Reverse Engineering of X265 Rate Control

Contents

1 General

We consider the following modes of x265 RC (Rate Control):

- **Constant Quality Mode (CRF)** - Rate Control attempts to keep near constant visual perception. For example, RC increases the QP for high motion frames (more quantization errors, since HVS (Human Vision System) is less sensitive in high motions) and lowers QP down for low motion. Frame start QP is specified by a **predefined (experimentally determined) map** from the crf-value and the frame type (I,p or B) to the start QP. This mapping function is hard-coded and it does not depend on video content at all (not best solution). This CRF Rate Control mode is specified in the command-line by '`--crf X`'.

- **Single Pass ABR (Adaptive Bitrate):** RC attempts to keep an instant bitrate close to the target one, but to some local bitrate are allowed to not compromise heavily visual quality. This mode is specified in the command-line '`--bitrate X`'

For CRF and ABR rate control modes the adaptive quantization can be optionally added by '`--aq-mode [1|2]`', for details on the adaptive quantization pls. refer https://www.ramugedia.com/on-qp-modulation . Two adaptive quantization modes are available: auto-variance ('`--aq-mode 2`') and aq_variance ('`--aq-mode 1`').

The adaptive quantization is divided into two stages: look-ahead and encoding.

In **aq-mode**=1 the aq-strength (QP fluctuation range within a frame) for all frames is the same.

In **aq-mode=2** the aq-strength is adapted for each frame, based on its energy (i.e. complexity).

In both modes the quantizer varies for each CTU, however the variability of the quantizer in the aq-mode=2 is adaptive.

For illustration, let's imagine the picture consisting of two stripes: the bottom is grass (high complexity) and the top is clear blue sky (low complexity), then aq-mode=2 would provide a bigger difference between higher and lower quantizers than that with aq-mode=1.

Note: the start QP for each frame can be set manually (i.e. via a corresponding txt-file). The txt-file is specified via command line parameter '`-qpfile`'. If the '`-qpfile`' is not specified then start QP and frame type are specified automatically.

In this paper we consider CRF and single pass ABR rate control modes without VBV.

1.1 X265 Presets

The following x265 command-line parameters are important:

`--no-amp` : disable assymetric motion partitions

`--no-rect` : disable rectangle partitions Nx2N and 2NxN

`--no-tskip` : disable transform skip mode

`-b` : maximal number of B-frames

`--b-adapt` : adaptive B-frame scheduling, 0 – disabled.

`-s` : maximal CU-size

`--tu-intra-depth` : maximal depth of transform tree rooted at intra-CU

`--tu-inter-depth` : maximal depth of transform tree rooted at inter-CU

`--rd` : R-D decision level

`--subme` : amount of subpel refinement to perform (0:least .. 7:most)

`--max-merge` : maximum number of merge MV candidates

`--me` : motion search method dia (0), hex (1), …

`--no-signhide` : disable hiding of sign bits

`--no-lft` : disable deblocking

1.1.1 Ultrafast

ultrafast=-s 32 --b-adapt 0 -b 4 --tu-inter-depth=1 --tu-intra-depth=1 --rd 0 --subme=0 --max-merge=1 --me=0 --no-amp --no-rect --no-tskip --early-skip --no-lft --no-sao --no-signhide --no-weightp

1.1.2 Superfast

superfast=-s 32 --b-adapt 0 -b 4 --tu-inter-depth=1 --tu-intra-depth=1 --rd 0 --subme=1 --max-merge=1 --me=1 --no-amp --no-rect --no-tskip --early-skip --no-signhide --no-weightp

1.1.3 Veryfast

veryfast=--b-adapt 0 -b 4 --tu-inter-depth=1 --tu-intra-depth=1 --rd 0 --subme=1 --me=1 --max-merge=2 --no-amp --no-rect --no-tskip --early-skip

1.1.4 Faster

faster=--b-adapt 0 -b 4 --tu-inter-depth=1 --tu-intra-depth=1 --rd 0 --subme=1 --me=1 --max-merge=2 --no-amp --no-rect --no-tskip --early-skip

1.1.5 Fast

fast=--tu-inter-depth=1 --tu-intra-depth=1 --rd 0 --subme=1 --me=1 --max-merge=2 --no-amp --no-rect --no-tskip

