



VSofts H.264 Codec 3.5

Error Resilience Features

Version 1.3

Vanguard Software Solutions, Inc.

895 Jordan Ave.,
Los Altos, CA 94022
(650) 961-3098 (voice)
(650) 292-2340 (fax)
sales@vsofts.com
<http://www.vsofts.com>

1. Version History

Version	Date	Comments
1.0	11/06	Initial version
1.1, 1.2	12/06	Corrections
1.3	01/08	Updated: FMO now works in MT, but disables slice mode#2

2. Introduction

Streaming compressed video over communication channels that introduce errors, losses and excessive delay requires countermeasures to preserve the quality of the viewing experience. This document describes the VSofts technology features that provide these countermeasures and suggested application methods for their use.

When the multimedia stream is subject to damage, what can be done to minimize the visual disruption to the viewer?

- The damage should be limited in extent, both spatial and temporal. In other words, minimize the areas of the screen that are affected, and do not allow the damage to persist or build up over time
- Sprinkle the damage across the screen rather than letting it concentrate in one large area
- Spread the damage over multiple frames rather than concentrating it all in one frame

We refer to the ability of the system to minimize and to quickly recover from the damage as its robustness. VSofts achieves robustness through two separate and complementary techniques: error resilience and error concealment.

Error resilience refers to mechanisms in the VSofts Encoder, that enhance the capability of the compressed bitstream to resist channel errors. When errors and losses arise in the stream, error concealment tools in the VSofts decoder analyze the losses and conceal them in the displayed video by minimizing the visual artifacts.

Error resilience conflicts with compression efficiency. Proper design requires a tradeoff analysis to select the optimal blend. Error resilience tools introduce redundancy in the data. On the other hand, the compression schemes aim to remove both spatial and temporal redundancies from the data. However, removing the redundant information from the video stream makes it more vulnerable to adverse network conditions. For example, in the case of an inter-prediction (a temporal redundancy removal method), a predicted frame (or P-frame) depends on a previous reference frame(s). When errors occur in the reference frame due to packet losses or delay, the subsequent predicted frame will degrade due to error propagation.

3. Error Resilience

3.1 Overview

Error resilience functionality in the encoder produces a bitstream that supports error recovery at the decoder. This overview section discusses three general principles for the implementation of error resilience. The follow on sections describe specific, more elegant methods that are available within VSofts encoders.

3.2 Selective Intra Coding

Frequent emplacement of I-frames in the stream is undoubtedly the most powerful tool to halt error propagation and to recover corrupted pictures. Since the I-frames are compressed based entirely on their own macroblocks, without prediction or reference to other frames, they are able to stand alone. I-frames establish new references for the subsequent prediction process.

The coding efficiency of I-frames is lower than the coding efficiency of predicted frames. Consequently, transmitting an I-frame requires higher bandwidth which results in:

- Delay Jitter – due to longer transmission period on narrow bandwidth channels,
- Bandwidth Excursions due to the suddenly larger bit rate

The encoder can compensate for these effects by drastically reducing the Q Factor in subsequent frames, reducing their quality in order to stay within the bandwidth limits.

3.2.1 VSofts Intra Coded Macroblocks

VSofts uses a more elegant implementation of this principle. Rather than inserting a whole I-frame, the encoder sprinkles a few intra coded macroblocks in the P and B frames. The number and location of these Intra MBs can be configured. The Intra MB update provides a reasonable tradeoff between the error-resiliency and coding efficiency, and can be used with other error resilience schemes. The benefits of this technique include:

- Stable output bit rate
- Less Delay Jitter
- Stable workload for the decoder

There are several methods for dispersing the Intra coded macroblocks in the output stream.

3.2.2 Motion Tracking

The goal of this method is to reduce the visible damage that result from losses to reference frames. In such cases the motion compensation system in the decoder produces comet tail artifacts. VSofts testing validated the effectiveness of this

method, and demonstrated that it provides good error resilience with minimal increase of bandwidth. Breaking and clean up of these “tales” is performed by inserting the intra coded macroblocks according to a motion tracking algorithm, which separately tracks fast motion regions, and updates them more frequently than stable regions.

3.2.3 Row based Distribution

This method accelerates the general recovery from packet losses without regard to motion. One or more horizontal rows of the macroblocks are inserted in the inter frames (P and B). The selected lines move downwards from frame to frame by one row. The number of frames required to “scan” the entire image space is: Frame Height / 16. For example a SD frame of 480 pixels consists of 40 rows. If this method is used with two rows, then a maximum of 20 frames will be needed to clean the corrupted pictures.

