Low complexity H.264 list decoder for enhanced quality real-time video over IP

F. Golaghazadeh[1], S. Coulombe[1], F-X Coudoux[2], P. Corlay[2]

[1] École de technologie supérieure [2] Université de Valenciennes

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Université de Valenciennes et du Hainaut-Cambrésis

NSERC CRSNG

CCECE 2017

Outline

- Introduction
- Problem statement
- Literature review
- Proposed approach
- Experimental setup and results
- Conclusions

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Problem statement

Unreliable Channels ⇨ **Transmission Errors** ⇨ **Visual Quality Degradation**

- Corrupted Video Packets (bit errors)
  (e.g. attenuation, multi-path fading)
- Lost Video Packets
  (e.g. network congestion)

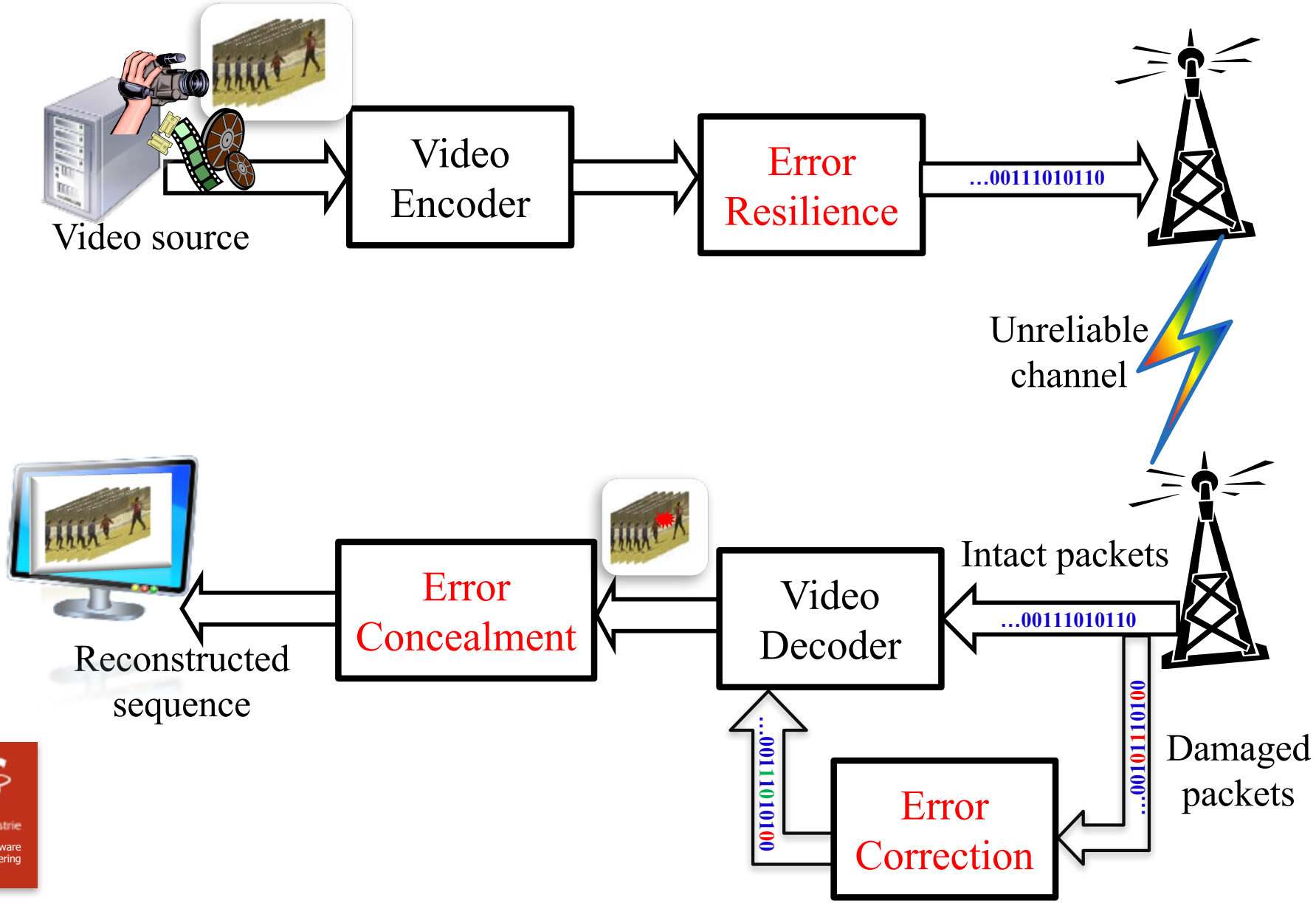
- **Compressed video streams are very vulnerable to errors**

Time

Motion-compensated prediction process ⇨ **Error propagation** in compressed video!

