实验报告

实验题目:h.261运动矢量搜索算法

姓名:任卓然

学号:151220089

【实验目的及要求】

探讨运动搜索算法的改进

提供的代码：

Main.cpp，CodeInter.cpp, CodeOneIntra.cpp, CountBit.cpp, Dct.cpp, IO.cpp, Not\_est.cpp, OutputVlc.cpp, Pred.cpp, RateControl.cpp, YUVto261.cpp, config.h, Global.h, Indices.h, OutputVlc.h。

代码的编译：与系统无关，可在linux或windows下编译，建议使用VisualStudio系统编译，也可以在其它编译器里编译。

在VS里编译过程：打开VS，创建新的空项目，将所有源文件添加，编译即可。

运行命令：程序名输入YUV文件名输出的文件名待编码的帧数帧频 P帧数帧内帧的量化阶帧间帧的量化阶

实验要求：将给定代码Mot\_est.cpp中MotionEstimation函数的全搜索算法替换成对数搜索或层次搜索，比较替换前后的视频质量和编码效率。

实验提交：Mot\_est.cpp文件和本实验报告

【实验原理】（简要说明本实验项目所涉及的理论知识）

全搜索算法是一种贪心算法，算法的核心思想是遍历搜索范围中的所有宏块，选择差距最小的宏块作为结果。对宏块的比较采用绝对误差和，其算法为：



在本实验中的实现为sad = SAD\_Macroblock(ii, act\_block, h\_length, Min\_FRAME[0]);

虽然此算法开销最大，但找到的宏块必为最佳匹配宏块。假设搜索范围的长为M,宽为N，则此算法的时间复杂度为O(M\*N)。

对数搜索法是一种类似于二分搜索的算法。算法的核心思想是在给定的搜索范围内，以初始搜索点为中心，设定一个正方形，取正方形中4个角和4条边中心点所代表的宏块，比较绝对误差和来确定最佳宏块。再以此宏块的中心点为正方形的中心点，正方形的边长变为上次的一半，不断循环下去求出最佳宏块。直到正方形边长不足1为止。

此算法的开销比全搜索算法小，假设搜索范围长为M,宽为N，设定的正方形边长为D，

则算法的时间复杂度为O(log(min(M,N))\*D^2)

【实验方案】（写出修改的位置和代码）

修改Most\_est.cpp文件的MotionEstimation函数如下：

void MotionEstimation(unsigned char \*curr, unsigned char \*prev, int x\_curr,

int y\_curr, int xoff, int yoff, int seek\_dist,

MotionVector \*MV[6][MBR+1][MBC+2], int \*SAD\_0)

{

/\*前部分省略\*/

/\* 螺旋搜索 \*/

int dist = sxy/2;//设定边长为2倍dist

i = x\_curr + xoff;//确定初始中心点

j = y\_curr + yoff;

while (dist >= 1)

{

int x0 = 0, y0 = 0;//修改后的中心点

for (int num = 0; num < 8; ++num)

{

if (i >= ilow && i <= ihigh && j >= jlow && j <= jhigh)

{

ii = search\_area + (i - ilow) + (j - jlow)\*h\_length;

sad = SAD\_Macroblock(ii, act\_block, h\_length, Min\_FRAME[0]);

if (sad < Min\_FRAME[0])

{

x0 = x\_curr;

y0 = y\_curr;

MV\_FRAME[0].x = i - x\_curr;

MV\_FRAME[0].y = j - y\_curr;

Min\_FRAME[0] = sad;

}

}

if (num < 2) i += dist;

else if (num < 4) j += dist;

else if (num < 6) i -= dist;

else j -= dist;

}

dist /= 2;

i = x0;

j = y0;

}

i = x\_curr/MB\_SIZE+1;

j = y\_curr/MB\_SIZE+1;

for (k = 0; k < 5; k++)

{

MV[k][j][i]->x = MV\_FRAME[k].x;

MV[k][j][i]->y = MV\_FRAME[k].y;

MV[k][j][i]->min\_error = Min\_FRAME[k];

}

free(act\_block);

free(search\_area);

return;

