Perlab 程序性能优化实验

实验要求

（1）修改 kernel.c 的源代码，对 rotate 和 smooth 函数进行优化。
（2）查看目录下的 Makefile 文件，清楚不同的 make 指令完成的功能。用命令 make clean 清除原来的内容，用命令 make 编译生成新的可执行文件 driver。键入命令 ./driver，测试修改后的 rotate 和 smooth 两个函数的加速比和 naive_rotate 和 naive_smooth 两个函数的加速比，并分析原因。
（3）针对 rotate 和 smooth，每个写出至少三种不同的优化版本。

实验目的

了解程序的时间局部性和空间局部性，并能根据这种性质对实验代码进行优化，获得更好的加速比。

时间安排

时间	内容
第十周第一次课	完成 rotate 第一个优化函数
第十周第二次课	完成 rotate 第二个优化函数
第十一周第一次课	完成 rotate 第三个优化函数
第十一周第二次课	完成 smooth 第一个优化函数
第十二周第一次课	完成 smooth 第二个优化函数
第十二周第二次课	完成 smooth 第三个优化函数

实验内容

1.优化 rotate 函数，rotate 函数是将图像实现逆时针旋转 90°，如下图所示：

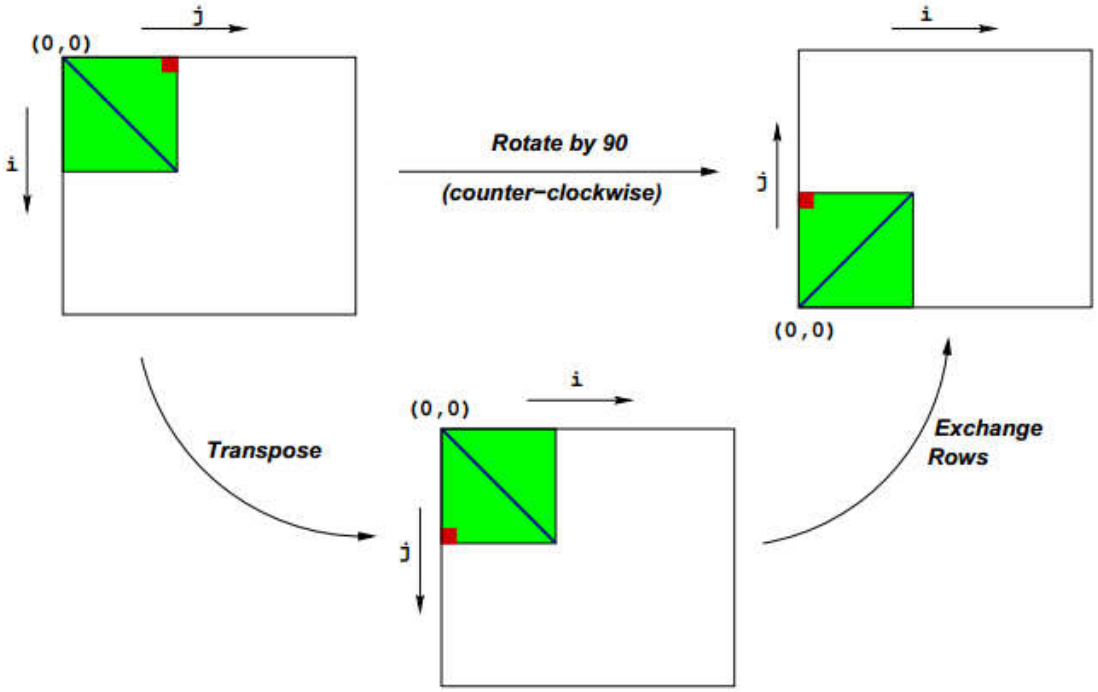

Figure 1: Rotation of an image by 90° counterclockwise

首先在 kernel.c 中找到 navie rotate 函数。

```
char naive_rotate_descr[] = "naive_rotate: Naive baseline implementation
void naive_rotate(int dim, pixel *src, pixel *dst)
{
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++)
            dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];
}
```

根据 dim-1-j 重复计算来进行优化。修改 rotate 函数

```
char rotate_descr[] = "rotate: Current working version";
void rotate(int dim, pixel *src, pixel *dst)
{
int i, j,tmp;

    for (j = 0; j < dim; j++)
    {
        tmp=dim-1-j;
        for (i = 0; i < dim; i++)
            dst[RIDX(tmp, i, dim)] = src[RIDX(i, j, dim)];
    }
}
```

