H.265/HEVC bitstream transmission simulator
User manual
(Version 0.1, February 2021)

Matteo Naccari

(matteo.naccari@gmail.com)

Introduction

This brief manual describes the software tool "`transmitter-simulator-hevc`" which simulates the transmission of an H.265/HEVC bitstream through an error prone channel. The software is written in C++ 11 and developed using MS Visual Studio 2019 (v142 toolset). It is believed that the source code easily be built using different compilers and platforms. When you clone the git repository associated with the software, the following relevant files will be present:

- *error_plr_x (x in {3,5,10,20})*: four examples of error pattern files corresponding to Packet Loss Rates (PLRs) of 3, 5, 10 and 20%. For further details over the generation of the aforementioned files the interested reader is referred to: "**S. Wenger, "Error patterns for Internet experiments", JVT- Q15-I-16r1, October 1999**".
- *config_file.txt:* an example of configuration file containing the parameters of the "`transmitter-simulator-hevc`" software tool.
- *transmitter_simulator_user_manual.pdf:* the pdf of this manual.
- The source code files of the "`transmitter-simulator-hevc`" software tool. These are:
    - packet.h
    - parameters.h
    - simulator.h
    - reader.h
    - syntax.
    - packet.cpp
    - parameters.cpp
    - simulator.cpp
    - main.cpp

For any bug or information, you can contact me at the following email addresses:

- matteo.naccari@polimi.it
- matteo.naccari@gmail.com
- matteo.naccari@lx.it.pt

Features and functioning

The "`transmitter-simulator-hevc`" software simulates the transmission of bitstreams compliant with the H.265/HEVC standard over error prone channels. The two main features of the "`transmitter-simulator-hevc`" software are:

- Different bitstream corruption modalities such as: all packets, all packets but those associated with intra coded slices and all packets associated with intra slices only.
- Generation of different channel realisations from a unique error pattern file.

The "`transmitter-simulator-hevc`" software simulates the transmission of H.265/HEVC bitstreams by discarding the coded packets according to a given error pattern file. Such an error pattern file contains a sequence of byte characters '0' and '1' associated with no packet loss full packet loss, respectively. A snippet of a typical error pattern file is reported in Figure 1:

```
00000010000000011100100000000010100000101000000010000000000001100
```

Figure 1: An example of error pattern contained in the error pattern file.

In order to simulate noisy transmission, the "`transmitter-simulator-hevc`" software needs the parameters listed in Table 1:

Parameter name	Description
Input bitstream	The H.265/HEVC bitstream being corrupted
Output bitstream	The H.265/HEVC corrupted bitstream. This file represents the bitstream that would be transmitted to the receiver if the channel drops coded packet according to the error pattern file
Error pattern file	The file containing the channel errors pattern
Offset	This is an unsigned integer number which represents the starting point where to read the error pattern file. Using the offset parameter, different channel realisations can be simulated, starting by the same error pattern file
Modality of corruption	The modality to corrupt an H.265/HEVC bitstream specified by the "Input bitstream" parameter. Allowed modalities are: • 0: corrupts all the packets according to the error pattern file • 1: corrupts all the coded packets but the ones containing intra coded slices • 2: corrupts only packets containing intra coded slices

Table 1: Description "`transmitter-simulator-hevc`" input parameters.

The bitstream transmission simulation is performed according to the following pseudo-code:

```
1.  Open the input bitstream
2.  Open the output bitstream
3.  Open the error pattern file and load it into array A
4.  set j to offset
5.  set mode to modality
6.  while there are packets to read from the input bitstream do
7.          read the next packet from the input bitstream and put into P
8.          if P contains an intra coded slice AND mode == 1 then
9.                  set writeable to 1
10.         end if
11.         if P does not contain an intra coded slice AND mode == 2 then
```

```
12.             set writeable to 1
13.         end if
14.         if the packet is not a video coding layer one then
15.             write P to the output bitstream
16.         else if A[j] == '0' then
17.             write P to the output bitstream
18.             j++
19.         else if A[j] == '1 then
20.             if writeable == 1 then
21.                 write P to the output bitstream
22.             else
23.                 j++
24.             end if
25.         end if
26.         if j >= length(A) then
27.             j = 0
28.         end if
29. end do
```

In the pseudo code above, a video coding layer packet refers to a packet containing data associated with the pixels of the compressed video. Examples of non video coding packets are the ones containing purely metadata information such as VPS, SPS and PPS headers or an SEI message.

Installation

A MS Visual Studio 2019 solution ships with the Git repository of the software and can be used to build the code. A CMakeList file is also provided to build the software with a different toolchain (e.g. g++ under Linux). The CMakeLists file has been automatically generated from the MS Visual Studio 2019 solution file and tested with g++ 7.5 under the `wsl` environment.

Usage

The "`transmitter-simulator-hevc`" software allows two types of usages:

- Input parameters passed through configuration file. config_file.txt contains an example of how a configuration file should be formatted. It is important to notice that the software expects the parameters in the file to follow the specific order:
    - Original bitstream file name [**string**]
    - Transmitted (corrupted) bitstream file name [**string**]
    - Corruption patter file name [**string**]
    - Offset where to start reading the corruption patter file [**integer**]
    - Corruption modality [**integer**]

  If the parameters are defined in the configuration file in an order different from the above one, an unexpected behaviour of the software may appear.
- Input parameters are passed through command line. Also, for this use the order of the parameters is fixed.

By typing `transmitter-simulator-hevc.exe` on a command prompt window the software will show a little online help about its usage (see Figure 2).

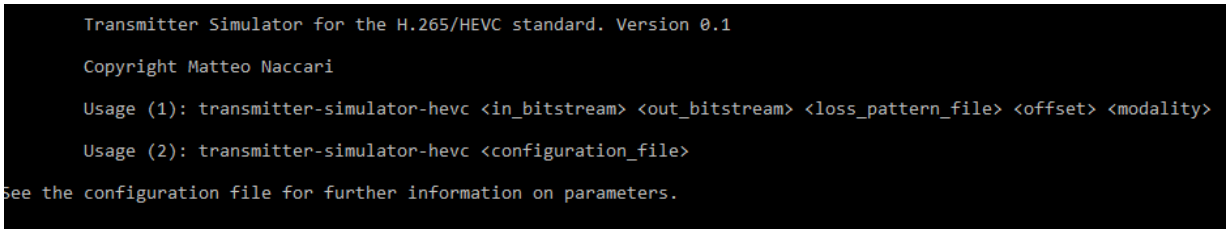

Figure 2: Online help provided by the "`transmitter-simulator-hevc`" software.

Potential issues and limitations

The software has been tested on a limited set of use cases. Accordingly, its correct functioning is not guaranteed. Some potential issues and limitations can be listed as follows:

- Error pattern files with a format different from the simple string of zero and ones depicted in Figure 1 will lead to an unpredictable behaviour of the "`transmitter-simulator-hevc`" software.
- In order to let the "`transmitter-simulator-hevc`" software to function properly, the order of the input parameters must be as the one specified in Table 1.
- The software has been used with bitstreams compliant with the first and second version of the H.265/HEVC standard (collectively known as V1 and Range Extensions, REext, respectively). It is believed that the software should be able to handle bitstreams compliant with the third version of the standard (collectively known as screen content coding extension).
- The software hasn't been tested using bitstreams associated with lengthy clips (e.g. a movie). Accordingly, its correct functioning for these cases cannot be guaranteed.
- The software hasn't been tested with bitstreams using the dependent slice feature. It is believed that its functioning should be correct anyway.