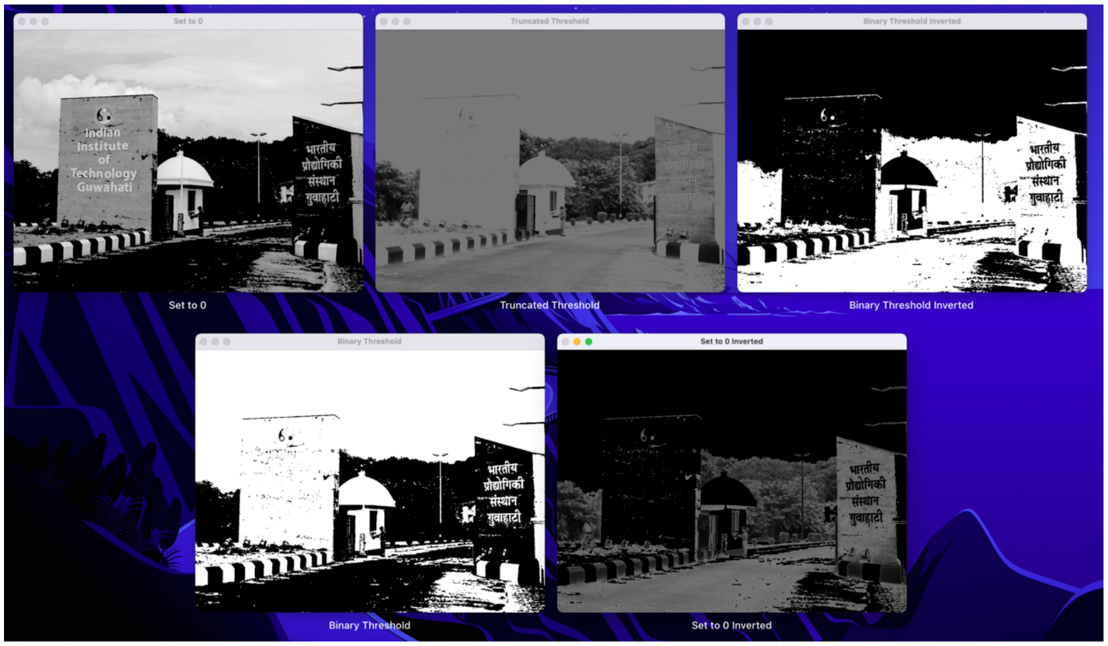
```
# techniques applied to the input image
cv2.imshow('Otsu Threshold', thresh1)
```

```
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

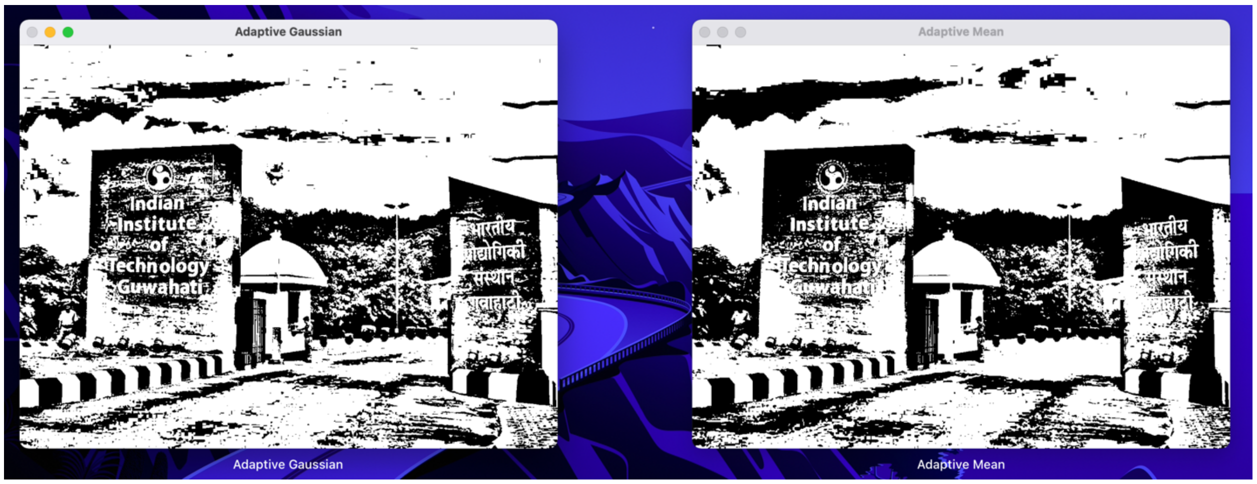
In Otsu Thresholding, a value of the threshold isn't chosen but is determined automatically. A bimodal image (two distinct image values) is considered. The histogram generated contains two peaks. So, a generic condition would be to choose a threshold value that lies in the middle of both the histogram peak values.

