

Title: Enhanced variable-length coding

Status: Input Document to JVT

Purpose: Proposal

Author(s) or Till Halbach

Contact(s):

Tel.: +47 - 73 59 44 88

Fax: +47 - 73 59 26 40

Email: halbach@tele.ntnu.no

Source: Norwegian University of Science and Technology (NTNU)
(Department of Telecommunications)
Trondheim, Norway

Abstract

This contribution investigates error resilience properties of two codes, UVLC and VLCD. UVLC has been used so far in H.26L standardization, and VLCD is a reversible VLC that offers error resilience/concealment possibilities which are superior to UVLC.

Intellectual Property Rights

The author is not aware of any IPRs that are connected to the proposed techniques. For more information, the JVT Patent Disclosure Form is attached to the document.

Introduction

Considering error-prone environments like IP-like data transport or 3GPP-specified mobile channels, H.26L is mainly focusing on packet losses. The philosophy is that, once a disturbed packet is recognized as such, the whole packet content should be discarded because of the difficulties to locate or even correct/conceal the errors.

For treatment of residual bit/burst errors, H.26L also specifies that its algorithms have to be error-resilient. It is further not obvious which approach – packet discarding or residual errors – outperforms the other. If there are delay constraints, however, residue errors are often imperative. Moreover, so-called UDP light may let decide the application if it would be wiser to retransmit or cope with errors.

Error resilience is mainly addressed by the inherent self-synchronizing properties of UVLC. However, these properties are limited, and – to the author's knowledge – no attempts have been made so far to exploit the advantages UVLC offers. A new code is therefore proposed to replace UVLC as entropy code.

The document is structured as follows: It begins with a summary of the self-synchronization properties of UVLC. It then investigates the bit stream and source symbol statistics of H.26L, and finally introduces VLCD and its properties. Necessary changes of the current draft document in order to employ VLCD and exploit its properties are described as well.

UVLC coding and decoding

Work has already been done on this topic in [VCEG-L23]. This section summarizes previous research and adds the results of the discussion VCEG had on its email reflector recently.

UVLC consists of synchronization (sync) bits, zeros with a terminating 1-bit to mark the end of the code word, that are interleaved with so-called info bits x_i in the form $(0 x_2 0 x_1 0 x_0 1)_2$. Hence, UVLC is a comma code. The info part consists of a plain binary code. The code words for the first eight indices are given in the following table.

Code book index i	Code symbol X
0	1
1	001
2	011
3	00001
4	00011
5	01001
6	01011
7	0000001
⋮	⋮

UVLC, first introduced in [VCEG-F11], is quite similar to exponential reversible Golomb Rice codes for coding of non-negative integers. It has self-synchronizing properties which are investigated in the following. To cut down complexity, only one single bit error, i.e. bit inversion, is considered.

It is noted that the code is systematic, which means there exists a mapping rule from the index to the binary code word; use of a code book under encoding and decoding is thus not necessary. Further, the code book is not limited in size. This makes UVLC a complete code. From the knowledge that UVLC code words are of odd length, one would expect that only an odd number of merged code words can make a new code word. This leads also to the fact that, after bit error occurrence, sync bits become info bits, and the other way round. The existence of a sync code word $(1 1)_2$ guarantees, i.e. is sufficient for, statistical synchronization of UVLC [Max85].

First, there is the trivial case when bit errors affect the code stream at info bit positions. This changes the value of a single code word but does not lead to sync loss, e.g. a bit error at position 2 in $(0, 0, 0, 0, 1)_2$ changes the index from (3) to (5). Since UVLC is a complete code, error detection cannot be made on a bit basis but only on semantics, i.e. the index might be out of range or a decoded motion vector could point outside the corresponding picture.

If a 0-sync bit is inverted and the info bits of the *affected* code word are all zero, then two symbols are altered, but sync is not lost: (1, 3) encoded gives $(0, 0, 1, 0, 0, 0, 0, 1)_2$, disturbed e.g. $(1, 0, 1, 0, 0, 0, 0, 1)_2$ and hereby the indices (0, 11). If there is at least one 1-info in the symbol, sync is first regained after three code words: (2, 1, 1) encoded becomes $(0, 1, 1, 0, 0, 1, 0, 0, 1)_2$, disturbed e.g. $(0, 1, 0, 0, 0, 1, 0, 0, 1)_2$, which results in only one index (25). Hence, three code words are merged to a single one, and one gets a symbol assignment problem.