1.1.6 Medium

medium=--tu-inter-depth=1 --tu-intra-depth=1 --rd 2 --max-merge=3 --no-amp --no-rect --no-tskip

1.1.7 Slow

slow=--b-adapt 2 -b 4 --tu-inter-depth=1 --tu-intra-depth=1 --rd 2 --max-merge=3 --no-tskip

1.1.8 Slower

slower=--b-adapt 2 --rc-lookahead 20 -b 5 --tu-inter-depth=2 --tu-intra-depth=2 --rd 2 --max-merge=4 --no-tskip --ref 3

1.1.9 Veryslow

veryslow=--b-adapt 2 --rc-lookahead 30 -b 9 --max-merge=5 --ref 5

1.1.10 Placebo

placebo=--b-adapt 2 -b 16 --max-merge=5 --ref 16 --merange 124 --rc-lookahead 60

1.2 x265 Parallelism

x265 supports two schemas of parallelizations: **frame-level** and **WPP**. It's worth mentioning that applying parallel coding might deteriorate coding efficiency (reported that WPP deteriorates the coding efficiency by 1%).

1.2.1 Frame-Level Parallelism

First thread starts the k-th frame, the second thread waits until a several ctu-rows of the k-th frame have been completed. Then the second thread can commence since the search area is already available.

To enable frame-level parallelism you need disable WPP (use `'--no-wpp'`) and apply `'--frame-threads'` (no co-existence of WPP and frame-threads) by setting `'--frame-threads N'`, where N is the number of threads.

Example [2 concurrently encoded frames]:

x265 --input a.yuv --input-res 3840x1744 --fps 24 --b-adapt 0 -b 0 --ref 1 ***--frame-threads 2*** ***--no-wpp*** *--rc-lookahead 2 -o test.h265*

1.2.2 WPP Parallelism

WPP-level (or MB-level) parallelism is realized by means of corresponding built-in tool of HEVC – Wavefront Coding (WPP). To enable WPP parallelism with N threads one needs disable the frame multi-threading (by setting `'--frame-threads 1'`) and to enable WPP `(--wpp)` respectively. Finally to set the number of threads *(--threads* N).

./x265 --input a.yuv --input-res 3840x1744 --fps 24 --b-adapt 0 -b 0 --ref 1
--threads N **--frame-threads 1** --wpp --rc-lookahead 2 -o test1.h265yuv