键入命令./driver，通过比较

```
limeng@limeng-X550JD:~/code/csapplab/csapp实验/LAB4$ ./driver
Teamname: limeng
Member 1: Meng
Email 1: 925762221@qq.com

Rotate: Version = naive_rotate: Naive baseline implementation:
Dim             64      128     256     512     1024    Mean
Your CPEs       2.3     3.3     5.3     8.7     11.3
Baseline CPEs   14.7    40.1    46.4    65.9    94.5
Speedup         6.3     12.1    8.8     7.6     8.3     8.4

Rotate: Version = rotate: Current working version:
Dim             64      128     256     512     1024    Mean
Your CPEs       2.0     2.2     2.9     4.3     7.5
Baseline CPEs   14.7    40.1    46.4    65.9    94.5
Speedup         7.2     17.9    16.1    15.2    12.6    13.2

Smooth: Version = smooth: Current working version:
Dim             32      64      128     256     512     Mean
Your CPEs       50.2    50.5    50.6    50.3    51.0
Baseline CPEs   695.0   698.0   702.0   717.0   722.0
Speedup         13.9    13.8    13.9    14.3    14.2    14.0

Smooth: Version = naive_smooth: Naive baseline implementation:
Dim             32      64      128     256     512     Mean
Your CPEs       50.0    50.5    50.5    50.6    51.4
Baseline CPEs   695.0   698.0   702.0   717.0   722.0
Speedup         13.9    13.8    13.9    14.2    14.1    14.0
```

比较 native_rotate 和 rotate 两个函数的加速比(speedup)和 CPE,改进你的 rotate 版本，继续优化。

2. 优化 smooth 函数，smooth 函数用来实现类似于求图片中每个元素所在的九宫格的平均值，然后将该值放到该元素中，编写 smooth 优化函数，达到最佳的加速比。

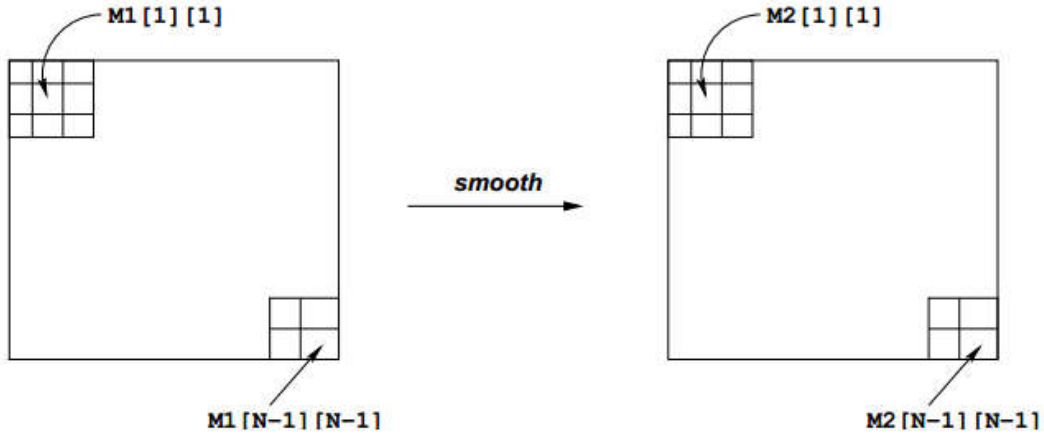

Figure 2: Smoothing an image

注意事项

（1）强烈建议安装 32 位的 ubuntu 或者 32 位的虚拟机，如果安装的是 64 位的 ubuntu,那么执行命令 sudo apt-get install libc6-dev-i386、 sudo apt-get install gcc-multilib 安装 32 位兼容包。

（2）初始 kernels.c 程序中下图中内容为作者信息，每个学生须根据自己情况修改相关内容，否则会编译报错。

```
/*
 * Please fill in the following team struct
 */
team_t team = {
    "bovik",              /* Team name */

    "Harry Q. Bovik",     /* First member full name */
    "bovik@nowhere.edu",  /* First member email address */

    "",                   /* Second member full name (leave blank if none) */
    ""                    /* Second member email addr (leave blank if none) */
};
```

（3）新建优化函数需在 kernel.c 中注册，如下图所示。

```
/*********************************************************************
 * register_rotate_functions - Register all of your different versions
 *     of the rotate kernel with the driver by calling the
 *     add_rotate_function() for each test function. When you run the
 *     driver program, it will test and report the performance of each
 *     registered test function.
 *********************************************************************/

void register_rotate_functions()
{
    add_rotate_function(&naive_rotate, naive_rotate_descr);
    add_rotate_function(&rotate, rotate_descr);
    /* ... Register additional test functions here */
}
```