Finally, if a 1-sync bit is inverted, there is again the case that sync is not lost if the info bits of the *following* code word contain at least one 1-info: (2, 5) produces the code $(0, 1, 1, 0, 1, 0, 0, 1)_2$ which may then be altered by a bit error to $(0, 1, 0, 0, 1, 0, 0, 1)_2$, which in turn gives (5, 1). If the code word's info bits contain only zeros, however, symbol splitting takes place, meaning sync loss, which is the inverse procedure of the symbol merging discussed above.

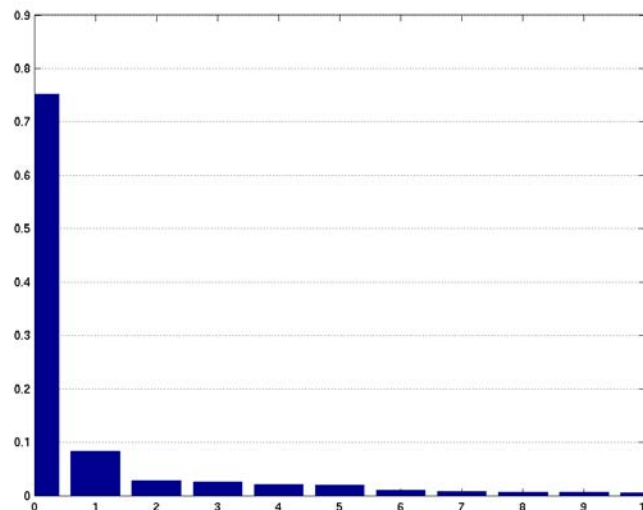
One might argue that UVLC is in addition non-instantaneously reversibly decodeable. As an example, decoding of the symbols (0, 1, 2, 0) is considered, following the procedure below. It can be seen that UVLC is not reversibly decodeable in an instantaneous manner.

Known code stream	Description
1 0 0 1 0 1 1 1	Original
x x x x x x x <u>1</u>	Parse each second bit backwards
x x x x x <u>1</u> x 1	Next bit to the right must be sync
x x x x x 1 <u>1</u> 1	'1' denotes code word end as usual
x x x x <u>0</u> 1 1 1	
x x <u>0</u> x 0 1 1 1	Continue backwards
<u>1</u> x 0 x 0 1 1 1	Turn again to forward decoding
1 <u>0</u> 0 x 0 1 1 1	
1 0 0 <u>1</u> 0 1 1 1	Finished

It is concluded that, given an exponential source distribution – which is shown to be a valid assumption in the following Section – UVLC is matched very good to the source statistics with regard to coding efficiency. However, its resync property of UVLC is useless since the error cannot be localized, for code word merging and splitting cannot be avoided, and this means that the whole bit stream between two known unique resynchronization markers has to be discarded. Backward decoding achieves no gain with respect to error resilience since it partly consists of forward decoding.

H.26L's source symbol and code stream statistics

The following statistics show a typical test case using TML-9.0, together with the coding efficiency conditions commonly used in H.26L standardization [VCEG-N81]. Transport markers and start codes have been removed from the bit stream beforehand and are not included in the calculations. 260 pictures of the sequence *Tempete* in CIF format are encoded, with frame skip 0 and quantization parameter 28. The distribution of code word indices is strongly exponential, which can be seen in the following figure which shows the source symbol probabilities over the code word index.