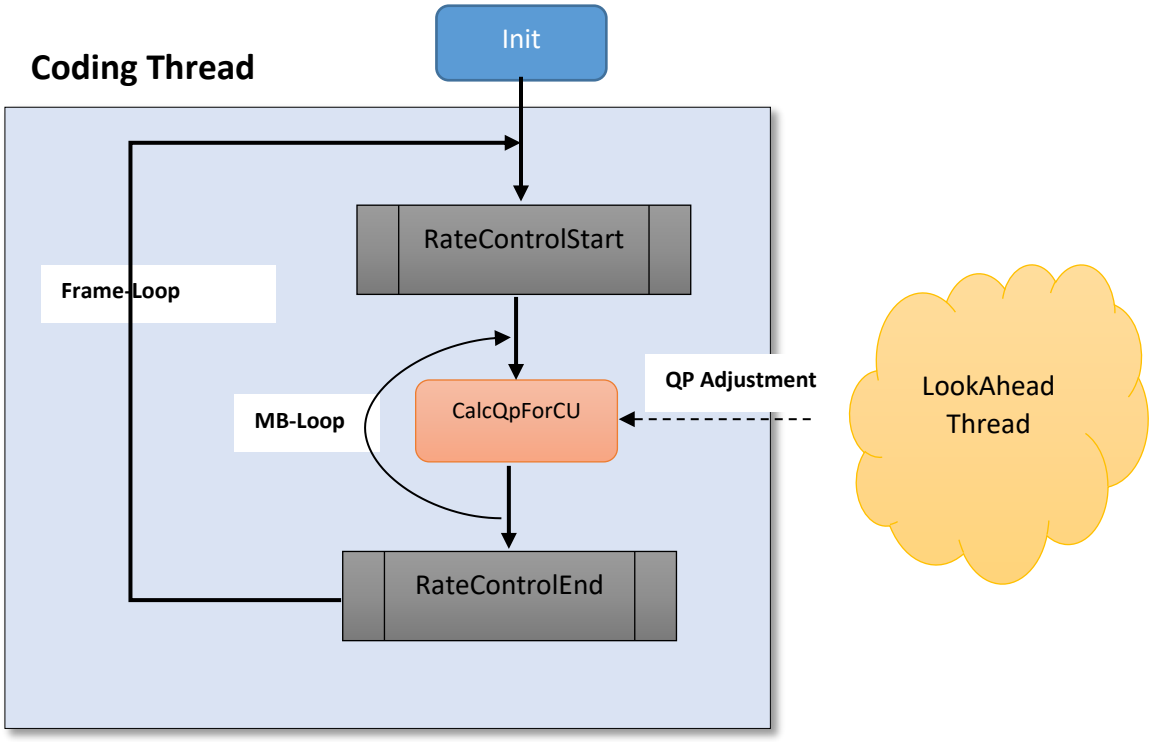
2 Rate Control: Coding Thread

x265 consists of two main threads: **coding thread** and **look-ahead** (**preprocessing**) thread (in low latency mode look-ahead is disabled).

Notice the **look-ahead thread** is always ahead of the coding thread by N frames, where N is specified by '`--rc-lookahead N`'.

The **look-ahead thread** gathers frame statistics for each picture and builds QP-adjustment map for RC (if adaptive quantization is on). In addition, Motion Estimation is performed on decimated frame to provide hints to real motion estimation in **Coding Thread**.

Rate Control is performed in the **coding thread** and it consists of two levels: frame-level and CTU (or MB) level:



2.1 Sequence-Level RateControl Init

At the sequence level **RateControl Init** is invoked once, at the beginning of encoding. This function performs the global resources allocations and parameters initialization, selected operations of Sequence Level **RateControl Init**:

- Specify the scaled picture size (for lookahead): *scaled[W|H] = ([Width|Height] / 2 + 7) >> 3*
- In case of CRF the particular constant parameter **RateFactor** is computed as follows:

$$RateFactor = \frac{1}{0.85 * 2^{(crf + 5.4 - 12)/6}}$$

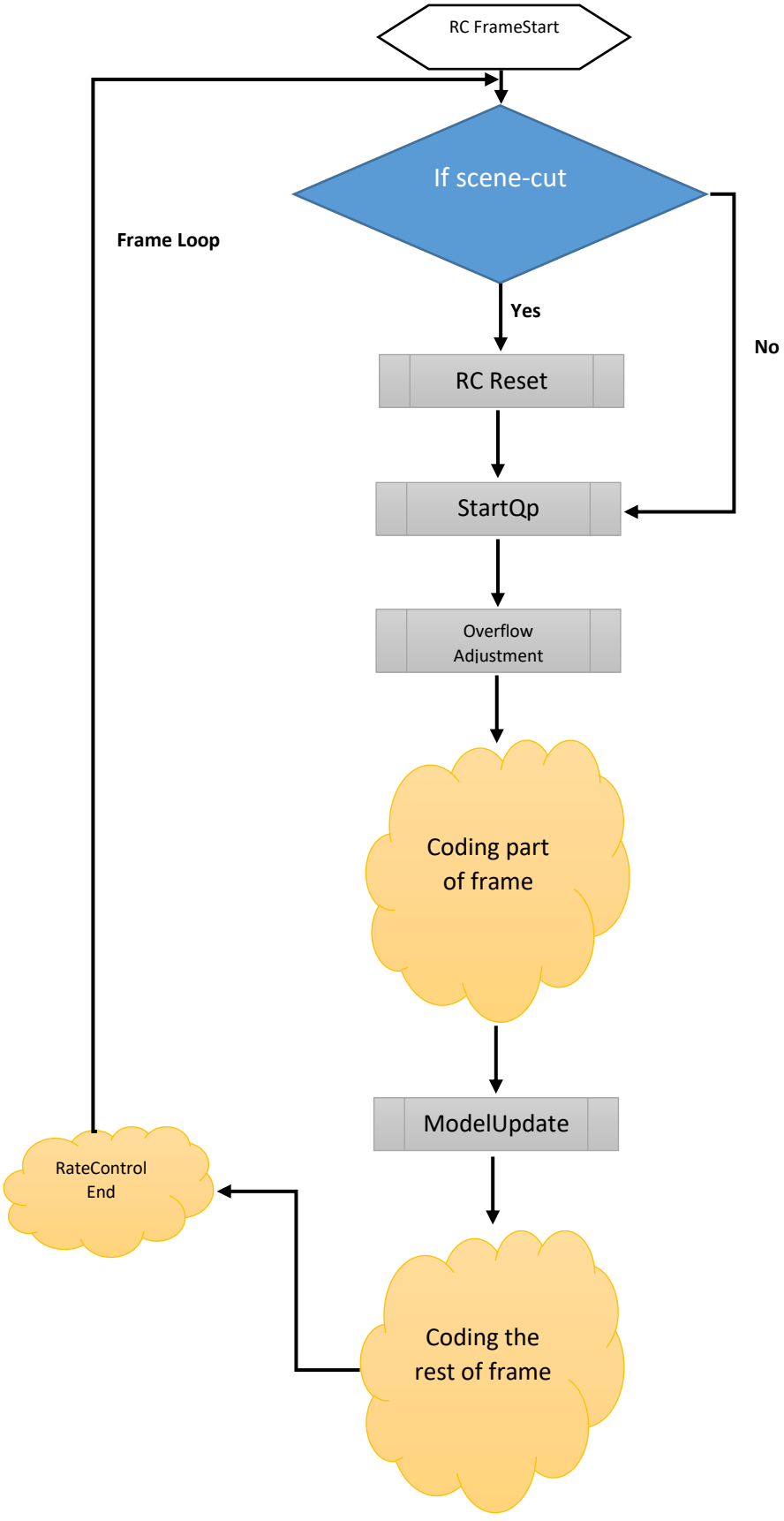
where '*crf*' is a number in command line (followed by '--crf')

- Rate Control Parameters Init
- VBV Init

2.2 Frame-Level RateControlStart

The function **RateControlStart** is invoked once per frame (at the start) and consists of three functional blocks:

- **Check RC Reset Conditions**
- **FrameStartQp** - compute the start QP unless qpfile mode is on
- **Bitrate Overflow Correction**
- **ModelUpdate** – update of RC model parameters


RC FrameStart
If scene-cut
Frame Loop
Yes
No
RC Reset
StartQp
Overflow
Adjustment
Coding part
of frame
ModelUpdate
RateControl
End
Coding the
rest of frame

2.2.1 Check RC Reset

Rate Control is reset at the scene cut.

If the following condition is met then RC is reset (i.e. current total satdCost is a 4xtimes peak, this is indication of a scene cut):

[Scene-cut condition]	*Current frame satdCost > 4 * moving average of satdCosts*

Update of moving average of satdCosts

2.2.2 FrameStartQp

The function **FrameStartQp** is invoked at the start of each frame and it's tailored to calculate the start QP which would have produced the desired bit-size if it had been applied to all frames from the very beginning or from the last RC reset.

2.2.2.1 ABR mode: I/P Frame Start QP

For ABR mode the R-D curve is approximated by the rule:

$$qscale = C*X/Bits$$

where

- **X** - called complexity and is actually approximated basing on previous results
- **Bits** - stands for target bit-size of the frame.
- ***C*** – constant specified as (0.04*fps)^0.4 , a magic number pulled out thin air.

For I/P frames we firstly determine ***qscale*** = ***C*X/Bits*** approximation and then convert ***qscale*** to QP units as follows:

$$startQP = 12.0 + 6.0*log2(qscale/0.85)$$

For B frames the start QP is determined in a different way, pls. see the section below.

