Homework1

杨加佳　　15331354

1.1.1

∵ 128 = 2^7

∴ 这个图片共有 7 个位平面

1.1.2

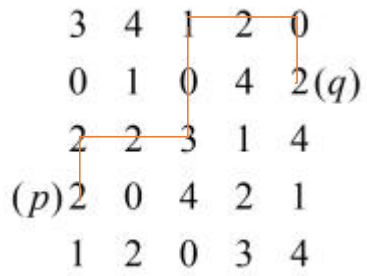
第 N 层是视觉上最重要的

1.1.3

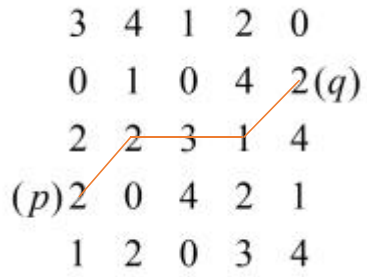
（1024×2048）* 7 / 8 = 1 835 008

1.2

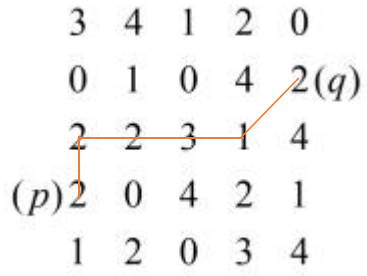
最短 4 邻接路径



最短 8 邻接路径



最短 m 邻接路径



1.3

1) $A \cap B \cap C$
2) $(A \cap B) \cup (A \cap C) \cup (B \cap C)$
3) $(A \cap C) \cup B$-$(\mathrm{A} \cap \mathrm{B}) \cup (\mathrm{B} \cap \mathrm{C})$

2.2.1

原图

192×128

96×64

48×32

24×16

12×8

2.2.2

300×200

2.2.3

450×300

2.2.4

500×200

2.2.5

关于双线性插值的算法，虽然能够理解其基本的思想和原理，但仅根据书上的公式的话，刚开始并没有想出来该如何实现这种算法，于是上网查找了相关资料，也查到了双线性插值算法的化简后的版本：

双线性插值，又称为双线性内插。在数学上，双线性插值是有两个变量的插值函数的线性插值扩展，其核心思想是在两个方向分别进行一次线性插值。

假如我们想得到未知函数 f 在点 $P = (x, y)$ 的值，假设我们已知函数 f 在 $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, 及 $Q_{22} = (x_2, y_2)$ 四个点的值。

首先在 x 方向进行线性插值，得到

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{Where} \quad R_1 = (x, y_1),$$
$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{Where} \quad R_2 = (x, y_2).$$

然后在 y 方向进行线性插值，得到

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

这样就得到所要的结果 $f(x, y)$,

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1).$$

如果选择一个坐标系统使得 f 的四个已知点坐标分别为 (0, 0)、(0, 1)、(1, 0) 和 (1, 1)，那么插值公式就可以化简为

$$f(x,y) \approx f(0,0)\,(1-x)(1-y) + f(1,0)\,x(1-y) + f(0,1)\,(1-x)y + f(1,1)xy.$$

于是，先算出目标图片大小与原图的比率，然后根据这个比率，对目标图片的每个像素，算出对应的原图片的像素坐标，这个坐标算出来一般是小数。对算出来的这个坐标进行向下取整的操作，然后将这个操作抹掉的小数部分的值当作权值，套入上面的公式当中，便实现了双线性插值算法。

因为之前实训学习与实现过对 BMP 格式的图片的处理，刚开始也打算对图片格式进行转换，按实训所学对图片进行操作，但操作起来确实太繁琐了。后来听取了一位师兄的建议，重新又安了 Matlab 上网自学了一下一些基础，便开始边学边完成作业。当中也遇到了许多诸如 Matlab 中图片描述长宽的顺序与我们习惯的相反以及下标从 1 开始这样的坑，但也都磕磕绊绊地解决了吧。后来也对比了其他同学用其他语言实现的算法，发现用 Matlab 的确简便了很多，今后也会多花些心思研究一下 Matlab 的。

2.3.1

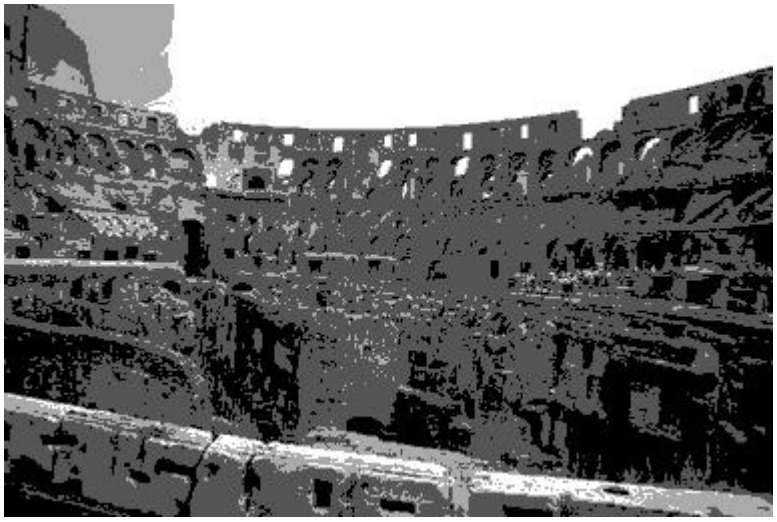
原图

128 灰度级

32 灰度级

8 灰度级

4 灰度级



2 灰度级

2.3.2

关于降低灰度级的操作，我的理解是，这相当于重新划分 0-255 这个大区间，至于如何决定像素点该落在哪个区间，我的算法是先算出每个对应灰度级区间的长度，然后用某个像素点的灰度值除以这个长度，对得到的结果进行四舍五入的操作，再乘以这个长度，就能得到某个像素点降低灰度级后的值。

在刚开始的时候，我对题目要求的出现了错误，我刚开始的理解是 256 灰度级相当于 8 个层叠加，降低灰度级就是抽出其中的几层，过程中还找了许多根据位运算完成这种操作的方法，但这种方法，在 128 和 64 灰度级还没什么大问题，灰度级很低之后，就会出现很严重的失真现象，我才意识到这种方法是错误的。与同学讨论之后，重新弄懂了题目的意思，才有了上面所述的新的想法，实现出来的效果也跟大家差不多。