Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG
Hong Kong, Jan, 2005

Document JVT-N046
File: JVT-N046r1.sxw
Generated: 2005-02-19

Title: **Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods**

Status: Approved Output Document

Purpose: Draft of Joint Model

Author(s) or Contact (s):

Keng-Pang Lim
Institute for Infocomm Research
20, Heng Mui Keng Terrace
Singapore 119613
Tel: +65 6874 4303
Fax: +65 6774 4998
Email: kplim@i2r.a-star.edu.sg

Gary Sullivan
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
Tel: +1 (425) 703-5308
Fax: +1 (425) 706-7329
Email: garysull@microsoft.com

Thomas Wiegand
Heinrich Hertz Institute (FhG),
Einsteinufer 37, D-10587 Berlin,
Germany
Tel: +49 - 30 - 31002 617
Fax: +49 - 30 - 392 72 00
Email: wiegand@hhi.de

Source: Editor

DRAFT INTERNATIONAL STANDARD
STUDY OF ISO/IEC 14496-10 and ISO/IEC 14496-5 / AMD6
STUDY OF ITU-T Rec. H.264 and ITU-T Rec. H.2.64.2
DRAFT ITU-T RECOMMENDATION

TABLE OF CONTENTS

LIST OF FIGURES

1

2 LIST OF TABLES

3 Foreword

Reference software and descriptions of reference encoding methods and non-normative reference decoding error concealment methods are useful in aiding users of a video coding standard to establish and test conformance and interoperability, and to educate users and demonstrate the capabilities of the standard. In this spirit, the methods described herein and the accompanying software modules are provided for implementers of ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10 advanced video coding.

4 Introduction

ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10 advanced video coding was developed as a joint project of the ITU-T SG16 Q.6 Video Coding Experts Group (VCEG) and the ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group (MPEG), with completion of the text of the video coding standard in 2003. Reference software was also developed jointly for approval soon thereafter, and the reference software accompanies this document. In addition to implementing the normative decoding process specified in the text of ITU-T Rec. H.264 | ISO/IEC 14496-10, software is provided to demonstrate effective (non-normative) encoding techniques and (non-normative) decoding error concealment techniques for use with the standard. These non-normative aspects are described in this document.

JOINT MODEL FOR NON-NORMATIVE ASPECTS OF ADVANCED VIDEO CODING

5 Scope

This document specifies non-normative reference encoding methods and methods of concealing errors and losses in decoders for video data conforming to ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10 advanced video coding.

6 Non-normative example encoding methods

1.1 Motion estimation and mode decision

6.1.1 Concepts and Techniques

In this section we introduce background information and techniques needed to determine which of the many possible modes to use for encoding a macroblock. The following are explained in the following sections:

a) The Lagrangian Cost

b) The 7 Inter modes

c) λ_{MODE} and λ_{MOTION}

d) SAD, the 4x4 Hadamard Transform, SATD, SA(T)D, and SSD

6.1.1.1 The Lagrangian Cost

Unlike other video coding standards, H.264 has many different Intra and Inter mode choices to code a macroblock. To choose the best mode, it is recommended to use the Lagrangian multiplier to compute the cost for each mode and select the mode that gives the smallest cost. Therefore, the mode decision process minimizes the following Lagrangian cost, *J*:

***J* = Distortion +** λ_{MODE} *** Rate**

The **Distortion** measurement quantifies the quality of the reconstructed pictures while the **Rate** measures the bits needed to code the macroblock using the particular mode. The best mode will be the one that gives small distortion and less bits to code the macroblock. Both factors are simultaneously quantified by the Lagrangian multiplier, λ_{MODE}

The above rate-distortion optimisation method is also used for choosing the best reference frame, motion vector and direction of prediction in B slice by using different lambda values and distortion measurement.

6.1.1.2 The 7 Inter mode partitions

Note that the Inter modes are related to the partitioning possibilities of a 16x16 luma macroblock as designated in the following list: (partition width x partition height)

Inter mode 1: 16 x16 (one partition)

Inter mode 2: 16 x 8 (two partitions)

Inter mode 3: 8 x16 (two partitions)

Inter mode 4: 8 x 8 (four partitions)

Inter mode 5: 8 x 4 (an 8x8 partition with two sub-partitions)

Inter mode 6: 4 x 8 (an 8x8 partition with two sub-partitions)

Inter mode 7: 4 x 4 (an 8x8 partition with four sub-partitions)

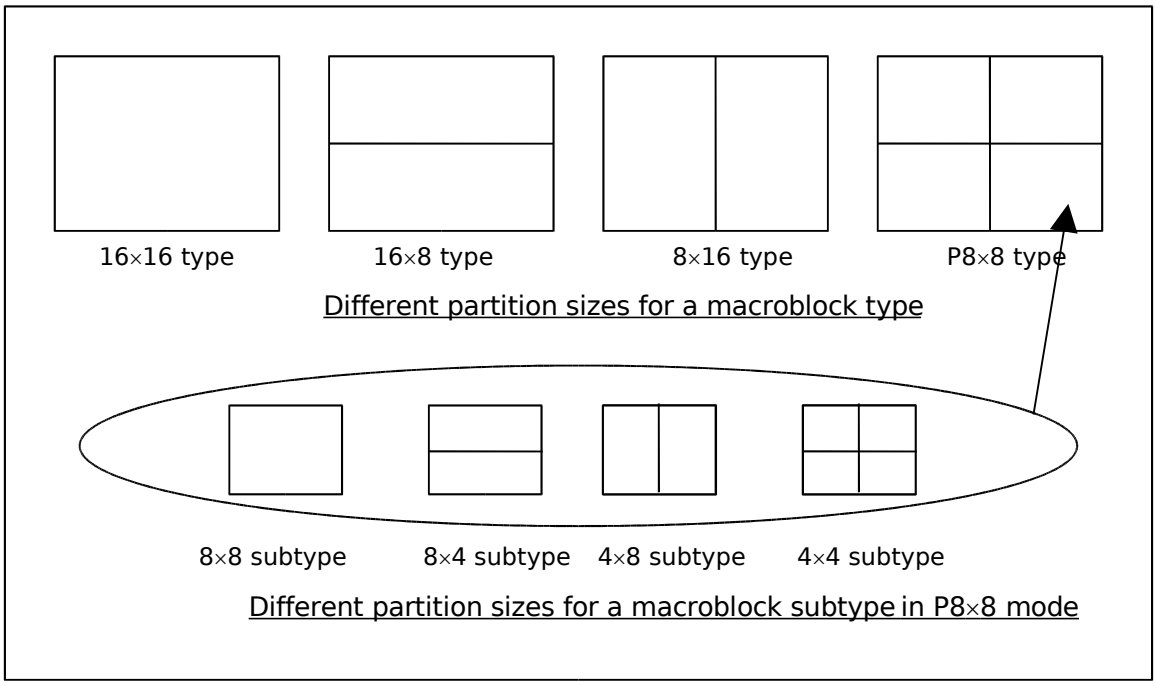


Figure. 2-6.1 Different partition sizes in a macroblock

When mode 4 is considered, then modes 5, 6, and 7 must be considered for each of the four individual
8x8 sub- macroblocks. Note that each partition has its own unique motion vector for the use of motion-compensated prediction of the partition.

6.1.1.3 Determining λ_{MODE} and λ_{MOTION}.

The Lagrangian multiplier λ_{MODE} is given by

$$\lambda_{MODE,I,P} = 0.85 \times 2^{(QP-12)/3} \text{ for Intra mode testing and P slice Inter mode testing} \quad (2\text{-}6.1)$$

$$\lambda_{MODE,B} = \max\left(2, \min(4, \frac{QP-12}{6})\right) \times \lambda_{MODE,I,P} \quad (2\text{-}6.2)$$

for B slice testing, where *QP* is the macroblock quantisation parameter.

λ_{MOTION} is used in computing the motion vector in P or B slices. The motion vector cost is computed by:

J = Distortion + λ_{MOTION} *** Rate,** then ...

When Distortion is computed using SSD, … $\lambda_{MOTION} = \lambda_{MODE}$ (2-6.3)

When Distortion is computed using SAD, or SA(T)D, … $\lambda_{MOTION} = \sqrt{\lambda_{MODE}}$ (2-6.4)

where the distortion measurements, SSD, SAD and SA(T)D are explained in the following section:

6.1.1.4 SAD, Hadamard Transform, SATD, SA(T)D, SSD

For any given block of pixels, the difference between the original and prediction values at location (i,j) is given by the expression

$$Diff(i,j) = \text{Original}(i,j) - \text{Prediction}(i,j) \quad (2\text{-}6.5)$$

SAD (Sum of Absolute Differences) is computed using the following equation:

$$SAD = \sum_{i,j} |Diff(i,j)| \quad (2\text{-}6.6)$$

Since ultimately the transformed coefficients are coded, we can achieve a better estimation of the cost of each mode by estimating the effect of the DCT with the 4x4 Hadamard transform. This technique is used when it may be too computationally intensive to perform complete RDO decisions using the DCT and IDCT.

The 4x4 Hadamard transform is illustrated in Figure 2-6.2 below. (not normalized):

H =

1	1	1	1
1	1	-1	-1
1	-1	-1	1
1	-1	1	-1

Figure 2-6.2 Hadamard Transform

Note that since **H** is a symmetric matrix, it is equal to its own transpose. So the Hadamard transform of a 4x4 block of difference (error) values, *Diff*, obtained by applying equation (2-6.5) to a 4x4 block, is given by multiplying *Diff* by **H** on both the left and right ...

$$DiffT = \text{H} * Diff * \text{H} \quad (2\text{-}6.7)$$

Then SATD (Sum of Absolute Transformed Differences) is computed using the following equation:

$$SATD = (\sum_{i,j} |DiffT(i,j)|) / 2 \quad (2\text{-}6.8)$$

In some places in the encoder, a Hadamard transform flag can be turned ON or OFF. **SA(T)D** refers to either **SAD** or **SATD** depending on the status of the Hadamard transform flag.

In the JM software, **SAD** is used when computing full-pel motion estimation while **SATD** is used for sub-pel motion estimation.

Finally, **SSD** is the sum of the squared differences between an original block *s* and its reconstruction *c* given by

$$SSD = \sum_{i,j} \left[s(i,j) - c(i,j) \right]^2 \qquad (2\text{-}6.9)$$

where *c*(i,j) is the reconstructed pixel value after transform, quantisation, inverse quantisation and inverse transform.

6.1.2 High-complexity mode

This is the most time consuming configuration of the encoder. The Rate Distortion Optimisation (RDO) option in the configuration file is ON and exhaustive search is used. The reconstructed video quality is the best.

6.1.2.0.1 Exhaustive Full-Pel Motion Search

The JM software uses spiral full search in high complexity mode. The full-pel search positions are organized in a spiral structure around a prediction vector. The full search range given by MC_range is used for all INTER-modes and reference frames. To speed up the search process, the prediction vector of the 16x16 block is used as the center of the spiral search for all INTER-modes.

In the JM implementation, the SAD values for 4x4 blocks are pre-calculated for all motion vectors of the search range and then used for fast SAD calculation for all larger blocks.

The prediction vector of the 16x16 block is determined using the technique described in the standard document (ISO/IEC International Standard ISO/IEC 14496-10 advanced video coding) in section 8.4.1.3.1. For the purpose of this document, denote this prediction vector as **p**.

The exhaustive full-pel motion search is computed by minimising the following equation for all full-pel positions in the search region.

$$J(\mathbf{m}, \lambda_{MOTION}) = SAD(s, c(\mathbf{m})) + \lambda_{MOTION} \cdot R(\mathbf{m} - \mathbf{p}) \qquad (2\text{-}6.10)$$

with $\mathbf{m} = (m_x, m_y)^T$ being the motion vector, $\mathbf{p} = (p_x, p_y)^T$ being the prediction for the motion vector, and λ_{MOTION} being the Lagrange multiplier. The rate term $R(\mathbf{m} - \mathbf{p})$ represents the predicted motion vector error information only and is computed by a table-lookup. In the JM implementation, the rate is estimated by using the universal variable length code (UVLC) table, even if CABAC is used.

For a M x N block, SAD is computed as

$$SAD(s, c(\mathbf{m})) = \sum_{x=1}^{M} \sum_{y=1}^{N} \left| s(x,y) - c(x - m_x, y - m_y) \right| \qquad (2\text{-}6.11)$$

with s being the original video block and note that *c* is now the predicted video block at the position designated by **m** in the reference picture considered.

6.1.2.0.2 Adaptive Motion Search Range

Motion search range decision plays as a role of a pre-processing step of a motion estimation to reduce encoder complexity. Search range is differently determined at each macro-block by using

its neighboring information.

