# **Digital Image Processing**

### 13331233 孙中阳

### 1.1

#### 1

128 = 2 ^ 7 8 bits

#### 2

plane 7, the highest-order bits

#### 3

bits = M \* N \* K = 1024 \* 2048 \* 7 = 14680064 bytes = 16777216 / 8 = 1835008 1835008 bytes

#### 1.2

没有4联通路径 8联通路径的最短距离为4 m联通的最短路径为5

### 1.3

A  $\cap$  B  $\cap$  C (B  $\cap$  C)  $\cup$  (A  $\cap$  B)  $\cup$  (A  $\cap$  C) B  $\cup$  (A  $\cap$  C) - (A  $\cap$  B) - (B  $\cap$  C)

# 2.2

### 我选择33.png

original 384 \* 256



### 1

192 \* 128



96 \* 64



48 \* 32



24 \* 16

12 \* 8

ad

2

300 \* 200



3

450 \* 300



500 \* 200



### 5

代码已有注释表明细节,这里主要讨论算法框架 本题涉及到的算法主要为双线性插值,具体逻辑如下:

- 1.求出缩放系数
- 2.对每个像素,根据缩放系数,找到原图中相对位置附近的四个像素
- 3.根据四个像素和相对位置的距离确定其系数(在新像素RGBA值中的比例)
- 4.计算得到新像素的RGBA值

由于JAVA使用的BufferedImage不能直接操纵灰度值,只能分别操纵RGBA值(存储在一个4byte的int类型中,最高到最低位的4个byte分别表示A,R,G,B),故在使用双线性差值前经过了将原图转化为三维数组的过程,具体为 img[row][col][4]。四个元素分别为RGBA,分别操作RGBA才能得到新的像素的RGBA。最终得到四个RGBA值后再将其还原为一维数组,输出图像

# 2.3

#### original

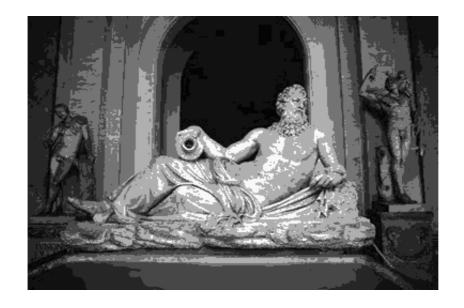


1

128











算法逻辑比较简单,有之前操作RGBA的经验后直接将RGBA范围映射到需要的值即可,主要逻辑为

- 1.根据新的灰度级别求出其相对256的间距,比如将256转换为128时间距为2
- 2.对每个像素,根据其现有灰度值计算新的灰度值

直接处理RGBA可能产生意想不到的效果,故在本题中首先采用一个心理公式 r \* 0.299 + g \* 0.587 + b \* 0.114 将RGBA值转换为一个灰度值,然后直接处理这一值即可