2.2.2.2 B-Frame Start QP

In case of B-frames in both CRF and ABR modes the initial QP is obtained by blending of avgQPs of forward and backward references (weighted by poc differences) + ***m_pbOffset***, where *m_pbOffset* is determined as follows:

```
m_pbOffset = 6.0 * LOG2(pbratio)

If the frame is used for reference then

    m_pbOffset = m_pbOffset/2
```

pbratio is specified by the command-line parameter `'--pbratio'`

2.2.2.3 CRF Mode: I/P-Frame

For I/P frames the initial *qscale* is computed constantly as follows:

*qscale = (0.04*fps)^0.4 / RateFactor*

where **RateFactor** is specified as:

$$RateFactor = \frac{1}{0.85 * 2^{(crf + 5.4 - 12)/6}}$$

Then **qscale** is converted to start QP:

*startQP = 12.0 + 6.0*log2(qscale/0.85)*

2.2.1 ModelUpdate

The complexity **X** is updated once per frame but not at the start of a frame, actually at the middle of a frame.

At the very beginning period of encoding (= '*2*frame_rate'* frames) the model parameter **X** is updated after the half of frame completed (precisely at the start of '*(m_numRows + 1) / 2'* row).

After the initial period the model parameter **X** is modified after each *'m_refLagRows'* have been completed.

Notice *m_refLagRows* takes usually small magnitudes (since search area vertical size is usually several CTUs), i.e. RC is updated after several ctu rows (almost at the start) of each frame.

m_refLagRows = 1 + (search_range + 63) / 64

2.3 CalcqpForCU

QP-adj map is specified in the **Look-ahead thread** for each 32x32 block (in the original resolution). However, QP-delta is signaled per 64x64 block. Therefore the average of four QP-adjustments (comprising the current 64x64 CTU) serves as QP-adjustment of the current CTU.

2.4 RateControlEnd (Frame-Level)

At the end of each frame the function **rateControlEnd** is called. This function performs the following operations:

- Update/Compute statistics
- Amortization of I-frame

2.4.1 Update/Compute Statistics

The following statistics are updated/computed:

- The average Qp (after HVS QP-adjustment), stored in *m_avgQpAq*
- Update *m_wantedBitsWindow* by *bitrate*frameDuration*
- Update accumulated complexity *m_cplxrSum* by '*bits*avgQp'*
- Update m_totalBits

2.4.2 Amortization of I-frame

To prevent from RC to make "panic" decision due to I-frame peaks, we cheat RC – instead of providing the actual I-frame bit-size, the reduced (faked) I-frame bit-size is fed to Rate Control and not-declared "hidden" bits are amortized afterwards among the following inter frames.

At the start of each I-frame the number of amortizing frames is determined:

m_residualFrames = min(75, keyframeMax)

Then the delta in bits (*m_residualCost*) to be added to each amortizing frame is specified:

*m_residualCost = (int)((bits *0.85) / m_residualFrames) // 85% of i-frame size is not declared!!!*

Then the actual bit-size (*bits*) is reduced:

*bits -= m_residualCost * m_residualFrames*

In other words, the hidden size is *m_residualCost * m_residualFrames* and the faked bit-size (*bits*) is fed to RC.