Let assume that E is a macro-block under motion estimation, and A, B, C are its neighboring 4×4 blocks, as shown in Fig. 1. Also, assume that motion vectors of A, B, and C are (MV_A_x, MV_A_y), (MV_B_x, MV_B_y), and (MV_C_x, MV_C_y), and input search range is input_search_range. The first step of the motion search range is to estimate the local maximum vector of E. It is defined as

$$\begin{aligned} \text{max\_MV\_E\_x} &= \max(\text{abs}(\text{MV\_A\_x}), \max(\text{abs}(\text{MV\_B\_x}), \text{abs}(\text{MV\_C\_x}))) \\ \text{max\_MV\_E\_y} &= \max(\text{abs}(\text{MV\_A\_y}), \max(\text{abs}(\text{MV\_B\_y}), \text{abs}(\text{MV\_C\_y}))) \end{aligned} \quad (2\text{-}6.12)$$

In the above equation, if a neighboring block is outside of image, the motion vector is replaced by input search range. The second step is to determine temporal search range by the local statistics as

$$\begin{aligned} \text{local\_search\_range\_x} &= \max(\text{k\_x}, 2 \times \text{max\_MV\_E\_x}) \\ \text{local\_search\_range\_y} &= \max(\text{k\_y}, 2 \times \text{max\_MV\_E\_y}) \end{aligned} \quad (2\text{-}6.13)$$

where k_x and k_y are determined as

$$\begin{aligned} \text{k\_i} &= \begin{cases} (\text{input\_search\_range} + 2)/4 & \text{if } \alpha_i \geq 2 \\ (\text{input\_search\_range} + 4)/8 & \text{otherwise} \end{cases} \\ \alpha_i &= \text{abs}(\text{MV\_A\_i}) + \text{abs}(\text{MV\_B\_i}) + \text{abs}(\text{MV\_C\_i}) \quad \text{for } i = x, y \end{aligned} \quad (2\text{-}6.14)$$

The maximum search range is subject to input search range, and therefore, the motion search range is constrained as

$$\begin{aligned} \text{new\_search\_range\_x} &= \min(\text{input\_search\_range}, \text{local\_search\_range\_x}) \\ \text{new\_search\_range\_y} &= \min(\text{input\_search\_range}, \text{local\_search\_range\_y}) \end{aligned} \quad (2\text{-}6.15)$$

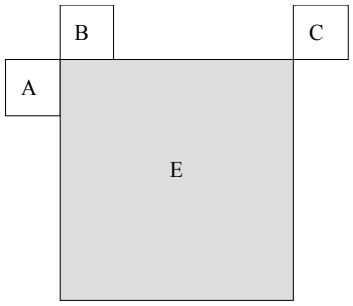


Figure 2-6.3 – Reference block location for motion search range

6.1.2.0.3 Fractional Sample Search

In the sub-sample search, the distortion measures SATD is calculated after the 4x4 Hadamard transform is applied as described in section 6.1.1.4. The Lagrangian multiplier λ_{MOTION} is given by

$$\lambda_{MOTION,I,P} = \sqrt{0.85 \times 2^{(QP-12)/3}} \quad \text{for I and P slices} \quad (2\text{-}6.16)$$

and

$$\lambda_{MOTION,B} = \sqrt{\max\left(2, \min(4, \frac{QP-12}{6})\right) \times \lambda_{MOTION,I,P}} \quad \text{for B slice} \quad (2\text{-}6.17)$$

where *QP* is the macroblock quantisation parameter.

Fractional sample search is performed in two steps. This is illustrated in Figure 2-6.4 where capital letters represent integer positions, numbers represent ½ sample positions and lower case letters represent ¼ sample positions.

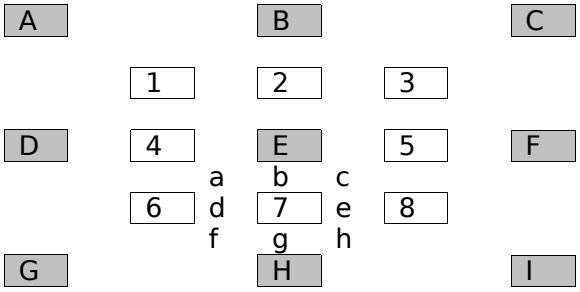


Figure 2-6.4 – Fractional sample search positions

Assume that the integer search points to position E. Then ½ sample positions 1, 2, 3, 4, 5, 6, 7, 8 are searched. Assume that 7 is the best position. Then the ¼ sample positions a, b, c, d, e, f, g, h are searched. If motion compensation with 1/8 sample accuracy is used, an additional sub-sample refinement step is performed in the described way. After fractional sample search has been performed for the complete macroblock, the SATD for the whole macroblock is computed.

Note: Hadamard_on_flag is always set to true for exhaustive high-complexity mode.

6.1.2.0.4 Finding the Best Reference Frame

The determination of the reference frame REF and the associated motion vectors for the various Inter modes is done after motion estimation wrt all Ref frames by minimizing

$$J(REF \mid \lambda_{MOTION}) = SATD(s, c(REF, \mathbf{m}(REF))) + \lambda_{MOTION} \cdot (R(\mathbf{m}(REF) - \mathbf{p}(REF)) + R(REF)) \quad (2\text{-}6.18)$$

The rate term $R(REF)$ represents the number of bits associated with choosing *REF* and is computed by table-lookup using UVLC.

6.1.2.0.5 Finding the Best Prediction Direction for B slice.

Each partition in an INTER mode B slice macroblock can be predicted from one or two reference frames. The reference frame can be in the future or from the past reconstructed frames. If the reference frame is in the future, it is called backward prediction whereas forward prediction refers to using past reference frame for prediction. It can have bi-directional prediction where both the two references are weighted and used together.

The determination of the prediction direction PDIR for the various Inter modes in B slices is done after motion estimation and reference frame decision by minimizing

$$\begin{aligned} J(PDIR \mid \lambda_{MOTION}) = {} & SATD(s, c(PDIR, \mathbf{m}(PDIR))) + \\ & \lambda_{MOTION} \times (R(\mathbf{m}(PDIR) - \mathbf{p}(PDIR)) + R(REF(PDIR))) \end{aligned} \quad (2\text{-}6.19)$$

6.1.2.1 The Algorithmic Steps to Encode a Macroblock

The procedure to encode one macroblock, s, of an I, P, or B slice, in the high-complexity mode uses the methods described in section 2.1.1.1 and is summarized as follows:

a) Perform motion estimation and reference frame selection for Inter modes 4, 5, 6, and 7 for each of the four 8x8 sub-macroblocks.

b) Perform motion estimation and reference frame selection for Inter modes 1, 2, and 3.

c) Choose the best combination of Intra modes

d) Use the results of a), b), and c) to determine the prediction mode for the current macroblock.

6.1.2.1.1 Determining the Best Combination of Intra Modes

The best Intra mode is selected by choosing the smaller cost computed in the best Intra16x16 and Intra 4x4 mode.

6.1.2.1.1.1 Selecting the Best *Intra4x4* Prediction Mode

For Intra4x4 prediction, the mode decision for each of the sixteen 4x4 subblocks is performed by minimizing

$$J(s,c,IMODE \mid QP,\lambda_{MODE}) = SSD(s,c,IMODE \mid QP) + \lambda_{MODE} \cdot R(s,c,IMODE \mid QP) \qquad (2\text{-}6.20)$$

$$IMODE \in \{\text{all nine Intra4x4 modes}\}$$

where QP is the macroblock quantiser, λ_{MODE} is the Lagrange multiplier for mode decision, and $IMODE$ indicates one of the nine Intra4x4 prediction modes.

SSD is the sum of the squared differences between the original 4x4 block luminance signal *s* and its reconstruction *c*, and $R(s,c,IMODE \mid QP)$ represents the number of bits associated with choosing *IMODE*. It includes the bits for the intra prediction mode and the transformed-coefficients for the 4x4 luminance block. The rate term is computed using the UVLC entropy coding, even if CABAC is used for entropy coding.

6.1.2.1.1.2 Selecting the best Intra16x16 Prediction Mode

As contrary to the rate distortion optimisation method for Intra4x4 mode determination, the best Intra16x16 prediction mode is determined by choosing the mode that results in the minimum SATD. The cost for each of the four Intra16x16 prediction modes is given by the following expression

$$SATD(s,c,IMODE) \qquad (2\text{-}6.21)$$

$$IMODE \in \{\text{all four Intra16x16 modes}\}$$

6.1.2.1.1.3 Lagrangian Cost for Intra Chroma Mode

Similarly, the best mode for the four Intra Chroma mode is given by minimizing the IntraChroma cost, J_{ch}

$$J_{ch}(s,c,IMODE \mid QP,\lambda_{MODE}) = SSD(s,c,IMODE \mid QP) + \lambda_{MODE} \cdot R(s,c,IMODE \mid QP) \qquad (2\text{-}6.22)$$

$$IMODE \in \{\text{all four chroma modes}\}$$

where s now represents the original U or V chrominance values of the 8x8 chroma macroblock and $R(s,c,IMODE \mid QP)$, in this case also includes only the bits for the DCT-coefficients.

6.1.2.1.1.4Lagrangian Cost for Intra Mode

The cost for the best Intra mode is computed using the following:

$$J(s,c,MODE \mid QP,\lambda_{MODE}) = SSD(s,c,MODE \mid QP) + \lambda_{MODE} \cdot R(s,c,MODE \mid QP) \quad (2\text{-}6.23)$$

SSD is the sum of the squared differences between the original block *s* and its reconstruction *c* given as

$$SSD(s,c,MODE \mid QP) = \sum_{x=1,y=1}^{16,16} \left(s_Y[x,y] - c_Y[x,y,MODE \mid QP]\right)^2 + \sum_{x=1,y=1}^{8,8} \left(s_U[x,y] - c_U[x,y,MODE \mid QP]\right)^2 + \sum_{x=1,y=1}^{8,8} \left(s_V[x,y] - c_V[x,y,MODE \mid QP]\right)^2, \quad (2\text{-}6.24)$$

and $R(s,c,MODE \mid QP)$ is the number of bits associated with choosing the best intra mode currently selected for the macroblock and includes the bits for the macroblock header and the bits for the transformed Y, U and V blocks.

$c_Y[x,y,MODE \mid QP]$ and $s_Y[x,y]$ represent the reconstructed and original luminance values; c_u, c_v, and s_u, s_v the corresponding chrominance values.

6.1.2.1.2 Determining the Best Inter Modes for each sub macroblock

First, compute the motion vector for each of the sub-macroblocks Then, the best sub mode for each 8x8 sub-macroblock is done by minimizing the Lagrangian function

$$J(s,c,MODE \mid QP,\lambda_{MODE}) = SSD(s,c,MODE \mid QP) + \lambda_{MODE} \cdot R(s,c,MODE \mid QP) \quad (2\text{-}6.25)$$

where *QP* is the macroblock quantisation parameter, λ_{MODE} is the Lagrange multiplier for mode decision, and *MODE* indicates a mode chosen from the set of potential prediction modes:

P slice: $$MODE \in [8x8(4),8x4(5),4x8(6),4x4(7)] \quad (2\text{-}6.26)$$

B slice: $$MODE \in [Direct,8x8(4),8x4(5),4x8(6),4x4(7)] \quad (2\text{-}6.27)$$

SSD is the sum of the squared differences between the original block *s* and its reconstruction *c* and $R(s,c,MODE \mid QP)$ is the number of bits associated with choosing *MODE*, including the bits for the macroblock header, the motion vector and reference picture, and the transformed blocks. Note that the number in the parenthesis indicate the number corresponding to the mode used in the reference software.

The cost for each sub block is then sum together to give the Lagrangian cost of the P8x8 mode.

6.1.2.1.3 Selecting the Best Inter Mode

Finally, the best Inter mode for the macroblock is done by minimizing the Lagrangian function

$$J(s,c,MODE \mid QP,\lambda_{MODE}) = SSD(s,c,MODE \mid QP) + \lambda_{MODE} \cdot R(s,c,MODE \mid QP) \quad (2\text{-}6.28)$$

$$MODE \in [16\text{x}16,16\text{x}8,8\text{x}16 \text{and P8x8 modes}]$$

where *QP* is the macroblock quantisation parameter, λ_{MODE} is the Lagrange multiplier for mode decision..

6.1.2.1.4 Final Macroblock Prediction Mode Decision

In summary, the macroblock mode decision is done by minimizing the Lagrangian functional

$$J(s,c,MODE\,|\,QP,\lambda_{MODE}) = SSD(s,c,MODE\,|\,QP) + \lambda_{MODE} \cdot R(s,c,MODE\,|\,QP) \qquad (2\text{-}6.29)$$

where *QP* is the macroblock quantiser, λ_{MODE} is the Lagrange multiplier for mode decision, and *MODE* indicates a mode chosen from the set of potential prediction modes:

$$\text{I frame:} \qquad MODE \in \{INTRA4x4,\ INTRA16x16\}, \qquad (2\text{-}6.30)$$

$$\text{P frame:} \qquad MODE \in \begin{Bmatrix} INTRA4x4,\ INTRA16x16,\ SKIP, \\ 16x16,\ 16x8,\ 8x16,\ 8x8 \end{Bmatrix}, \qquad (2\text{-}6.31)$$

$$\text{B frame:} \qquad MODE \in \begin{Bmatrix} INTRA4x4,\ INTRA16x16,\ \ DIRECT, \\ 16x16,\ 16x8,\ 8x16,\ 8x8 \end{Bmatrix} \qquad (2\text{-}6.32)$$

SSD is the sum of the squared differences between the original block *s* and its reconstruction *c* given as

$$SSD(s,c,MODE\,|\,QP) = \sum_{x=1,y=1}^{16,16} \left(s_Y[x,y] - c_Y[x,y,MODE\,|\,QP]\right)^2 + \sum_{x=1,y=1}^{8,8} \left(s_U[x,y] - c_U[x,y,MODE\,|\,QP]\right)^2 + \sum_{x=1,y=1}^{8,8} \left(s_V[x,y] - c_V[x,y,MODE\,|\,QP]\right)^2, \qquad (2\text{-}6.33)$$

and $R(s,c,MODE\,|\,QP)$ is the number of bits associated with choosing *MODE* and *QP*, including the bits for the macroblock header, the motion vector, the reference pictures, and the transformed Y, U and V coefficients.