The drawbacks of this method are:

- It amounts to a GOP change. In this example the equivalent of an I Frame is sent every 20 frames, significantly decreasing the compression efficiency
- Assuming that the damage is uniformly distributed (any macroblock can be lost with equal probability) then the expected number of frames to correct the error is $\frac{1}{2}$ the interval – or 10 in this example. This number is fairly high.

3.2.4 Random Distribution

This method inserts intra coded macroblocks in every inter coded frame, based on the expected packet loss probability estimate. Greater weight is assigned to the central portion of the frame, because macroblocks in this area are more likely to be references for other predictions. Lesser weight is assigned to the peripheral macroblocks.

3.2.5 Decoder Indicated Loss

This method relies on the decoder to detect missing macroblocks. A separate signaling and control protocol informs the encoder side about missing macroblocks. Upon receiving the indication of the missing macroblock, the encoder inserts an I coded macroblock, or I frame, in the output stream.

This technique can be combined with the others. On its own it is an effective method for low loss, low delay networks because the missing I coded macroblock will be supplied in roughly a network round trip time, which ranges from 50-200ms in most networks. It has the benefit of providing fairly rapid updates without incurring unnecessary overhead of redundant macroblocks.

3.3 Frame segmentation into Slices

A slice is a contiguous collection of macroblocks, encoded from top left to bottom right, such that the group of all slices covers the whole frame. Intra compression requires the decoder to reconstruct spatially encoded macroblocks by reference to

adjacent macroblocks. The extent of referenced macroblocks is limited to their slice. The encoder makes compression decisions on a per slice basis.

Slices carry their own headers. Each slice is sent in its own NAL unit. In support of the resilience objective, the NAL units are transmitted one per network packet.

Because the extent of prediction is limited to one slice, then a single packet loss leads to errors only in that particular slice. The errors do not propagate to other slices. It may appear that decreasing of slice size (so that it contains fewer macroblocks) limits the extent of damage. However, smaller slices exhibit lower coding efficiency:

- Smaller packets carry higher overhead (in the form of slice headers), requiring higher bitrates to carry the same amount of information
- Fewer macroblocks means a lesser opportunity to establish prediction similarities, therefore the residuals will be larger and will require more bandwidth for transmission.

The VSofts encoder provides fine grained control over these parameters in order to regulate the following tradeoffs.

3.4 Constrained Texture Prediction

Inside Inter (P or B) frames there can be some number of intra coded macroblocks, whose texture is predicted from the neighboring macroblocks' texture. The neighboring macroblocks can be inter or intra coded. The inter coded macroblocks are derived from motion compensation and are subject to corruption due to packet losses. In such networks, it is desirable to limit texture prediction by the decoder to using only a set of neighboring intra coded macroblocks.

Using this option one can divide the picture onto the several horizontal stripes in such a way that the macroblocks of each stripe may be predicted only from the co-located stripe of the reference frame. So if the decoding error occurs inside one stripe, then it cannot propagate outside of its boundaries. This method provides a small quality improvement (**~0.5 – 1.0 dB PSNR**) for low levels (<**5%**) of packet loss.

3.5 Flexible Macroblock Order

The FMO tool splits all macroblocks into slice groups, consisting of one or more non contiguous slices. Because they may not be adjacent to each other, the macroblocks may be transmitted out of raster scan order. However, the macroblocks of a particular slice group are transmitted in raster order.

VSofts implements 2 types of slice grouping: checkerboard and interleaving.

- In the checkerboard grouping, alternate macroblocks belong to two different groups. The "white" macroblocks are transmitted in first slice group, and the "black" ones in the second. This randomizes the effects of losses throughout the frame.

- In interleaving, the frame is divided into with 2 to 7 groups. The macroblocks are transmitted in row order within each group. For example, with 2 groups, the even macroblocks rows are transmitted in first slice group, and the odd rows in the second.

3.6 Arbitrary Send Order

The ASO tool randomizes the transmission order of macroblocks in network packets. Rather than building the frame in scan order, the tool attempts to spread the frame over the interval of frame transmission time.

VSofts measurements determined that the ASO tool requires many cpu cycles but that it does not provide a commensurate increase in resilience over the other techniques described here. Therefore, VSofts does not support this tool.

4. Error Concealment

Error concealment refers to the actions taken by the decoder to restore in some way the texture coded in lost packets, by concealing their artifacts. The concealment schemes can be spatial, temporal or combined. In spatial interpolation, the values of missing pixels are estimated from the surrounding pixels of the same frame, without using the temporal information. On the other hand, the temporal interpolation is based on the corresponding regions of the reference frames. If a motion vector is missing, it can be calculated based on the motion vectors of the surrounding regions. Combined scheme use some adaptive mechanism to choose best concealment for each lost macroblock.

The VSofts decoder uses the following schemes. They render the best results in combination with error resilience implementations, but they can also be useful on their own.

4.1 Spatial Texture Interpolation

The spatial texture reconstruction method recovers from macroblock losses by rapidly estimating the texture from adjacent macroblocks. It consists of numerous interpolation methods and provides fast and efficient texture gaps recovery for arbitrarily shaped regions, and arbitrary structure of the surrounding admissible areas. By its nature this technique is best for interspersed macroblock losses, such as would result from FMO. Large contiguous areas do not recover as well.

4.2 Motion Compensation

This technique recovers by interpolating the motion vectors for all macroblocks of the lost slice from the motion vectors of the neighboring decoded slices. Thus the motion of the lost texture regions is recovered and the texture is reconstructed by the corresponding motion compensation. This technique yields satisfactory results, smooth appearance and no disruptive tails.

4.3 Severe Loss Compensation

Faced with severely corrupted high motion sequences, coded without proper error resilience, the decoder faces a dilemma. If the decoder suspects that the lost slices will cause noticeable visual artifacts in future frames, from which it cannot recover, then it has the choice:

- It can freeze the image area till the next point of full recovery, or
- It can continue to show the results of decoding with as much error concealment as is available.

The first path yields correctly decoded frames at some future point. In the intervening period all motion updates are lost, and the actual output frame rate for large image areas may drop to match the rate of key frames. In the second case the output may contain ugly texture regions but it will reflect the events happening on the motion picture more or less well.

The current version of the decoder always selects the second option. Future versions will provide a configuration parameter for this behavior, because for low motion sequences, the first behavior is more pleasing.

5. Resilience Implementation in the Encoder

5.1 Intra macroblocks update based on motion tracking

To enable this method set **error_resilience.intra_update_method = 1** in the encoder settings.

The intensity and motion sensitivity of the intra update is controlled by three encoder settings parameters:

- **error_resilience.initial_expected_loss_percent**,
- **error_resilience.fast_motion_update_period**,
- **error_resilience.full_motion_update_period**

The intra update algorithm tracks separately fast motion regions and the regions of any nonzero motion, as follows:

5.1.1 Fast motion tracking

The encoder inserts intra macroblocks every **error_resilience.fast_motion_update_period** frames or doesn't insert any if **error_resilience.fast_motion_update_period = 0**. VSofts recommends keeping this parameter value 1 or 2.

5.1.2 Slow nonzero motion tracking

The encoder inserts intra macroblocks every **error_resilience.full_motion_update_period** frames or doesn't insert them at all if **error_resilience.full_motion_update_period = 0**.

Slow motion tracking implies more intensive intra macroblocks insertion, so this parameter value should be sufficiently high (≥ 5) or zero.

5.2 Row Based Distribution

To enable this method set `error_resilience.intra_update_method = 2`.

5.3 Random Distribution

To enable this method set `error_resilience.intra_update_method = 3`.

5.4 Frame Segmentation into Slices

The following parameters control how the frame is decomposed into slices:

- `slice.mode`
- `slice.param`
- `slice.i_param`
- `slice.b_param`.

Value of <code>slice.mode</code>	Meaning
0	Disabled - No slicing
1	Specifies the slice size as a number of macroblocks
2	Specifies the slice size in bytes
3	Specifies the number of slices

The `slice.param` settings do not apply to `slice.mode "0"`.

For `slice.mode` values **1,2**, and **3**, the value of `slice.param` is applied to all frame types (I,P, and B) unless the next two parameters are also specified, in which case it will apply only to the P frames. The meaning of the `slice.param` depends on the meaning implied by the `slice.mode`.

- `slice.i_param` implies the value for I frames
- `slice.b_param` implies the value for B frames

The recommended setting for error resilience is `slice.mode = 2`. In this case encoder will construct the slices not to exceed the specified byte limit. The desired number of bytes in slice is set in `slice.param`. This parameter should be set depending on typical size of the network packets used in the application.

