Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Student Name

***Sanatkumar Rajmogali Ippalpalli***

Title of Project Report

***Guided Project 5 – Adaptive Thresholding Edge Detection***

Table of Contents

[Introduction 4](#_Toc82509385)

[Eckovation theme & Question 6](#_Toc82509386)

[Prerequisites before starting coding 7](#_Toc82509387)

[program DEVELOPMENT steps 9](#_Toc82509388)

[Dataset/Image requirements 9](#_Toc82509389)

[Technique – ADAPTIVE THRESHOLDING & OTSU THRESHOLDING 9](#_Toc82509390)

[PROGRAM / CODE DEVELOPMENT 11](#_Toc82509391)

[Analysis 19](#_Toc82509392)

[CONCLUSION 20](#_Toc82509393)

[Figure 1 Import libraries and datasets/modules 11](#_Toc82509441)

[Figure 2 Load Datasets/image 11](#_Toc82509442)

[Figure 3 Visualization of Image 13](#_Toc82509443)

[Figure 4 Code for estimate shape of the images 14](#_Toc82509444)

[Figure 5 results for plotting images 14](#_Toc82509445)

[Figure 6 Adaptive Thresholding (image1) 14](#_Toc82509446)

[Figure 7 Adaptive Thresholding (image2) 15](#_Toc82509447)

[Figure 8 Adaptive Thresholding (image3) 16](#_Toc82509448)

[Figure 9 OTSU Thresholding (image1) 17](#_Toc82509449)

[Figure 10 OTSU Thresholding (image2) 18](#_Toc82509450)

[Figure 11 OTSU Thresholding (image3) 18](#_Toc82509451)

[Figure 12 Masking the images 19](#_Toc82509452)

***EXECUTIVE SUMMARY***

Science and technology improved many technologies and has guided numerous innovative features which advanced image processing technology.

As part of guided project, one of the feature is the edge detection technology at identifying **edges**, [curves](https://en.wikipedia.org/wiki/Curve) in a [digital image](https://en.wikipedia.org/wiki/Digital_image) at which the [image brightness](https://en.wikipedia.org/wiki/Luminous_intensity) changes sharply or, more formally, has [discontinuities](https://en.wikipedia.org/wiki/Discontinuity_(mathematics)) as the details mentioned in the question.

Edge detection is a fundamental tool in [image processing](https://en.wikipedia.org/wiki/Image_processing), [machine vision](https://en.wikipedia.org/wiki/Machine_vision) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), particularly in the areas of [feature detection](https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)) and [feature extraction](https://en.wikipedia.org/wiki/Feature_extraction)

# Introduction

Science and technology improved many technologies and has guided numerous innovative features which advanced the techniques in deep learning impacting computer vision, image processing.

Deep learning has had a tremendous impact on various fields of technology in the last few years. One of the hottest topics buzzing in this industry is computer vision, the ability for computers to understand images and videos on their own. Self-driving cars, biometrics and facial recognition all rely on computer vision to work. At the core of computer vision is image processing.

An image is represented by its dimensions (height and width) based on the number of pixels. For example, if the dimensions of an image are 500 x 400 (width x height), the total number of pixels in the image is 200000. This pixel is a point on the image that takes on a specific shade, opacity or color. It is usually represented in one of the following:

* Grayscale - A pixel is an integer with a value between 0 to 255 (0 is completely black and 255 is completely white).
* RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).
* RGBA - It is an extension of RGB with an added alpha field, which represents the opacity of the image.

Image processing requires fixed sequences of operations that are performed at each pixel of an image. The image processor performs the first sequence of operations on the image, pixel by pixel. Once this is fully done, it will begin to perform the second operation, and so on. The output value of these operations can be computed at any pixel of the image.

In blurring, we simple blur an image. An image looks more sharp or more detailed if we are able to perceive all the objects and their shapes correctly in it. For example. An image with a face, looks clear when we are able to identify eyes, ears, nose, lips, forehead e.t.c very clear. This shape of an object is due to its edges. So in blurring, we simple reduce the edge content and makes the transition form one color to the other very smooth.

Sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

Generally edges are of three types:

* Horizontal edges
* Vertical Edges
* Diagonal Edges

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Bpp or bits per pixel denotes the number of bits per pixel. The number of different colors in an image is depends on the depth of color or bits per pixel.

Hence Eckovation includes this guided project in the courseware for students to understand, implementation / execute the code themselves.

This report includes the 5W1H about the theme of development of code and running the code with database available over the internet. At the end of the report, the conclusions share the adaptive thresholding & OTSU thresholding features extracted and useful for next course of activities to gain advantages in the edge detection activities development.

# Eckovation theme & Question

**Theme : Adaptive Thresholding Edge Detection**

Edges define the boundaries between different regions in an image, which helps in matching the pattern, segment, and recognize an object. In simple thresholding, the threshold value is global, which is prone to fail in many cases. **Adaptive thresholding** is a modified method where the threshold value is calculated for each pixel based on a smaller region around it. Therefore, there will be different threshold values for different regions which gives better results for images with varying **illumination**.

**Question:**

Using OpenCV, first convert any image with varying High condition to a grayscale image. Now implement edge detection first using the canny edge detection. Then apply simple thresholding and also **Adaptive/OTSU thresholding using OpenCV** to see the working of each of these methods. Once you obtain good results, use the obtained edge detection result as a mask to give color to all the edges (if edges use the color from the original image, else leave it black only).

# Prerequisites before starting coding

1. Who - Software needed?
2. What - Version / Release of software?
3. Any Prerequisites
4. How - to install the software
5. Which -libraries are needed to execute the problem statement
6. Where – dataset requirements, path location to include in the code
7. When – to use the above feature extraction
8. Who – Software neeed?