$c_Y[x,y,MODE\,|\,QP]$ and $s_Y[x,y]$ represent the reconstructed and original luminance values; c_u, c_v, and s_u, s_v are the corresponding chrominance values.

Note that the *SKIP* mode refers to the 16x16 mode where no motion and residual information is coded.

The mode corresponding to the smallest Lagrangian cost will be the mode selected for the macroblock.

6.1.3 Fast High-complexity mode

When RDO option is turned ON, it needs to compute the Lagrangian cost for all possible modes. High complexity mode is computationally intensive because full exhaustive search in motion estimation is used. Some tools are explained in these section that speed up the original encoding process while maintaining the quality of the reconstructed video. The Fast High-complexity mode describe these tools.

In the following sections, fast motion estimation is described first follows by fast sub-sample search. Depending on the spatio-temporal characteristics of the macroblock, some computations of the Lagrangian cost for the possible modes can be excluded thereby speeding up the encoding process. The sections after fast motion estimation will explain these fast Intra and fast Inter mode prediction processes.

6.1.3.1 Fast Motion Estimation

There are four steps in the proposed fast integer sample search algorithm:

Step_1: Initial search point prediction
A motion prediction set Σ is defined including all the candidate predictors described in section 6.1.3.1.1, then:

1) $\mathbf{m}_{\min} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \mathrm{S}$ (*s.t means" subject to"*)

2) Early_Termination(see section 6.1.3.1.2)
Step_2: Asymmetrical-cross search

An asymmetrical-cross search pattern is defined by set Ω_1, which is:

$\Omega_1 = \{\mathbf{m} = (m_x, m_y)^T \,|\, \mathbf{m} = (cm_x \pm 2i, cm_y)^T, i = 0,1,2,....,\frac{W}{2}; \mathbf{m} = (cm_x, cm_y \pm 2j)^T, j = 1,2,....,\frac{W}{4}\}$, where $\mathbf{cm} = \mathbf{m}_{\min}$, and W is the search range. Then:

1) $\mathbf{m}_{\min 2} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \Omega_1$

2) $\mathbf{m}_{\min} = \arg[\min(J(\mathbf{m}_{\min}, \lambda_{MOTION}), J(\mathbf{m}_{\min 2}, \lambda_{MOTION}))]$

3) Early_Termination(see section 6.1.3.1.2)
Step_3: Uneven Multi-Hexagon-grid Search

Two sub-steps in this search step:

Step_3-1:

A square search pattern with search range equals 2 is defined by Ω_2, which is:

$\Omega_2 = \{\mathbf{m} = (m_x, m_y)^T \,||\, m_x - cm_x| \le 2, | m_y - cm_y| \le 2\}$, where $\mathbf{cm} = \mathbf{m}_{\mathbf{min}}$. Then:

1) $\mathbf{m}_{\min 3} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \Omega_2$

2) $\mathbf{m}_{\min} = \arg[\min(J(\mathbf{m}_{\min}, \lambda_{MOTION}), J(\mathbf{m}_{\min 3}, \lambda_{MOTION}))]$
Step_3-2:

A 16 points hexagon search pattern is defined by Ω_{16-HP}, which is:

$\Omega_{16-HP} = \{\mathbf{m} = (x, y)^T \,|\, \mathbf{m} = (\pm 4, \pm 2)^T, (\pm 4, \pm 1)^T, (\pm 4, 0)^T, (\pm 2, \pm 3)^T, (0, \pm 4)^T\}$. By extending the search pattern with a scale factor *k*, a multi-hexagon-grid search scheme is taken.

for k= 1, k <= W/4, k++

{

Search pattern after scaling with factor k is Π_k, which is:

$\Pi_k = \{\mathbf{m} = (m_x, m_y) \,|\, m_x = cm_x + k\text{g}x', m_y = cm_y + k\text{g}y', (x', y') \in \Omega_{16-HP}\}$, then:

1) $\mathbf{m}_{\min k} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \Pi_k$

2) $\mathbf{m}_{\min} = \arg[\min(J(\mathbf{m}_{\min}, \lambda_{MOTION}), J(\mathbf{m}_{\min k}, \lambda_{MOTION}))]$

}

Step_4: Extended Hexagon-based Search

Two sub-steps in this search step:

Step_4-1:

A hexagon search pattern is defined by Ω_3, which is:

$\Omega_3 = \{\mathbf{m} = (m_x, m_y)^T \,|\, \mathbf{m} = (cm_x \pm 2, cm_y)^T, (cm_x \pm 1, cm_y \pm 2)^T\}$, where $\mathbf{cm} = \mathbf{m}_{\min}$. Then:

1) $\mathbf{m}_{\min 4} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \Omega_3$

2) $\mathbf{m}_{\min} = \arg[\min(J(\mathbf{m}_{\min}, \lambda_{MOTION}), J(\mathbf{m}_{\min 4}, \lambda_{MOTION}))]$

3) if $\mathbf{m}_{\min} == \mathbf{cm}$, goto Step_4-2
else goto Step_4-1

Step_4-2:

A diamond search pattern is defined by Ω_4, which is:

$\Omega_4 = \{\mathbf{m} = (m_x, m_y)^T \,|\, \mathbf{m} = (cm_x \pm 1, cm_y)^T, (cm_x, cm_y \pm 1)^T\}$, where $\mathbf{cm} = \mathbf{m}_{\min}$. Then:

1) $\mathbf{m}_{\min 5} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \Omega_4$

2) $\mathbf{m}_{\min} = \arg[\min(J(\mathbf{m}_{\min}, \lambda_{MOTION}), J(\mathbf{m}_{\min 5}, \lambda_{MOTION}))]$

3) if $\mathbf{m}_{\min} == \mathbf{cm}$, stop
else goto Step_4-2

6.1.3.1.1 Motion prediction set for fast search

Three prediction modes are used to predict current block's motion vector, they are:

1) Median Prediction:
mvLXN is the motion vector (with N being either the neighboring partition A, B, or C, see Figure 2-6.3) of the neighboring partitions.

refIdxLXN is the reference indices (with N being either A, B, or C) of the neighboring partitions, and refIdxLX is the reference index of the current partition.

mvMpLX is the median motion prediction of current partition.

The following rules are applied in sequential order to determine the motion prediction mvMpLX:

- If both partitions B and C are not available and A is available, then mvLXB = mvLXA and mvLXC = mvLXA. As well as refIdxLXB = refIdxLXA and refIdxLXC = refIdxLXA.
- If one and only one of the reference indices refIdxLXA, refIdxLXB and refIdxLXC is equal to the reference index refIdxLX of the current partition, the following applies. Let refIdxLXN be the reference index that is equal to refIdxLX, then the motion vector mvLXN is assigned to the motion prediction mvMpLX: mvMpLX = mvLXN.

  - Otherwise, each component of the motion prediction mvMpLX is given by the median of the

  corresponding vector components of the motion vector mvLXA, mvLXB, and mvLXC:

  $$mvMpLX[0] = Median(mvLXA[0], mvLXB[0], mvLXC[0])$$

  $$mvMpLX[1] = Median(mvLXA[1], mvLXB[1], mvLXC[1])$$

2) Uplayer prediction

The recommendation supports motion compensation block size ranging from 16x16 to 4x4 luminance samples, and these block partition modes can be named from Mode1 to Mode7.

Assuming the prediction mode of current partition is $Mode_{curr}$, the uplayer mode $Mode_{Up}$ is:

$$Mode_{Up} = \begin{cases} unavailable, & if \quad Mode_{curr} == Mode1 \\ Mode1, & if \quad Mode_{curr} == Mode2,\ Mode3 \\ Mode2, & if \quad Mode_{curr} == Mode4 \\ Mode4, & if \quad Mode_{curr} == Mode5,\ Mode6 \\ Mode5, & if \quad Mode_{curr} == Mode7 \end{cases}$$

Then if refIdxLX == refIdxLXU, the uplayer predicted vector is:

$$mvUpLX = mvLXU$$

where mvLXU and refIdxLXU are the motion vector and the reference indice of the uplayer partition respectively, and refIdxLX is the reference index of the current partition.

3) Neighboring Ref-frame Prediction

mvNrpLX is defined as the motion pediction of current partition, and mvNrpLX is not available while refIdxLX<1, otherwise mvNrpLX = mvLXNRP*(refIdxLX+1)/(refIdxLXNRP+1), where, refIdxLX is the reference index of the current partition; mvLXNRP is the motion vector of the current partition when reference index refIdxLXNRP==refIdxLX-1.

The final motion prediciton set is based on the above three predictions 1), 2), and 3)

Let Ψ be a set:

$$\Psi(CMV) = \{\mathbf{MV} = (MV[0], MV[1])^T \mid \mathbf{MV} = (CMV[0] \pm 1, CMV[1])^T, (CMV[0], CMV[1] \pm 1)^T\},$$

Then set:

$$S_1 = \{mvMpLX, mvUpLX, mvNrpLX, \underbrace{(0,0)}, \Psi(mvMpLX), \Psi(\underbrace{(0,0)})\}$$

Let $\mathbf{m}_{S\min} = \arg[\min_{\mathbf{m}_i} J(\mathbf{m}_i, \lambda_{MOTION})], s.t. \mathbf{m}_i \in \mathrm{S}_1$, the motion prediction set is:

$$\mathrm{S} = \mathrm{S}_1 + \Psi(\mathbf{m}_{S\min})$$

6.1.3.1.2 Early_termination

6.1.3.1.2.1 Cost Prediction of the Current Paritition

Three cost prediction modes are used to predict current partition's cost, they are:

1) Median Prediction:

- If both partitions B and C are not available and A is available, then csLXB = csLXA and csLXC = csLXA, as well as refIdxLXB = refIdxLXA and refIdxLXC = refIdxLXA.

  - If one and only one of the reference indices refIdxLXA, refIdxLXB and refIdxLXC is equal to the reference index refIdxLX of the current partition, the following applies. Let refIdxLXN, with N being A, B, or C, be the reference index that is equal to refIdxLX. Then the cost csLXN, with N being the correspondent A, B or C, is assigned to the cost prediction: J_{pred_MP} = csLXN.

  - Otherwise, each component of the motion prediction is given by the median of the corresponding vector components of the motion vector mvLXA, mvLXB, and mvLXC:

    mvMpLX[0] = Median(mvLXA[0], mvLXB[0], mvLXC[0])

    mvMpLX[1] = Median(mvLXA[1], mvLXB[1], mvLXC[1])

Let mvLXN (with N being either A, B, or C) be the motion vector that satisfies mvLXN[0]== mvMpLX[0], and let mvLXM(with M being either A, B, or C) be the motion vector that satisfies mvLXM[1]== mvMpLX[1]. Then the median predicted cost is: J_{pred_MP} = min(csLXN, csLXM)

2) Uplayer Prediction

If refIdxLX == refIdxLXU, the uplayer cost predictor is: J_{pred_UP} = csLXU/2, where, csLXU is the cost of the uplayer partition, refIdxLXU is the reference indices of the uplayer partition, index refIdxLX is the reference of the current partition.

3) Neighboring Ref-frame Prediction

J_{pred_NRP} is cost predictor of current partition. If refIdxLX<1, J_{pred_NRP} is not available, otherwise J_{pred_NRP} = csLXNRP, Where, refIdxLX is the reference index of the current partition, csLXNRP is the cost of the current partition when reference indices is refIdxLXNRP==refIdxLX-1.

6.1.3.1.2.2 Early termination

Based on above three cost predictors, the final cost predictor of current partition is:

$$J_{pred} = \begin{cases} J_{pred\_NRP}, & if \quad refIdxLX > 0 \\ J_{pred\_MP}, & if \quad refIdxLX == 0 \,\&\&\, Mode_{curr} == 1 \\ J_{pred\_UP}, & if \quad refIdxLX == 0 \,\&\&\, Mode_{curr} > 1 \end{cases}$$

The Early_Termination is described as following:

if $J(\mathbf{m},\lambda_{MOTION})_{curr} - J_{pred} < J_{pred} * \beta_1$, then jump to a motion search strategy with small diamond search pattern to do a minor refinement, see section 6.1.3.1 step_4-2. And else if $J(\mathbf{m},\lambda_{MOTION})_{curr} - J_{pred} < J_{pred} * \beta_2$, then jump to a motion search strategy with hexagon search pattern to do a preferable refinement, see section 6.1.3.1 step_4-1.

Where two thresholds β_1 and β_2 are set for each partition mode according to the error judgement possibility, and their values are chosen from the experiments. β_1 = {0.01, 0.01, 0.01, 0.02, 0.03, 0.03, 0.04} from Mode1 to Mode7 and β_2 = {0.06, 0.07, 0.07, 0.08, 0.12, 0.11, 0.15} from Mode1 to Mode7 are adopted in fast integer-sample search.

Assuming $J_{th1} = J_{pred} * (1+\beta_1)$ and $J_{th2} = J_{pred} * (1+\beta_2)$, the macro definition for Early_Termination in section 6.1.3.1 can be derived:

#define Early_Termination

if $J(\mathbf{m}_{\min},\lambda_{MOTION}) < J_{th1}$ goto Step_4-2

else if $J(\mathbf{m}_{\min},\lambda_{MOTION}) < J_{th2}$, goto Step_4-1

else continue

#endif

6.1.3.2 Fast Fractional Pel Motion Estimation

Assume $\mathbf{m}$ is the motion vector of the current partition after integer sample search, then $\mathbf{fm} = \mathbf{m} \times 4$, which is the motion vector in fractional sample unit after integer sample motion estimation.