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Literature review

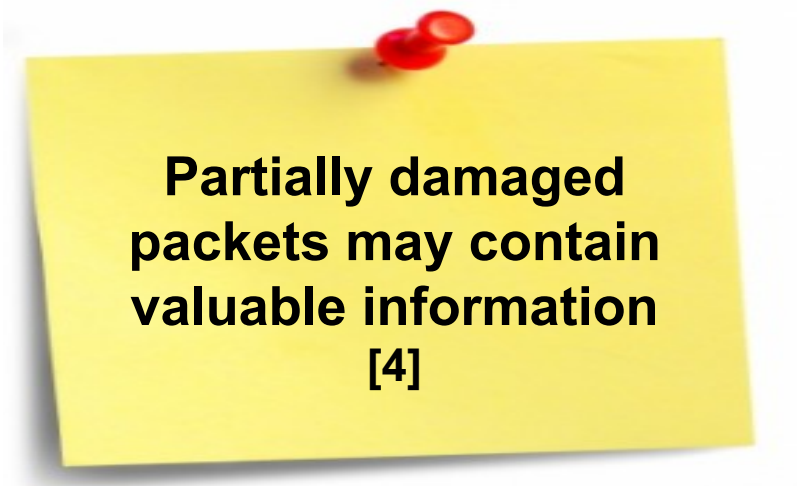


Literature review

❑ Error Concealment:

Estimates missing pixels by using

- **Spatial correlation [1]**
- **Temporal correlation [2]**
- **Both spatial and temporal correlation**
  - ➢ - spatio-temporal boundary matching and partial differential equation **(STBMA) [3]**