Python

1. What- Version / Release of software?

Python version 3.6 (latest version of python)

1. Any Prerequisites

RAM space availability & hard disk space availability

Admin rights to install the software

1. How - to install the software
2. The following url <https://www.python.org/downloads/>can be referred to download python.
3. Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/>
4. Which -libraries are needed to execute the problem statement
5. Import cv2 ( pip install OpenCV)
6. Numpy (pip install numpy)
7. Matplotlib (pip install matplotlib)
8. Where – dataset requirements, path location to include in the code
9. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: [https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-external-](https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/) [command/](https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/).
10. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.
11. When – to use the above feature extraction
12. When – to use the above technique

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

# program DEVELOPMENT steps

* Dataset/Image requirement
* Technique selections
* Program / code development
* Analysis

### Dataset/Image requirements

The image sources used for this project are downloaded / collected from internet.

Purple Flower. JPG – significant flower image with color may help to find edges

Maheshbabu.JPG – Famous south Indian Tollywood actor

Allfruits.jpg – All color combinations

### Technique – ADAPTIVE THRESHOLDING & OTSU THRESHOLDING

Thresholding is the simplest way to segment objects from a background. If that background is relatively uniform, then you can use a global threshold value to binarize the image by pixel-intensity. Here, we binarize an image using the threshold\_adaptive function, which calculates thresholds in regions of size block\_size surrounding each pixel (i.e. local neighborhoods). Each threshold value is the weighted mean of the local neighborhood minus an offset value.

If the intensity of a pixel in the input image is greater than a threshold, the corresponding output pixel is marked as white (foreground), and if the input pixel intensity intensity is less than or equal to the threshold, the output pixel location is marked black (background).

In OpenCV, you can perform Adaptive threshold operation on an image using the method **adaptiveThreshold()** of the **Imgproc** class.

Syntax -

adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)

This method accepts the following parameters −

* src − An object of the class Mat representing the source (input) image.
* dst − An object of the class Mat representing the destination (output) image.
* maxValue − A variable of double type representing the value that is to be given if pixel value is more than the threshold value.
* adaptiveMethod − A variable of integer the type representing the adaptive method to be used. This will be either of the following two values
  + ADAPTIVE\_THRESH\_MEAN\_C − threshold value is the mean of neighborhood area.
  + ADAPTIVE\_THRESH\_GAUSSIAN\_C − threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.
* thresholdType − A variable of integer type representing the type of threshold to be used.
* blockSize − A variable of the integer type representing size of the pixelneighborhood used to calculate the threshold value.
* C − A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean).

The Otsu’s technique named after its creator Nobuyuki Otsu is a good example of auto thresholding. A problem with simple thresholding is that you have to manually specify the threshold value. We can manually check how good a threshold is by trying different values but it is tedious and it may break down in the real world.

Let us hop to the inscribing carving!

### PROGRAM / CODE DEVELOPMENT

As explained step by step during the lecture by mentor, we would approach steps and understand the basics with brief explanation as needed.

#### Step 1: Import the relevant libraries and applicable datasets/modules

Graphical user interface, text, application, email

Description automatically generated

Figure 1 Import libraries and datasets/modules

#### Step 2: Load dataset/Image and convert Numpy array

Download the image over internet and copy into the disk.

Text

Description automatically generated

Figure 2 Load Datasets/image

Visualize Images

Find the image into the Numpy arrays to learn the shape of the pictures. Using the NumPy shape attribute that returns a tuple with each index having the number of corresponding elements.

Graphical user interface, text, application, email

Description automatically generated

A close-up of a fetus

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with medium confidence

A picture containing text, indoor, vegetable

Description automatically generated

Figure 3 Visualization of Image

#### Step 3: Estimate shape of the images

Graphical user interface, text, application

Description automatically generated

Figure 4 Code for estimate shape of the images

Figure 5 results for plotting images

#### Step 4: Adaptive Thresholding

Apply the adaptive thresholding technique and regenerate the images

Graphical user interface, text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Figure 6 Adaptive Thresholding (image1)

Text

Description automatically generated

A picture containing text

Description automatically generated

Figure Adaptive Thresholding (image2)

Graphical user interface, text, application

Description automatically generated

A picture containing tree

Description automatically generated

Figure Adaptive Thresholding (image3)

#### Step 5: OTSU THRESHOLDING

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

Figure 9 OTSU Thresholding (image1)

Text

Description automatically generated

A picture containing text, person, outdoor, posing

Description automatically generated

Figure OTSU Thresholding (image2)

Graphical user interface, text, application

Description automatically generated

A picture containing vegetable

Description automatically generated

Figure 11 OTSU Thresholding (image3)

Text

Description automatically generated

Figure 12 Masking the images

This is continued to achieve desired results as per requirements.

### Analysis

The adaptive thresholding is completed on three different images featuring different dimensions on the color, type, variable, pixel size, etc.

Edges were able to generate accurately using Adaptive thresholding and OTSU thresholding successfully.

This entire program runs within few seconds.

# CONCLUSION

In this guided project, we built a adaptive thresholding and OTSU technique to find the edge detection.

This is done in first attempt. Hence, the improvements in the code with time with multiple attempts may be checked and justified for the accuracy score.

This entire program runs within few seconds.

references:

1. [https://towardsdatascience.com/building-a-facial-recognition-model-using-pca-svm-algorithms-c81d870add16](https://towardsdatascience.com/building-a-facial-recognition-model-using-pca-svm-algorithms-c81d870add16 )
2. <https://learnopencv.com/otsu-thresholding-with-opencv/>
3. <https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm>
4. <https://www.tutorialspoint.com/opencv/opencv_colored_images_to_grayscale.htm>
5. <https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-2-adaptive-thresholding/>