HEVC、H.264 视频编解码处理实验

姓名：鲁为涛 学号：519021910162

摘要:

本次试验任务中，我首先基于课上所学的编码技术，查阅文献学习了 HEVC、H.264 的基本组成结构和功能。在此基础上，我学习了参考代码，通过博客、github 学习了配置环境、修改参数的基本方法，最终根据实验要求设计实验过程、记录实验结果，再结合理论对结果进一步分析。我使用不同的内置编码参数和自己设计的参数，对不同纹理、运动特征和空间分辨率的测试序列做编码测试，并使用 H.264 做编解码。最终发现 randomaccess 是最适合 HEVC 编码的模式，并且 HEVC 在编码性能上显著优于 H.264。

关键词： HEVC、H.264 视频编解码

实验基本原理：

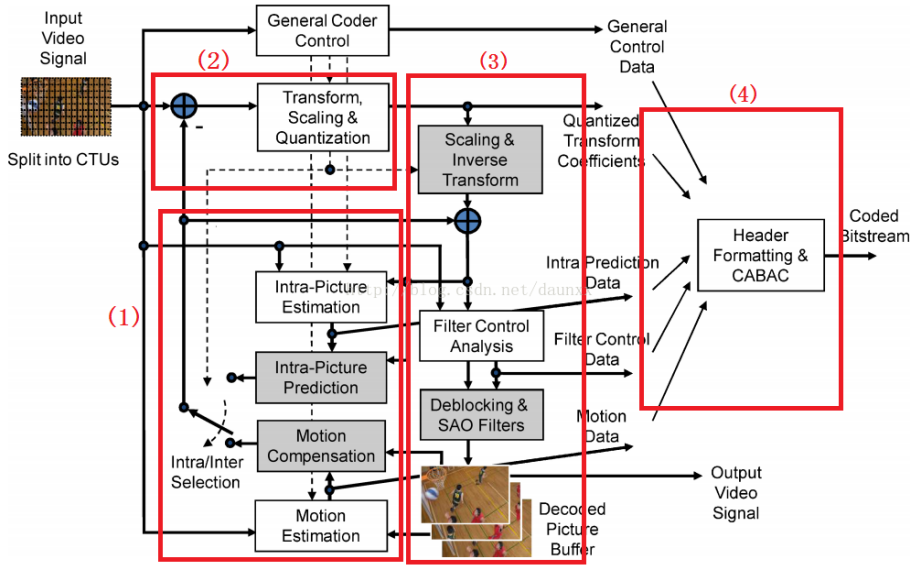
预测编码旨在除去相邻像素之间的冗余度，只对不能预测的信息进行编码。“相邻”：分为帧内和帧间。

1.HEVC 基本介绍：

HEVC 是 High Efficiency Video Coding 的缩写，是一种新的视频压缩标准，用来以替代 H.264/AVC 编码标准，2013 年 1 月 26 号，HEVC 正式成为国际标准。

类似于 H.264 和 MPEG-2，HEVC 采用 3 种帧——一个图像组内的 I、B 和 P 帧，包含帧间和帧内压缩的元素。

HEVC 的基本框图如下所示：（以下参考博客：https://blog.csdn.net/yanceyxin/article/details/81941396）



（1）代表的是预测模块，包括帧内预测和帧间预测，帧内预测用于消除空间冗余；帧间预测则主要用于消除时间上的冗余。帧内预测所使用的帧我们称为 I 帧;帧间预测所使用的帧分为 P 帧和 B 帧，P 帧指的是向前预测帧，而 B 帧指的是双向预测帧。

（2）代表的是变换和量化，在（2）模块的左边有一个特殊的数据处理符号，这个符号下面还有一个减号，此处代表的是求残差，就是说，我们从（1）模块中得到的预测后的图片和原始图片的差值，称为残差。可以看到，变换和量化的输入就是刚刚得到的残差。