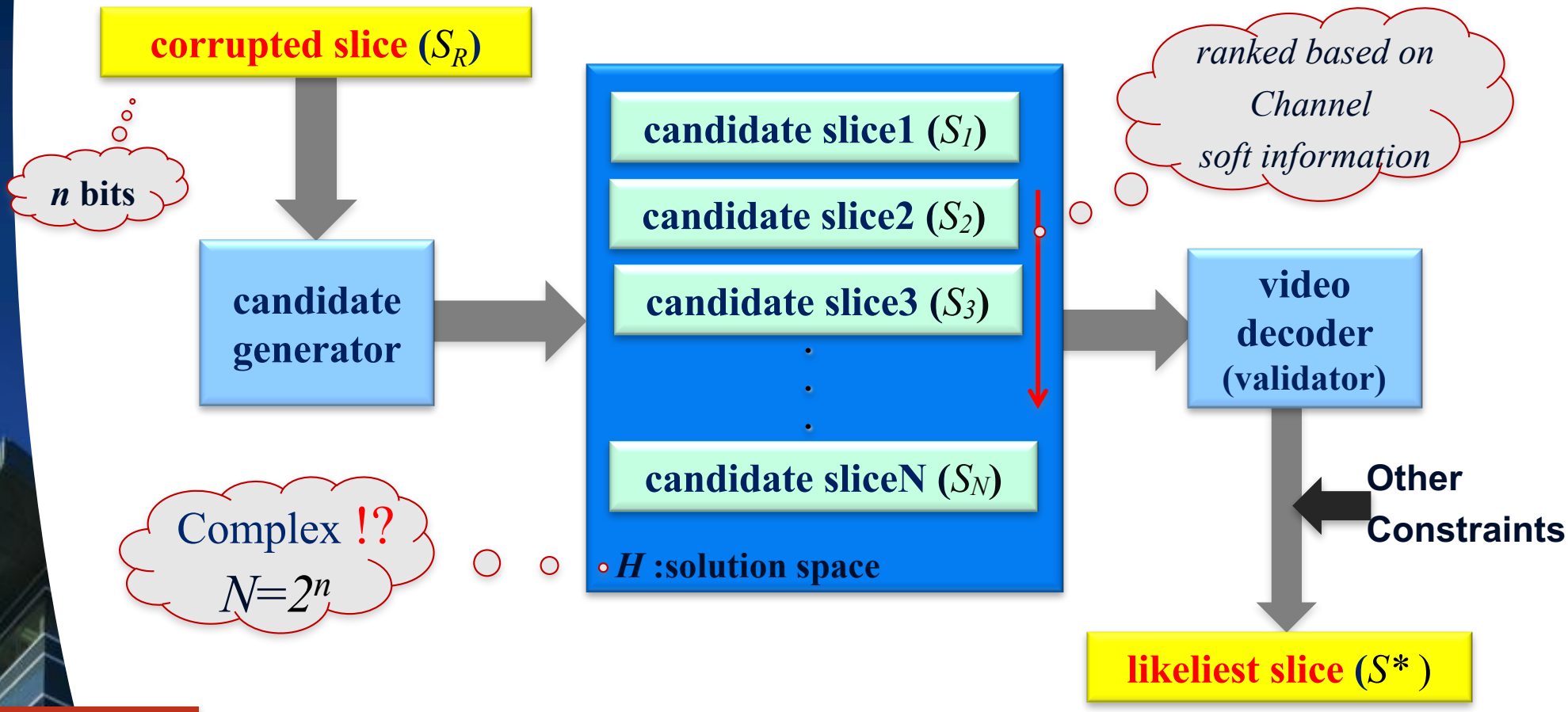


ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

[1] J. Liu, et al., "Spatial error concealment with an adaptive linear predictor", 2015.
[2] J. Zhou, et al., "Efficient motion vector interpolation for error concealment of H.264/AVC", 2011.
[3] Y. Chen, et al., "Video error concealment using spatio-temporal boundary matching and partial differential equation", 2008.
[4] L. Superiori, et al., "Performance of a H.264/AVC error detection algorithm based on syntax analysis", 2006.

Literature review

❑ Error Correction: List decoding



[5] D. Levine, et al., "Iterative joint source-channel Decoding of H.264 compressed video", 2007.

[6] N. Nguyen, et al., "Iterative joint source- channel decoding for H.264 video transmission using virtual checking method at source decoder", 2010.

[7] G. Sabeva, et al., "Robust decoding of H.264 encoded video transmitted over wireless channels", 2006.

[8] C. Weidmann, et al.," Combined sequential decoding and error concealment of H.264 Video", 2004.

[9] R. A. Farrugia , et al.," Robust decoder-based error control strategy for recovery of H.264/AVC video content", 2011.

Literature review

❑ Error Correction: Syntax Level

Soft/Hard Output Maximum Likelihood Decoding (SO/HO-MLD) [10]

- Generate only a set of valid candidates at the correction process of each syntax element
- Any mistake in the decoding of a syntax element will propagate and reduce the performance

Lack of access to the soft information in the application layer

(complex to implement)

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

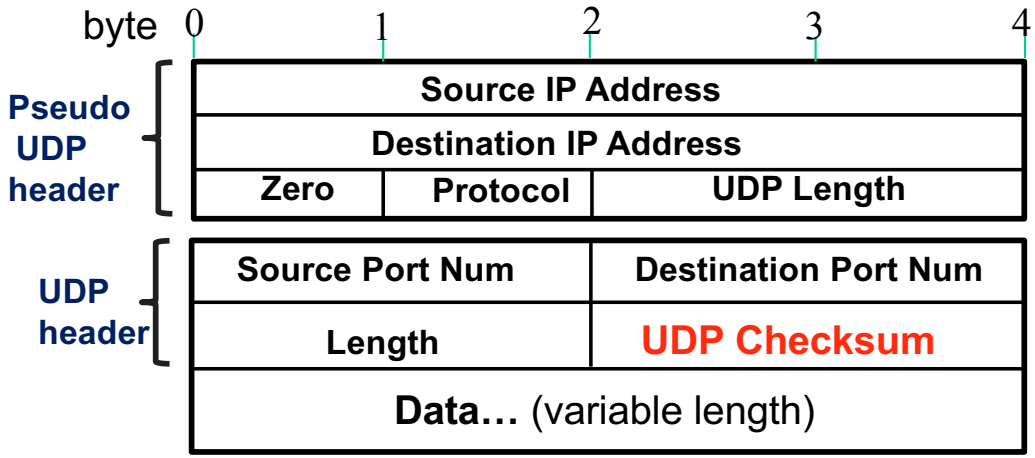
[10] F. Caron, et al., "Video error correction using soft-output and hard-output maximum likelihood decoding applied to an H.264 baseline profile", 2015.

Proposed approach

- Reduction candidate list by analyzing the calculated UDP checksum value at the receiver side.

Checksum: ***verify the integrity*** of the delivered message

UDP checksum: one's complement of the one's complement sum of a *pseudo UDP header*, *UDP header* and application *data message*.



Proposed approach

TRANSMISSION

Bit Position	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet 1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**

Bit Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Second 16-bit word	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	1
First 16-bit word	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0
Ones's compl. Sum	1	1	1	1	0	0	1	1	0	1	1	0	1	0	0	1
Checksum (C_T)	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**

$$C_T = \overline{\sum_{i=0}^{m-1} W_i}$$

16-bit word

UDP checksum calculation:

- Divide the packet into chunks of 16-bit words.
- Compute one's complement sum over all the words.
- Flip all the bits of the final sum (one's complement).

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Proposed approach

RECEPTION

Bit Position	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet 1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**

Bit Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Second 16-bit word	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	1
First 16-bit word	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0
Ones's compl. Sum	1	1	1	1	0	0	1	1	0	1	1	0	1	0	0	1
Checksum (C_T)	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Checksum(C_R)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The zero value of C_R validates the received packet.