At the start of non-I frame the hidden bits are added if *m_residualFrames!=0*

bits + = m_residualCost

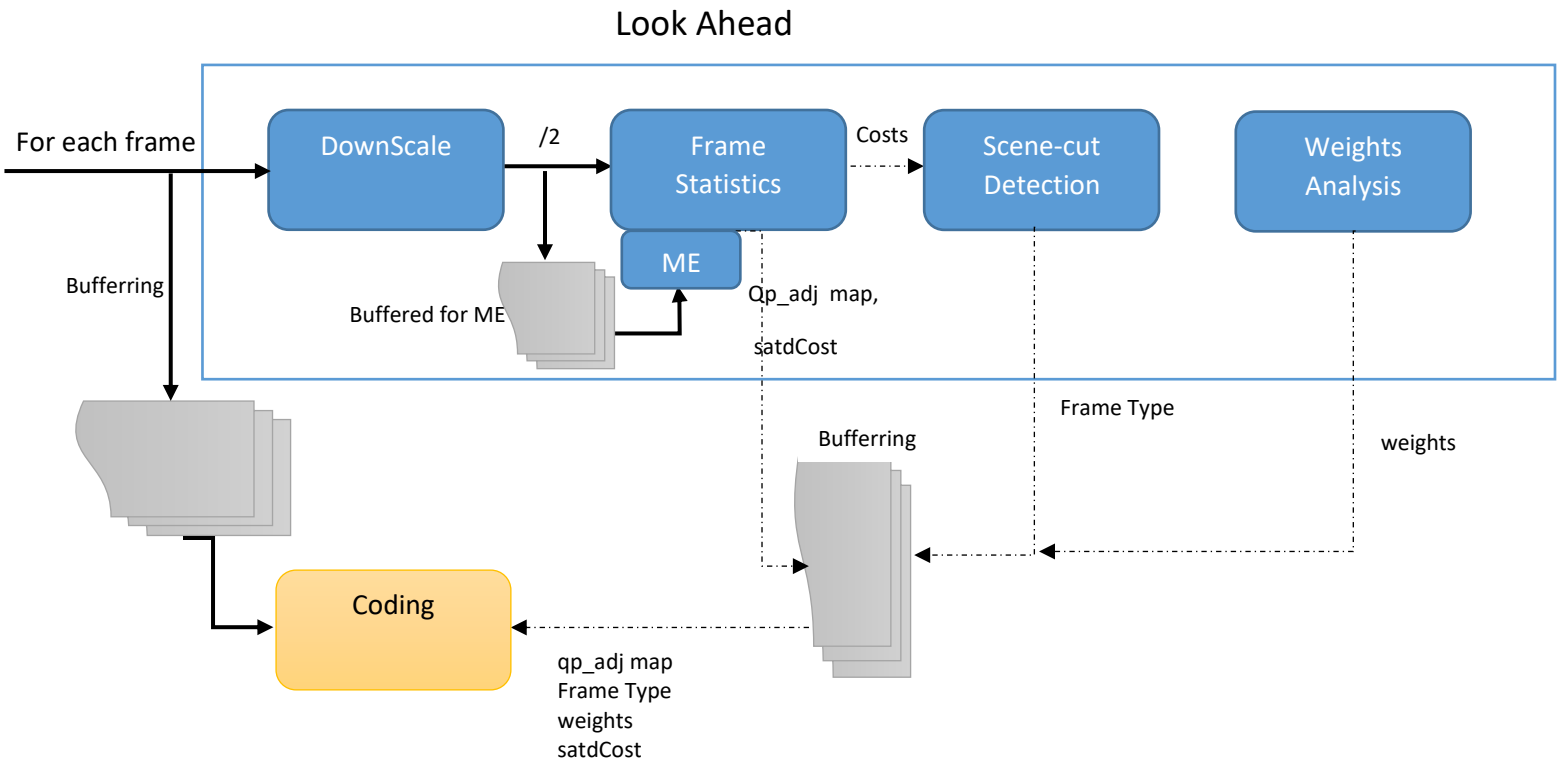
m_residualFrames -= 1

3 Rate Control: Look-ahead Thread

Look-ahead consists off the following main stages applied to each frame prior to the coding:

- Downscale the frame (by the factor two)
- Compute frame statistics (total satdCost etc.) on scaled frames.
- Determine qp_offset (or qp-adjustment) for each 16x16 scaled block (= 32x32 in the original resolution), if adaptive quantization (*rc.aqMode!=0)* is on.
- Scene-cut Detection on scaled frames for adaptive I-frame placement
- Weights Analysis on scaled frames if the weighted prediction (**--weightp)** enabled

The look-ahead process is performed in a separate thread in reverse-scan order within a frame (unlike to coding).


Look Ahead
For each frame
DownScale
/2
Frame Statistics
Costs
Scene-cut Detection
Weights Analysis
ME
Bufferring
Buffered for ME
Qp_adj map,
satdCost
Frame Type
weights
Bufferring
Coding
qp_adj map
Frame Type
weights
satdCost

3.1 Frame Statistics

3.1.1 CTU Cost (SATD-based)

For each 16x16 scaled block (CU) the residual cost is computed and stored at *lowresCosts* array.

