Computer Vision HW6 Report
蕭恩慈 / B07902095

The task is to write a program which counts the Yokoi connectivity number on the downsampled data of the original image(lena.bmp). The language that is used to finish the task is Python language, with PIL and numpy library.
The way to do this task is to first make a binary version of the original image (binary-lena.bmp), then using 8x8 blocks as a unit, take the topmost-left pixel as the downsampled data.
To create the binary image, open the original image in the binary format, set the threshold to 128.
Then, get the downsampling data by calculating the width and height of the original image by 8 times (since we want to use 8x8 blocks), then "replace" each original image pixel with the already calculated downsample image.
Now, we want to get the neighborhood pixel. First, allocate memory space of the neighborhood pixels, then set the position to x and y. Then, scan the distance of x and y from -1 to 1 to calculate the destination of x and y, and set out the out of range pixel. The way we do this is by locating the destination x and y, if it is within the size, then get the pixel of x and y in the original image. Otherwise, if it is not within the range, set x and y to 0.
Downsampling and getting neighborhood pixels code snippet:

```
#downsampling
def downsampling(ori):
    w = int(ori.size[0] / 8)
    h = int(ori.size[1] / 8)
    downsampled = Image.new("1", (w, h))
    for c in range(downsampled.size[0]):
        for r in range(downsampled.size[1]):
            ori_px = ori.getpixel((c * 8, r * 8))
            downsampled.putpixel((c, r), ori_px)mo

    return downsampled

#get neighborhood px
def neighbor(ori, position):
    neighborPixel = np.zeros(9)
    x, y = position
    for dx in range(3):
        for dy in range(3):
            target_x = x + (dx - 1)
            target_y = y + (dy - 1)
            if ((0 <= target_x < ori.size[0]) and (0 <= target_y < ori.size[1])):
                neighborPixel[3 * dy + dx] = ori.getpixel((target_x, target_y))
            else:
                neighborPixel[3 * dy + dx] = 0

    neighborPixel = [
        neighborPixel[4], neighborPixel[5], neighborPixel[1],
        neighborPixel[3], neighborPixel[7], neighborPixel[8],
        neighborPixel[2], neighborPixel[0], neighborPixel[6]
    ]
```

To set the neighborhood pixels, sort the original order `[[x0, x1, x2], [x3, x4, x5], [x6, x7, x8]]` to `[[x7, x2, x6], [x3, x0, x1], [x8, x4, x5]]` order.
into the order we want. For this task, we use

```
neighborPixel = [
    neighborPixel[4], neighborPixel[5], neighborPixel[1],
    neighborPixel[3], neighborPixel[7], neighborPixel[8],
    neighborPixel[2], neighborPixel[0], neighborPixel[6]
]
```

Lastly, after getting *qrs* location, implement the original image to the Yokoi connectivity number function.

Code snippet:

```
def h(b, c, d, e):
    if ((b == c) and (b != d or b != e)):
        return "q"
    if ((b == c) and (b == d and b == e)):
        return "r"
    if (b != c):
        return "s"

def f(a1, a2, a3, a4):
    if ([a1, a2, a3, a4].count("r") == 4): #if a1 = a2 = a3 = a4 = r
        return 5
    else:
        return [a1, a2, a3, a4].count("q") #else count q

def yokoi(ori):
    yokoi = np.full(downsampled.size, " ")
    for c in range(ori.size[0]):
        for r in range(ori.size[1]):
            if (ori.getpixel((c, r)) != 0):
                neighborPixel = neighbor(ori, (c, r))
                yokoi[c, r] = f(
                    h(neighborPixel[0], neighborPixel[1], neighborPixel[6], neighborPixel[2]),
                    h(neighborPixel[0], neighborPixel[2], neighborPixel[7], neighborPixel[3]),
                    h(neighborPixel[0], neighborPixel[3], neighborPixel[8], neighborPixel[4]),
                    h(neighborPixel[0], neighborPixel[4], neighborPixel[5], neighborPixel[1])
                )
            else:
                yokoi[c, r] = " "

    return yokoi
```

Driver's code:

```
if __name__ == "__main__":
    ori = Image.open("lena.bmp")

    bin = binary(ori)
    bin.save("binary-lena.bmp")

    downsampled = downsampling(bin)
    downsampled.save("downsampled-lena.bmp")

    yokoi = yokoi(downsampled)
    np.savetxt("yokoi-lena.txt", yokoi.T, fmt = "%s", delimiter = "")
```

The result will be a **text file** containing Yokoi connectivity numbers of the original (lena.bmp) image.