The whole fast fractional sample motion estimation algorithm can be described in the following two steps:

Step_1. Initial search point prediction

Set the predicted fractional sample motion vector $\mathbf{fm}_{pred} = (\mathbf{fm}_{pred\_MP} - \mathbf{fm})\%4$, where $\mathbf{fm}_{pred\_MP} = \mathbf{m}_{pred\_MP} \times 4$ (see section) and % is the mod operation. Then:

$\mathbf{fm}_{\min} = \arg[\min_{\mathbf{m}_i} J(\mathbf{fm}_i,\lambda_{MOTION})], s.t. \mathbf{fm}_i \in T$, where $T = \{\mathbf{fm}_i \,|\, \mathbf{fm}_{pred}, (0,0)\}$.

Step_2. Diamond search strategy

For (i=0; i<7; i++)

{

Let diamond search pattern is: $\Omega_5 = \{\mathbf{fm}_i = (fm_{i,x}, fm_{i,y}) \,|\, \mathbf{fm}_i = (fcm_x \pm 1, fcm_y)^T, \mathbf{fm}_i = (fcm_x, fcm_y \pm 1)^T\}$, where $\mathbf{fcm} = \mathbf{fm}_{\min}$, then

1) $\mathbf{fm}_{\min f} = \arg[\min_{\mathbf{m}_i} J(\mathbf{fm}_i,\lambda_{MOTION})], s.t. \mathbf{fm}_i \in \Omega_5$

2) $\mathbf{fm}_{\min} = \arg[\min(J(\mathbf{fm}_{\min f}, \lambda_{MOTION}), J(\mathbf{fm}_{\min}, \lambda_{MOTION}))]$

3) if ($\mathbf{fm}_{\min} == \mathbf{fcm}$) stop

}

6.1.3.3 For interlace frame and B frame cases

The fast search motion estimation can be done for interlace and B frame cases. For interlace case, motion estimation of current partition is done on the corresponding field of reference frames. For B frame case, when choosing motion prediction set and cost prediction set for backward prediction , Median Prediction mode and Uplayer Prediction mode is the same as that in forward prediction. Neighboring Ref-frame Prediction mode is not used for backward prediction.

6.1.3.4 Fast Intra and Inter Mode Selection

Under the fast high-complexity mode, macroblock mode decision is made in rate-distortion optimized way but having computational efficiency in mind. As indicated in Figure 2-3, inter mode prediction is executed first, and if decided necessary, intra mode decision follows.

To choose the best macroblock mode under the high complexity mode, the encoder needs to calculate the RDcost (rate distortion cost) of every possible mode and chooses the mode having the minimum value. To compute RDcost associated with each mode, same operation of forward and inverse integer transform/quantization and variable-length coding is repetitively performed. Moreover, although the probability of having intra mode is much less than that of inter modes in inter-coded slices, encoder computes the RDcost of all the possible intra modes at each macroblock simply because the RD optimization technique needs to make sure which mode produces the minimum RDcost. Under the fast high-complexity mode, the computational efficiency is enhanced in two ways. First, in inter-coded slices, it investigates the possibility of early decision of SKIP mode as the final coding mode by checking the skip mode condition first since a macroblock satisfying the skip condition has very high probability of ending up with the SKIP mode as the best coding mode. Secondly, it selectively investigates intra coding modes in inter-coded slices after deciding the best inter coding mode. In this way, the encoder can avoid quite a number of unfruitful calculation of RDcost values without costing coding performance.

The procedure of fast high-complexity mode decision is explained in accordance to Figure 2-3.

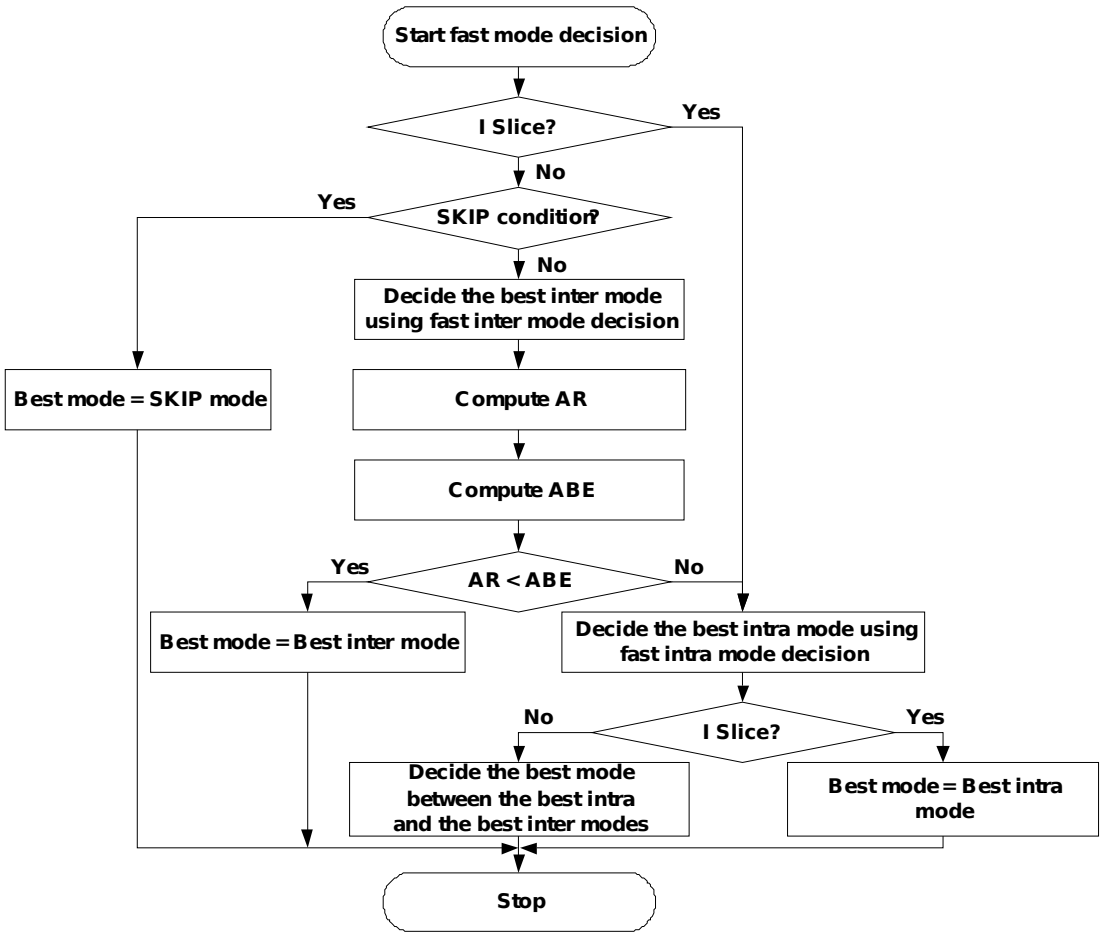


Figure 2-6.5 Flow chart of macroblock mode decision under fast high-complexity mode

6.1.3.4.1 Edge Map Generation

Prior to mode decision, Sobel edge operators are applied to the current picture to generate edge map, where each pixel is associated with an edge vector containing its edge direction and amplitude. For a pixel $p_{i,j}$, in a luminance (or chrominance) picture, the corresponding edge vector, $D_{i,j} = \{dx_{i,j}, dy_{i,j}\}$, is defined as,

$$\begin{aligned} dx_{i,j} &= p_{i-1,j+1} + 2\times p_{i,j+1} + p_{i+1,j+1} - p_{i-1,j-1} - 2\times p_{i,j-1} - p_{i+1,j-1} \\ dy_{i,j} &= p_{i+1,j-1} + 2\times p_{i+1,j} + p_{i+1,j+1} - p_{i-1,j-1} - 2\times p_{i-1,j} - p_{i-1,j+1} \end{aligned} \quad (2\text{-}6.34)$$

where $i = 0, 1, \ldots, N$, $j = 0, 1, \ldots, M$, and $N \times M$ is the picture dimension.

The amplitude of the edge vector is decided by,

$$Amp(D_{i,j}) = |dx_{i,j}| + |dy_{i,j}| \quad (2\text{-}6.35)$$

The direction of the edge (in degree) is decided by the hyper-function,

$$Ang(D_{i,j}) = \frac{180^0}{\pi} \times \arctan\left(\frac{dy_{i,j}}{dx_{i,j}}\right) \quad (2\text{-}6.36)$$

This is used for Fast inter mode decision in 2.1.2.3 and Fast intra mode decision 2.1.2.5.

6.1.3.4.2 Early SKIP mode decision in inter-coded slices

This is to check whether the given macroblock is likely to choose the SKIP mode as the best mode or not. To have a SKIP mode in P-slices, a macroblock should meet following conditions all together:

(i) the motion compensation block size is 16x16,
(ii) reference frame is just one previous one,
(iii) motion vector is the same as its PMV, and
(iv) its transform coefficients are all quantized to zero.

To check these conditions, motion vector and reference frame for the given 16x16 block are estimated.

To be decided as the SKIP in B slice, both condition of direct mode and CBP(coded block pattern) = 0 should be satisfied. Therefore, motion vector under direct mode should be calculated.
Once the condition of the SKIP mode is satisfied in either P- or B- slices, the other inter and intra modes are not investigated anymore. Unless the SKIP mode condition is satisfied, following fast inter mode decision procedure is performed

6.1.3.4.3 Fast inter mode decision

It is observed that when video objects move, various parts of the video objects move together. The so called "spatial smoothness of motion" arises from this observation and has been commonly used in computing motion field. From the observation, it is further noticed that homogeneous regions in natural or synthetic video sequences probably belong to the same video object and thus move together as well. One of the main reasons for using variable block sizes is to represent motion of video objects more adequately. Since homogeneous regions tend to move together, homogeneous blocks in a picture should have similar motion and should not be split further into smaller blocks. The following fast inter mode prediction technique detects the homogeneous blocks so as to avoid further computation in splitting the blocks.

6.1.3.4.3.1 Homogeneous block detection

We determine if a block is homogeneous using the pre-computed amplitude of the edge vector in equation (2-6). If the sum of the magnitude of the edge vector at all pixel location in a block is less than Thd_H, it is classified as homogeneous block, otherwise, it is non-homogeneous. If n and m refers to the index of the row and column of the macroblock, $MB_{n,m}$, then:

$$H_{n,m} = \begin{cases} 1 & \sum_{i,j \in \text{postions in the macroblock}} Amp\ (D_{i,j}) < Thd_H \\ 0 & \sum_{i,j \in \text{postions in the macroblock}} Amp\ (D_{i,j}) \geq Thd_H \end{cases}$$

where $H_{n,m} = 1$ indicates that the macroblock $MB_{n,m}$, is a homogeneous block and is non-homogeneous otherwise.

6.1.3.4.3.2 Mode decision for homogeneous and non-homogeneous blocks

When a 16×16 block is detected as non-homogeneous block, RD optimisation on 16×16, 16×8, 8×16 and 8×8 block sizes are used. During the determination of the sub-types in the non-homogeneous 16×16 block, only 8×8 block size is used when the 8×8 block is homogeneous.

However, when the 16×16 block is detected as homogeneous block, fewer block sizes are used in

RD optimisation depending on the preferred prediction direction which is determined in the same way as the intra prediction mode decision in 2.1.2.5. If the preferred prediction direction is horizontal prediction, then 16×16 and 16×8 block sizes are used. If the preferred prediction direction is vertical prediction, then 16×16 and 8×16 block sizes are used. For other intra mode, only 16×16 block size is used.

Finally, for zero-motion-block with motion compensated error less than Thd_{mc}, 16×16 block size is used.

6.1.3.4.4 Selective intra mode decision

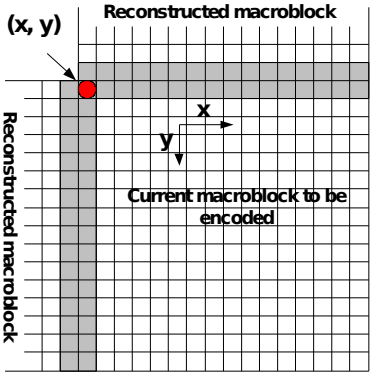
Inter-coded slices generally have only a few intra macroblocks. Therefore, for the sake of computational efficiency, intra mode prediction is investigated only when the result of selected best inter mode indicates such necessity. This test is performed by checking the inequality below:

Investigate the intra coding modes unless AR < ABE

where, ABE is average error between pixels at boundary of the current and its adjacent encoded blocks, and AR is the average number of bits consumed to encode the motion-compensated residual data under the best inter mode. AR is computed as below :

$$AR = \frac{\lambda}{384}\left(\#bits\ of\ texture\ data\right)\ \ wherer\ \ \lambda = 0.85\times 2^{Qp/3}$$

ABE is computed in following way:



$$ABE = \frac{1}{64}\sum_{i=0}^{15}\left[\left|Y_{Orig}(x,y+i) - Y_{Rec}(x-1,y+i)\right| + \left|Y_{Orig}(x+i,y) - Y_{Rec}(x+i,y-1)\right|\right]$$
$$+\sum_{i=0}^{7}\left[\left|C_{b,Orig}(cx,cy+i) - C_{b,Rec}(cx-1,cy+i)\right| + \left|C_{b,Orig}(cx+i,cy) - C_{b,Rec}(cx+i,cy-1)\right|\right]$$
$$+\sum_{i=0}^{7}\left[\left|C_{r,Orig}(cx,cy+i) - C_{r,Rec}(cx-1,cy+i)\right| + \left|C_{r,Orig}(cx+i,cy) - C_{r,Rec}(cx+i,cy-1)\right|\right]$$

Figure 2-6.6 Computing average boundary error

, where $Y_{Orig}, C_{b,Orig}, C_{r,Orig}$ are the pixel values of of current macroblock to be encoded; $Y_{Rec}, C_{b,Rec}, C_{r,Rec}$ are the reconstructed data of the macroblock; and $(x,y),(cx,cy)$ respectively indicate coordinates of the upper-most left pixel in luminance (Y) and chrominance block(Cb, Cr). According to the position of a block, following exception rule applies:

Case 1: [Inside of slice : Left, Right, Upper and Bottom blocks are available]

ABE is computed as Figure 2-4

Case 2: [Boundary of slice]

If AR=0, skip investigating intra prediction modes; else *ABE = 0*

Once further intra mode prediction deems necessary, following fast intra mode decision is carried out.