The symbol with index zero occurs in about 75 of 100 cases. The first 16 code words cover a probability of occurrence of 97.46%, and the accumulated probability is 99.74% for the first 128 code words in the code table. The maximum 'outlier' has the value 513, which relates to a code word length of 19 with the info being $(0, \dots, 0, 1, 0)_2$. The first source symbol never used has the index 268. It should further be mentioned that the relative occurrence of 1-sync bits is approximately 43% for 1-sync, 35% for 0-sync, and 22% for info bits. Code words with all info bits equal to zero occur statistically in about 12 of 100 cases. This allows calculation of the probability of synchronization loss, given a single bit error:

$$\Pr_{\text{sync loss}} = \Pr_{1\text{-sync}} \Pr_{0\text{-info}} + \Pr_{0\text{-sync}} \Pr_{1\text{-info}} \cong 9.7\%,$$

where e.g. $\Pr_{1\text{-sync}}$ denotes the probability of occurrence of a 1-sync bit (accordingly for the other probabilities). This is significantly below the probability of sync loss for e.g. a Huffman code, which is close to one. It is concluded that UVLC cuts down the probability for sync loss just by its inherent code structure. However, the encoded test sequence's entropy – which can easily be computed since the source symbols are uncorrelated – is 1.7646 bit/SS while UVLC yields an overall code rate of 2.3112 bit/SS, which means a gap of 0.5476 bit/SS.

This leaves space for improvements since, by closing this gap and hereby increasing compression efficiency, on the average 3.4 Kbit/s in bandwidth could be saved, assuming 25 fps video transmission.

VLCD encoding and decoding

As a promising remedy to enhance error localization in the bit stream, it is proposed to replace UVLC by a reversible code, called here VLCD. This is accomplished by replacing UVLC's sync bits format by $(0, x_2, 1, x_1, 1, x_0, 0)_2$, the rest of the structure is maintained. The code table is shown in the following:

Code book index i	Code symbol X
0	1
1	000
2	010
3	00100
4	00110
5	01100
6	01110
7	0010100
⋮	⋮

VLCD's sync bits are symmetric, and VLCD code words fulfill the biprefix condition, which makes the code (instantaneously) reversibly decodeable. Systematic code word generation is maintained, further, the code book may become infinitely large. VLCD is also used for coding motion vector data in H.263, Annex D.

VLCD has obviously the same coding efficiency like UVLC, but, as shown below, not its self-synchronizing property. In fact, sync cannot be regained after occurrence of a single bit error before the entropy decoder reaches the end of the bit stream. This information should hence be known at the decoder side either by knowing the exact number of bits to decode or by reaching

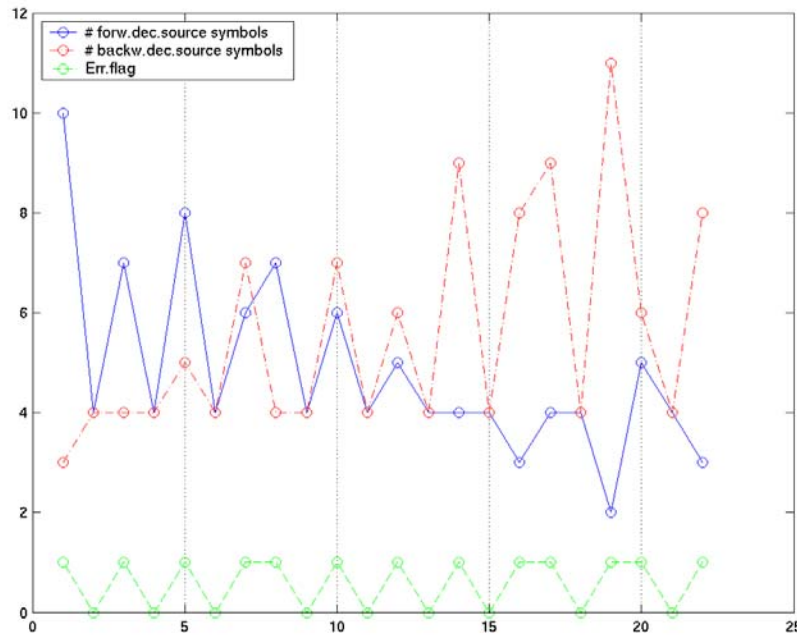
a unique long resynchronization marker. Assuming the decoder is out of sync, two cases are possible to attempt to re-establish synchronization:

a) If the decoder checks y_2 as a sync bit in $(\dots, y_2, y_1)_2$, the next bit that resyncs decoding is a $(0)_2$.

b) If y_1 is assumed to be sync bit, then a resync symbol $(z, 0)_2$ has to be appended, where z could be chosen 0 to cover the first case as well.