（3）这个一个完整的解码器，包括反量化、反变换、滤波。之所以编码器中要有一个完整的解码器，是因为编码器中最重要的是模块（1）预测模块，而做预测是要有参考帧的，基于已有的参考帧才可以对当前帧做帧内或者帧间预测。为了实现编码器和解码器的一致性（简单点说，就是我们用电脑或者其他的什么东东看 HEVC 的视频的时候，我们得到 HEVC 的比特流后，只需

要一个解码器就可以了，通过解码器，我们得到一幅幅的图片，解码器在解码的时候其参考图片是前面已解码的图片，为了预测的准确性，编码器和解码器做预测的时候所参考的图片必须是一样的，所以在编码器中寸在了一个完整的解码器，这个解码器的作用就是为了实现参考帧的一致性）。

（4）这个模块是从码流的视角进一步对视频流压缩，主要是消除编码冗余。

2. HEVC 和 H.264 的对比

HEVC 与 H.264 相比，有许多的进步

编码树块：在 H.264 采用有最大 16x16 尺寸的宏块的场合，HEVC 则采用有最大 64x64 像素尺寸的编码树块，或 CTB。在编码更大的帧尺寸时（如 4K 分辨率），更大的块尺寸更高效。

帧内预测方向：在 H.264 采用 9 个帧内预测方向的场合，HEVC 能够使用超过 35 个方向，增加更多提升更高效帧内压缩的可能的参考像素块（参见图 2）。明显的代价是在增加的方向中搜索需要更多编码时间。

其它进展包括：

- 自适应运动矢量 ，允许此编解码发现更多的帧间冗余
- 优异的并行化工具，包括在多核环境内更高效编码的波前并行处理
- 熵编码为 CABAC 独有，CAVLC 不再有
- 去块效应滤波器的改进以及建立一个进一步限制块边缘失真的称为取样自适应补偿的第二滤波器

本实验中所使用的 HM16.20 编码器，提供了 3 种内置编码模式，仅码内预测的 Intra 模式、Lowdelay 模式和 RandomAccess 模式。

1 实验任务一：HEVC 视频编码和解码

实验过程：

本实验中我主要通过学习网络上的博客经验、github 的公开代码和操作手册，并结合本学期所学课堂知识完成。详细的流程可见：https://blog.csdn.net/daunxx/article/details/38048605

具体实验步骤为：1，通过 Github 下载 Hm16.20 软件、安装 VS 环境。

2：通过 Vs2022 执行 build 中 vs2015 中的 sln 软件，将 TAppEncoder 设置为启动项目，然后编译生成可执行文件。

3：**参数分析**：首先从官网下载测试序列，在 Hm16.20 中找到内置的参数，包括各测试序列对应的参数 xxx.cfg 文件和三种编码模式参数.cfg 文件。其中测试序列的.cfg 文件中参数有：

```
InputBitDepth                 : 8          # Input bitdepth
InputChromaFormat             : 420        # Ratio of luminance to chrominance samples
FrameRate                     : 50         # Frame Rate per second
FrameSkip                     : 0          # Number of frames to be skipped in input
SourceWidth                   : 416        # Input  frame width
SourceHeight                  : 240        # Input  frame height
FramesToBeEncoded             : 500      # Number of frames to be coded
```

InputBitDepth 决定 YUV 文件使用多少位来定义一个像素点，其余参数决定了输出尺寸、播放每秒帧数、编码文件的总帧数等。

内置模式的编码参数则较多，有 Intra、lowdelay、Randomaccess 三种模式，具有不同的编码参数，三种模式的区别主要体现在以下模块：

```
#======== Coding Structure =============
IntraPeriod                   : 1          # Period of I-Frame ( -1 = only first)
DecodingRefreshType           : 1          # Random Accesss 0:none, 1:CRA, 2:IDR, 3:Recovery Point SEI
GOPSize                       : 1          # GOP Size (number of B slice = GOPSize-1)
ReWriteParamSetsFlag          : 1          # Write parameter sets with every IRAP
```