If no error:

$$C_R = \overline{\sum_{i=0}^{m-1} \hat{W}_i + \hat{C}_T}$$

$$= \overline{\sum_{i=0}^{m-1} \hat{W}_i + \overline{\sum_{i=0}^{m-1} W_i}} = \overline{\sum_{i=0}^{m-1} (W_i + \overline{W}_i)}$$

$$= \overline{FFFF} = 0000$$

the received versions of W_i and C_T are denoted as $\hat{W}_i$ and $\hat{C}_T$

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Proposed approach

TRANSMISSION

Bit Position	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet 1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**

RECEPTION — **Corrupted**

Bit Position	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet 1	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**

Bit Position	15	14	13	12	11	**10**	9	8	7	6	5	4	3	2	1	0
Second 16-bit word	0	0	1	0	0	**0**	1	0	1	1	0	1	0	0	0	1
First 16-bit word	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0
Ones's compl. sum	1	1	1	1	0	**1**	1	1	0	1	1	0	1	0	0	1
C_T	0	0	0	0	1	1	0	0	1	0	0	1	0	1	1	0
	1	1	1	1	1	**0**	1	1	1	1	1	1	1	1	1	1
Checksum(C_R)	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Non-zero value of C_R : received packet is corrupted.

The checksum is indication the position of the error in modulo 16.

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Proposed approach

Error Type	C_R	C_R Patterns
$1_j \longrightarrow 0_j$	000 . . . 010 . . . 000 Col. 15 j 0	1000 0000 0000 0000 0100 0000 0000 0000 . . 0000 0000 0000 0001
$0_j \longrightarrow 1_j$	111 . . . 101 . . . 111	0111 1111 1111 1111 1011 1111 1111 1111 ⋮ 1111 1111 1111 1110

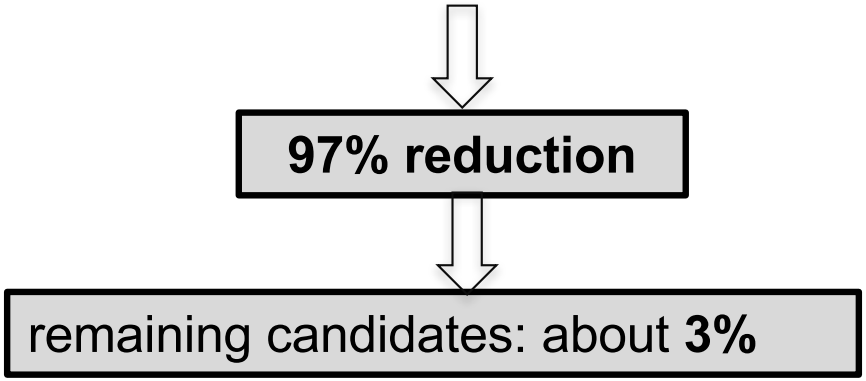
The C_R pattern :

- **potential error columns in the words**
- **type of the flipped bits**

Candidate Reduction

One bit in error in a packet of *N* bits:

- Traditional list decoding approaches: ***N*** candidates
  - proposed approaches: ***N/32*** candidates.



Packet length (bits)	Average number of candidates		Eliminated candidates
	List Decoding	Proposed	
272	272	9	97%
768	768	24	97%
1120	1120	35	97%
5280	5280	165	97%

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Simulation setup

❖ Coding Parameters:

- **Baseline** profile of H.264 (simulator: **JM** software v.18.5)
- First 60 Frames of sequence: *Foreman* (352x288), *Crew* (704x576), *Opening Ceremony, Walk* (720x576).
- Coded in **IPPP…** format, Intra refresh rate of **30** frames, quantization parameters (QPs): 22, 27, 32, and 37
- Each slice (= a single row of MBs), encapsulated into RTP packets.
- UDP checksum calculated on RTP packet

❖ **Corrupted frame: 31th-59th randomly**

❖ **Channel bit error rate (BER): [10^{-7}, 10^{-6}]: one bit error**

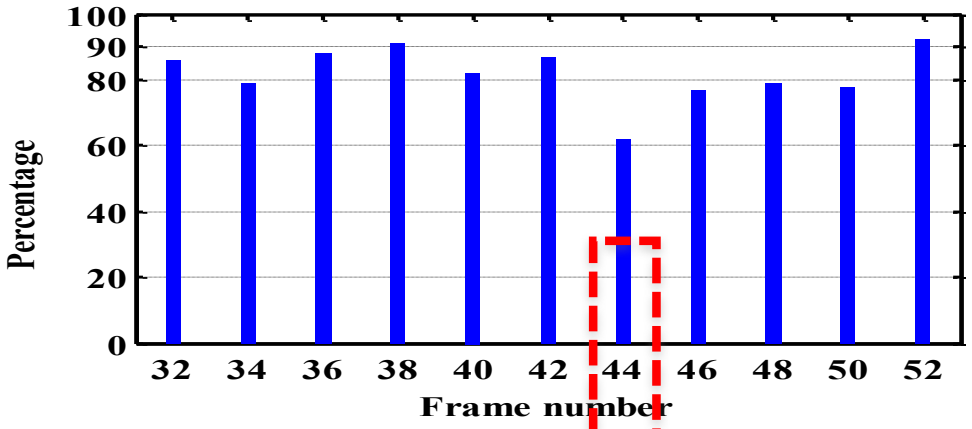
❖ Decoding Options :

