Computer Vision

Home Work 7

Name – ROHIT DAS Student ID – NTNU_61047086s

Project – Thinning of Image.

Language and library used: Python, Pillow, Numpy.

Description: This program will perform the following functions while executing lena.bmp image file:

1. Convert image to Binary
2. Find Yokoi Connectivity Number
3. Find interior image
4. Dilate interior image for generating marked image
5. Applying thinning function by deleting marked image and yokoi connectivity number and put the values in new image until the image becomes idempotent.

Parameters: None. Please Copy-paste the image path inside the program.

Algorithms Used –

Part 1: Convert Image to Binary

Principal Code:

```
def img_binarize(img_in):
    image_binary = np.copy(img_in)
    for h_row in range(0, row):
        for w_col in range(0, col):
            if img_in[h_row,w_col] > 127:
                image_binary[h_row, w_col] = 255
            else:
                image_binary[h_row, w_col] = 0
    return (image_binary)
```

Part 2: Finding Yokoi Connectivity Number using h function , w function, get neighbours and get value functions

Principal Code:

```
############################ Part 3 ############################
############################ Setup the Yokoi Connectivity Number
##################


def yokoi_connectivity_number(downsampled_image):
# Create a blank canvas of size 64*64 for sketching the Yokoi Number
    YokoiConnectivityNumber = np.full(downsampled_image.size, ' ')
    # Scan each column in original image.
    for h_row in range(downsampled_image.size[0]):
        # Scan each row in original image.
```

```
        for w_col in range(downsampled_image.size[1]):
            if (downsampled_image.getpixel((h_row,w_col)) != 0):
                # Get neighborhood pixel values.
                neighborhoodPixels =
getNeighborhoodPixels(downsampled_image, h_row,w_col)
                # Calculating the pattern of relation between the
neighboring pixel
                YokoiConnectivityNumber[h_row,w_col] =
four_connectivity_fFunction(
                    four_connectivity_hFunction(neighborhoodPixels[0],
neighborhoodPixels[1],
                                                neighborhoodPixels[6],
neighborhoodPixels[2]),
                    four_connectivity_hFunction(neighborhoodPixels[0],
neighborhoodPixels[2],
                                                neighborhoodPixels[7],
neighborhoodPixels[3]),
                    four_connectivity_hFunction(neighborhoodPixels[0],
neighborhoodPixels[3],
                                                neighborhoodPixels[8],
neighborhoodPixels[4]),
                    four_connectivity_hFunction(neighborhoodPixels[0],
neighborhoodPixels[4],
                                                neighborhoodPixels[5],
neighborhoodPixels[1]))
            # This point is background.
            else:
                YokoiConnectivityNumber[h_row,w_col] = ' '

    # Return Yokoi Connectivity Number.
    return YokoiConnectivityNumber


def getNeighborhoodPixels(downImage, h_row,w_col):
    """
    Corners neighborhood (for corresponding ith values in x)
    x7,x2,x6
    x3,x0,x1
    x8,x4,x5
    """
    return [getValue(downImage, h_row, w_col), getValue(downImage, h_row +
1, w_col), getValue(downImage, h_row, w_col - 1),
            getValue(downImage, h_row - 1, w_col), getValue(downImage,
h_row, w_col + 1), getValue(downImage, h_row + 1, w_col + 1),
            getValue(downImage, h_row + 1, w_col - 1), getValue(downImage,
h_row - 1, w_col - 1), getValue(downImage, h_row - 1, w_col + 1)]


def getValue(img, row, col):
    # Helper function to get the value separately for better code
readability
    if row >= img.size[0] or row < 0 or col >= img.size[1] or col < 0:
        return 0
    return img.getpixel((row,col))
```

Part 3: Find Interior Image

Principal Code –

```
def getInteriorImage(YokoiConnectivityNumber):
    # New image with the same size as Yokoi connectivity number and
'binary' format.
    interiorImage = Image.new('1', YokoiConnectivityNumber.shape)
    # Scan each column in interior image.
    for h_row in range(interiorImage.size[0]):
        # Scan each row in interior image.
        for w_col in range(interiorImage.size[1]):
            # If this pixel is interior(label = 5).
            if (YokoiConnectivityNumber[h_row, w_col] == '5'):
                # Put white pixel to interior image.
                interiorImage.putpixel((h_row, w_col), 1)
            # If this pixel is not interior.
            else:
                # Put black pixel to interior image.
                interiorImage.putpixel((h_row, w_col), 0)
    # Return interior image.
    return interiorImage
```

