本人使用x264主要想调整的参数，

第一，profile，也就是使用baseline还是main,还是high编码。

可以通过该接口设置

x264\_param\_apply\_profile();

第二, 编码复杂度

param.i\_level\_idc=30;

第三,图像质量控制

 param.rc.f\_rf\_constant = 25;

 param.rc.f\_rf\_constant\_max = 45;

rc.f\_rf\_constant是实际质量，越大图像越花，越小越清晰。

param.rc.f\_rf\_constant\_max ，图像质量的最大值。

第四, 码率控制

一开始我使用恒定码流设置，无论我怎么设置，都无法控制实际码流，后来换成平均码流后，就行了。

param.rc.i\_rc\_method = X264\_RC\_ABR;//参数i\_rc\_method表示码率控制，CQP(恒定质量)，CRF(恒定码率)，ABR(平均码率)

param.rc.i\_vbv\_max\_bitrate=(int)((m\_bitRate\*1.2)/1000) ; // 平均码率模式下，最大瞬时码率，默认0(与-B设置相同)

param.rc.i\_bitrate = (int)m\_bitRate/1000;

x264使用的bitrate需要/1000。

第五，使用实时视频传输时，需要实时发送sps,pps数据

param.b\_repeat\_headers = 1;  // 重复SPS/PPS 放到关键帧前面

该参数设置是让每个I帧都附带sps/pps。

第六. I帧间隔

我是将I帧间隔与帧率挂钩的，以控制I帧始终在指定时间内刷新。

以下是2秒刷新一个I帧

 param.i\_fps\_num = (int)m\_frameRate;

 param.i\_fps\_den = 1;

 param.i\_keyint\_max = m\_frameRate \* 2;

第七,编码延迟

在使用中，开始总是会有编码延迟，导致我本地编码立即解码回放后也存在巨大的视频延迟，

后来发现设置x264\_param\_default\_preset(&param, "fast" , "zerolatency" );后就能即时编码了。

主要是zerolatency该参数。

基本的就这样了，完整的设置代码

 x264\_param\_t param;

 x264\_param\_default\_preset(&param, "fast" , "zerolatency" );

 param.i\_width = m\_width;

 param.i\_height = m\_height;

 param.b\_repeat\_headers = 1;  // 重复SPS/PPS 放到关键帧前面

 param.b\_cabac = 1;

 param.i\_threads = 1;

 param.i\_fps\_num = (int)m\_frameRate;

 param.i\_fps\_den = 1;

 param.i\_keyint\_max = m\_frameRate \* 2;

 // rc

 //param.rc.b\_mb\_tree=0;//这个不为0,将导致编码延时帧...在实时编码时,必须为0

 param.rc.f\_rf\_constant = 25;

 param.rc.f\_rf\_constant\_max = 45;

 param.rc.i\_rc\_method = X264\_RC\_ABR;//参数i\_rc\_method表示码率控制，CQP(恒定质量)，CRF(恒定码率)，ABR(平均码率)

 //param.rc.f\_rate\_tolerance=0.1;

 param.rc.i\_vbv\_max\_bitrate=(int)((m\_bitRate\*1.2)/1000) ; // 平均码率模式下，最大瞬时码率，默认0(与-B设置相同)

 param.rc.i\_bitrate = (int)m\_bitRate/1000;

 x264\_param\_apply\_profile(&param, "baseline");

 param.i\_level\_idc=30;

 param.i\_log\_level = X264\_LOG\_NONE;

 if(( m\_p264Handle = x264\_encoder\_open(&param)) == NULL)

看看别人写的对x264结构体的说明

typedef struct x264\_param\_t

{

  /\* CPU 标志位 \*/

  unsigned int cpu;

  int i\_threads; /\* 并行编码多帧 \*/

  int b\_deterministic; /\*是否允许非确定性时线程优化\*/

  int i\_sync\_lookahead; /\* 线程超前缓冲 \*/

  /\* 视频属性 \*/

  int i\_width; /\* 宽度\*/

  int i\_height; /\* 高度\*/

  int i\_csp; /\* 编码比特流的CSP,仅支持i420，色彩空间设置 \*/

  int i\_level\_idc; /\* level值的设置\*/

  int i\_frame\_total; /\* 编码帧的总数, 默认 0 \*/

/\*Vui参数集视频可用性信息视频标准化选项 \*/

  struct

  {

  /\* they will be reduced to be 0 < x <= 65535 and prime \*/

  int i\_sar\_height;

  int i\_sar\_width; /\* 设置长宽比 \*/

  int i\_overscan; /\* 0=undef, 1=no overscan, 2=overscan 过扫描线，默认"undef"(不设置)，可选项：show(观看)/crop(去除)\*/