6.1.3.4.5 Fast intra mode decision

Fast intra mode decision algorithm is implemented based on the local edge direction information generated in 6.1.3.4.1. By determining the edge direction of the current block to be coded, only a few most probable INTRA modes are selected for RDO computations.

As there is only a limited number of modes that intra prediction could be applied, a thresholding technique is used to substitute Equation (???) [Ed. Note: Missing equation no.] in the actual implementation of the algorithm. Therefore the edge direction histogram of a 4×4 luma block is decided by the followings,

$$
\begin{aligned}
&For\ all\ the\ pixels\ p_{i,j}\ in\ the\ 4\times 4\ luma\ block,\\
&Histo(k) = \sum_{(i,j)\in SET(k)} Amp(D_{i,j}),\\
&\text{where,}\\
&\qquad SET(k) \in \left\{(i,j) \mid Ang\left(\vec{D}_{i,j}\right) \in a_k\right\},\\
&\text{and}\\
&\qquad a_0 = (-103.3^0, -76.6^0]\\
&\qquad a_1 = (-13.3^0, 13.3^0]\\
&\qquad a_3 = (35.8^0, 54.2^0]\\
&\qquad a_4 = (-54.2^0, -35.8^0]\\
&\qquad a_5 = (-76.7^0, -54.2^0]\\
&\qquad a_6 = (-35.8^0, -13.3^0]\\
&\qquad a_7 = (54.2^0, -76.7^0]\\
&\qquad a_8 = (13.3^0, 35.8^0]
\end{aligned}
\tag{2-6.37}
$$

Similarly, the edge direction histogram of an 8×8 luma block or 16×16 chroma block is decided as follows,

$$
\begin{aligned}
&For\ all\ the\ pixels\ p_{i,j}\ in\ the\ 8\times 8\ chroma\ bbck\ or\ 16\times 16\ luma\ block,\\
&Histo(k) = \sum_{(i,j)\in SET(k)} Amp(D_{i,j}),\\
&\text{where,}\\
&\qquad SET(k) \in \left\{(i,j) \mid Ang\left(\vec{D}_{i,j}\right) \in a_k\right\},\\
&\text{and}\\
&\qquad a_0 = (-112.5^0, -67.5^0]\\
&\qquad a_1 = (-22.5^0, 22.5^0]\\
&\qquad a_3 = (-67.5^0, -22.5^0]
\end{aligned}
\tag{2-6.38}
$$

Figure 2-6.7 shows an example of the edge direction histogram of a 4×4 luma block.

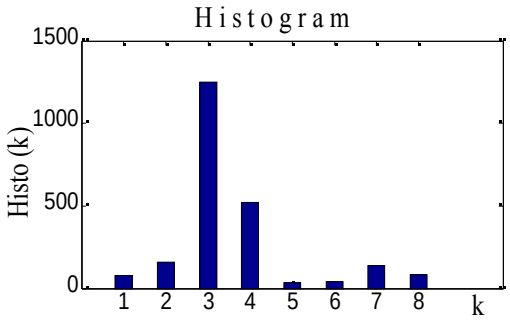


Figure 2-6.7 Edge direction histogram

According to (???)[Ed. Note: Missing equation no.] and (???)[Ed. Note: Missing equation no.], each cell in the edge direction histogram sums up the amplitudes of the edge with the similar direction in that block. Since the pixels along edge direction are likely to have similar values, the best prediction mode is probably in the edge direction whose cell has the maximum amplitude, or the directions close to the maximum amplitude cell.

6.1.3.4.5.1 4×4 luma block prediction modes

The histogram cell with the maximum amplitude and the two adjacent cells are considered as candidates for the best prediction mode. In addition, DC mode is chosen as the fourth candidate. Thus, for each 4×4 luma block, only 4 prediction modes are performed.

6.1.3.4.5.2 16×16 luma block prediction modes

Only the histogram cell with maximum amplitude is considered as a candidate of best prediction mode. In addition, DC mode is chosen as the next candidate. Thus, for each 16×16 luma block, only 2 prediction modes are performed.

6.1.3.4.5.3 8×8 chroma prediction modes

In the case of chroma blocks, there are two different histograms, one from component U and the other from V. The histogram cells with maximum amplitude from the two components are both considered as candidate modes. In addition, DC mode is chosen as another candidate. Note that if the candidates from the two components are the same, there is 2 candidate prediction modes; otherwise, it is 3.

Table 2-6.1 summarizes the number of candidates selected for RDO calculation based on edge direction histogram.

Table 2-6.1 Number of selected modes

	Block size	Total No. of modes	No. of modes selected
Luma (Y)	16×16	4	2
Luma (Y)	4×4	9	4
Chroma (U, V)	8×8	4	3 or 2

6.1.4 Low-complexity mode

In low-complexity mode, the Rate Distortion Optimisation (RDO) is turned off. The cost of each mode is computed using some biases and Sum of Absolute Difference (SAD) of either the prediction errors or the Hadamard transformed coefficients of the difference as shown in Figure 2-6.8

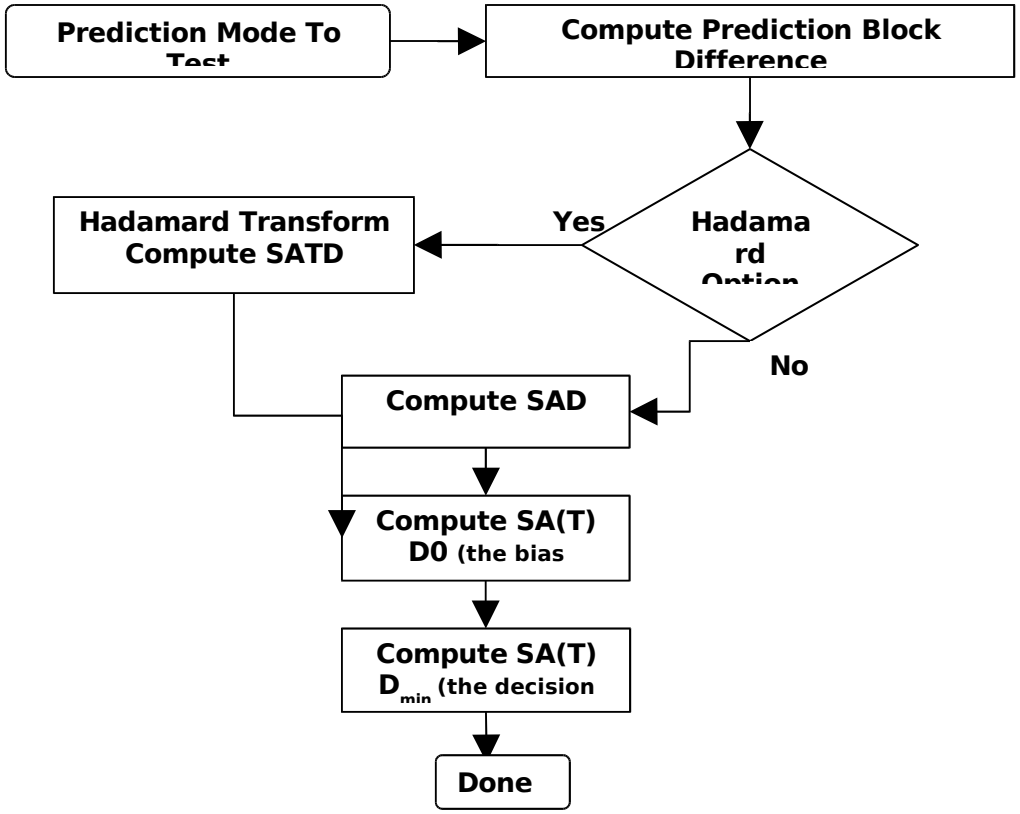


Figure 2-6.8 Flow Chart for Low-Complexity Prediction Mode Decision

Note that when computing the cost for each inter mode, the distortion term of the Lagrange cost function is actually SAD or SATD in low complexity mode.

6.1.4.1 Prediction Blocks and Computing SA(T)D

Depending on the Hadamard flag, SA(T)D is computed from the prediction block of each mode as specified in section 6.1.1.4.

6.1.4.2 Computing SA(T)D0

The SA(T)D to be minimised is given a 'bias' value SA(T)D0 initially in order to favour prediction modes that need fewer bits to be signalled. This bias is basically a parameter representing bit usage multiplies by $QP_0(QP)$ where $QP_0(QP)$ is a table lookup with input QP, the quantisation parameter.

The calculation of SA(T)D0 at each mode is as follows.

(i) Forward prediction mode (PRED_L0):

$$SA(T)D0 = QP_0(QP) \times (2 \times code\_number\_of\_ref\_idx\_fwd + Bits\_to\_code\_MVDFW) \quad (2\text{-}6.39)$$

(ii) Backward prediction mode (PRED_L1):

$$SA(T)D0 = QP_0(QP) \times Bits\_to\_code\_MVDBW \quad (2\text{-}6.40)$$

(iii) Bi-directional prediction mode (BiPred):

$$SA(T)D0 = QP_0(QP) \times (2 \times code\_number\_of\_ref\_idx\_fwd + Bits\_to\_code\_forward\_Blk\_size + Bits\_to\_code\_backward\_Blk\_size + Bits\_to\_code\_MVDFW + Bits\_to\_code\_MVDBW) \quad (2\text{-}6.41)$$

[Ed. Note: Pls specify highlighted]

(iv) Direct prediction mode (Direct):

For flat regions having zero motion, B pictures basically fail to make effective use of zero motion and instead are penalized in performance by selecting 16x16 intra mode. Therefore, in order to prevent assigning 16x16 intra mode to a region with little details and zero motion, SA(T)D of direct mode is subtracted by 16xQP0(QP) to bias the decision toward selecting the direct mode..

$$SA(T)D0 = -16 \times QP_0(QP) \quad (2\text{-}6.42)$$

(v) Intra 4x4 mode:

For the whole intra 4x4 macroblock, $24 \times QP_0(QP)$ is added to the SA(T)D before comparison with the best SA(T)D for inter prediction. This is an empirical value to prevent using too many intra blocks

$$SA(T)D0 = 24 \times QP0(QP) \quad (2\text{-}6.43)$$

(vi) Intra 16x16 mode:

$$SA(T)D0 = 0 \quad (2\text{-}6.44)$$

In Intra 16x16 mode, SATD is used irregardless of Hadamard flag.

[Ed. Note: Inter mode bias missing in here. i.e. Mvd bits]

6.1.4.3 Final Mode Selection by Determining $SA(T)D_{min}$

The prediction mode that results in the minimum value $SA(T)D_{min} = \min(SA(T)D+SA(T)D0)$ is chosen.

6.2 Transform and quantisation

6.2.1 Quantisation tables

The coefficients Q(m,i,j), used in the formulas below, are defined in pseudo code by:

$Q(m,i,j) = M_{m,0}$ for $(i,j) = \{(0,0),(0,2),(2,0),(2,2)\}$,

$Q(m,i,j) = M_{m,1}$ for $(i,j) = \{(1,1),(1,3),(3,1),(3,3)\}$,

$Q(m,i,j) = M_{m,2}$ otherwise;

where

$$M = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix} \qquad (2\text{-}6.45)$$

6.2.2 4x4 spatial block processing

Instead of a discrete cosine transform (DCT), an integer transform with a similar coding gain as a 4x4 DCT is used. The transformation of input samples $X=\{x_{00} \ldots x_{33}\}$ to output coefficients Y is defined by

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \qquad (2\text{-}6.46)$$

Multiplication by two can be performed either through additions or through left shifts, so that no actual multiplication operations are necessary. Thus, we say that the transform is multiplier-free.

For input samples with 9-bit dynamic range (because they are residuals from 8-bit sample data), the transform coefficients are guaranteed to fit within 16 bits, even when the second transform for DC coefficients is used. Thus, all transform operations can be computed in 16-bit arithmetic.

The transform and inverse transform matrices above have orthogonal basis functions. Unlike the DCT, though, the basis functions don't have the same norm. Therefore, for the inverse transform to recover the original samples, appropriate normalization factors must be applied to the transform coefficients before quantisation and after inverse quantisation. Such factors are absorbed by the quantisation and inverse quantisation scaling factors.