其中，GOP 参数决定了两个 I 帧数的距离，GOP 参数越大，意味着 I 帧数越小。其余参数分共同决定了所属模式。**Intra 模式仅有帧内预测，因此 GOP=1，Lowdelay 仅有第一帧为 I 帧，而**

RandomAccess 则是每 32 帧出现一帧 I 帧。

其中关键的参数有 QP，为量化参数

```
QP                                : 32          # Quantization parameter(0-51)
```

QP 参数决定了量化步长，越低的 QP 意味着量化越精细，越大的 QP 则意味着量化越粗糙，大 QP 将使图像质量更低。但低的 QP 值将占用更多编码时的内存及更大的码流。

4：修改 TAppEncoder 属性页 debuger 中的命令参数和目标路径，使其和设置的参数文件匹配。我首先使用三种内置模式参数，完整编码了 500 帧的 416*240 的吹泡泡视频，对比三种模式的区别。

5：**修改参数**。对于视频参数，我修改了编码文件的帧数，减少为 35 帧，从而减小编码耗时。我修改了 QP 值，改变量化程度，探索 QP 值对视频质量的影响。此外，我选取了运动属性、纹理不同的测试序列，探索不同测试序列在上述同样参数下的编解码结果。

实验结果：

本实验中需要使用多种编码参数和两组测试序列，我使用的编码参数是由 HM16 内置的 encoder_X_main.cfg。其中 X 分别为 intra、lowdelay、randomprocess。整体实验过程如前文实验步骤 4 和 5 所示，下为 4，5 对应的实验结果及分析。

（1）采用 3 种内置模式的实验结果

如图为采用内置的 encoder_intra_main.cfg 参数对吹泡泡视频编解码后结果。Intra 只包含 I 帧（关键帧）、不包含 B 帧和 P 帧。我修改了文件的输出帧数，使其和原始视频保持一致（500 帧），并随机截取了实验结果的 129、474 两帧。（图 1）

图 1 Intra 参数模式下的两帧关键帧实验结果（左：原始视频，右：编码解码后的视频）

两帧图像的 PSNR 值分别如下所示：

POC 129 71704 bits [Y 32.8741 dB U 37.3101 dB V 37.4967 dB]
POC 474 99064 bits [Y 32.4104 dB U 36.1919 dB V 36.3828 dB]

该编解码过程的总统计运行结果如图 2：由于对应的 intra 模式只有帧内预测，无帧间预测，因此结果仅 I 帧，无 B 帧和 P 帧，且每一帧的码率相似。平均 PSNR 值为 33.7209.

```
SUMMARY --------------------------------------------------------
      Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
           500   a    4053.7496   32.7417   36.6866   37.1132   33.7209


I Slices--------------------------------------------------------
      Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
           500   i    4053.7496   32.7417   36.6866   37.1132   33.7209


P Slices--------------------------------------------------------
      Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             0   p    -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)


B Slices--------------------------------------------------------
      Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             0   b    -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)

RVM: 0.000
Bytes written to file: 5067187 (4053.750 kbps)

 Total Time:     1377.642 sec.
```