  /\*见以下的值h264附件E \*/

  Int i\_vidformat;/\* 视频格式，默认"undef"，component/pal/ntsc/secam/mac/undef\*/

  int b\_fullrange; /\*Specify full range samples setting，默认"off"，可选项：off/on\*/

  int i\_colorprim; /\*原始色度格式，默认"undef"，可选项：undef/bt709/bt470m/bt470bg，smpte170m/smpte240m/film\*/

  int i\_transfer; /\*转换方式，默认"undef"，可选项：undef/bt709/bt470m/bt470bg/linear,log100/log316/smpte170m/smpte240m\*/

  int i\_colmatrix; /\*色度矩阵设置，默认"undef",undef/bt709/fcc/bt470bg,smpte170m/smpte240m/GBR/YCgCo\*/

  int i\_chroma\_loc; /\* both top & bottom色度样本指定，范围0~5，默认0 \*/

  } vui;

  int i\_fps\_num;

  int i\_fps\_den;

/\*这两个参数是由fps帧率确定的，赋值的过程见下：

{ float fps;

if( sscanf( value, "%d/%d", &p->i\_fps\_num, &p->i\_fps\_den ) == 2 )

  ;

  else if( sscanf( value, "%f", &fps ) )

  {

  p->i\_fps\_num = (int)(fps \* 1000 + .5);

  p->i\_fps\_den = 1000;

  }

  else

  b\_error = 1;

  }

Value的值就是fps。\*/

  /\*流参数 \*/

  int i\_frame\_reference; /\* 参考帧最大数目 \*/

  int i\_keyint\_max; /\* 在此间隔设置IDR关键帧 \*/

  int i\_keyint\_min; /\* 场景切换少于次值编码位I, 而不是 IDR. \*/

  int i\_scenecut\_threshold; /\*如何积极地插入额外的I帧 \*/

  int i\_bframe; /\*两个相关图像间P帧的数目 \*/

  int i\_bframe\_adaptive; /\*自适应B帧判定\*/

  int i\_bframe\_bias; /\*控制插入B帧判定，范围-100~+100，越高越容易插入B帧，默认0\*/

  int b\_bframe\_pyramid; /\*允许部分B为参考帧 \*/

/\*去块滤波器需要的参数\*/

  int b\_deblocking\_filter;

  int i\_deblocking\_filter\_alphac0; /\* [-6, 6] -6 light filter, 6 strong \*/

  int i\_deblocking\_filter\_beta; /\* [-6, 6] idem \*/

  /\*熵编码 \*/

  int b\_cabac;

  int i\_cabac\_init\_idc;

  int b\_interlaced; /\* 隔行扫描 \*/

  /\*量化 \*/

  int i\_cqm\_preset; /\*自定义量化矩阵(CQM),初始化量化模式为flat\*/

  char \*psz\_cqm\_file; /\* JM format读取JM格式的外部量化矩阵文件，自动忽略其他—cqm 选项\*/

  uint8\_t cqm\_4iy[16]; /\* used only if i\_cqm\_preset == X264\_CQM\_CUSTOM \*/

  uint8\_t cqm\_4ic[16];

  uint8\_t cqm\_4py[16];

  uint8\_t cqm\_4pc[16];

  uint8\_t cqm\_8iy[64];

  uint8\_t cqm\_8py[64];

  /\* 日志 \*/

  void (\*pf\_log)( void \*, int i\_level, const char \*psz, va\_list );

  void \*p\_log\_private;

  int i\_log\_level;

  int b\_visualize;

  char \*psz\_dump\_yuv; /\* 重建帧的名字 \*/

  /\* 编码分析参数\*/

  struct

  {

  unsigned int intra; /\* 帧间分区\*/

  unsigned int inter; /\* 帧内分区 \*/

  int b\_transform\_8x8; /\* 帧间分区\*/

  int b\_weighted\_bipred; /\*为b帧隐式加权 \*/

  int i\_direct\_mv\_pred; /\*时间空间队运动预测 \*/

  int i\_chroma\_qp\_offset; /\*色度量化步长偏移量 \*/

  int i\_me\_method; /\* 运动估计算法 (X264\_ME\_\*) \*/

  int i\_me\_range; /\* 整像素运动估计搜索范围 (from predicted mv) \*/

  int i\_mv\_range; /\* 运动矢量最大长度(in pixels). -1 = auto, based on level \*/

  int i\_mv\_range\_thread; /\* 线程之间的最小空间. -1 = auto, based on number of threads. \*/

  int i\_subpel\_refine; /\* 亚像素运动估计质量 \*/

  int b\_chroma\_me; /\* 亚像素色度运动估计和P帧的模式选择 \*/

  int b\_mixed\_references; /\*允许每个宏块的分区在P帧有它自己的参考号\*/

  int i\_trellis; /\* Trellis量化，对每个8x8的块寻找合适的量化值，需要CABAC，默认0 0：关闭1：只在最后编码时使用2：一直使用\*/

  int b\_fast\_pskip; /\*快速P帧跳过检测\*/

  int b\_dct\_decimate; /\* 在P-frames转换参数域 \*/

  int i\_noise\_reduction; /\*自适应伪盲区 \*/

  float f\_psy\_rd; /\* Psy RD strength \*/

  float f\_psy\_trellis; /\* Psy trellis strength \*/

  int b\_psy; /\* Toggle all psy optimizations \*/

  /\*，亮度量化中使用的无效区大小\*/

  int i\_luma\_deadzone[2]; /\* {帧间, 帧内} \*/

  int b\_psnr; /\* 计算和打印PSNR信息 \*/

  int b\_ssim; /\*计算和打印SSIM信息\*/

  } analyse;

  /\* 码率控制参数 \*/

  struct

  {

  int i\_rc\_method; /\* X264\_RC\_\* \*/

  int i\_qp\_constant; /\* 0-51 \*/

  int i\_qp\_min; /\*允许的最小量化值 \*/

  int i\_qp\_max; /\*允许的最大量化值\*/

  int i\_qp\_step; /\*帧间最大量化步长 \*/

  int i\_bitrate; /\*设置平均码率 \*/

  float f\_rf\_constant; /\* 1pass VBR, nominal QP \*/

  float f\_rate\_tolerance;

  int i\_vbv\_max\_bitrate; /\*平均码率模式下，最大瞬时码率，默认0(与-B设置相同) \*/

  int i\_vbv\_buffer\_size; /\*码率控制缓冲区的大小，单位kbit，默认0 \*/

  float f\_vbv\_buffer\_init; /\* <=1: fraction of buffer\_size. >1: kbit码率控制缓冲区数据保留的最大数据量与缓冲区大小之比，范围0~1.0，默认0.9\*/

  float f\_ip\_factor;

  float f\_pb\_factor;

  int i\_aq\_mode; /\* psy adaptive QP. (X264\_AQ\_\*) \*/

  float f\_aq\_strength;

  int b\_mb\_tree; /\* Macroblock-tree ratecontrol. \*/

  int i\_lookahead;

  /\* 2pass 多次压缩码率控制 \*/

  int b\_stat\_write; /\* Enable stat writing in psz\_stat\_out \*/

  char \*psz\_stat\_out;

  int b\_stat\_read; /\* Read stat from psz\_stat\_in and use it \*/

  char \*psz\_stat\_in;

  /\* 2pass params (same as ffmpeg ones) \*/

  float f\_qcompress; /\* 0.0 => cbr, 1.0 => constant qp \*/

  float f\_qblur; /\*时间上模糊量化 \*/

  float f\_complexity\_blur; /\* 时间上模糊复杂性 \*/

  x264\_zone\_t \*zones; /\* 码率控制覆盖 \*/

  int i\_zones; /\* number of zone\_t's \*/

  char \*psz\_zones; /\*指定区的另一种方法\*/

  } rc;

  /\* Muxing parameters \*/

  int b\_aud; /\*生成访问单元分隔符\*/

  int b\_repeat\_headers; /\* 在每个关键帧前放置SPS/PPS\*/

  int i\_sps\_id; /\* SPS 和 PPS id 号 \*/

  /\*切片（像条）参数 \*/

  int i\_slice\_max\_size; /\* 每片字节的最大数，包括预计的NAL开销. \*/

  int i\_slice\_max\_mbs; /\* 每片宏块的最大数，重写 i\_slice\_count \*/

  int i\_slice\_count; /\* 每帧的像条数目: 设置矩形像条. \*/

  /\* Optional callback for freeing this x264\_param\_t when it is done being used.

  \* Only used when the x264\_param\_t sits in memory for an indefinite period of time,

  \* i.e. when an x264\_param\_t is passed to x264\_t in an x264\_picture\_t or in zones.

  \* Not used when x264\_encoder\_reconfig is called directly. \*/

  void (\*param\_free)( void\* );

} x264\_param\_t;

这个是老版本的，新的x264好像新增了一些参数。