Quantisation is performed according to the following equation:

$$Y_Q(i,j) = \left[Y(i,j) \cdot Q\left((QP+12)\%6, i, j \right) + f \right] / 2^{15+(QP+12)/6}, \quad i,j = 0,\text{K},3 \qquad (2\text{-}6.47)$$

where Y are the transformed coefficients, Y_Q are the corresponding quantised values, $Q(m,i,j)$ are the quantisation coefficients listed below, with f having the same sign as $Y(i,j)$, and |f| is in the range 0 to $2^{14+(QP+12)/6/2}$. Specifically,

$$|f| = \begin{cases} (2 \cdot 2^{15+(QP+12)/6}) / 6 & \text{for intra coefficient quantization} \\ (1 \cdot 2^{15+(QP+12)/6}) / 6 & \text{for inter coefficient quantization} \end{cases} \qquad (2\text{-}6.48)$$

NOTE – while the intermediate value inside square brackets in Equation (2-6.47) has a 32-bit range, the final value Y_Q is guaranteed to fit in 16 bits.

6.2.3 DC luminance coefficients in 16x16 intra mode

To minimize the dynamic range expansion due to transformations (and thus minimize rounding errors in the quantisation scale factors), a simple scaled Hadamard transform is used. The direct transform is defined by:

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{D00} & x_{D01} & x_{D02} & x_{D03} \\ x_{D10} & x_{D11} & x_{D12} & x_{D13} \\ x_{D20} & x_{D21} & x_{D22} & x_{D23} \\ x_{D30} & x_{D31} & x_{D32} & x_{D33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) // 2 \qquad (2\text{-}6.49)$$

where the symbol // denotes division with rounding to the nearest integer:

$$a // 2^b = \text{sign}(a) \times \left[\left(\text{abs}(a) + 2^{b-1} \right) >> b \right] \qquad (2\text{-}6.50)$$

Quantisation is performed according to the following equation:

$$Y_Q(i,j) = [Y(i,j) \cdot Q((QP+12)\%6, i, j) + 2f] / 2^{16+(QP+12)/6}, \quad i,j = 0,\ldots,3 \qquad (2\text{-}6.51)$$

where *f* has the same sign as *Y*(*i,j*) and |f| is given by Equation (2-6.48).

6.2.4 DC chrominance coefficients

The 2 dimensional 2x2 transform procedure is:

$$Y = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} DC_{00} & DC_{01} \\ DC_{10} & DC_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (2\text{-}6.52)$$

Quantisation is performed according to the following equation:

$$Y_Q(i,j) = \left[Y(i,j) \cdot Q((QP+12)\%6, i, j) + 2f \right] / 2^{16+(QP+12)/6}, \quad i,j = 0,\ldots,3 \qquad (2\text{-}6.53)$$

where *f* has the same sign as *Y*(*i,j*) and |f| is given by Equation (2-6.48).

6.3 Elimination of single coefficients in inter macroblocks [Ed. Note: Do we still use this?]

6.3.1 Luminance

With the small 4x4 blocks, it may happen that for instance a macroblock has only one nonzero coefficient with |Level| =1. This will probably be a very "expensive" coefficient in terms of bit usage and it could have been better to set it to zero. For that reason a procedure to check single coefficients has been used for inter luma blocks. During the quantisation process, a parameter Single_ctr is accumulated depending on Run and Level according to the following rule:

- If Level = 0 or (|Level| = 1 and Run > 5) nothing is added to Single_ctr.
- If |Level| > 1, 9 is added to Single_ctr.
- If |Level| = 1 and Run < 6, a value T(Run) is added to Single_ctr. where T(0:5) =(3,2,2,1,1,1)
- If the accumulated Single_ctr for a 8x8 block is less than 4, all coefficients of that luma block are set to zero. Similarly, if the accumulated Single_ctr for the whole macroblock is less than 6, all coefficients of that luma macroblock are set to zero.

6.3.2 Chrominance

A similar method to the one for luma is used. Single_ctr is calculated similarly for each chroma component, but for AC coefficients only and for the whole macroblock.

If the accumulated Single_ctr for each chroma component of a macroblock is less than 7, all the AC chroma coefficients of that component for the whole macroblock are set to zero.

6.4 S-Pictures

6.4.1 Encoding of secondary SP-pictures

This section suggests an algorithm that can be used to create a secondary SP-picture with identical sample values to another SP-picture (called the target SP-picture). The secondary SP-picture is typically used for switching from one bitstream to another and thus has different reference frames for motion compensation than the target SP-picture.

Intra-type macroblocks from the target SP-picture are copied into the secondary SP-picture without alteration. SP-macroblocks from the target SP-pictures will be replaced by the following procedure: Let Lrec be the quantised coefficients, see subclause 8.2, that are used to reconstruct the SP-block. Define the difference levels Lerr by:

$$Lerr = Lrec - L_P \qquad (2\text{-}6.54)$$

where L_P is the quantised transform coefficients of the predicted block from any of the SP coding modes. Then a search over all the possible SP modes is performed and the mode resulting in the minimum number of bits is selected.

6.4.2 Encoding of SI-pictures

The following algorithm is suggested to encode an SI-picture such that the reconstruction of it is identical to that of an SP-picture. SP-pictures consist of intra and SP-macroblocks. Intra-type macroblocks from SP-pictures are copied into SI-pictures with minor alteration. Specifically, the MB_Type is altered to reflect Intra Modes in SI-pictures, i.e., MB_Type is incremented by one, See Table 15. The SP-macroblocks from SP-pictures will be replaced by SIntra 4x4 modes. Let Lrec be the quantised coefficients, see subclause 8.2, that are used to reconstruct the SP-block. Define the difference levels Lerr by:

$$Lerr = Lrec - L_I \qquad (2\text{-}6.55)$$

where L_I is the quantised transform coefficients of the predicted block from any of the intra prediction modes. Then a search over the possible intra prediction modes is performed and the mode resulting in the minimum number of bits is selected.

6.5 Encoding with anticipation of slice losses

Low delay video transmission may lead to losses of slices. The decoder may then stop decoding until the next I picture or P picture may conduct a concealment, for example as explained in Appendix IV, and continue decoding. In the latter case, spatio-temporal error propagation occurs if the concealed picture content is referenced for motion compensation. There are various means to stop spatio-temporal error propagation including the usage of multiple reference pictures and Intra coding of macroblocks. For the latter case, a Lagrangian mode selection algorithm is suggested as follows.

Since transmission errors occur randomly, the decoding result is also a random process. Therefore, the average decoder distortion is estimated to control the encoder for a specified probability of packet losses p. The average decoding result is obtained by running *N* complete decoders at the encoder in parallel. The statistical process of loosing a slice is assumed to be independent for each of the *N* decoders. The slice loss process for each decoder is also assumed to be independent and identically distributed, and a certain slice loss probability *p* is assumed to be known at the encoder. Obviously for large *N* the decoder gets a very good estimate of the average decoder distortion. However, with increasing *N* a linear increase of storage and decoder complexity in the encoder is incurred. Therefore, this method might not be practical in real-time encoding processes and complexity and memory efficient algorithms are currently under investigation.

To encode a macroblock in a P picture, the set of possible macroblock types is given as

$$S_{MB} = \{ \text{MBskip, MBinter16x16, MBinter16x8, MBinter8x16, MBinter8x8, MBintra16x16, MBintra4x4} \} \quad (2\text{-}6.56)$$

For each macroblock the coding mode m' is selected according to

[Ed. Note: something is wrong for the symbols below the min sign]

$$m' = \min_{m \in S_{MB}} \{ D_m + \lambda R_m \} \quad (2\text{-}6.57)$$

with D_m being the distortion in the current macroblock when selecting macroblock mode m and R_m being the corresponding rate, i.e. the number of bits. For the COPY_MB and all INTER_MxN types, the distortion D_m is computed as

$$D_m = \frac{1}{N} \sum_{n=1}^{N} \sum_{i} (f_i - \hat{f}_{i,n,m}(p))^2 \quad (2\text{-}6.58)$$

with f_i being the original sample value at position i within the macroblock and $\hat{f}_{i,n,m}$ being the reconstructed sample value at position i for coding macroblock mode m in the simulated decoder n. The distortion for the intra macroblocks remains unchanged. Since the various reconstructed decoders also contain transmission errors, the Lagrangian cost function for the COPY_MB and all INTER_MxN types increases making Intra_NxN types more popular.

The λ parameter for mode decision depends on the quantisation parameter q as follows

[Ed. Note: Change "l" to Lambda below.]

$$\lambda = (1 - p) 0.85 \times 2^{QP/3} \quad (2\text{-}6.59)$$

A reference frames restriction is introduced to avoid the prediction from areas which have been intra-updated in later frames. This restriction is only effective for P-frames where multiple reference frames are used, rate-distortion optimized mode selection is applied and the reference restriction flag (RRF) is set to 1 or 2. The RRF has the following effects on the encoded video:

RRF	Effect on Encoded Video
0	No restriction on reference frames for any MB is applied
1	Disallows combinations of macroblock modes and reference frames which reference pixels previous to a macroblock intra-updated due to error resilience reasons.
2	Disallows combinations of macroblock modes and reference frames which reference pixels previous to any intra-updated macroblock

The detailed algorithm on the reference frame restriction is as follows. Let us define a intra pixel map *ip(i,j), i=1,..., img_width, j=1,..., img_height*, which specifies for each pixel the lowest past frame number in which the reference chain referenced an intra-updated pixel. The generation of *ip (i,j)* will be discussed later. In addition we define for each 8x8 block a intra-refresh flag *r(i,j), i=1,..., img_width/8, j=1,..., img_height/8*, which signals if the block at position *(i,j)* restricts further prediction. The generation of *r(i,j)* will also be discussed later.

The reliability of a certain combination of MB coding mode *mode*, reference frame *ref* and 8x8block position *block* is checked and only if the combination is reliable, it is valid to be selected in the encoding process. A combination is not valid if any referenced pixel at position *(i,j)* or any corresponding quarter-pel position has been intra-updated in a frame later than the selected reference frame, i.e. it is not valid if *ip(i,j) < ref*.

The intra-refresh flag *r(i,j)* is updated for each 8x8 block after encoding a macroblock depending of the selected reference restriction flag. If the RRF is set to 2, then *r(i,j)* is set to 1 if this block is intra-coded (MB mode = INTRA16 or block mode = IBLOCK). If the RRF is set to 1, then *r(i,j)* is set

to 1 if this block is not intra-coded due to compression reasons, but intra-coded (MB mode = INTRA16 or block mode = IBLOCK) due to error-resilience reasons (forced intra-update by regular or random intra-update, optimized intra-updates).

After encoding one frame the intra pixel map is updated. If the current pixel position is intra updated, the intra pixel map is set to 1 at this position. Otherwise, the intra pixel map is increased by one as the last intra update is moved further to the past. The maximum number is the number of possible reference frames specified. The pseudo-code is as follows:

```
// for all blocks
for (my=0; my<img->height/8; my++)
  for (mx=0; mx<img->width/8;  mx++)  {
       j = my*8 + 8;
       i = mx*8 + 8;
       // for all pixels
       for (y=my*8; y<j; y++)
          for (x=mx*8; x<i; x++)
             if (r[my][mx])
                ip[y][x] = 1;
             else
                ip[y][x] = min(ip[y][x]+1, input->no_multpred+1);
```

6.6 Rate Control

This section presents the rate control algorithm with an assumption that there exists a predefined pattern for the order of the coded pictures. The algorithm is used to create the stream satisfying the available bandwidth provided by a channel and is also compliant to hypothetical reference decoder (HRD). It consists of three tightly consecutive components: GOP level rate control, picture level rate control and the optional basic unit level rate control. The basic unit is defined as a group of successive macroblocks in the same frame. When the basic units arise, each shall contain at least a macroblock.

6.6.1 GOP level rate control

GOP level rate control calculates the total bits for the rest pictures in this GOP and the initial quantization parameter of instantaneous decoding refresh (IDR) picture and that of the first stored picture.

When the j[th] picture in the i[th] GOP is coded, the total bits for the rest pictures in this GOP are computed as follows

$$B_i(j)=\begin{cases}\dfrac{R_i(j)}{f}\times N_i - V_i(j) & j=1\\ B_i(j-1)+\dfrac{R_i(j)-R_i(j-1)}{f}\times(N_i-j+1)-b_i(j-1) & j=2,3,...,N_i\end{cases}. \qquad (2\text{-}6.60)$$

(1) For the first picture in a GOP (i.e. j = 1), the total bits are calculated from the upper formula in (2-6.60). f is the predefined coding frame rate. N_i is the total number of pictures in the i[th] GOP. $R_i(j)$ and $V_i(j)$ are the instant available bit rate and the occupancy of the virtual buffer, respectively, when the j[th] picture in the i[th] GOP is coded.

(2) For other pictures, the total bits are calculated from the bottom formula in (2-6.60). $b_i(j\text{-}1)$ is the actual generated bits in the (j-1)[th] picture. Considering the case of the dynamic channels, $R_i(j)$ may vary at different frames and GOPs. But, in the case of constant bit rate, $R_i(j)$ is always equal to $R_i(j\text{-}1)$. The formula can be simplified as

$$B_i(j)=B_i(j-1)-b_i(j-1) \qquad (2\text{-}6.61)$$

(3) $V_i(j)$ is updated after coding each picture as

$$V_i(1) = \begin{cases} 0 & i = 1 \\ V_{i-1}(N_{i-1}) & other \end{cases}$$
$$V_i(j) = V_i(j-1) + b_i(j-1) - \frac{R_i(j-1)}{f} \qquad j = 2,3,...,N_i \qquad (2\text{-}6.62)$$

An initial quantization parameter $QP_i(1)$ is set for the IDR picture and the first stored picture of the i^{th} GOP.