Error Concealment: Frame Copy (**FC-JM**)
Spatio Temporal Boundary Matching (**STBMA**)

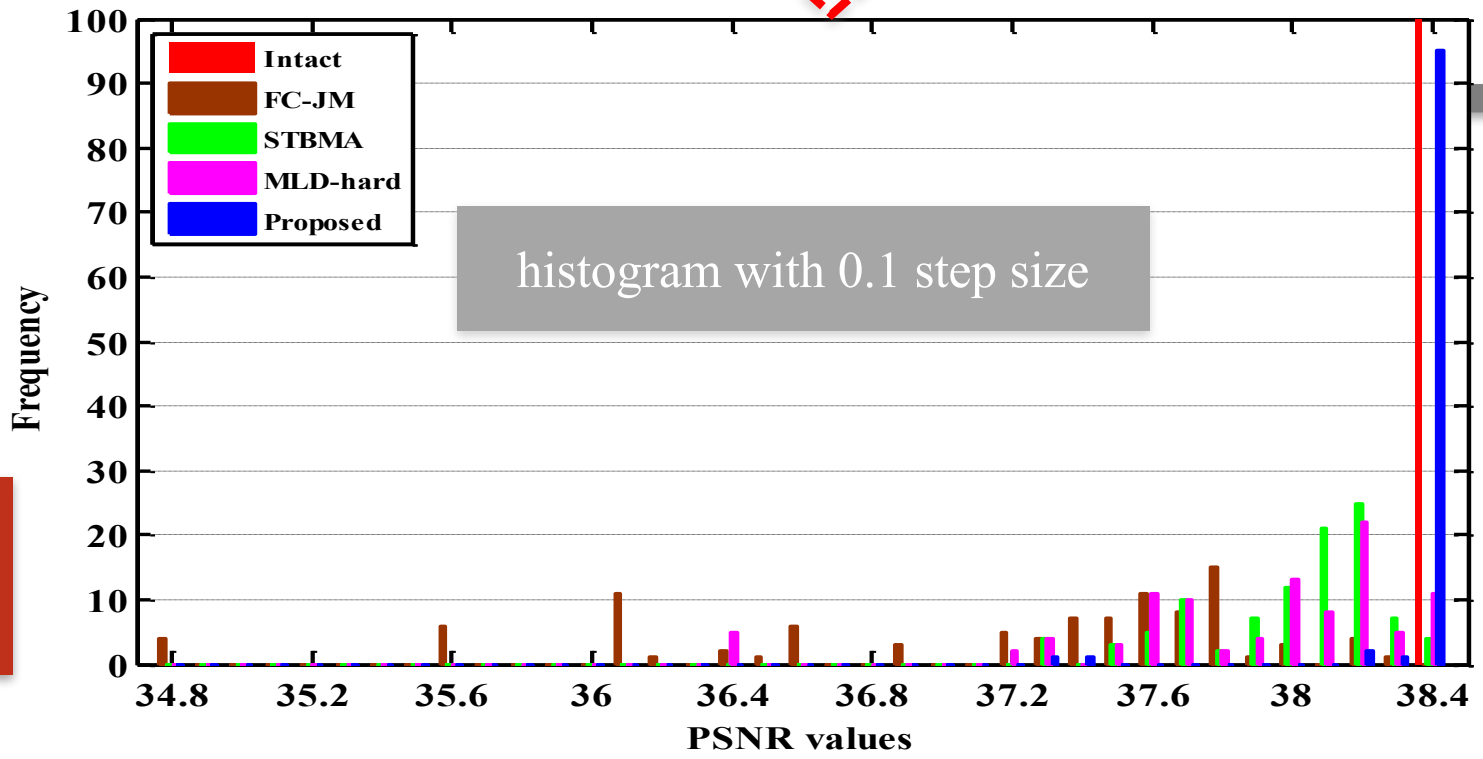
Error Correction: Maximum likelihood Decoding (**HO-MLD**)
Proposed List Decoding (**Proposed**)