- **Results:**

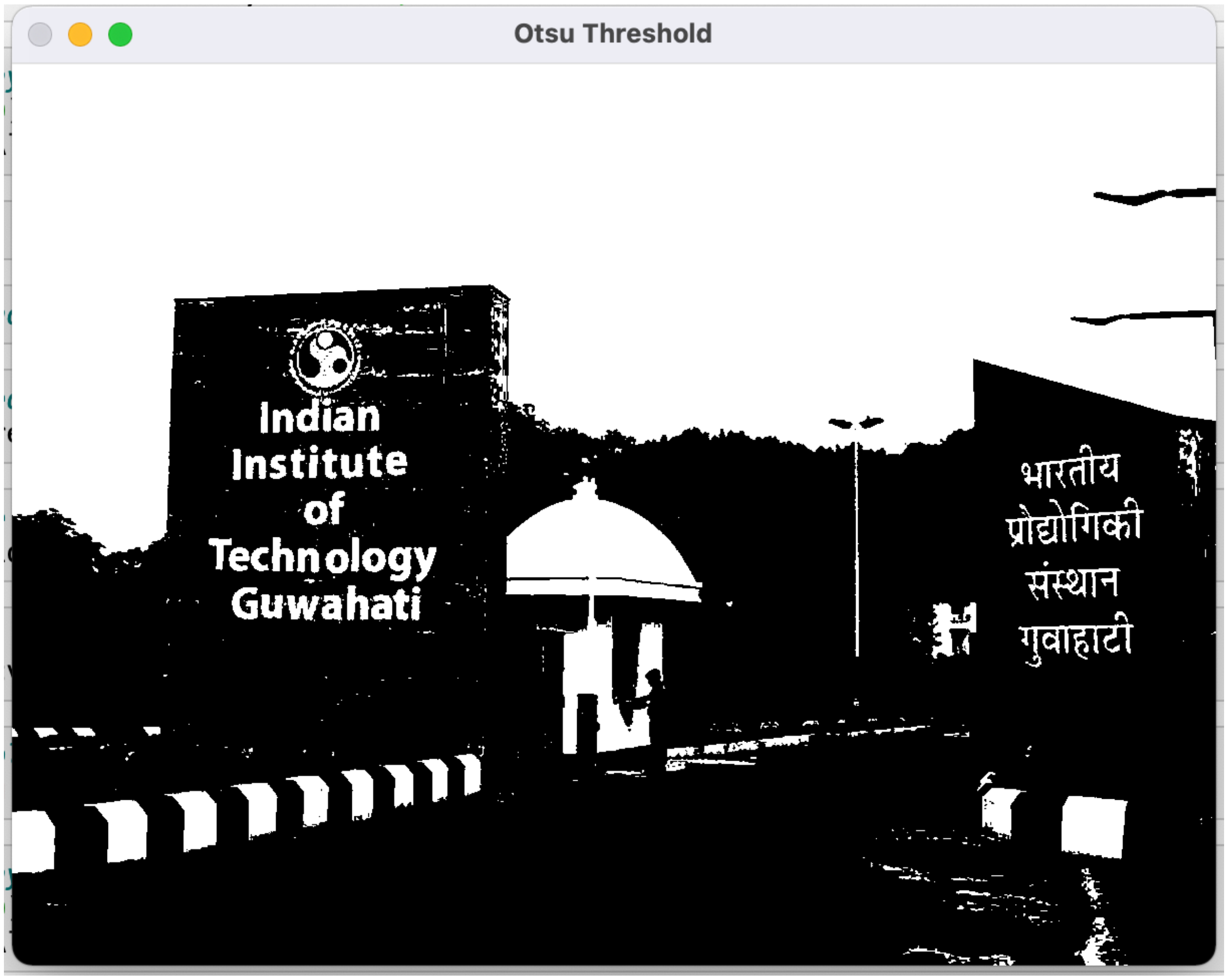
1. Simple thresholding



2. Adaptive thresholding



3. OTSU thresholding


Otsu Threshold
Indian Institute of Technology Guwahati
भारतीय प्रौद्योगिकी संस्थान गुवाहाटी