(1) For the first GOP, $QP_1(1)$ is predefined based on the available channel bandwidth as follows,

$$QP_1(1) = \begin{cases} 40 & bpp \le l1 \\ 30 & l1 < bpp \le l2 \\ 20 & l2 < bpp \le l3 \\ 10 & bpp > l3 \end{cases} \qquad (2\text{-}6.63)$$

where $bpp = \dfrac{R_1(1)}{f \times N_{pixel}}$

N_{pixel} is the number of pixel in a picture. $l1$=0.15, $l2$=0.45, $l3$=0.9 is recommended for QCIF/CIF, and $l1$=0.6, $l2$=1.4, $l3$=2.4 is recommended for the picture size larger than CIF.

(2) For the other GOPs,

$$QP_i(1) = \max\left\{QP_{i-1}(1) - 2, \min\left\{QP_{i-1}(1) + 2, \frac{SumPQP(i-1)}{N_p(i-1)} - \min\left\{2, \frac{N_{i-1}}{15}\right\}\right\}\right\} \qquad (2\text{-}6.64)$$

$N_P(i\text{-}1)$ is the total number of stored pictures in the (i-1)th GOP, and $SumPQP(i\text{-}1)$ is the sum of average picture quantization parameters for all stored pictures in the (i-1)th GOP. It's further adjusted by

$$QP_i(1) = QP_i(1) - 1 \quad if \quad QP_i(1) > QP_{i-1}(N_{i-1} - L) - 2 \qquad (2\text{-}6.65)$$

$QP_{i-1}(N_{i-1} - L)$ is the quantization parameter of the last stored picture in the previous GOP, and L is the number of successive non-stored pictures between two stored pictures.

6.6.2 Picture level rate control

The picture level rate control consists of two stages: pre-encoding and post-encoding.

6.6.2.1 Pre-encoding stage

The objective of this stage is to compute a quantization parameter of each picture. Different methods are proposed for stored and non-stored pictures

6.6.2.1.1 Non-stored pictures

The quantization parameters for non-stored pictures are computed by a simple interpolation method as follows:

Suppose that the j[th] and (*j*+*L*+1)[th] pictures are stored pictures and the quantization parameters for these stored pictures are $QP_i(j)$ and $QP_i(j+L+1)$, respectively. The quantization parameter of the i[th] non-stored pictures is calculated according to the following two cases:

Case 1: When L=1, there is only one non-stored picture between two stored pictures. The quantization parameter is computed by

$$QP_i(j+1)=\begin{cases}\dfrac{QP_i(j)+QP_i(j+2)+2}{2} & \text{if } QP_i(j)\neq QP_i(j+2)\\ QP_i(j)+2 & \text{Otherwise}\end{cases} \quad (2\text{-}6.66)$$

Case 2: when L>1, there are more than one non-stored picture between two stored pictures. The quantization parameters are computed by

$$QP_i(j+k)=QP_i(j)+\alpha+\max\{\min\{\frac{(QP_i(j+L+1)-QP_i(j)}{L-1},2\times(k-1)\},-2\times(k-1)\} \quad (2\text{-}6.67)$$

where *k =1, …, L*, and α is given by

$$\alpha=\begin{cases}-3 & QP_i(j+L+1)-QP_i(j)\le -2\times L-3\\ -2 & QP_i(j+L+1)-QP_i(j)=-2\times L-2\\ -1 & QP_i(j+L+1)-QP_i(j)=-2\times L-1\\ 0 & QP_i(j+L+1)-QP_i(j)=-2\times L\\ 1 & \mathrm{QP_i}(j+L+1)-QP_i(j)=-2\times L+1\\ 2 & \text{Otherwise}\end{cases} \quad (2\text{-}6.68)$$

The quantization parameter $QP_i(j+k)$ is further bounded by 0 and 51.

6.6.2.1.2 Stored pictures

The quantization parameter of stored picture is computed via the following two steps.

Step 1: Determine target bits for each stored picture.

(1) Determine the target buffer level for each stored picture in the current GOP.

A target buffer level is predefined for each stored picture according to the coded bits of the first IDR picture and the first stored picture, and the average picture complexity. After coding the first stored picture in the i[th] GOP, the initial value of target buffer level is set to,

$$S_i(2)=V_i(2) \quad (2\text{-}6.69)$$

The target buffer level for the subsequent stored picture is determined by

$$S_i(j+1)=S_i(j)-\frac{S_i(2)}{N_p(i)-1}+\frac{\overline{W}_{p,i}(j)\times(L+1)\times R_i(j)}{f\times(\overline{W}_{p,i}(j)+\overline{W}_{b,i}(j)\times L)}-\frac{R_i(j)}{f} \quad (2\text{-}6.70)$$

where $\overline{W}_{p,i}(j)$ is the average complexity weight of stored pictures, $\overline{W}_{b,i}(j)$ is the average

complexity weight of

non-stored pictures. They are computed by

$$\begin{aligned}
\overline{W}_{p,i}(j) &= \frac{W_{p,i}(j)}{8} + \frac{7 \times \overline{W}_{p,i}(j-1)}{8} \\
\overline{W}_{b,i}(j) &= \frac{W_{b,i}(j)}{8} + \frac{7 \times \overline{W}_{b,i}(j-1)}{8} \\
W_{p,i}(j) &= b_i(j) \times QP_{p,i}(j) \\
W_{b,i}(j) &= \frac{b_i(j) \times QP_{b,i}(j)}{1.3636}
\end{aligned} \qquad (2\text{-}6.71)$$

When there is no non-stored picture between two stored pictures, Equation (2-6.70) is simplified as

$$S_i(j+1) = S_i(j) - \frac{S_i(2)}{N_p(i) - 1} \qquad (2\text{-}6.72)$$

(2) Compute the target bits for the current stored picture.

The target bits allocated for the j^{th} stored picture in the i^{th} GOP are determined based on the target buffer level (2-6.70), the frame rate, the available channel bandwidth, and the actual buffer occupancy of (2-6.62) as follows:

$$\widetilde{T}_i(j) = \frac{R_i(j)}{f} + \gamma \times (S_i(j) - V_i(j)) \qquad (2\text{-}6.73)$$

where γ is a constant and its typical value is 0.5 when there is no non-stored picture and 0.25 otherwise.

Meanwhile, the number of remaining bits should also be considered when the target bit is computed.

$$\hat{T}_i(j) = \frac{W_{p,i}(j-1) \times B_i(j)}{W_{p,i}(j-1) \times N_{p,r} + W_{b,i}(j-1) \times N_{b,r}} \qquad (2\text{-}6.74)$$

where $N_{p,r}$ and $N_{b,r}$ are the number of the remaining stored pictures and the number of the remaining non-stored pictures, respectively.

The target bits are a weighted combination of $\widetilde{T}_i(j)$ and $\hat{T}_i(j)$:

$$T_i(j) = \beta \times \hat{T}_i(j) + (1-\beta) \times \widetilde{T}_i(j) \qquad (2\text{-}6.75)$$

where β is a constant and its typical value is 0.5 when there is no non-stored picture and is 0.9 otherwise.

To conform with the HRD requirement, the target bits are bounded by

$$T_i(j) = \max\{Z_i(j), \mathrm{T_i}(j)\}$$
$$T_i(j) = \min\{U_i(j), \mathrm{T_i}(j)\} \quad (2\text{-}6.76)$$

where $Z_i(j)$ and $U_i(j)$ are computed by

$$Z_i(j) = \begin{cases} B_{i-1}(N_{i-1}) + \dfrac{R_i(j)}{f} & j = 1 \\ Z_i(j-1) + \dfrac{R_i(j)}{f} - b_i(j) & other \end{cases} \quad (2\text{-}6.77)$$

$$U_i(j) = \begin{cases} (B_{i-1}(N_{i-1}) + t_{r,1}(1)) \times \varpi & j = 1 \\ U_i(j-1) + (\dfrac{R_i(j)}{f} - b_i(j)) \times \varpi & other \end{cases} \quad (2\text{-}6.78)$$

$t_{r,1}(1)$ is the removal time of the first picture from the coded picture buffer. ϖ is a constant with typical value of 0.9.

Step 2: Compute the quantization parameter and perform RDO.

The MAD of the current stored picture, $\tilde{\sigma}_i(j)$, is predicted by a linear model (2-6.79) using the actual MAD of the previous stored picture, $\sigma_i(j-1-L)$.

$$\tilde{\sigma}_i(j) = a_1 \times \sigma_i(j-1-L) + a_2 \quad (2\text{-}6.79)$$

where a_1 and a_2 are two coefficients. The initial value of a_1 and a_2 are set to 1 and 0, respectively. They are updated by a linear regression method similar to that of MPEG-4 Q2 after coding each picture or each basic unit.

The quantization step corresponding to the target bits is then computed by using the following quadratic :

$$T_i(j) = c_1 \times \frac{\tilde{\sigma}_i(j)}{Q_{step,i}(j)} + c_2 \times \frac{\tilde{\sigma}_i(j)}{Q^2_{step,i}(j)} - m_{h,i}(j) \quad (2\text{-}6.80)$$

where $m_{h,i}(j)$ is the total number of header bits and motion vector bits, c_1 and c_2 are two coefficients.

The corresponding quantization parameter $QP_i(j)$ is computed by using the relationship between the quantization step and the quantization parameter of AVC. To maintain the smoothness of visual quality among successive frames, the quantization parameter $QP_i(j)$ is adjusted by

$$QP_i(j) = \min\{QP_i(j-L-1) + 2, \max\{QP_i(j-L-1) - 2, QP_i(j)\}\} \quad (2\text{-}6.81)$$

The final quantization parameter is further bounded by 51 and 0. The quantization parameter is then used to perform RDO for each MB in the current frame.

6.6.2.2 Post-encoding stage

After encoding a picture, the parameters *a1* and *a2* of linear prediction model (2-6.79), as well as c_1 and c_2 of quadratic R-D model (2-6.80) are updated. A linear regression method similar to MPEG-4 Q2 is used to update these parameters.

Meanwhile, the actual bits generated are added to the buffer. To ensure that the updated buffer occupancy is not too high, a number of pictures may be skipped by using the method similar to MPEG-4 Q2.

6.6.3 Basic unit level rate control

Suppose that a picture is composed of N_{mbpic} MBs. A basic unit is defined to be a group of continuous MBs, and consists of N_{mbunit} MBs, where N_{mbunit} is a fraction of N_{mbpic}. If N_{mbunit} equals to N_{mbpic}, it would be a picture level rate control, and if N_{mbunit} equals to 1, it falls back to a macroblock level rate control,

The total number of basic units in a frame, N_{unit}, is computed by

$$N_{unit} = \frac{N_{mbpic}}{N_{mbunit}} \tag{2-6.82}$$

If the basic unit is not selected as a frame, an additional basic unit layer rate control for the stored picture should be added.

Same as the frame layer rate control, the quantization parameters for IDR picture and non-stored pictures are the same for all basic unit in the same picture. It is computed the similar way as that at frame layer provided that $QP_i(j)$ and $QP_i(j+L+1)$ are replaced by the average values of quantization parameters of all basic units in the corresponding picture.

The basic unit layer rate control selects the values of quantization parameters of all basic units in a frame, so that the sum of generated bits is close to the frame target $T_i(j)$.

The following is a step-by-step description of this method.

Step 1 Predict the MADs, $\tilde{\sigma}_{l,i}(j)$, of the remaining basic units in the current stored picture by model (2-6.79) using the actual MADs of the co-located basic units in previous stored picture.

Step 2 Compute the number of texture bits b_l for the l^{th} basic unit. This step is composed of the following three sub-steps:

Step 2.1 Compute the target bits for the l^{th} basic unit.

Let T_r denote the number of remaining bits for the current frame and its initial value is set to $T_i(j)$. The target bits for the l^{th} basic unit are given by

$$\tilde{b}_l = T_r \times \frac{\tilde{\sigma}_{l,i}^2(l)}{\sum_{k=l}^{N_{unit}} \tilde{\sigma}_{k,i}^2(j)} \tag{2-6.83}$$

Step 2.2 Compute the average number of header bits generated by all coded basic units:

$$\begin{aligned} \widetilde{m}_{hdr,l} &= \widetilde{m}_{hdr,l-1} \times (1-\frac{1}{l}) + \frac{\hat{m}_{hdr,l}}{l} \\ m_{hdr,l} &= \widetilde{m}_{hdr,l} \times \frac{l}{N_{unit}} + m_{hdr,1} \times (1-\frac{l}{N_{unit}}); 1 \le l \le N_{unit} \end{aligned} \quad (2\text{-}6.84)$$

where $\hat{m}_{hdr,l}$ is the actual number of header bits generated by the l^{th} basic unit in the current stored picture. $m_{hdr,1}$ is the

estimation from all basic units in the previous stored picture.

Step 2.3 Compute the number of texture bits $\hat{b}_l$ for the l^{th} basic unit.

$$\hat{b}_l = \widetilde{b}_l - m_{hdr,l} \quad (2\text{-}6.85)$$

Step 3 Compute the quantization step for the l^{th} basic unit of j^{th} picture in i^{th} GOP by using the quadratic R-D model (2-6.80) and it is converted to the corresponding quantization parameter $QP_{l,i}(j)$ by using the method provided by AVC. We need to consider the following three cases:

Case 1 The first basic unit in the current frame.

$$QP_{1,i}(j) = \overline{QP}_i(j-L-1) \quad (2\text{-}6.86)$$

where $\overline{QP}_i(j-L-1)$ is the average value of quantization parameters for all basic units in the previous stored picture.