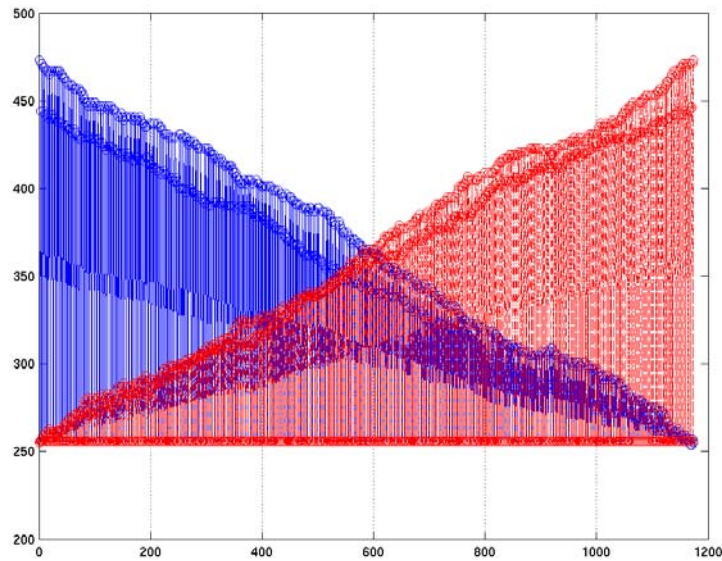
However, the second zero in $(0, 0)_2$ would start decoding a new code word in a). Hence, a short resync symbol does not exist. This guarantees in turn that resync does not happen [Max85].

The decoder will flag an error because some bits will remain at the end of the bit stream.

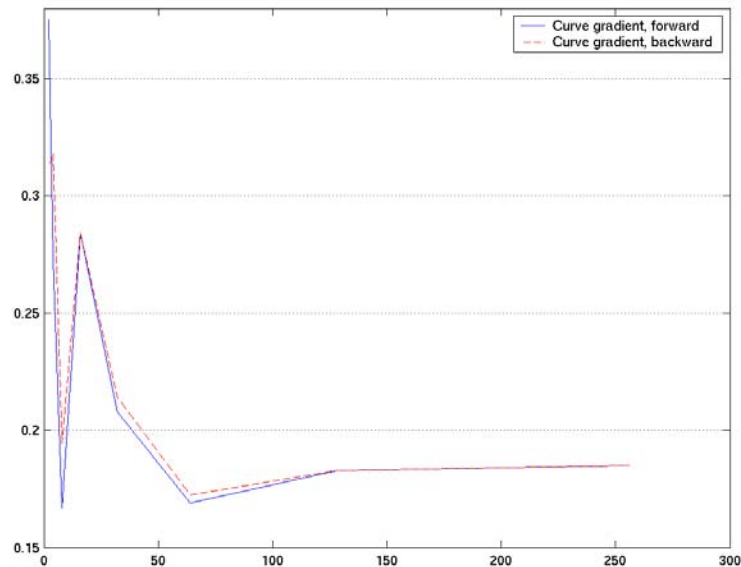


As an example, the last Figure shows three curves for the decoding of four VLCD symbols when a single bit error has damaged the code stream: The number of source symbols counted in forward and backward direction, and the error flag position over the error's bit position in the code stream. It is stressed that, because of the guaranty of nonresync and the symmetry of the sync bits, the error flag takes on the same values for both directions. It can be seen that the flag always detects the bit error when it hits a sync bit position. Hence, it can be used for error detection purposes.

One can further observe the tendency of the decoder to decode shorter code symbols when the decoder is in out-of-sync state. The earlier the error hits, the more source symbols are produced by decoding. This effect is favoured by the statistical distribution of zeros and ones in sync and info bits. After a bit error, info bits in the following code word are treated as sync bits and the vice versa. For more code words or, respectively, more bits, the tendency can be observed very clearly, see next Figure. Here, the typical UVLC code stream as introduced above is VLCD encoded. For the Figure's plot, 512 source symbols or approximately 1200 bits are extracted.