The following operations (for each lowres CU) are executed to compute lowresCost:

1. [Inter-Cost] Perform motion estimation with the search area 16x16 on previous scaled frame and return 'mcost', where mcost= SATD + λ*R(MV)

2. [Intra-Cost] For all 35 intra modes, select the smallest SATD cost – 'icost'
    a. Disfavor intra-cost: *icost = icost + 9*
    b. Store *icost* in *intraCost* array

3. lowresCost = mcost

4. [Decision] if icost<mcost then
    a. lowresCost = icost and m_intraMBs++

5. lowresCosts[cu_num] = lowrescost

Notes:

- At the stage (2) the best intra mode is ignored (because we are in look-ahead thread), only minimal cost is exploited.
- In order to exploit MV prediction (remind that MV is also estimated) the 16x16 blocks are traversed in reverse order (from right to left, from bottom to top).
- In addition to lowrescost, *icost* and *m_intraMbs* are further used.

3.1.2 QP-adj Map

For each 16x16 scaled block qp_adj (or *qp_offset)* is determined basing on the block energy, the sum of 16x16 luma and two 8x8 chroma variances is called 'energy'. The energy is computed by the utility 'acEnergyCu(pic, block_x, block_y)'.

qp_adj formula:

qp_adj = (energy+1)^0.1

The qp adjustment is stored in the buffer *qpCuTreeOffset* to be used lately (during encoding).

Note:

- qp_adj is monotonically increasing function of the energy, the more energy the coarser quantization.
- Because *qp_delta* signaling granularity is 64x64 and qp_adj is specified for each 32x32 block (in the original resolution), the average of *qp_offsets* of all 32x32 blocks comprising a given 64x64 CTU is taken as the final qp-adjustment for the 64x64 CTU.

3.1.3 Frame-Level

For each frame two costs are calculated: *icost* (sum of intra costs) and *pcost* (sum of inter residuals). These costs are further used in scene-cut detection.

3.2 Scene Detection

The purpose of scene-cut detection - adaptive placement of I-frames if the new scene is detected.

Scene detection is performed on the sliding window of successive frames. The size of the window equals to look-ahead depth, by default the look-ahead depth is 20 frames.

3.2.1 Flashlights Recognition

To minimize false detection rate, the scene detection mechanism contains a sort of flashlights discernment: if the previous scene-cut has been recognized *(num_bframes+1)* frames ago then the current scene detection is flashlight and false alarm.

However, flashlight-detection is applied only in case *num_bframes>0*. So, for IP-only streams flashlights might cause false scene-cut detections and as a result plenty irrelevant I-frame placements. It's worth to modify this code.

3.2.2 Scene-cut Detection Bias

In the command line the scene-cut detection can be disabled by '*--no-scenecut*' or '--scenecut 0', by default scenecut enabled.

According to the position of current frame in GOP (*curGopPos*), the bias is determined as follows:

```
threshMax = scenecutThreshold / 100.0
threshMin = threshMax * 0.25    /* magic numbers pulled out of thin air */
  if ( keyframeMin == keyframeMax )
      threshMin = threshMax
  if ( curGopPos <= keyframeMin / 4)
       bias = threshMin / 4
  else if (curGopPos <= keyframeMin)
      bias = threshMin * curGopPos / keyframeMin
  else
  {
      bias = threshMin  + (threshMax - threshMin)*( curGopPos - keyframeMin) / (keyframeMax - keyframeMin)
  }
```

Here

- *scenecutThreshold* – specified in the command line `'--scenecut X'`, default 40
- *keyframeMin* – specified by `'--min-keyint'` in the command line and denotes the minimal GOP size (relevant if I-frame placement at scene-cuts enabled).
- *keyframeMax* – specified by `--keyint`, equal to GOP size.

Let's consider the common case when GOP size (GopSize) is fixed, i.e. *keyframeMin = keyframeMax* .

```
if  the current frame is at the first quarter of GOP then
      bias= threshMax/4
else
    bias = threshMax * curGopPos / GopSize
```

Note:

The bias for frames in the first quarter of GOP is less than that in the rest of GOP.

3.2.3 Scene-cut Detection Mechanism

To detect scene-cut *icost* (sum of intra costs) and *pcost* (sum of inter residuals) are compared as follows:

If pcost/icost >= 1.0 - bias then

Scene-cut detected

Note:

The maximal magnitude of the bias depending on *scenecutThreshold* – the greater *scenecutThreshold* the greater *bias* . Hence, the fewer *pcost/icost* ratio is required to declare scene-cut.