Case 2 When the number of remaining bits is less than 0, the quantization parameter should be greater than that of previous basic unit such that the sum of generated bits is close to the target bits, i.e.

$$QP_{l,i}(j) = QP_{l-1,i}(j) + \Delta_{Bu} \quad (2\text{-}6.87)$$

where Δ_{Bu} is the varying range of quantization parameter along basic units, the initial value of Δ_{Bu} is 1 if N_{unit} is greater than 8, and 2 otherwise. It is updated after coding each basic unit as follows:

$$\Delta_{Bu} = \begin{cases} 1; & \text{if } QP_{l-1,i}(j) > 25 \\ 2; & \text{otherwise} \end{cases} \quad (2\text{-}6.88)$$

To maintain the smoothness of perceptual quality, the quantization parameter is further bounded by

$$QP_{l,i}(j) = \max\{0, \overline{QP}_i(j-L-1) - \Delta_{Fr}, \min\{51, \overline{QP}_i(j-L-1) + \Delta_{Fr}, QP_{l,i}(j)\}\} \quad (2\text{-}6.89)$$

where Δ_{Fr} is the varying range of quantization parameter along frames, and is defined by

$$\Delta_{Fr} = \begin{cases} 2; & \text{if } N_{unit} > 18 \\ 4; & \text{if } 18 \ge N_{unit} > 9 \\ 6; & \text{otherwise} \end{cases} \quad (2\text{-}6.90)$$

Case 3 Otherwise, we shall first compute a quantization step by using the quadratic model (2-6.80),

and convert it into the corresponding quantization parameter $QP_{l,i}(j)$. Similar to case 2, it is bounded by

$$QP_{l,i}(j) = \max\{QP_{l-1,i}(j) - \Delta_{Bu}, \min\{QP_{l,i}(j), QP_{l-1,i}(j) + \Delta_{Bu}\}\} \quad (2\text{-}6.91)$$

Meanwhile, in order to maintain the smoothness of visual quality, it is further bounded by (2-6.35).

Step 4 Perform RDO for all MBs in the current basic unit and code them by AVC.

Step 5 Update the number of remaining bits, the coefficients of the linear prediction model (2-6.79), and those of the quadratic R-D model (2-6.80).

To obtain a good trade-off between average PSNR and bit fluctuation, Nmbunit is recommended to be the number of MBs in a row for field coding, adaptive field/frame coding, or MB-AFF coding, and Nunit is recommended to be 9 for other cases.

7 Non-normative decoder error concealment description

7.1 Introduction

It is assumed that no erroneous or incomplete slices are decoded. When all received slices of a picture have been decoded, skipped slices are concealed according to the presented algorithms. In practice, record is kept in a macroblock (MB) based status map of the frame. The status of an MB in the status map is "Correctly received" whenever the slice that the MB is included in was available for decoding, "Lost" otherwise. After the frame is decoded if the status map contains "Lost" MBs, concealment is started.

Given the slice structure and MB-based status map of a frame, the concealment algorithms were designed to work MB-based. The missing frame area (samples) covered by MBs marked as "Lost" in the status map are concealed MB-by-MB (16x16 Y samples, 8x8 U, V samples). After an MB has been concealed it is marked in the status map as "Concealed". The order in which "Lost" MBs are concealed is important as also the "Concealed", and not only the "Correctly received" MBs are treated as reliable neighbours in the concealment process whenever no "Correctly received" immediate neighbour of a "Lost" MB exists. In such cases a wrong concealment can result in propagation of this concealment mistake to several neighbour concealed MBs. The processing order chosen is to take the MB columns at the edge of the frame first and then move inwards column-by-column so to avoid a concealment mistake made in the usually "difficult" (discontinuous motion areas, large coded prediction error) center part of the frame propagate to the "easy" (continuous motion area, similar motion over several frames) side parts of the frame.

Figure 3-7.1 shows a snapshot of the status map during the concealment phase where already concealed MBs have the status of "Concealed", and the currently processed (concealed) MB is marked as "Current MB".

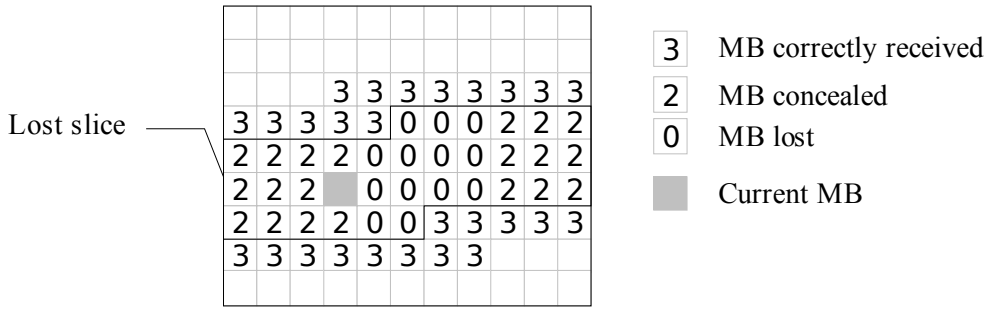


Figure 3-7.1– MB status map at the decoder

7.2 Intra frame concealment

Lost areas in intra frames have to be concealed spatially as no prior frame may resemble the intra frame. The selected spatial concealment algorithm is based on weighted sample averaging presented in A. K. Katsaggelos and N. P. Galatsanos (editors), “Signal Recovery Techniques for Image and Video Compression and Transmission”, Chapter 7, P. Salama, N. B. Shroff, and E. J. Delp, “Error Concealment in Encoded Video Streams”, Kluwer Academic Publishers, 1998.

Each sample value in a macroblock to be concealed is formed as a weighted sum of the closest boundary samples of the selected adjacent macroblocks. The weight associated with each boundary sample is relative to the inverse distance between the sample to be concealed and the boundary sample. The following formula is used:

$$\text{Sample value} = (\sum a_i B\text{-}d_i)) / \sum (B\text{-}d_i) \qquad (3\text{-}7.1)$$

where a_i is the sample value of a boundary sample in an adjacent macroblock, B is the horizontal or vertical block size in samples, and d_i is the distance between the destination sample and the corresponding boundary sample in the adjacent macroblock.

In [Ed. Note: Something missing]

Figure 3-7.2, the shown destination sample is calculated as follows

Sample value = (15x(16-3) + 21x(16-12) + 32x(16-7) + 7x(16-8)) / (13 + 4 + 9 + 8) 18

Only "Correctly received" neighbouring MBs are used for concealment if at least two such MBs are available. Otherwise, neighbouring "Concealed" MBs are also used in the averaging operation.

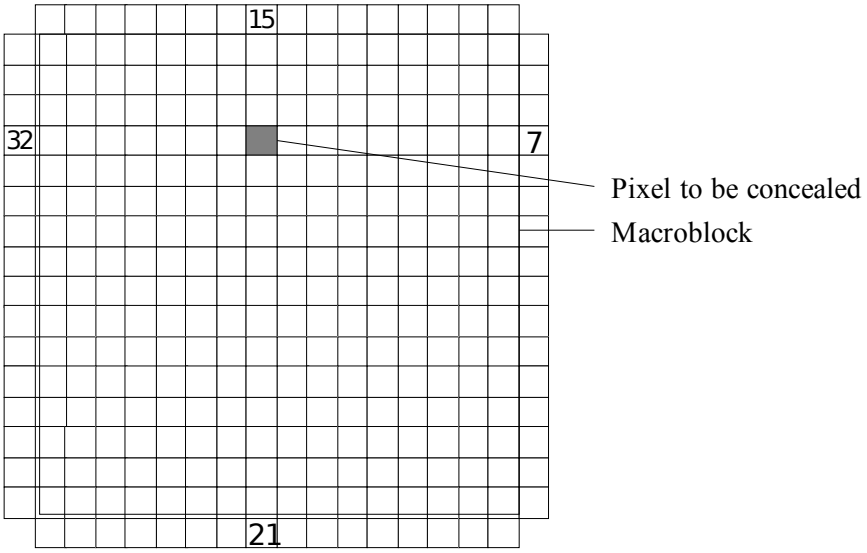


Figure 3-7.2 – Spatial concealment based on weighted sample averaging

7.3 Inter and SP frame concealment

7.3.1 General

Instead of directly operating in the sample domain a more efficient approach is to try to "guess" the motion in the missing sample area (MB) by some kind of prediction from available motion information of spatial or temporal neighbours. This "guessed" motion vector is then used for motion compensation using the reference frame. The copied sample values give the final

reconstructed sample values for concealment, and no additional sample domain operations are used. The presented algorithm is based on W.-M. Lam, A. R. Reibman, and B. Liu, “Recovery of lost or erroneously received motion vectors,” in Proc. ICASSP’93, Minneapolis, Apr. 1993, pp. V417–V420.

7.3.2 Concealment using motion vector prediction

The motion activity of the correctly received slices of the current picture is investigated first. If the average motion vector is smaller than a pre-defined threshold (currently ¼ samples for each motion vector component), all the lost slices are concealed by copying from co-located positions in the reference frame. Otherwise, motion-compensated error concealment is used, and the motion vectors of the lost macroblocks are predicted as described in the following paragraphs.

The motion of a "Lost" MB is predicted from a spatial neighbour MB's motion relying on the statistical observation, that the motion of spatially neighbour frame areas is highly correlated. For example, in a frame area covered by a moving foreground scene object the motion vector field is continuous, which means that it is easy to predict.

The motion vector of the "Lost" MB is predicted from one of the neighbour MBs (or blocks). This approach assumes, that the motion vector of one of the neighbour MBs (or blocks) models the motion in the current MB well. It was found in previous experiments, that median or averaging over all neighbours' motion vectors does not give better results. For simplicity, in the current implementation the smallest neighbour block size that is considered separately as predictor is set to 8x8 Y samples. The motion of any 8x8 block is calculated as the average of the motion of the spatially corresponding 4x4 or other shaped (e.g. 4x8) blocks.

The decision of which neighbour's motion vectors to use as prediction for the current MB is made based on the smoothness of the concealed (reconstructed) image. During this trial procedure the concealment sample values are calculated using the motion vector of each candidate (motion compensated sample values). The motion vector, which results in the smallest luminance change across block boundaries when the block is inserted into its place in the frame is selected. (see Figure 3-7.3). The zero motion vector case is always considered and this copy concealment (copy sample values from the co-located MB in the reference frame) is evaluated similarly as the other motion vector candidates.

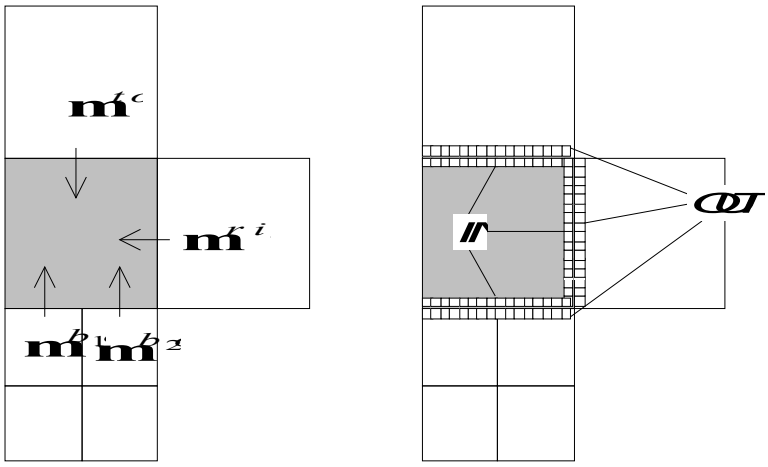


Figure 3-7.3 – Selecting the motion vector for prediction

The winning predictor motion vector is the one which minimizes the side match distortion , which is the sum of absolute Y sample value difference of the IN-block and neighbouring OUT-block samples at the boundaries of the current block:

[Ed. Note: Missing equation]

When "Correctly received" neighbour MBs exist the side match distortion is calculated only for

them. Otherwise all the "Concealed" neighbour MBs are included in the calculation.

7.3.3 Handling of multiple reference frames

When multiple references are used, the reference frame of the candidate motion vector is used as the reference frame for the current MB. That is, when calculating the side match distortion , the IN-block samples are from the reference frame of the candidate motion vector.

7.4 B frame concealment

A simple motion vector prediction scheme according to the prediction mode of the candidate MB is used as follows:

If the prediction mode of the candidate MB is

- forward prediction mode, use the forward MV as the prediction the same way as for P frames.
- backward prediction mode, use the backward MV as the prediction.
- bi-directional prediction mode, use the forward MV as the prediction, and discard the backward MV.
- direct prediction mode, use the backward MV as the prediction.

Note that 1) Each MV, whether forward or backward, has its own reference frame. 2) An Intra coded block is not used as a motion prediction candidate.

7.5 Handling of entire frame losses

TML currently lacks H.263 Annex U type of reference picture buffering. Instead, a simple sliding window buffer model is used, and a picture is referred using its index in the buffer. Consequently, when entire frames are lost, the reference buffer needs to be adjusted. Otherwise, the following received frames would use wrong reference frames. To solve this problem, the reference picture ID is used to infer how many frames are lost, and the picture indices in the sliding window buffer.

8 Acknowledgements

The authors would like to thank the following people for their comments and proposals in this document. They are:

Harvey Hanna II, Alexis Tourapis, Byeungwoo Jeon, Karsten Suehring, Feng Pan, Zhengguo Li, Yun He, Zhibo Chen.