}

【实验结果】（比较转换结果，原有转换值与自己方法的比较，包括数值比较和显示输出比较）

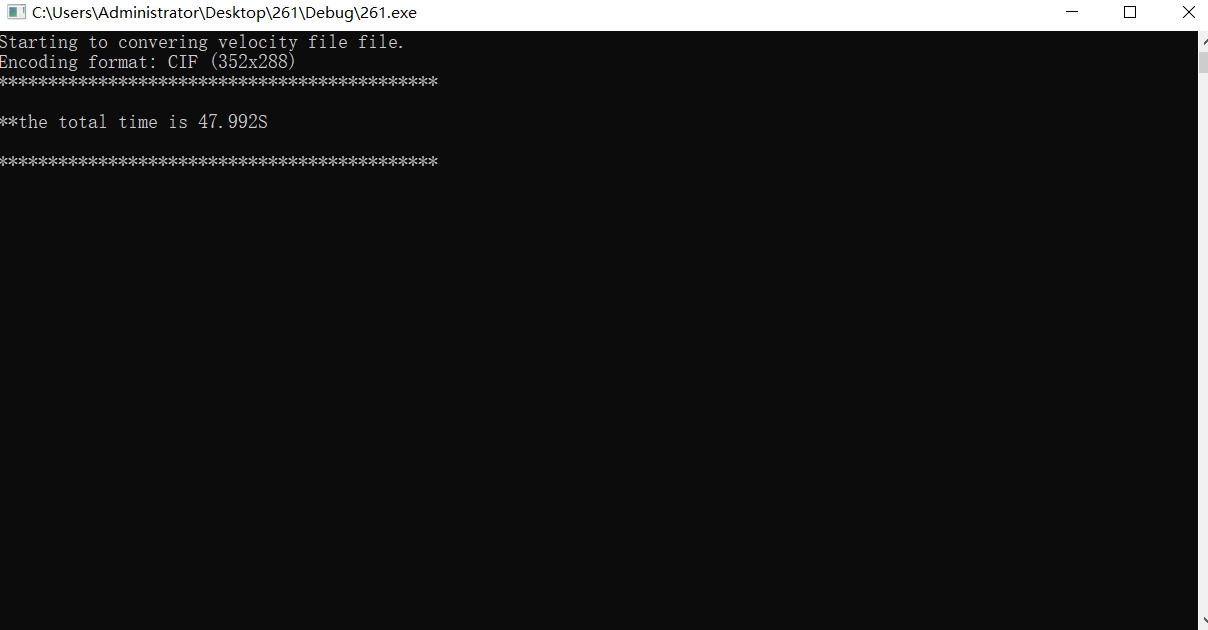
全搜索算法：原视频文件为44MB，在压缩之后大约为2.5MB，花费时间为48s。

对数搜索算法：原视频文件为44MB，在压缩之后大约为3.2MB，花费时间为10s。

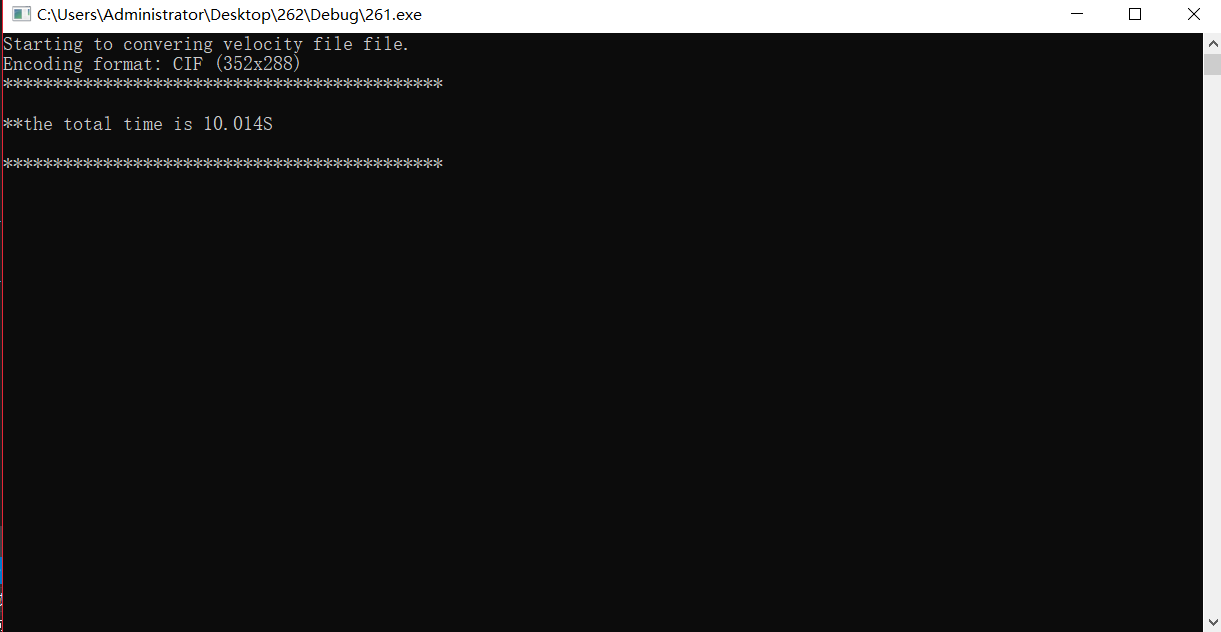
两种算法相比较而言，对数搜索算法运行较快，花费时间远低于全搜索算法。

在压缩之后，对数搜索算法的压缩视频大小比全搜索算法稍微大一点，但两个视频的质量没有明显的差别。

全搜索算法运行结果：



对数搜索算法运行结果：



【实验小结】（对本次实验的心得体会、思考和建议）

1. 实验前准备越充分，实验过程就越轻松。之前需要了解2种算法的具体区别和运行过程，实验中的问题也就轻而易举。
2. 实验中要找准中心点。首先从main函数入手，快速了解整个程序的运行过程，最后针对的Most\_est文件的中的函数进行详细了解。
3. 压缩效率和压缩时间不可兼得。