The last observation leads to the definition of a source symbols number gradient. Looking at one of the last two Figures, it is striking that the curves that represent the number of source symbols have an almost linear increasing or decreasing gradient. This suggests the gradient to be equal to the difference of maximum and minimum source symbol number and the number of bits. If the number of source symbols in both directions is approximately equal, then the error is found in the middle of the bit stream. The gradient allows an estimation of where the error has hit the bit stream for possible error concealment attempts. However, the value of the gradient depends of course on distribution of source symbols and encoder parameters. It might hence be necessary to signal the gradient's value in the header data.



Another surprising fact is that the gradient converges for a sufficient number of bits in the code stream. This is visualized in the last Plot, where the gradient's absolute value is plotted over the number of bits in the code stream. This means, the less bits there are in the stream, the more the gradient is influenced by statistical code word distribution variations. More, the gradients in forward and backward direction are equal in their absolute value. After computation of the

number of decoded source symbols in one direction – assuming a single bit error and a raised error flag – the decoder is enabled to know approximately where the error position actually is. Decoding in forward and backward direction can then be terminated with a certain safety margin to the error's location.

Also, a possible error concealment technique is illustrated for data recovery after random bit error occurrence. It is a brute-force attempt and is demonstrated for a single bit error. The decoder tries to locate the error position by cycling through the bit stream in a bit-by-bit manner and simulating that another error has hit, i.e. it adds a bit on a certain location in mod-2 arithmetic. It then decodes e.g. forwardly and keeps track of the position of the estimated error and of the error flag. Some examples shall be considered here:

Error position 05: 5
Error position 12: [12, 20, 24]
Error position 14: [14, 17, 23, 28, 31, 34, 37]
Error position 16: [16, 22, 26, 29]
Error position 19: [19, 41]
Error position 20: [12, 20, 24]
Error position 21: 0

.
. .
.

[12, 20, 24] means for example that the possible error positions 12, 20 and 24 have been computed. It is obvious that a second error can wipe out the original one only when it hits the neighborhood. This allows to compute the safety margin mentioned above to cancel decoding operation.

Impact on today's H.26L draft model

In TML-9.0, there are mainly three data type families. Header data like e.g. the macroblock type is entropy-encoded as is, i.e. assumed memory-free. Motion vector data is differentially encoded, and the quantized and scanned transform coefficients are variable-length-encoded after prior run length coding.

VLCD can be employed easily by replacing the old index-to-CW mapping of UVLC with a mapping that orders the sync bits as shown in the table for VLCD above. The code word's info part and its length stay the same. To be able to exploit the possibilities VLCD offers, i.e. first of all to detect and locate the error roughly and then help recover parts of the bit stream, some changes have to be made to encoding of the data types mentioned above. It is assumed subsequently that symbol errors are subject to propagation due to prior info bit errors.

Header data is most sensitive to transmission errors. Here, it would make sense to spend computation time and complexity for computing an accurate error margin as discussed above. However, single header source symbols may have to be discarded, which does not mean that decoding of the current picture/slice necessarily becomes impossible.

Run length encoded quantized and scanned transform coefficients offer by means of the EOB symbol – the code word with index 0 – a natural short-length marker which is suited to separate the coefficients of different subblocks. A slightly changed length symbol is not very visually striking, but if a run symbol is contaminated, there will be significant error propagation throughout the whole reverse transform. Here, it might not be efficient to use the brute-force attempt as for the header data, but rather to discard all symbols between two EOB symbols. In very low-bit-rate code streams, the run length symbol sequences will not be long, and the loss in PSNR by discarding a transform is not very remarkable.

Motion vector data is probably most sensitive to errors since it is DPCM-encoded. Errors in one coefficient will thus propagate throughout the bit stream to the partition end. It is proposed to utilize the error locating capabilities of VLCD and make use of reversible DPCM coding. Reversible DPCM coding appends a single additional motion vector at the end of the existing MVD and is therefore only slightly less efficient than ordinary DPCM.

