数字图像处理第七次作业

南京大学电子科学与工程学院
李超凡 091180066

2012-4-16

1 图像相对信息冗余的计算

图像的相对信息冗余 R 表示一个图像经过压缩后数据体积减少的百分比，用公式来表示即：

$$R = 1 - \frac{1}{C}$$

其中的 C 代表数据压缩比，显然从这个公式可以看出相对冗余信息是针对具体的压缩方法而言的。

根据我的 PC 上自带的压缩方法，Lenna 图像的体积为 257.1 KB，最好的压缩比为 1.4379:1，采用的是 bz2 算法。故而对这一压缩算法来说，相对信息冗余为 0.3045。

2 游程编码的实现

游程编码的原理非常简单，简单地说就是用相同的数据及其长度来表示一串数据，这种编码适合用来去除图像的空间冗余，也就是空间中一连串的相同像素点。尤其适合处理黑白图像，传真即使用了这种编码。

我使用 C 语言实现了一个简单的可读的针对二值图像的游程编码工具。它可以将 8 位 *.bmp 图像编码为一个文本文件，文本文件中存放为数据长度串，用换行区别不同的行。代码如下：

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>

typedef struct bmpfile_header {
    uint32_t filesz;
    uint16_t creator1;
    uint16_t creator2;
    uint32_t bmp_offset;
} FILEH;
typedef struct {
    uint32_t header_sz;
    int32_t width;
    int32_t height;
    uint16_t nplanes;
    uint16_t bitspp;
    uint32_t compress_type;
    uint32_t bmp_bytesz;
    int32_t hres;
    int32_t vres;
    uint32_t ncolors;
    uint32_t nimpcolors;
} INFOH;

int main(int argc , char * argv[]){
    INFOH ih ;
    FILEH fh ;
    FILE* pic ;
    FILE* output ;
    char* o_str ;
    int i ,j , color, counts, color_tmp;
    pic = fopen(argv[1], "r") ;
```

```
    fseek(pic, 0x2, SEEK_SET);
    fread(&fh.filesz, 4, 1, pic);
    fprintf(stderr,"fh.filesz:0x%X\n",fh.filesz);
    fseek(pic, 0xa, SEEK_SET);
    fread(&fh.bmp_offset, 4, 1, pic) ;
    fprintf(stderr,"fh.bmp_offset:0x%X\n",fh.bmp_offset);
    fseek(pic, 0x1c, SEEK_SET);
    fread(&ih.bitspp, 2, 1, pic);
    fprintf(stderr,"ih.bitspp:0x%X\n",ih.bitspp);
    fseek(pic, 0x12, SEEK_SET);
    fread(&ih.width, 4, 1, pic);
    fprintf(stderr,"ih.width:%d\n",ih.width);
    fseek(pic, 0x16, SEEK_SET);
    fread(&ih.height, 4, 1, pic);
    fprintf(stderr,"ih.height:%d\n",ih.height);
    fseek(pic, 0x1e, SEEK_SET);
    fread(&ih.compress_type, 4, 1, pic);
    fprintf(stderr,"ih.compresstype:0x%X\n",ih.compress_type);
    o_str = malloc(strlen(argv[1] + 5));
    strcpy(o_str, argv[1]);
    strcat(o_str, ".rle");
    output = fopen(o_str, "w");
    free(o_str);


    fseek(pic, fh.bmp_offset ,SEEK_SET);
    for(i = 1 ; i < ih.height + 1; i++){
//      fprintf(output, "%d:", i);
        counts = 0 ;
        for(j = 1 ; j < ih.width + 1; j ++){
            fread(&color,1,1,pic);
            if(color == color_tmp ){
                counts ++ ;
                if(j == ih.width ){
                    fprintf(output, "(%d,%d)", color, counts);
                }
            }else{
                if(counts){
                    fprintf(output, "(%d,%d)",color_tmp, counts);
                }
                counts = 1 ;
            }
            color_tmp = color ;
        }
        putc('\n', output);
    }
}
```

代码 1. rle.c

试验图像为图 1 中的图像。这个图像的体积为 136918 B，经过压缩得到的编码的第 26 行（对应图像的底部第 10 行像素点）如代码 2 所示，所得文件的总大小为 34616 B，压缩比为 $136918 : 34616 \approx 3.96$。

```
(0,382)(255,2)(0,95)
```

代码 2. 游程编码后的文件

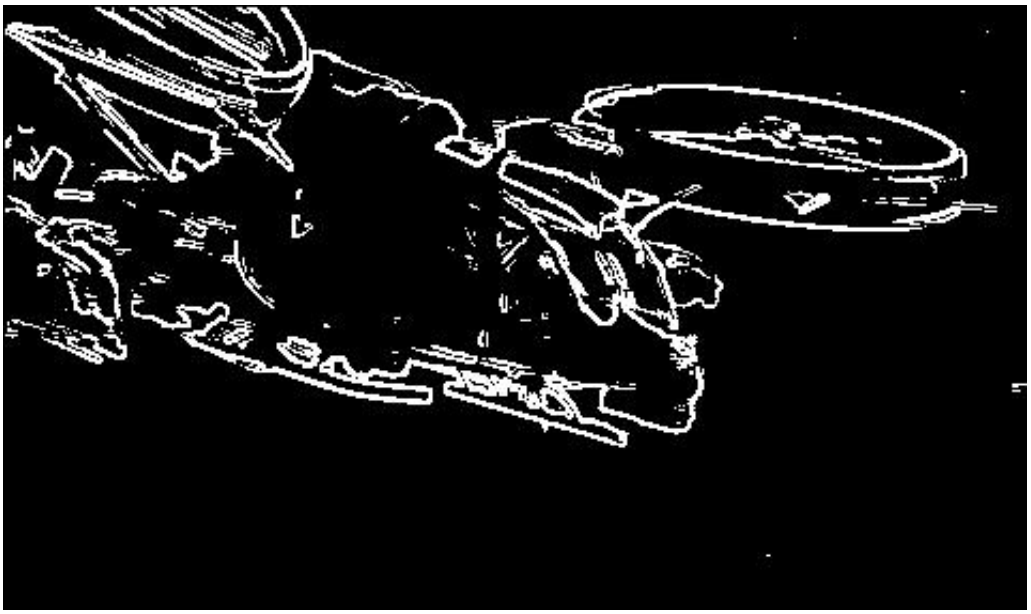

图 1. 试验图像