13331314 叶嘉祺

1 Exercises

Answer the following questions in your report.

1.1 Storage (3*3=9Points)

If we consider an N -bit gray image as being composed of N 1-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image and plane N all the highest-order bits, then given a 2048*2048, 256-level gray-scale image:

1.How many bit planes are there for this image?

```
256 = 2^8, there are 8 bit planes for this image.
```

2.Which panel is the most visually significant one?

```
The plane N is the most visually significant one because it carry the
highest-order bits which make the largest effect to the gray value in the
image.
```

3.How many bytes are required for storing this image? (Don’t consider image headers and compression.

```
2048 * 2048 * 8 = 2^25 (bits) = 2^22 (bytes)
```

1.2 Adjacency (3*3=9Points)

Figure 1 is a 5*5 image. Let V = {1,2,3} be the set of pixels used to define adjacency. Please report the lengths of the shortest 4-, 8-, and m- path between p and q . If a particular path does not exist, explain why?

```
    3 4 1 2 0
    0 1 0 4 2 (q)
    2 2 3 1 4
(p) 2 0 4 2 1
    1 2 0 3 4
```

1. 4-path:

   It's unable to find a 4-path from p-q since the four positions whose values are not in V block each of the posible 4-path from p-q:

   ```
       3 x 1 2 0
       0 1 x x 2 (q)
       2 2 3 1 x
   (p) 2 0 4 2 1
       1 2 0 3 4
   ```

2. 8-path:

   The shortest is shown below, the length of the path is 4.

   p (3,0):2 -> (2,1):2 -> (2,2):3 -> (2,3):1 -> (1, 4):2 q

3. m-path

   The shortest m-path is shown below, the length of the path is 5.
   p (3,0):2 -> (2,0):2 -> (2,1):2 -> (2,2):3 -> (2,3):1 -> (1,4):2 q

1.3 Logical Operations (3*3=9Points)

Figure 2 are three different results of applying logical operations on sets A, B and C. For each of the result, please write down one logical expression for generating the shaded area. That is, you need to write down three expressions in total.

1. A∩B∩C
2. (A∩B)U(A∩C)U(B∩C)
3. ((A∩C) - (A∩B∩C)) U (B∩(AUC^c))

2 Programming Tasks

Scaling (45 Points)

1. Down-scale to 192 × 128 (width: 192, height: 128), 96 × 64, 48 × 32, 24 × 16 and 12 × 8, then manually plot your results in the report. (10 Points)

192*128

96*64

48*32

24*16

16*8

2. Down-scale to 300 × 200, then plot your result. (5 Points)

3. Up-scale to 450 × 300, then plot your result. (5 Points)

4. Scale to 500 × 200, then plot your result. (5 Points)

5. Detailedly discuss how you implement the scaling operation, i.e., the "scale" function, in less than 2 pages. Please focus on the algorithm part. If you have found interesting phenomenons in your scaling results, analyses and discussions on them are strongly welcomed and may bring you bonuses. But please don't widely copy/paste your codes in the report, since your codes are also submitted. (20 Points)

First, we create an image object which has the desired input size. The calculat the Scaling Rate(size_new/size_origin) for both height(`m`) and width('n'). Then we will have a nested 'for' loop to calculate the gray value for each pixel for the new image.
For down scaling, Isometric sampling and Mean sampling will be applied (respectively) to the image.
For Isometric sampling, there is a corresponding relationship, and we can easily calculate the gray value of the pixel.

```
x = x = int(i/m)
y = int(j/n)
new_image[i, j] = old_img[x,y]
```

For mean sampling,we will split the image into serveral small pieces which equals to the new down scaling image and then we will calculate a area of pixels and then sum up then, and then calculate the

average of the new pixel.

```
(x1,y1)...(x1,yn)
.              .
.              .
(xn,y1)...(xn,yn)

average = sum(xi,yi) / n^2
```

Well, it's very surprised to find that mean scaling is not always better than the Isometric sampling.

For up scaling , I use the Isometric sampling, for example, if we scale width = 2*width, the matrix of the image will become:

```
11 12 13     11 11 12 12 13 13
14 15 16     14 14 15 15 16 16
17 18 19     17 17 18 18 19 19
```

2.3 Quantization (28 Points)

1. Reduce gray level resolution to 128, 32, 8, 4 and 2 levels, then plot your results respectively. Note that, in practice computers always represent "white" via the pixel value of 255, so you should also follow this rule. For example, when the gray level resolution is reduced to 4 levels, the resulting image should contain pixels of 0, 85, 170, 255, instead of 0, 1, 2, 3. (8 Points)

128

32

8

4

2. Detailedly discuss how you implement the quantization operation, i.e., the "quantize"function, in less than 2 pages. Again, please focus on the algorithm part. Analyzing

and discussing interesting experiment results are also welcomed, but please don't widely copy/paste your codes in the report (20 Points).

For quantization, we similarily create a new image firstly. Then, we will calculat two values level and segment. For a given gray level for example 4. It will divided the rgb set into 4 segemnts (0~64, 64~128, 128~172, 172~256), numbering with 0, 1, 2, 3 respctively. The variable segement represent the number. Then we will have 4 gray values for 4-l level(0, 85, 170, 255). Then for each numbered position, there is a solution for the new image:

```
segement = 256 / level
level = 255.0 / float(level - 1)

new_image[i,j]
= int(old_image[i.j] / segement) * level
```

Then we can calculate all the gray values for other pixels.