Part 4: Binary Dilation for generating marked Image

Principal Code-

```
def binary_Dilation(image,Kernel):
    # Copy the Image
    dilated_Image = np.zeros(image.shape, dtype=np.uint8)
    # Centre value of the Kernel
    kernel_centre_point = tuple(x//2 for x in Kernel.shape)
    # Iterating over each pixel in original image
    for h_row in range(0, row):
        for w_col in range(0, col):
            # If the pixel is white
            if image[h_row,w_col] != 0:
                # Iterating over Kernel Shape
                for K_row in range(0,Kernel.shape[0]):
                    for K_col in range(0,Kernel.shape[1]):
                        # Get kernel value 1 from each iteration
                        if Kernel[K_row,K_col] == 1:
                            # Get the destination pixel location to be
filled out
                            n_row_val = h_row+(K_row-
kernel_centre_point[0])
                            n_col_val = w_col+(K_col-
kernel_centre_point[1])
                            # Avoiding out of range and putting the value
255 to destination pixel
                            if n_row_val < image.shape[0] and n_col_val <
image.shape[1]:
                                dilated_Image[n_row_val,n_col_val] = 255

    return dilated_Image
```

Part 5: Applying thinning function by deleting marked image and yokoi connectivity number and put the values in new image until the image becomes idempotent.

Principal Code-

```
########################## Part 6 ##########################
########################## Applying thinning function by deleting marked
image
########################## and yokoi connectivity number and put the
values in new image until the image
########################## becomes idempotent #################

def getThinningImage(originalImage, YokoiConnectivityNumber, markedImage):
    thinningImage = Image.new('1', originalImage.size)
    # Scan each column in thinning image.
    for h_row in range(thinningImage.size[0]):
        # Scan each row in thinning image.
        for w_col in range(thinningImage.size[1]):
            # If this point is removable(label = 1) and is in marked image.
            if (YokoiConnectivityNumber[h_row, w_col] == '1' and
markedImage.getpixel((h_row, w_col)) != 0):
                # Remove this pixel from original image.
                thinningImage.putpixel((h_row, w_col), 0)
            else :
                thinningImage.putpixel((h_row, w_col),
originalImage.getpixel((h_row, w_col)))
    # Return thinning image.
    return thinningImage


########################## Part 7 ##########################
########################## Check if the image produced from repeated
iterations is idempotent #################


def isEqualImage(image1, image2):
    from PIL import ImageChops
    return ImageChops.difference(image1, image2).getbbox() is None
```

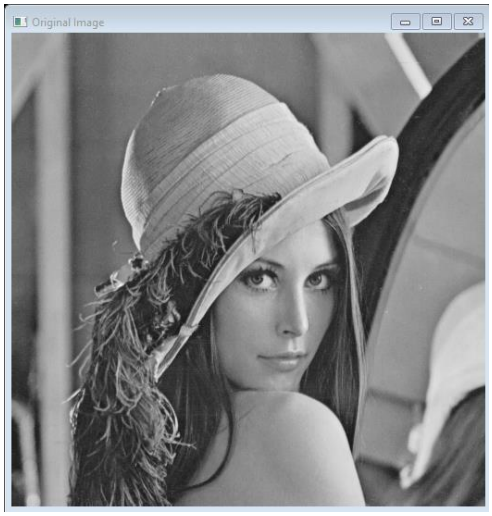
Part 6 : Main Function

```
if __name__ == "__main__":
    # Converted the image to Binary in OpenCV for better readability of
Code
    bin_image = img_binarize(gray_image)
    thinningImage = Image.fromarray(bin_image)
    while True:
        # Get Yokoi Connectivity Number.
        YokoiConnectivityNumber = yokoi_connectivity_number(thinningImage)
        # Get interior image from Yokoi Connectivity Number.
        interiorImage = getInteriorImage(YokoiConnectivityNumber)
        # Get marked image by dilation of interior image..
        markedImage_cv = binary_Dilation(np.array(interiorImage), kernel)
        markedImage_pil = Image.fromarray(markedImage_cv)
        # Get thinning image.
        tempImage = getThinningImage(thinningImage,
YokoiConnectivityNumber, markedImage_pil)
        # If this iteration reaches idempotence.
        if (isEqualImage(tempImage, thinningImage)):
            break
```

```
        # Update thinning image.
        thinningImage = tempImage
    thinningImage.show()
```

Example:

- Original image



- Image after applying thinning