|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO CODING TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 159** |
| **Mainz – 20–28 October 2022** |
| |  |  | | --- | --- | | **Title:** | **Working draft 3 of ISO/IEC TR 23002-9 Film grain synthesis technology for video applications** | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Type:** | **Project** | | **Subtype:** | **Draft** | | **Status:** | **Approved** | | **Date:** | **2022-12-29** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **No. of pages** | **48** (without this cover page) | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  28th Meeting, Mainz, DE, 20–28 October 2022 | Document: JVET- AB2020-v1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Film grain synthesis technology for video applications (Draft 3)** | | |
| *Status:* | Output contribution | | |
| *Purpose:* | Draft text | | |
| *Author(s) or Contact(s):* | D. Grois  Y. He  W. Husak  P. de Lagrange  A. Norkin  M. Radosavljević  A. Tourapis  W. Wan | Email: | [Dan\_Grois@comcast.com](mailto:Dan_Grois@comcast.com)  [yonghe@qti.qualcomm.com](mailto:yonghe@qti.qualcomm.com)  [wjh@dolby.com](mailto:wjh@dolby.com)  [philippe.delagrange@interdigital.com](mailto:philippe.delagrange@interdigital.com)  anorkin@netflix.com  [milosr@xiaomi.com](mailto:milosr@xiaomi.com)  [atourapis@apple.com](mailto:atourapis@apple.com)  [wade.wan@broadcom.com](mailto:wade.wan@broadcom.com) | |
| *Source:* | Editors | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **Summary**  This draft technical report provides guidance on the use of film grain synthesis technology for video applications. Such technology may be used in conjunction with metadata signalling mechanisms, such as the supplemental enhancement information messages available in several video coding standards. The purpose of this document is to provide a publicly referenceable overview of the end-to-end processing steps for film grain synthesis, which may include content analysis, noise/film grain removal and film grain model parameter estimation, parameter encoding, encapsulation, and decoding, and film grain synthesis and blending for consumer distribution applications.  This version reorganized the technical report in the following order: overview, synthesis, analysis, parameter descriptions and examples while also adding text in areas that were thinly populated. Several figures and pictures were added to provide visual examples to better instruct the reader. Examples have been moved to an Annex. |

|  |
| --- |
| **Keywords**  Film Grain, Synthesis, Analysis, Modelling, Metadata. |

**Table of Contents**

[1 Scope 4](#_Toc122099190)

[2 References 5](#_Toc122099191)

[3 Definitions 6](#_Toc122099192)

[4 Abbreviations and acronyms 6](#_Toc122099193)

[5 Conventions 6](#_Toc122099194)

[5.1 General 6](#_Toc122099195)

[5.2 Arithmetic operators 7](#_Toc122099196)

[5.3 Bit-wise operators 7](#_Toc122099197)

[5.4 Assignment operators 8](#_Toc122099198)

[5.5 Relational, logical and other operators 8](#_Toc122099199)

[5.6 Range notation 8](#_Toc122099200)

[5.7 Mathematical functions 8](#_Toc122099201)

[5.8 Order of operations 9](#_Toc122099202)

[6 Overview of film grain technologies 9](#_Toc122099203)

[6.1 General 9](#_Toc122099204)

[6.2 Film grain technical characteristics 10](#_Toc122099205)

[6.3 Use cases and applications 12](#_Toc122099206)

[6.4 Film grain modelling 12](#_Toc122099207)

[7 Film grain synthesis 14](#_Toc122099209)

[7.1 General 14](#_Toc122099210)

[7.2 General description of film grain synthesis 15](#_Toc122099211)

[7.2.1 General 15](#_Toc122099212)

[7.2.2 Grain template generation 15](#_Toc122099213)

[7.2.2.1 General 15](#_Toc122099214)

[7.2.2.2 Frequency filtering model 15](#_Toc122099215)

[7.2.2.3 Autoregressive model 16](#_Toc122099216)

[7.2.3 Randomization 16](#_Toc122099217)

[7.2.3.1 General 16](#_Toc122099218)