The concept is as follows: Consider the index sequence (9, 2, 5, 0, 1, 3) for encoding. The sequence is filtered according to the system function $H_{\text{enc}} = 1 + z^{-1}$, i.e. delayed one symbol and added to itself: $X_1 = (9, 2, 5, 0, 1, 3)$, $X_2 = (0, 9, 2, 5, 0, 1, 3)$, $X_3 = X_1 + X_2 = (9, 11, 7, 5, 1, 4, 3)$ which can be VLCD-encoded. Forward de-DPCM after an error-free transmission is accomplished by inverse filtering – the system function being here $H_{\text{dec}} = (1 + z^{-1})^{-1} = z/(z + 1)$ – or the following procedure: Having received $Y = (y_1, y_2, \dots, y_7)$, the original source symbol sequence $W = (w_1, \dots, w_6)$ is computed in a forward manner by means of $w_1 = y_1$, $w_2 = y_2 - w_1$, and so on. The last two symbols w_6 and y_7 have to be equal in the error-free case; they can be compared to check additionally for error occurrence. Backward de-DPCM works accordingly: $w_6 = y_7$, $w_5 = y_6 - w_6$, and so on. Here, y_0 and w_0 should be equal. If the decoder now points to an error exactly in the symbol 7, two symbols can be recovered in forward, and four in backward direction. For this example, it is assumed that the error could be localized with one-symbol accuracy.

Summary and outlook

This document showed some advantages when using the reversible code VLCD instead of the self-synchronizing UVLC. The position of bit errors can be determined approximately by means of the number of decoded source symbols, and, with increased computational effort, with a small margin, as far as single bit errors are concerned. The results lead to the proposal of replacing UVLC by VLCD. Concerning today's draft model of H.26L, header symbols could be saved, as well as blocks of run-length-encoded DFD coefficients, which are edge blocks of macroblocks. Only MVD coding has to be changed slightly, i.e. coded by the proposed reversible differential coding technique, for the cost of a somewhat reduced efficiency. Replacing UVLC by VLCD affects only the ordering of sync bits.

The source symbol number gradients are subject to source variations. Extensive testing involving VLCD decoding and video sources of various character has not been done so far due to long simulation times. So, this task remains.

During work on this contribution, there has been a thread on the email reflector that reported previous investigations in '*error propagation of the RVLC in H.263 Annex D with transmitted over BSC (so more than single error)*'. Future contributions should take these bursty conditions into account as well. Also, the VLCD decoder's decoding behavior and possibilities with regard to increased error resilience and concealment could be investigated for the case of several random bit errors. The author believes, however, that VLCD performs superior there as well.

Finally, when packet losses probably with retransmission requests are assumed, the error-resilient properties of UVLC or VLCD are superfluous. The respective code table should then be replaced by an appropriate code with better coding efficiency like e.g. a Huffman code.

Abbreviations

3GPP	Third-generation partnership project
BSC	Binary symmetric channel
CW	Code word
DPCM	Differential Pulse Code Modulation
EOB	End of block
IP	Internet protocol

IPR	Intellectual property right
ITU	International Telecommunications Union
JVT	Joint Video Team
MVD	Motion vector data
PSNR	Peak-signal-to-noise ratio
RVLC	Reversible variable-length coding
SS	Source symbol
Sync	Synchronization
TML	Test Model Long-Term
UDP	User Datagram Protocol
UVLC	Universal variable-length code
VCEG	Video Coding Experts Group
VLCD	Variable-length code from H.263, Annex D
VLC	Variable-length coding/code

References

[VCEG-L23]

Louis Kerofsky. Bit errors within the UVLC. Eibsee (Germany), Jan.2001

[VCEG-F11]

Gisle Bjøntegård. Response to Call for Proposals for H.26L. Oct.1998

[Max85]

James C. Maxted and John P. Robinson. Error Recovery for Variable Length Codes. IEEE Trans.Inf.Theory, volume IT-31, no. 6, pp. 794-801, Nov.1985

[VCEG-N81]

Gary Sullivan and Gisle Bjøntegård. Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material. Santa Barbara (CA, USA), Sep.2001

JVT Patent Disclosure Form

International Telecommunication Union
Telecommunication Standardization Sector



International Organization for Standardization



International Electrotechnical Commission



Joint Video Coding Experts Group - *Patent Disclosure Form*

(Typically one per contribution and one per Standard | Recommendation)

Please send to:

JVT Rapporteur Gary Sullivan, Microsoft Corp., One Microsoft Way, Bldg. 9, Redmond WA 98052-6399, USA
Email (preferred): Gary.Sullivan@itu.int Fax: +1 425 706 7329 (+1 425 70MSFAX)

This form provides the ITU-T | ISO/IEC Joint Video Coding Experts Group (JVT) with information about the patent status of techniques used in or proposed for incorporation in a Recommendation | Standard. JVT requires that all technical contributions be accompanied with this form. *Anyone* with knowledge of any patent affecting the use of JVT work, of their own or of any other entity (“third parties”), is strongly encouraged to submit this form as well.

This information will be maintained in a “living list” by JVT during the progress of their work, on a best effort basis. If a given technical proposal is not incorporated in a Recommendation | Standard, the relevant patent information will be removed from the “living list”. The intent is that the JVT experts should know in advance of any patent issues with particular proposals or techniques, so that these may be addressed well before final approval.

This is not a binding legal document; it is provided to JVT for information only, on a best effort, good faith basis. Please submit corrected or updated forms if your knowledge or situation changes.

This form is *not* a substitute for the *ITU ISO IEC Patent Statement and Licensing Declaration*, which should be submitted by Patent Holders to the ITU TSB Director and ISO Secretary General before final approval.

Submitting Organization or Person:

Organization name _____ Identical to the document’s author and his originating institution.

Mailing address _____

Country _____

Contact person _____

Telephone _____

Fax _____

Email _____

Place and date of submission _____

Relevant Recommendation | Standard and, if applicable, Contribution:

Name (ex: “JVT”) _____

Title _____

Contribution number _____

(Form continues on next page)

Disclosure information – Submitting Organization/Person (choose one box)

2.0 The submitter is not aware of having any granted, pending, or planned patents associated with the technical content of the Recommendation | Standard or Contribution.

or,

The submitter (Patent Holder) has granted, pending, or planned patents associated with the technical content of the Recommendation | Standard or Contribution. In which case,

2.1 The Patent Holder is prepared to grant – on the basis of reciprocity for the above Recommendation | Standard – a free license to an unrestricted number of applicants on a worldwide, non-discriminatory basis to manufacture, use and/or sell implementations of the above Recommendation | Standard.

2.2 The Patent Holder is prepared to grant – on the basis of reciprocity for the above Recommendation | Standard – a license to an unrestricted number of applicants on a worldwide, non-discriminatory basis and on reasonable terms and conditions to manufacture, use and/ or sell implementations of the above Recommendation | Standard.

Such negotiations are left to the parties concerned and are performed outside the ITU | ISO/IEC.

2.2.1 The same as box 2.2 above, but in addition the Patent Holder is prepared to grant a “royalty-free” license to anyone on condition that all other patent holders do the same.

2.3 The Patent Holder is unwilling to grant licenses according to the provisions of either 2.1, 2.2, or 2.2.1 above. In this case, the following information must be provided as part of this declaration:

- patent registration/application number;
- an indication of which portions of the Recommendation | Standard are affected.
- a description of the patent claims covering the Recommendation | Standard;

*In the case of any box **other than 2.0** above, please provide the following:*

Patent number(s)/status _____

Inventor(s)/Assignee(s) _____

Relevance to JVT _____

Any other remarks: _____

(please provide attachments if more space is needed)

(form continues on next page)

Third party patent information – fill in based on your best knowledge of relevant patents granted, pending, or planned by other people or by organizations other than your own.

Disclosure information – Third Party Patents (choose one box)

3.1 The submitter is not aware of any granted, pending, or planned patents *held by third parties* associated with the technical content of the Recommendation | Standard or Contribution.

3.2 The submitter believes third parties may have granted, pending, or planned patents associated with the technical content of the Recommendation | Standard or Contribution.

For box 3.2, please provide as much information as is known (provide attachments if more space needed) - JVT will attempt to contact third parties to obtain more information:

3rd party name(s) _____

Mailing address _____

Country _____

Contact person _____

Telephone _____

Fax _____

Email _____

Patent number/status _____

Inventor/Assignee _____

Relevance to JVT _____

Any other comments or remarks: