3-D DWT Software Manual

Author: Evgeny Belyaev
Version: 3-D DWT 2.2
Last update: January 23, 2025.

1 Introduction

This document contains a description of the usage of software for video encoding, decoding and bit stream extraction for video codec based on three-dimensional discrete wavelet transform (3-D DWT).

2 Video encoding

2.1 Running using command line and encoding parameters

The encoder should be run in a command line as it is shown in this example:

WaveletCoder.exe config.cfg [encoding parameters]

Here *config.cfg* is configuration file name, and *[encoding parameters]* are parameters list which are different that those in the configuration file. The configuration file is a text file consisting one parameter or one comment per line. A comment line should start from symbol %, while an encoding parameter should be written in the following way: *ParameterName=Value* (without other symbols, for example, space symbol). In a command line encoding parameters should be written in the following way *ParameterName1=Value1 ParameterName2=Value2* and so on.

Encoding parameters are illustrated in Table 1.

Table 1: Encoding parameters

Parameter name	Meaning	Value
InputFile	Input file name	Input file should be in YUV 4:2:0 colorspace format
OutputFile	Output file name	
ViewsNumber	Number of video sources which have to be jointly encoded	$0, ..., 255$
SourceWidth	Frame width	*SourceWidth*%16 should be zero
SourceHeight	Frame height	*SourceHeight*%16 should be zero
FramesToBeEncoded	Number of frames which should be encoded	
FrameRate	Frames per second in an input video	
GlobalMotionCompensation	1 enables Motion-Compensated Temporal Filtering (MCTF) based on global motion	0 or 1
Spatial97Wavelets	0 enables 5/3 spatial DWT, while 1 enables 9/7 spatial DWT	0 or 1
NumWaveletDecom	Number of spatial wavelet decompositions (3, 4 or 5). Be sure that the input resolution can be decomposed	3,4 or 5
rcmode	0 enables constant quality mode, 1 enables constant bit rate mode	0 or 1
Target	If *rcmode*=0, then *Target* defines quality level (lower values means better quality). In case of *Target*=0 and 5/3 spatial DWT an input video will be encoded in lossless mode. If *rcmode*=1, then Target defines target bit rate in kbps.	
LossProtectionMode	0 – redundant packets are not inserted, 1 - authomatic unequal loss protection via Reed-Solomon (RS) codes, 2 – manual loss protection via RS codes	0,1 or 2
ploss	Target packet loss rate. If *ploss*>0 and $LossProtectionMode = 1$, then encoder is trying to find the best inter-packet RS codes in order to maximize the quality for the target *ploss* value	0,...,1.0
RScode1,*RScode2*, *RScode3*, *RScode4*	RS codes for four classes of packets. They are used only if $LossProtectionMode = 2$	0,..,12
MaxPacketSize	Compressed bit stream is represented as sequence of packets, so that each packet size is less than *MaxPacketSize*+7 bytes	200-1500

2.2 Example of configuration file

Configuration file "soccer.cfg" for video "soccer.yuv", $704 \times 576@60Hz$.

```
%Input file name (yuv 4:2:0 only supported)
InputFile=soccer.yuv
%Input file name (yuv 4:2:0 only supported)
OutputFile=stream.wfc
%Number of views
ViewsNumber=1
%Input frame width
SourceWidth=704
%Input frame height
SourceHeight=576
%Number of frames to be coded
FramesToBeEncoded=600
%Input video frame rate, Hz
FrameRate=60
%Global motion compensationed temporal filtering (0 - disable, 1 - enable)
GlobalMotionCompensation=1
%Spatial 97 lifting wavelet decomposition (0 - disable, 1 - enable)
Spatial97Wavelets=0
%Rate Control Mode (0 - constant quality, 1 - constant bitrate)
rcmode=1
%Target bitrate in (kbps) or quality 0...10000 (higher value means lower quality)
Target=4000.0
%Loss protection mode (0 - no protection, 1 - Unequal LP, 2 - Equal LP)
LossProtectionMode=0
%Target packet loss for inter-packet Reed-Solomon loss protection
ploss=0.0
%Maximum packet size, bytes
MaxPacketSize=800
```

Using mentioned above configuration file the video encoder for video file soccer.yuv can be run using the following command line:

WaveletCoder.exe soccer.cfg

To encode this video at bit rate 2000 kbps (not as in "soccer.cfg"), the same configuration file can be used, but parameter *Target* should be added in command line:

WaveletCoder.exe soccer.cfg Target=2000

In case of multi-video encoding, *InputFile=basename.yuv* should contain a base name for all videos, while each video should be in the files with names *basename_0.yuv*, *basename_1.yuv*, *basename_2.yuv* and so on.

3 Video decoding

The decoder should be run in a command line as it is shown in this example:

WaveletCoder.exe decoder.cfg [decoding parameters]

Decoding parameters are illustrated in Table 2.

Table 2: Decoding parameters

Parameter name	Meaning	Value
InputFile	Input bit stream size name	
OutputFile	Output reconstructed yuv file name	
SpatialLevel	Decoder extracts all spatial layers with ID<=SpatialLevel	0,1,2,3,4. SpatialLevel=4 can be used only for videos with resolution 1920 × 1088 and higher
TemporalLevel	Decoder extracts all temporal layers with ID<=TemporalLevel	0,1,2,3,4
QualityLevel	Decoder extracts all quality layers with ID<=QualityLevel	0.0, 0.25, 0.5, 0.75, 1.0, ..., 3.0
ISTADecoding	ISTA recovery at the decoder (off-line decoding with higher quality)	0,1
ISTAiter	Number of ISTA iterations	> 1

In Table 2 all layers are numbered starting from basic (more important) ones. For example, SpatialLevel=0 corresponds to the smallest frame resolution, SpatialLevel=1 means frame resolution so that frame width and height are twice higher than for SpatialLevel=0.