[7.2.3.2 Choice of initialization parameters for the pseudo-random generator. 17](#_Toc122099219)

[7.2.3.3 Block size 17](#_Toc122099220)

[7.2.3.4 Offset randomness 18](#_Toc122099221)

[7.2.4 Local adaptation 19](#_Toc122099222)

[7.2.4.1 General 19](#_Toc122099223)

[7.2.4.2 Adaptation of grain shape 19](#_Toc122099224)

[7.2.4.3 Adaptation of grain amplitude 20](#_Toc122099225)

[7.2.5 Block boundary treatment 20](#_Toc122099226)

[7.2.6 Blending 20](#_Toc122099227)

[7.3 Examples of film grain synthesis using the frequency filtering model 20](#_Toc122099228)

[7.3.1 SMPTE RDD-5 20](#_Toc122099229)

[7.3.1.1 General 20](#_Toc122099230)

[7.3.1.2 Grain template generation 21](#_Toc122099231)

[7.3.1.3 Randomization 21](#_Toc122099232)

[7.3.1.4 Local adaptation 22](#_Toc122099233)

[7.3.1.5 Deblocking 22](#_Toc122099234)

[7.3.2 Variants based on of RDD-5 22](#_Toc122099235)

[7.3.2.1 Implementation in VTM and HM 22](#_Toc122099236)

[7.3.2.2 Resolution-adaptive block size 22](#_Toc122099237)

[7.3.2.3 Single-line block 22](#_Toc122099238)

[7.4 Examples of film grain synthesis using the autoregressive models 22](#_Toc122099239)

[7.4.1 FGC SEI message based autoregressive model 22](#_Toc122099240)

[7.4.2 AFGS1 model 23](#_Toc122099241)

[7.4.2.1 General 23](#_Toc122099242)

[7.4.2.2 Grain template generation 23](#_Toc122099243)

[7.4.2.3 Randomization 24](#_Toc122099244)

[7.4.2.4 Local adaptation 24](#_Toc122099245)

[7.4.2.5 Block overlap 25](#_Toc122099246)

[7.4.2.6 Blending 25](#_Toc122099247)

[7.5 Example of film grain synthesis supporting both the frequency filtering and autoregressive models 26](#_Toc122099248)

[7.5.1 General 26](#_Toc122099249)

[7.5.2 Grain template generation 26](#_Toc122099250)

[7.5.3 Randomization 26](#_Toc122099251)

[7.5.4 Local adaptation 26](#_Toc122099252)

[7.5.5 Deblocking 26](#_Toc122099253)

[7.5.6 Blending 26](#_Toc122099254)

[8 Film grain analysis 27](#_Toc122099255)

[8.1 General 27](#_Toc122099256)

[8.2 Denoising and image analysis 28](#_Toc122099257)

[8.2.1 Denoising 28](#_Toc122099258)

[8.2.2 Edge and texture analysis 28](#_Toc122099259)

[8.3 Determination of grain scaling function 29](#_Toc122099260)

[8.3.1 General 29](#_Toc122099261)

[8.3.2 An example of FGC SEI message scaling factor estimation 30](#_Toc122099262)

[8.3.2.1 General 30](#_Toc122099263)

[8.3.2.2 Data points regularization and discarding potential outliers 30](#_Toc122099264)

[8.3.2.3 Curve fitting and curve quantization 30](#_Toc122099265)

[8.3.2.3.1 General 30](#_Toc122099266)

[8.3.2.3.2 Sub-range averaging 31](#_Toc122099267)

[8.3.2.3.3 Discarding potential outliers (1st pass) 31](#_Toc122099268)

[8.3.2.3.4 Extreme points extrapolation 31](#_Toc122099269)

[8.3.2.3.5 Discarding potential outliers (2nd pass) 31](#_Toc122099270)

[8.3.2.3.6 Curve fitting (1st pass) 31](#_Toc122099271)

[8.3.2.3.7 Discarding potential outliers (3rd pass) 32](#_Toc122099272)

[8.3.2.3.8 Curve fitting (2nd pass) 32](#_Toc122099273)