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Seq: Crew, QP 27

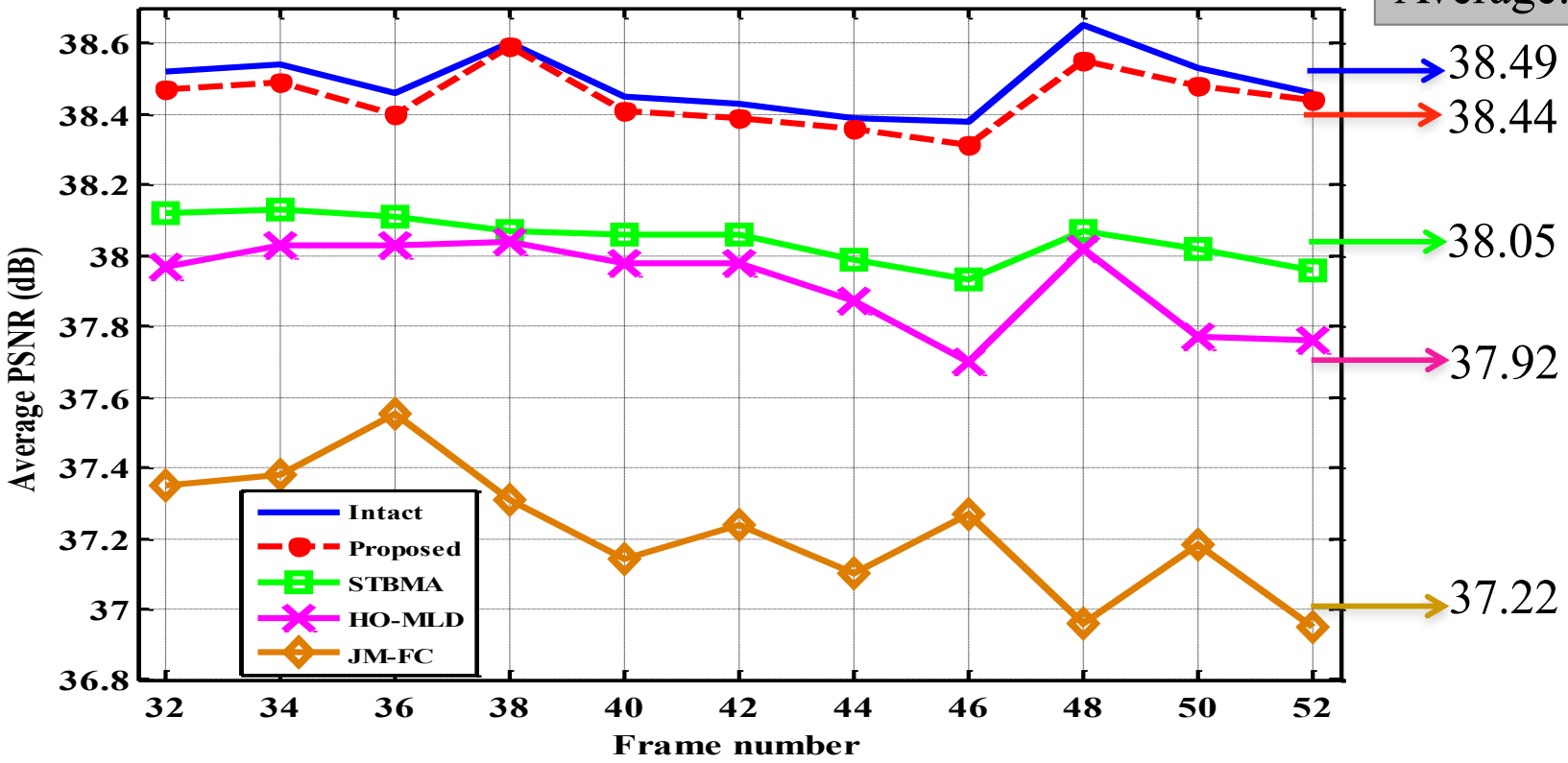


95% very close PSNR to the intact one.



histogram with 0.1 step size

Seq: Crew, QP 27



Average:

38.49

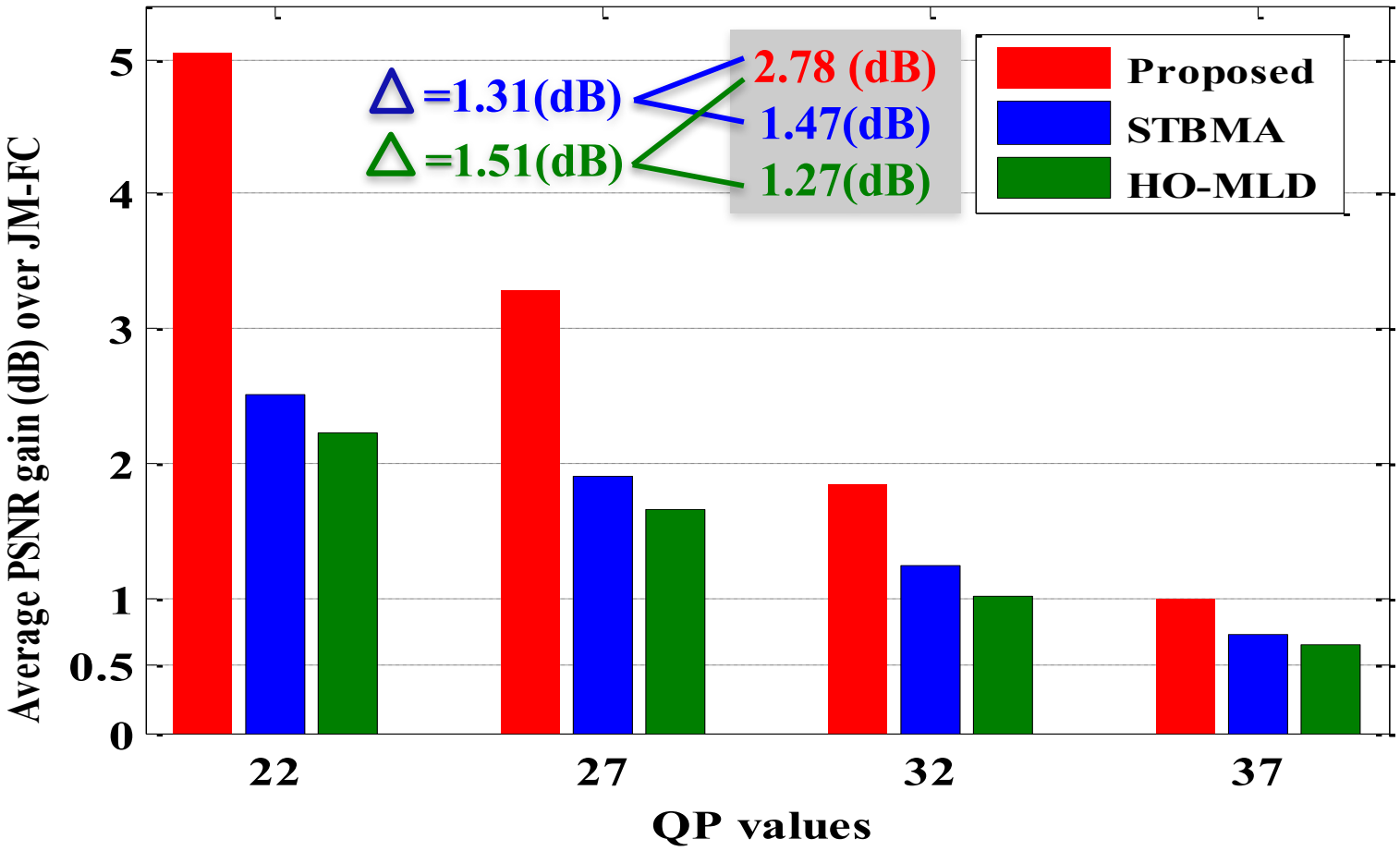
38.44

38.05

37.92

37.22

PSNR gain over JM-FC

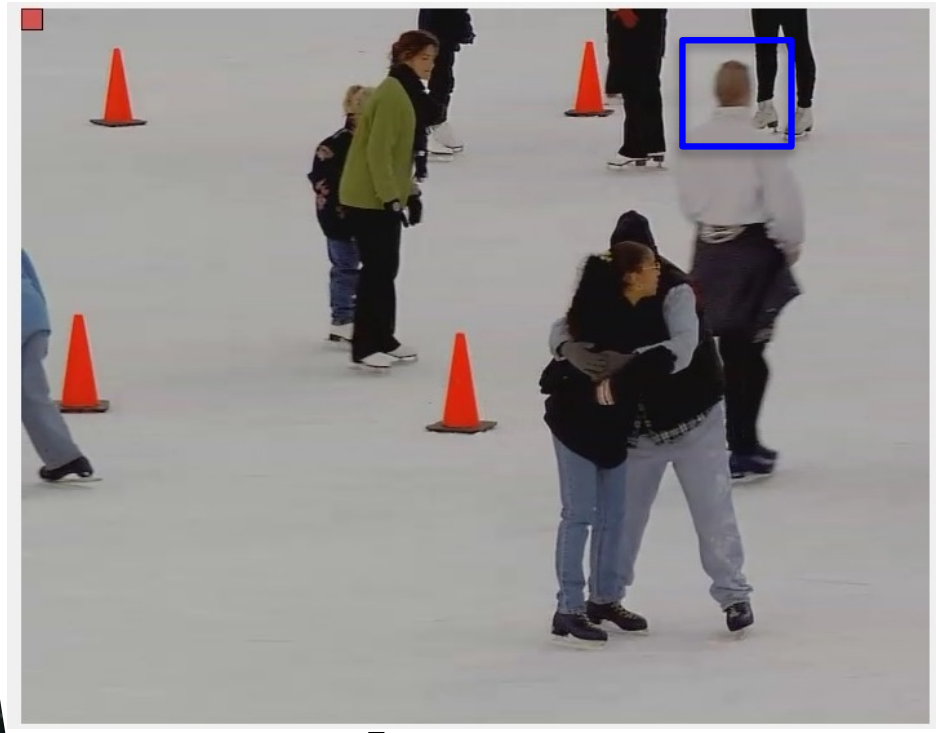


- Percentage of fully correcting bitstream:

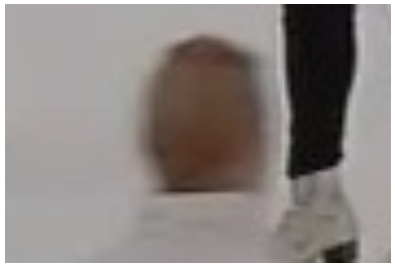
  Proposed approach: **80%**

  HO-MLD approach: **6%**

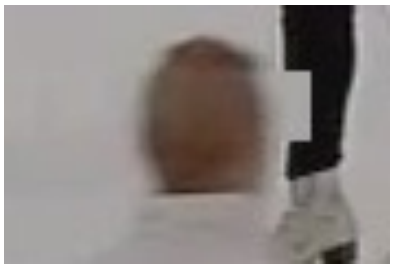
Visual quality comparison



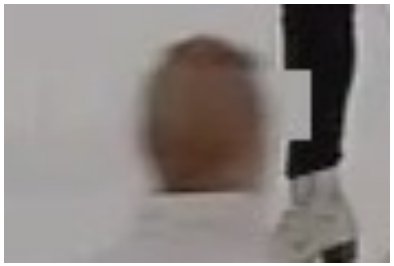
Intact



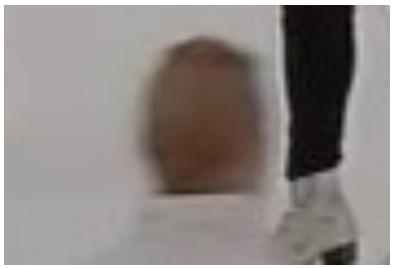
JM-FC



STBMA



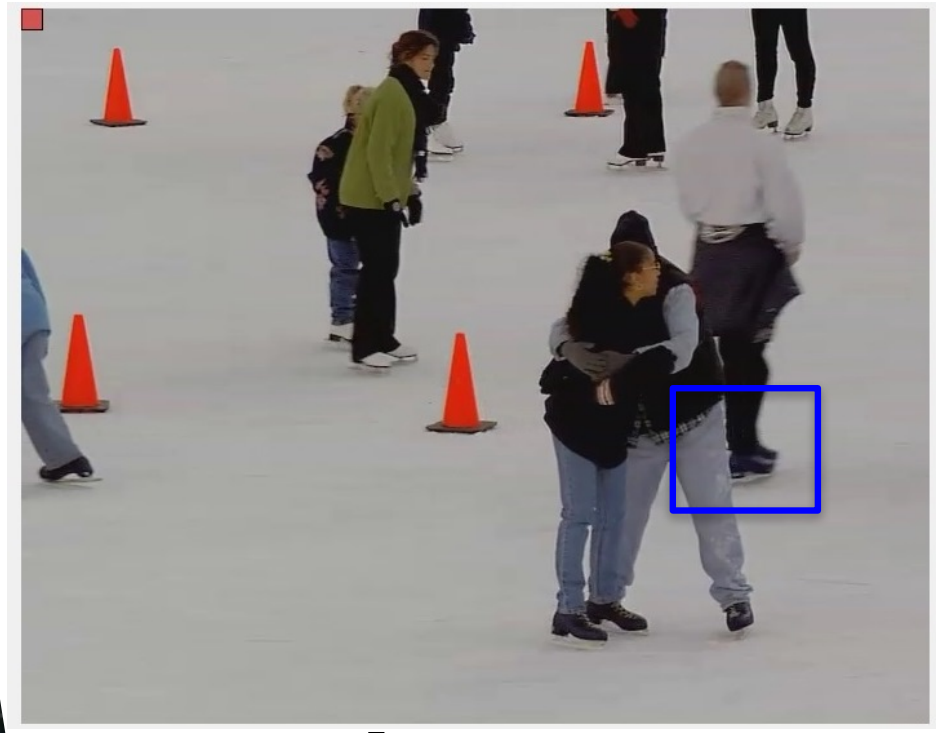
HO-MLD



Proposed

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Visual quality comparison



Intact

JM-FC

STBMA

HO-MLD

Proposed

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

Conclusions

One bit in error:

- Candidate reduction from *N* to *N/32*;
- Complexity reduction : **97%**

On average**:**

- ***2.78*** **dB** gains over **JM-FC**
- ***80%*** fully corrected packet

❑ Future step:

- The proposed approach can also be applied on HEVC coded sequences, and it can be extended to the case of more than one bit in error [1].

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering

[1] F. Golaghazadeh,, et al., "Checksum-filtered list decoding applied to H.264 and H.265 video error correction", 2017.

This research was funded by

NSERC
CRSNG

The Natural Science And Engineering Research Council of Canada

ÉTS
Le génie pour l'industrie
Department of Software and IT Engineering