图 2 Intra 参数下的编解码结果

此外，我尝试了内置的 Lowdelay 参数做编解码。与 intra 参数相比，该编码方式采用了帧间预测，大大减少了平均码率（在 I 帧间插入多个 B 帧，如图 2 所示），并且除了第一帧为 I 帧，码率为 71200bits，其余帧均为 B 帧，码率均较 Intra 模式明显变小。

```
POC    0 TId: 0 ( I-SLICE, nQP 31 QP 31 )     71200 bits [Y 34.1906 dB    U 37.2573 dB    V 38.8997 dB] [ET    2 ] [L0 ] [L1 ]
POC    1 TId: 0 ( B-SLICE, nQP 40 QP 40 )      2120 bits [Y 32.5833 dB    U 36.8909 dB    V 38.6126 dB] [ET    3 ] [L0 0 ] [L1 0 ]
POC    2 TId: 0 ( B-SLICE, nQP 39 QP 39 )      3416 bits [Y 32.0003 dB    U 36.8047 dB    V 38.4823 dB] [ET    5 ] [L0 1 0 ] [L1 1 0 ]
POC    3 TId: 0 ( B-SLICE, nQP 40 QP 40 )      2704 bits [Y 31.2792 dB    U 36.6170 dB    V 38.2559 dB] [ET    5 ] [L0 2 1 0 ] [L1 2 1 0 ]
POC    4 TId: 0 ( B-SLICE, nQP 33 QP 33 )     15520 bits [Y 32.9787 dB    U 36.9175 dB    V 38.5050 dB] [ET    9 ] [L0 3 2 1 0 ] [L1 3 2 1 0 ]
POC    5 TId: 0 ( B-SLICE, nQP 40 QP 40 )      1344 bits [Y 31.8933 dB    U 36.6454 dB    V 38.1208 dB] [ET    6 ] [L0 4 3 2 0 ] [L1 4 3 2 0 ]
POC    6 TId: 0 ( B-SLICE, nQP 39 QP 39 )      2808 bits [Y 31.5209 dB    U 36.6863 dB    V 38.0574 dB] [ET    7 ] [L0 5 4 3 0 ] [L1 5 4 3 0 ]
POC    7 TId: 0 ( B-SLICE, nQP 40 QP 40 )      2088 bits [Y 31.1833 dB    U 36.5473 dB    V 38.0662 dB] [ET    6 ] [L0 6 5 4 0 ] [L1 6 5 4 0 ]
```

图 3 Lowdelay 模式下的编码过程，由于存在 B 帧，每帧信息量减少

同样使用第 129、474 帧。与 Intra 模式相比，采用帧间预测后，信息量明显下降，而视频质量并未有明显变化。编解码后的两帧实验结果如图 4 所示：

POC 129 968 bits [Y 31.3748 dB U 36.8851 dB V 37.0771 dB] POC 4745800 bits [Y 29.6152 dB U 35.4948 dB V 35.6007 dB]

```
SUMMARY ------------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             500    a     350.5256   30.8569   36.0507   36.5810   31.9656

I Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
               1    i    3560.0000   34.1906   37.2573   38.8997   35.1311

P Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
               0    p    -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)

B Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             499    b     344.0938   30.8503   36.0482   36.5763   31.9610
```

图 4 Lowdelay 模式参数下的编解码结果和统计 PSNR 值

最后，我尝试了第三种参数模式，即 Randomaccess 模式。编解码后的对应两帧如下所示：

POC 129 640 bits [Y 33.7014 dB U 38.2445 dB V 38.5258 dB]

POC 474 4344 bits [Y 30.2312 dB U 36.4530 dB V 36.8273 dB]

```
SUMMARY ------------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             500    a     365.1016   32.0315   37.1318   37.8078   33.1099

I Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
              16    i    5695.5500   35.1054   37.7646   38.2631   35.8607

P Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
               0    p    -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)  -nan(ind)

B Slices-----------------------------------------------------
       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             484    b     188.8884   31.9299   37.1109   37.7927   33.0430

RVM: 0.000
Bytes written to file: 456377 (365.102 kbps)

 Total Time:     2522.621 sec.
```

图 5 Random 模式参数下的编解码结果和统计 PSNR 值

综合三种模式压缩 500 帧的平均 PSNR 值，结果如表 1 所示：

编码模式	I 帧个数	码率	Y-PSNR	U-PSNR	V-PSNR	YUV-PSNR	运行时间(s)	文件大小
Intra	500	4053.7496	32.7417	36.6866	37.1132	33.7209	1377	4949KB
LowDelay	1	350.526	30.8569	36.0507	36.5810	31.9656	6450	428KB
RandomAccess	16	365.1016	32.0315	37.1318	37.8078	33.1099	2522	446KB

表 1 三种内置编码模式运行结果统计

（1）**结果的理论分析**：由关键帧解码图像对比可知，三种模式的编码解码后的视频质量与原视频几乎没有肉眼可见的差异，均较好的保证了压缩后视频的质量。对于表 1 所示统计结果，由于 Intra 模式仅使用帧内预测，没有帧间预测，因此不需要调用其他帧的信息，运行时间最短，并且信噪比最高。但同时也因 Intra 没有帧间预测，压缩效果最差。对于 Lowdelay 和 Randomaccess 两种模式，均采用了帧间预测，区别在于 Lowdelay 仅有第一帧是 I 帧，而

Randomaccess 则每隔 32 帧为 1 个 I 帧，因此 Randomaccess 模式的运行时间远小于 Lowdelay 模式。Lowdelay 模式运用了最多的帧间预测，结果是信噪比下降，并且运行时间大大上升，好处是压缩效果相较于 Intra 模式提升很大。RandomAccess 模式则更像是 Lowdelay 和 Intra 的一种折中方案。

综合而言，Randomaccess 模式无论是在运行时间、压缩效果、图像质量上都比较好，是三种模式中最适合做视频压缩的模式。

（2）采用不同的测试序列和改变参数的实验结果

由实验 1（1）可知，Randomaccess 模式在时间和效果上都较佳，并且有多个 I 帧。因此我在实验 1（2）中均采用该模式。

（2.0）减少编码帧数

通过修改测试序列参数文件的参数实现减少编码帧数，从而减少运行时间。我在实验 1（2）中均使用 30 帧视频用于测试参数变化对实验的影响。

（2.1）测试不同 QT 值对图像质量的影响

QP 参数决定了量化步长,越低的 QP 意味着量化越精细,越大的 QP 则意味着量化越粗糙。我分别测试了 QP=13，32，51 时的实验结果，并选取了其中的 I 帧作为关键帧来对比原始和经编码解码后的图像质量。

实验结果如表 2 所示：

QP	码率	YUV-PSNR	运行时间(s)	文件大小
13	7071.7269	45.0119	436.537	590KB
32	440.1257	33.3324	193.681	38KB
51	39.6000	25.5850	145.770	4KB

表 2 不同 QT 值的运行结果统计

图 6 3 种 QP 参数对应的关键帧（I）质量（1：13；2：32；3：51；4：原图）

三种 QP 值对应图像分别的 PSNR:

POC32 465560 bits [Y 52.9899 dB U 52.0168 dB V 51.9930 dB]

POC32 79168 bits [Y 35.9189 dB U 38.7249 dB V 39.2601 dB]

POC32 5976 bits [Y 25.1907 dB U 32.5098 dB V 33.0979 dB] 、

（2.2）更换 HEVC 测试序列后，三种 QT 值的实验结果

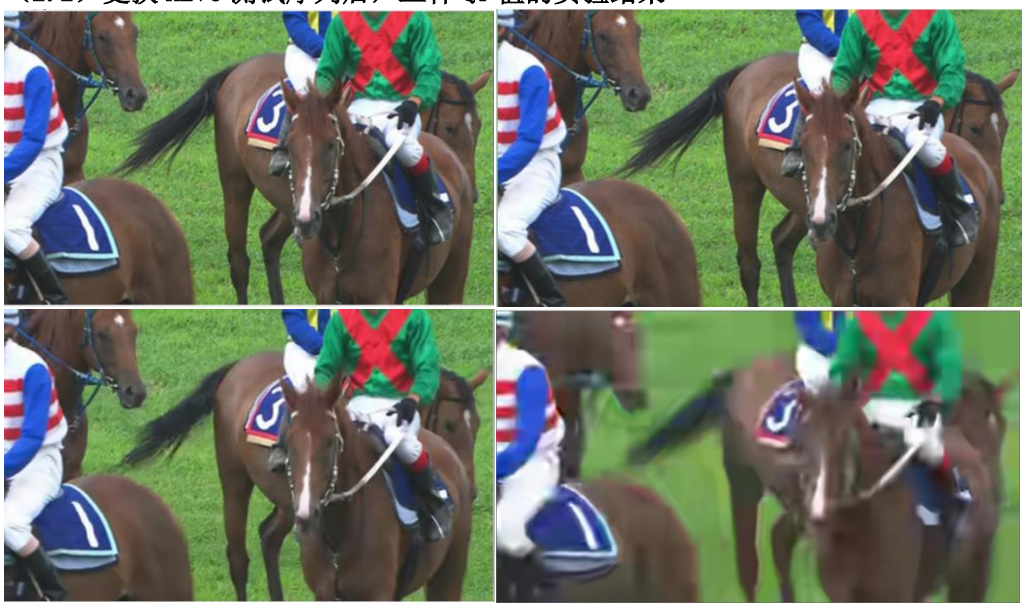


图 7 骑马序列 3 种 QP 参数对应的关键帧（I）质量（上 1：13；上 2：原图；下 1：32；下 2：51；）

（2）结果的理论分析：

测试序列 RaceHorese.yuv 和 Bubbleblow.yuv 在运动、纹理、空间分辨率均有较大区别。就运动特性而言，吹泡泡的动作较复杂且速度较快，骑马的动作特性较缓慢且速度较慢。就纹理特性而言，吹泡泡视频的纹理特性较复杂，骑马视频的空间分辨率大于吹泡泡。

当 QP=13 时，吹泡泡视频编码 35 帧耗时 436s，骑马视频编码 35 帧耗时 508.92s，相同 QP 值编码后的储存空间占用，骑马视频也高于吹泡泡，这是因为骑马视频的空间分辨率较大，原视频同样帧数时也大于吹泡泡。

采用不同的 QP 值做参数，较小的 QP 值意味着量化更精确，但代价是信息量增大。随着 QP 值的增加，编码量化更加粗糙，编码解码后的图像质量也明显下降。当 QP=32 时，PSNR 值已明显降低，但肉眼仍无法区分出明显差别，如图 7 所示，但局部放大图像，可以发现在纹理较复杂、对量化需求大的图像部分，图像质量已经显著下降，如图 6 种局部放大后被泡泡遮住的脸部，及图 7 中左下的马尾巴（QP=32）明显模糊。当 QP=51 时，量化已经非常粗糙，此时纹理特征为大面积色块。

2 实验任务二： HEVC、H.264 视频编解码对比实验

实验过程

我选取的测试序列为 RaceHorses 的前 35 帧，使用 HEVC 的默认 randomaccess 模式和 lowdelay_P 模式的默认参数，将两种参数与 H.264 的 encoder_baseline 模式编码做比较。

实验结果

参数配置：HEVC 采用 randomaccess 和 lowdelay_P 的默认参数设置，encoder_baseline 中修改了输入输出的参数设置，使其与 HEVC 保持一致，其余参数均使用默认值。

编码性能分析：

编码模式	帧类	比特率	Y-PSNR	U-PSNR	V-PSNR	YUV-PSNR	用时	总比特数
HEVC Random	平均	382.1074	31.8304	37.0344	37.2858	32.8703	268.122	55724
	I 帧 (2)	2574.2400	36.3224	38.4885	39.1183	37.0001		
	B 帧 (33	249.2509	31.5581	36.9463	37.1748	32.7118		
HEVC LowdelayP	平均	399.0103	31.7158	36.6404	36.7653	32.7080	233.025	58189
	I 帧 (1	2192.4	34.8481	37.8336	38.3001	35.6796		
	P 帧 (34	346.2635	31.6237	36.6053	36.7201	32.6451		
H264	平均	1356.51	36.511	38.239	38.534	NAN	646.232	1582592
	I 帧 (1	3665.42	36.751	38.808	39.094	NAN		
	P 帧 (34	1288.60	36.561	38.050	38.364	NAN		

表 3 HEVC 和 H.264 的 PSNR 结果和编码器性能

我选取了两种 HEVC 的第 1 帧（关键帧）和另一个 PSNR 值较大的 B（P）帧第 28 帧，并选择了 H.264 对应的两帧。原始图像、解码图像及对应 PSNR 值如图 8，9，10，11 所示：

图 8　原始图像（左第 1 帧，右第 28 帧）

Bit：123352，36.751Y　38.808 U　39.094V　　Bit：　44344　36.436Y　38.310U　38.648V

图 9　H.264 解码结果（左第 1 帧，右第 28 帧）

Bit：73080，Y 34.8481　U 37.8336　V 38.3001　　Bit：　27912　Y 33.3056　U 37.2253　　V 37.4241

图 10　HEVC_lowdelayP 解码结果（左第 1 帧，右第 28 帧）

Bit：89144，Y 36.3114　　U 38.3783　V 38.9737　　Bit：　14480　Y 36.3114　U 38.3783　V 38.9737

图 10　HEVC_Randomaccess 解码结果（左第 1 帧，右第 28 帧）

实验二结果分析

从图像质量而言，两种 HEVC 编码模式相较于 H.264 主要在 Y 通道的信噪比较低，平均信噪比略低于 H.264。说明在明亮度上的质量下降。但这并不会影响肉眼识别。三种编码模式都与原图相差无几。

从编码速率而言，两种 HEVC 编码模式均远快于 H.264，主要的原因是 HEVC 采用复杂的算法，如更多的帧内预测方向，找到了更好的压缩方式，使得需要编码的字节量减小了。第二各原因可能是因为 HEVC 采用的是优化后的并行计算方式，因此计算效率更好。

从压缩效果而言，两种 HEVC 编码的字节总数均远小于 H.264,仅 H.264　1/3 的比特数，HEVC 便可达到相似的压缩效果。

分别比较 I 帧和 B/P 帧，由于更好的帧内预测，HEVC 的 I 帧 bit 小于 H264，但可以发现由于 HEVC 使用了更好的帧间预测，如自适应运动矢量 ，允许此编解码发现更多的帧间冗余，因此在 B/P 帧上的压缩效果远超过 H.264，在 B/P 帧上的提升大于 I 帧。

此外一个与实验一中有区别的发现是此处采用的 lowdelay_P 模式运行时间竟然小于 Randomaccess（实验一中发现，lowdelay 比 Randomaccess 慢两倍左右），我觉得这是因为 P 帧

对应的是前项预测，B 帧对应的是双向预测，B 帧需要的帧间信息更多，因此在 HEVC 算法中采用双向预测 B 帧耗用的时间更长。

3 实验总结

本次试验任务中，我首先基于课上所学的编码技术，查阅文献学习了 HEVC、H.264 的基本组成结构和功能。在此基础上，我学习了参考代码，通过博客、github 学习了配置环境、修改参数的基本方法，最终根据实验要求设计实验过程、记录实验结果，再结合理论对结果进一步分析。我使用不同的内置编码参数和自己设计的参数，对不同纹理、运动特征和空间分辨率的测试序列做编码测试，并使用 H.264 做编解码。最终发现 randomaccess 是最适合 HEVC 编码的模式，并且 HEVC 在编码性能上显著优于 H.264。

通过本次实验，我进一步熟悉了 VS 配置环境的过程，提高了自己阅读代码、datasheet、从博客和 github 学习代码的能力。此外，我此前只是学习了理论，借助本次实验，我对于帧间预测、帧内预测的作用有了直观的了解，更进一步理解了 HEVC 相对于 H.264 的提升。

在本实验中，我也遇到过一些小“坎”，比如一开始不会播放 Yuv 文件，无法下载测试序列，以及参考的博客例子使用了错误的 cfg 文件，导致我一开始使用了 10bits/s 的速率跑出了色彩奇妙的图像却一直找不到 bug。不过这些问题都最终被我一一解决。

感谢解蓉老师在课堂上的教学，让我对本次实验打下了扎实的理论基础，也感谢助教提供的样例代码和在我遇到困难时的热心帮助。