[8.3.2.4 Final scaling function approximation 32](#_Toc122099274)

[8.3.3 An example of AFGS1 scaling factor estimation 32](#_Toc122099275)

[8.4 Determination of cut-off frequencies for frequency filtering model 33](#_Toc122099276)

[8.4.1 General 33](#_Toc122099277)

[8.4.2 An example of FGC SEI message cut-off frequency estimation 34](#_Toc122099278)

[8.5 Determination of autoregressive model coefficients 34](#_Toc122099279)

[9 Film grain metadata 35](#_Toc122099280)

[9.1 General 35](#_Toc122099281)

[9.2 Film grain characteristics SEI message 35](#_Toc122099282)

[9.2.1 General 35](#_Toc122099283)

[9.2.2 Interpretation of FGC SEI message syntax 35](#_Toc122099284)

[9.3 AFGS1 metadata (SEI message) 37](#_Toc122099285)

[9.3.1 Interpretation of AFGS1 metadata syntax 37](#_Toc122099286)

[Annex A : Considerations on the derivation of x/y offsets from random generator 40](#_Toc122099287)

[A.1 Preserving uniform distribution when offset range is not a power of two 40](#_Toc122099288)

[A.2 Considerations on left of right-shifting LFSR 40](#_Toc122099289)

[A.3 Specific considerations when offset range is a power of two 41](#_Toc122099290)

[Annex B : Example implementations of film grain synthesis technologies 42](#_Toc122099291)

[B.1 FGC SEI message insertion and manipulation 42](#_Toc122099295)

[B.1.1 Explicit insertion of FGC SEI message with VTM and HM encoder 42](#_Toc122099296)

[B.1.2 Manipulation of FGC SEI message (bitstream post-processor) 43](#_Toc122099297)

[B.2 Film grain synthesis example implementations 44](#_Toc122099298)

[B.2.1 FGC SEI message implementation – frequency filtering model 44](#_Toc122099299)

[B.2.2 Versatile film grain synthesis software 44](#_Toc122099300)

[B.3 Film grain analysis example implementations 45](#_Toc122099301)

[B.3.1 Denoising 45](#_Toc122099302)

[B.3.2 FGC SEI message implementation – frequency filtering model 46](#_Toc122099303)

[B.3.3 AFGS1 parameters estimation 47](#_Toc122099304)

**Film Grain Synthesis Technology for Video Applications**

# Scope

This report provides guidance on the use of film grain synthesis technology in video applications, including for use in the ITU-T H.264/AVC (MPEG-4 Part 10), ITU-T H.265/HEVC (MPEG-H Part 2), and ITU-T H.266/VVC (MPEG-I Part 3) standards. Such technology can provide subjective quality benefits for certain video applications and thus can effectively achieve improved video compression. This can be achieved by commonly removing or reducing the amount of noise that may be present in a video or image signal prior to compression, and by resynthesizing and adding back an approximation of the removed noise, given a set of signalled parameters, during decoding. Such parameters may be signalled using appropriate mechanisms, such as the supplemental enhancement information messages that are supported by several video coding standards. The purpose of this document is to provide a publicly referenceable overview of the end-to-end processing steps for film grain and sensor noise removal, estimation, parameterization, synthesis, and blending for consumer distribution applications. This document includes examples of encoder-side processing steps and post-decoding processing steps for grain blending for some of the currently defined technologies.