Example of configuration file "decoder.cfg"

```
InputFile=stream.wfc
OutputFile=dec.yuv
SpatialLevel=3
TemporalLevel=4
QualityLevel=3
```

Using mentioned above configuration file the video decoder can be run using the following command line:

WaveletCoder.exe decoder.cfg

To decode this video with half frame resolution, the same configuration file can be used, but parameter *SpatialLevel* should be added in command line:

WaveletCoder.exe decoder.cfg SpatialLevel=2

4 Bit stream structure

Output bit stream contains sequence of packets ready for the transmission over a packet network. Each packet has 7 bytes of header part and payload part. The header structure is presented in Table 3.

Table 3: Packet header structure

Parameter	Meaning	Value definition using C	Range
GOPIndex	All frames of an input video are splitted into group of frames (GOP). Each GOP consist of 16 frames. This field indicated a GOP index	buff[0]	0,..,255
PacketNumber	Number of packet in GOP. If PacketNumber%16<4 then the packet is Reed-Solomon parity packet	(buff[2]≪8)+buff[1]	$0,..,2^{16}-1$
PacketSize	Packet size in bytes, including a packet header	(buff[4]≪8)+buff[3]	$0,..,2^{16}-1$
QualityLayerIndex	Quality layer	(buff[5]≫6)	$0,..,3$
SpatialLayerIndex	Spatial layer	((buff[5]≫3)&7)	$0,..,4$
TemporalLayerIndex	Temporal layer	(buff[5]&7)	$0,..,4$
Protection level	This parameter shows Reed-Solomon protection level for the packet	buff[6]	0,..,11

Payload part consist of sequence of subpackets. Each subpacket has 7 bytes of header.

Table 4: Subpacket header structure

Parameter	Meaning	Value definition using C	Range
SubPacketSize	Subpacket size in bytes, including subpacket header	(buff[1]≪8)+buff[0]	$0,..,2^{16}-1$
ViewIndex	View (camera) index	buff[2]	$0,..,255$
GOPIndex	See GOPIndex in Table 3	buff[3]	$0,..,255$
SubPacketIndex	Subpacket index	buff[4]	$0,..,255$
FrameIndex	Frame index within GOP	buff[5]&0xF	$0,..,15$
YUV	Color component	(buff[5]≫4)&0x3	0,1,2
SubbandIndex	Wavelet subband index	(buff[5]≫6)&0x3	0,1,2,3
DecompLevel	Wavelet decomposition index	buff[6]&7	0,1,2,3

If *ViewIndex*=0, *FrameIndex*=0, *YUV*=0, *SubbandIndex*=3, and *SubPacketIndex*=0 then the following additional 10 bytes of the stream header are placed after subpacket header.

Table 5: Stream header structure

Parameter	Meaning	Value definition using C	Range
FrameWidth	Video frame width	buff[0]+(buff[1]≪8)	
FrameHeight	Video frame height	buff[2]+(buff[3]≪8)	
DecompLevelNumber	Number of wavelet decomposition levels	buff[4]	3,4
ViewsNumber	Number of views (cameras) in the stream	buff[5]	0,..,255
MotionCompensation	Global motion compensation flag	buff[6]	0,1
Lifting97Transform	Spatial wavelet transform mode	buff[7]	0,1
MaxPacketSize	Maximum packet size in bytes	buff[8]+(buff[9]≪8)	200,..,1500

5 Bit stream extracting

As it was described above, each packet has *QualityLayerIndex*, *SpatialLayerIndex* and *TemporalLayerIndex*, so that the video bit stream can be truncated according to the required spatial resolution, frame rate and frame quality. To realize it, all packets which have *QualityLayerIndex*> *QualityLevel*, *SpatialLayerIndex*>*SpatialLevel* and *TemporalLayerIndex*> *TemporalLevel* should be dropped. In this case, 4–5 frame different resolutions, 5 frame rates and 4 frame quality levels can be achieved.

In some cases, a higher frame quality levels can be required. To achieve it, more important wavelet subbands can be decoded at quality $\lfloor QualityLevel \rfloor+1$, while the remaining subbands at quality $\lfloor QualityLevel \rfloor$. Using this approach 12 different frame quality levels can be achieved. In this case *QualityLevel* should be a floating point number between 0 and 3 with step 0.25. Algorithm 1 illustrates bit stream extraction for given *QualityLevel*, *SpatialLevel* and *TemporalLevel* with more details.

Algorithm 1 Bit stream extraction. Input: *QualityLevel*, *SpatialLevel*, *TemporalLevel*

```
1: L ← getNumDecompLevels()
2: q_min ← ⌊QualityLevel⌋
3: if QualityLevel−q_min > 0 then
4:    q_max ← q_min + 1
5: else
6:    q_max ← q_min
7: end if
8: s_max ← ⌊(L + 1) × (QualityLevel − q_min)⌋
9: for each packet do
10:   q ←ReadPacketQualityLayerIndex()
11:   s ←ReadPacketSpatialLayerIndex()
12:   t ←ReadPacketTemporalLayerIndex()
13:   skip ← 0
14:   if s > SpatialLevel then
15:      skip ← 1
16:   else
17:      if s < s_max then
18:         if q > q_max then
19:            skip ← 1
20:         end if
21:      else
22:         if q > q_min then
23:            skip ← 1
24:         end if
25:      end if
26:   end if
27:   if t > TemporalLevel then
28:      skip ← 1
29:   end if
30:   if skip == 0 then
31:      StorePacket ()
32:   end if
33: end for
```