5.5 Constraints for inter texture prediction

This constraint flag is set automatically when Error Resilience is enabled, because without it, error recovery at the decoder is not effective.

It can be invoked separately, without enabling error resilience, by setting for encoder `intra_pred = 1`. However, it is not an effective resilience tool, used on its own.

To specify the rectangular motion constraints set the parameter **error_resilience.me_region_height_in_mbs** to an integer 1..N representing the vertical number of macroblocks per stripe.

5.6 FMO¹

FMO is an H.264 Baseline profile feature. It specifies a pattern that assigns macroblocks in a picture to one or several slice groups. Each slice group is transmitted separately. If a slice group is lost, the samples in spatially neighboring macroblocks in other slice groups can be used for efficient error concealment. The supported patterns range from rectangular patterns to regular scattered patterns, such as chess boards, or to completely random scatter patterns.

The **fmo.num_slice_groups** parameter enables FMO if a value greater than "1" is selected. We recommend a value of "2".

Value of fmo.type	Meaning
0	Interleaved
1	Dispersed (recommended)

The following parameters have the same meaning as the corresponding slice parameters above,

- **fmo.internal_group_slice_mode**
- **fmo.internal_group_slice_param** for P and I frames (unless next parameter is specified)
- **fmo.internal_group_slice_i_param** for I frames only
- Since there are no B frames in the Baseline Profile, there is no need for a value of 3.

Value of fmo.internal_group_slice_mode	Meaning
1	Specifies the slice size in macroblocks
2	Specifies the slice size in bytes

6. Error Concealment Settings in the Decoder

6.1 Standard Concealment

In order to correctly initialize the decoder, it must first receive the following NAL units: **SPS**, **PPS** and at least one **Intra** slice (not necessarily the whole **Intra** frame). If this packet is lost, the stream will appear to halt, and no output will be generated.

To enable the error concealment set

¹ Release note: FMO disables slice mode #2, mode #3 is recommended to be used instead

- **settings.full_error_concealment_enable = 1**
- **settings.mt_num_threads = 0**

Note: The decoder reconstructs and outputs only the frames for which at least one slice was successfully received. If complete frames are lost, the upper layer application must determine the course of action.

6.2 Lost Macroblock Indications

This feature will be available at a future time.

7. Example of Encoder Settings with Error Resilience

This is just one of the possible encoder configurations showing error resilience settings. It contains most of the encoder parameters (see the encoder settings manual for the full description of all the possible settings of the version 3.5. The values of the parameters, which are not included here, will be considered as default in the encoder). They can be changed according to the coding requirements.

7.1 General Settings

```
profile_idc = 77 // main profile
level_idc   = 32
sym_mode    = 1
intra_pred  = 1
```

```
high_compression_mode = 0x061 //or 0x60, or 0x20,or 0x21.
gop.keyframes         = 30
gop.num_units         = 10001
gop.time_scale        = 300000 // frame rate
```

```
rc.type           = 2 // one can also try VBR – the value 1.
rc.kbps           = 400
rc.auto_qp        = 1
```

```
me.subdiv         = 7 // 7 or 9, or 15
me.flags          = 3 // should be set to 3 for now
me.range          = 16 // 16 seems to be the optimal almost everywhere
me.max_refs       = 1
```

```
slice.mode        = 2 // bytes per slice
slice.param       = 1200
```

7.2 Resilience Settings

```
error_resilience.enable = 1 //set to zero to disable error resilient coding
error_resilience.initial_expected_loss_percent = 10 // expected loss percent -
5,15, etc.
```

```
error_resilience.intra_update_method = 1
```

```
error_resilience.fast_motion_update_period = 1 //one can try also 2 or 0
```

```
error_resilience.full_motion_update_period = 6 // one can try also 8, 10, or 0
```

error_resilience.me_region_height_in_mbs = 0

7.2.1 Lossy Networks Example

On lossy networks, VSofts recommends to set

- **error_resilience.full_motion_update_period** to a value between 6-10.

For higher visual quality set

- **error_resilience.full_motion_update_period** = 10
- **error_resilience.fast_motion_update_period** = 2

For higher resilience (and lower compression efficiency) set

- **error_resilience.full_motion_update_period** = 6
- **error_resilience.fast_motion_update_period** = 1.