# References

1. Rec. ITU-T H.266 | ISO/IEC 23090-3, *Versatile Video Coding*.
2. Rec. ITU-T H.274 | ISO/IEC 23002-7, *Versatile supplemental enhancement information messages for coded video bitstreams*
3. Rec. ITU-T H.265 | ISO/IEC 23008-2, *High efficiency video coding*.
4. Rec. ITU-T H.264 | ISO/IEC 14496-10, *Advanced video coding for generic audiovisual services*.
5. Allen, Elizabeth, and Sophie Triantaphillidou. *The manual of photography*. CRC Press, 2012.
6. SMPTE RDD 5-2006, *Film grain technology specifications for H.264 | MPEG-4 AVC bitstreams*
7. C. Gomila and A. Kobilansky, “SEI message for film grain encoding.” document JVT-H022, May 2003.
8. D. Grois, and A. Giladi, "Perceptual quantization matrices for high dynamic range H.265/MPEG-HEVC video coding ," *Proc. SPIE 11137, Applications of Digital Image Processing XLII*, 111370O (24 February 2020);<https://doi.org/10.1117/12.2525406>.
9. D. Grois, and A. Giladi, “HVS-Based Perceptual Quantization Matrices for HDR HEVC Video Coding for Mobile Devices”, *International Broadcasting Convention (IBC) 2020*, Online, 11-15 Sept. 2020.
10. T. Chen, "Elimination of Subband-Coding Artifacts Using the Dithering Technique", *Proceeding of International Conference on Image Processing (ICIP)*, pp. 874-877, 1994.
11. X. Jin, S. Goto and K. N. Ngan, "Composite Model-Based DC Dithering for Suppressing Contour Artifacts in Decompressed Video," in *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2110-2121, Aug. 2011, doi: 10.1109/TIP.2011.2114356.
12. J. M. Boyce and A. M. Tourapis, "Comfort noise for compressed video," *International Conference on Consumer Electronics (ICCE)*, 2005, pp. 323-324, doi: 10.1109/ICCE.2005.1429848.
13. Enhorn, J., Sjöberg, R., & Wennersten, P. (2020, October). A temporal pre-filter for video coding based on bilateral filtering. In 2020 IEEE International Conference on Image Processing (ICIP) (pp. 1161-1165). IEEE.
14. Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Transactions on image processing, 16(8), 2080-2095.
15. Ameur, Z., Hamidouche, W., François, E., Radosavljević, M., Menard, D. and Demarty, C.H., 2022. Deep-based Film Grain Removal and Synthesis. arXiv preprint arXiv:2206.07411.
16. Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., & Barron, J. T. (2019). Unprocessing images for learned raw denoising. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11036-11045).
17. Tian, C., Fei, L., Zheng, W., Xu, Y., Zuo, W., & Lin, C. W. (2020). Deep learning on image denoising: An overview. Neural Networks, 131, 251-275.
18. Cheong, H. Y., Tourapis, A. M., Llach, J., & Boyce, J. (2004, October). Adaptive spatio-temporal filtering for video denoising. In 2004 International Conference on Image Processing, 2004. ICIP'04. (Vol. 2, pp. 965-968). IEEE.
19. Kokaram, A., Kelly, D., Denman, H., & Crawford, A. (2012, September). Measuring noise correlation for improved video denoising. In 2012 19th IEEE International Conference on Image Processing (pp. 1201-1204). IEEE.
20. VVC reference software (VTM) GitLab, https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\_VTM
21. HEVC reference software (HM) GitLab, https://vcgit.hhi.fraunhofer.de/jvet/HM
22. FFmpeg GitLab, <https://github.com/FFmpeg/FFmpeg> (used version 4.2.4-1ubuntu0.1)
23. Marc Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method", Image Processing On Line, 2012, http://www.ipol.im/pub/art/2012/l-bm3d/article\_lr.pdf
24. Ymir Makinen, et al, "Image and video denoising by sparse 3D transform-domain collaborative filtering", https://webpages.tuni.fi/foi/GCF-BM3D/
25. AOMedia film grain synthesis specification 1 (AFGS1) draft, <https://aomediacodec.github.io/afgs1-spec/>
26. A. Norkin, N. Birkbeck, “Technical report on AOMedia film grain synthesis technology (draft)”, CWG-C050o\_v1, July 2020, <https://aomedia.org/docs/CWG-C051o_TR_AOMedia_film_grain_synthesis_technology_v2.pdf>
27. AV1 Bitstream & Decoding Process Specification, Version 1.0.0 with errata, <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>
28. A. Norkin and N. Birkbeck, Film Grain Synthesis for AV1 Video Codec, in Proc. IEEE Data Compression Conference (DCC), Snowbird, Utah, Mar. 2018
29. libaom – AV1 reference software, https://aomedia.googlesource.com/aom/

# Definitions

For the purpose of this document, the definitions given in [1] [2] [3] apply.

