/

Khulna University Of Engineering & Technology

KUET

SESSIONAL REPORT

Course No. CSE-4128

Department Of CSE

Experiment No. 01

Name of the Experiment Image segmentation, threesholding, edge detection and its implementation.

Remarks

Name Md. Abu Saeed

Roll No 1907057

Group No. B2

Year 4th

Semester 1st

Date of Performance 22/02/24

Date of Submission 06/03/24

Objectives:

(i) To learn about image segmentation.

(ii) To learn about edge detection.

(iii) To learn about threesholding and global threesholding techniques.

(iv) To implement Canny edge detection.

Introduction: Segmentation sub-divides an image into regions and may provide level to identify individual objects. It groups the strongly corellated pixels to extract objects. It is a crucial technique in image processing, and enables further analysis, like object detection or measurement. For example: extracting cat image from an image.

In edge detection, it detects the abrupt changes in intensity. Here, the abrupt changes basically indicates the edges. Many types of edge models are available like: step, ramp, line, point etc.

edge can be detected by 1st order derivative, since it can provide high value in abrupt changes.

gradient of an image can be ~~calculated~~ represented by magnitude and direction.

(i) magnitude can be calculated as

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

and $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ are partial derivatives with respect to x, and y respectively.

(ii) gradient direction can be calculated as

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

In, x-derivative, vertical edges are highlighted and in y-derivative horizontal edges are highlighted. They can be merged to detect all edges of an image.

kernel can be used for finding edges directly. Steps are:

(i) Gaussian kernel

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)}$$

(ii) Partial derivative:

$$\frac{\partial G}{\partial x} = -\frac{x}{\sigma^2} G(x,y) \quad [\text{wrto } x]$$

$$\frac{\partial G}{\partial y} = -\frac{y}{\sigma^2} G(x,y) \quad [\text{w.r.to } y]$$

These two partial derivatives can generate two kernels, and we can use it to find x and y derivatives of an image.

Threesholding on an image is a processing technique that can convert a grayscale image in binary image.

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

Here, T = Threesholding value

Threesholding value of an image can be calculated using global threeshold determination. Steps are as follows:

a) Assume an initial threesholding value, T. Normally the average value ie used.

b) Divide the image into two segments based on T and find average on each segments μ_1, μ_2.

c) Update the old threeshold value using

$$T = \frac{\mu_1 + \mu_2}{2}$$

d) Above b-c steps is continues until difference in two successive iteration ie less than some predefined value.

Classwork pseedocode:

```
function generateGaussiankernel(sigma, MUL):
        kernel_size = find kernel size
        kernel = initialize kernel
        for each element in kernel:
                kernel[i,j] = calculate using sigma
        return the kernel

function convolve(image, kernel):
    pad_image = calculate padding and add to image
    for each pixel (x,y) in image:
        sum = 0
        for each element (kx,ky) in kernel:
            sum += element * pixel after calculation
      pad image(x,y) <- sum
    return pad_Image
```

```
function get-kernel():
    sigma ← 0.7
    kernel = generateGaussiankernel(sigma, 7)
    h = length(kernel), cx = h//2
    kernel_x, kernel_y = zero matrices of hxh
    for each x,y in range(h):
        act_x, act_y = x-cx, y-cx
        c1, c2 = -actx/sigma^2, -act_y/sigma^2
        kernel_x[x,y], kernel_y[x,y] = c1 * kernel[x,y], c2 * kernel[x,y]
    return kernel_y, kernel_x

function merge(image_1, image_2):
    height, width from image_1,
    out = zeros of shape Image_1
    for each pixel (x,y) in image_1:
        out[x,y] = sqrt(image_1[x,y]^2 + image_2[x,y]^2)

function find_next_threeshold(image, t):
    initialize total1, total2, c1, c2 = 0
    for each pixel (x,y) in Image:
        if pixel > t, add to total2 and increment c2
        else add to total1 and increment c1
    compute average for mu1, mu2
    return average mu1, mu2

function find_threeshold(image):
    compute initial t as mean pixel value of image
    loop until abs(new and old) < 0.000001:
        update t with find_next_threeshold(image, t)
    return t
```

function make_binary(t, image, low, high):
    out = copy of image
    for each pixel (x, y) in image:
        out[x, y] = high if image[x, y] > t else low
    return out

main method:
    load image from path in grayscale
    apply gaussian blur with (3x3) kernel and sigma = 0
    generate kernel_x, kernel_y using get_kernel
    convolve image with kernels to get conv_x and conv_y

    generate gaussian kernel with sigma = 0.7
    convolve image to smooth them

    merge conv_x, conv_y into out
    normalize out into out_nor
    Display merged images.

    find optimal threeshold t using find_threeshold
    apply threeshold to make binary using make_binary
    Display threesholded image

Discussion: Image segmentation, edge detection and threesholding are important process in image processing that allows partioning an image into multiple segment. These techniques have wide application in computer vision, medical imaging, recognition system etc. It helps to visualize data for different operation.

Conclusion: Segmentation, edge detection and threesholding are fundamental components of computer vision. segmentation helps to extracts object from ~~dat~~ image. edge detection helps to detect boundary which is important for segmentation and further analysis. Threesholding provides a quicker way to distinguish objects. Together these methods can help to perform different image processing operation.

Reference:

(i) Documents from LAB (Lab2_B2.ppt).

(ii) https://encord.com/blog/image-threesholding-image-processing