# Abbreviations and acronyms

For the purpose of this document, the abbreviations and acronyms given in [1] [2] [3] and the following apply.

FGS Film Grain Synthesis

FGC Film Grain Characteristics

SEI Supplemental Enhancement Information

AFGS1 AOMedia Film grain synthesis model

# Conventions

## General

The mathematical operators used in this document are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

## Arithmetic operators

The following arithmetic operators are defined as follows:

|  |  |
| --- | --- |
| + | Addition |
| − | Subtraction (as a two-argument operator) or negation (as a unary prefix operator) |
| \* | Multiplication, including matrix multiplication |
| xy | Exponentiation. Denotes x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation. |
| / | Integer division with truncation of the result towards zero. For example, 7 / 4 and (−7) / (−4) are truncated to 1 and (−7) / 4 and 7 / (−4) are truncated to −1. |
| ÷ | Used to denote division in mathematical formulae where no truncation or rounding is intended. |
|  | Used to denote division in mathematical formulae where no truncation or rounding is intended. |
|  | The summation of f( i ) with i taking all integer values from x up to and including y. |
| x % y | Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0. |

## Bit-wise operators

The following bit-wise operators are defined as follows:

|  |  |
| --- | --- |
| & | Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| | | Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| ^ | Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0. |
| x >> y | Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation. |
| x << y | Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0. |

## Assignment operators

The following assignment operators are defined as follows:

|  |  |
| --- | --- |
| = | Assignment operator |
| ++ | Increment, i.e., x+ + is equivalent to x = x + 1; when used in an array index, evaluates to the value of the variable prior to the increment operation. |
| − − | Decrement, i.e., x− − is equivalent to x = x − 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation. |
| += | Increment by amount given, i.e., x += 3 is equivalent to x = x + 3, and x += ( −3 ) is equivalent to x = x + ( −3 ). |
| −= | Decrement by amount given, i.e., x −= 3 is equivalent to x = x − 3, and x −= ( −3 ) is equivalent to x = x − ( −3 ). |

## Relational, logical and other operators

The following operators are defined as follows:

|  |  |
| --- | --- |
| == | Equality operator |
| != | Not equal to operator |
| !x | Logical negation "not" |
| > | Larger than operator |
| < | Smaller than operator |
| ≥ | Larger than or equal to operator |
| ≤ | Smaller than or equal to operator |
| && | Conditional/logical "and" operator. Performs a logical "and" of its Boolean operators, but only evaluates the second operand if necessary. |
| | | | Conditional/logical "or" operator. Performs a logical "or" of its Boolean operators, but only evaluates the second operand if necessary. |
| a ? b : c | Ternary conditional. If condition a is true, then the result is equal to b; otherwise the result is equal to c. |

## Range notation

|  |  |
| --- | --- |
| x = y..z | x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers and z being greater than or equal to y. |

## Mathematical functions

The following mathematical functions are defined as follows:

Abs( x )

Ceil( x ) the smallest integer greater than or equal to x.

Floor( x ) the smallest integer lower than or equal to x.

Clip3( x, y, z ) =

Min( x, y )

## Order of operations

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– Operations of a higher precedence are evaluated before any operation of a lower precedence.

– Operations of the same precedence are evaluated sequentially from left to right.

Table 1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this document is the same as that used in the C programming language.

Table 1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

|  |  |
| --- | --- |
| **operations (with operands x, y, and z)** | |
| "x++", "x−−" |  |
| "!x", "−x" (as a unary prefix operator) |  |
| xy |  |
| "x \* y", "x / y", "x  y", "", "x % y" |  |
| "x + y", "x − y" (as a two-argument operator), |  |
| "x << y", "x >> y" |  |
| "x < y", "x <= y", "x > y", "x >= y" |  |
| "x == y", "x != y" |  |
| "x & y" |  |
| "x | y" |  |
| "x && y" |  |
| "x || y" |  |
| "x ? y : z" |  |
| "x..y" |  |
| "x = y", "x += y", "x −= y" |  |

# Overview of film grain technologies

## General

This section provides an overview of film grain technologies in the context of video compression and video distribution. It includes historical information on the development of such technologies in clause 6.2, a high level description of film grain modelling in clause 6.3, and information on some of the use cases and applications clause 6.4. More details are provided in subsequent clauses as follows:

* Section 7 describes some of the already defined film grain synthesis technologies, and in particular the frequency filtering and autoregressive models.
* Section 8 provides examples of technologies that can be used for film grain analysis, including techniques for video denoising, edge and complex texture detection, film grain characteristic analysis, and model parameter estimation.
* Section 9 describes some of the film grain metadata that have already been defined in current video coding standards and specifications and how each metadata is interpreted, if appropriate, in the context of the frequency filtering and autoregressive models.
* Example implementations of such technologies are then described in Annex A.

## Film grain technical characteristics

The multimedia distribution industry began using celluloid film as the medium for capture, editing and distribution. Content distribution evolved to analogue technology and then digital technologies. Throughout the evolution, there has been an attraction to the “film look”.

Film grain is considered a primary contributor to the film look or also known as the cinematic look. Film grain is a product of the physical characteristics of analogue film. It referred to the spatial temporal variations in optical density of processed film that resulted from photographically developing the light-exposed silver-halide crystals dispersed in photographic emulsion [5]. . Images are thus formed by exposure and the development of these crystals. In colour images, where the silver is chemically removed after development, dye clouds (like soft tiny coloured grains) are formed on the sites where the silver crystals have been exposed. Grains are randomly distributed in the resulting image because of the random formation of silver crystals in the original emulsion. The naked eye cannot distinguish individual grains, which are about 0.002 mm down to about a tenth of that in size. Instead, the eye resolves groups of grains in an image, that an observer identifies as a grainy look that is commonly called just “film grain”. This is illustrated in Figure 1. Another example is shown in Figure 2, with a different colour image formation process called “autochrome”. This is one of the first colour image techniques invented by Lumiere brothers, where a classical black and white photo emulsion is exposed through a colour filter made of a fine dust of potato starch dyed with different colours.

The larger the image resolution, the higher the perception of the film grain. Film grain is clearly noticeable in cinema and HD images; it progressively loses importance in SDTV and becomes imperceptible in smaller formats [7].

A close up of a person's eye

Description automatically generated with medium confidence A picture containing blurry

Description automatically generated

Figure 1. Fuji S 400 film (left: 4000 dpi scan of 2x2mm area; right: raw negative 500x microscope view of 0.1x0.1mm area)

A group of people standing together

Description automatically generated with medium confidence 

Figure 2. 24x24mm and 2.2x2.2mm crops of a 1916 autochrome picture (13x18cm glass plate) from A. Samama, courtesy of Ministère de la Culture / RMN-GP (France)

Film grain appearance is therefore inevitable because of the physical nature of the process embedded in the film design itself. However, historically, it was considered as noise, and as such, technological advances have gone in the direction of its elimination.

The silver-halide crystals were engineered to be smaller and less visible, however due to the physical design and characteristics of an analogue film, it was not possible to complete eliminate grainy look. With an advancement of digital camera sensors, and its widespread utilization, grainy look was eliminated. Although digital sensors have brought many possibilities in terms of visual quality and visual processing, “analogue look” lived-on among professionals and film enthusiasts. Within the new era film grain turns into a visual tool and not just a by-product of chemical processing as in analogue film stock.

Note that the term film grain also includes synthetic film grain that is added in post-production to digitally captured high-value content for artistic effect or to mask imperfections (sometimes too sharp and clear) digital footage. The term film grain can also be used to refer to image sensor noise, particularly in low light and high-speed captures.

Therefore, perception of moderate grain texture is a desirable, often sought after, feature in motion pictures. Although the exact effect of the grain is not clear, it is considered a requirement in the motion picture industry to preserve the grainy appearance of images throughout the image processing and delivery chain. Intuitively, the presence of film grain helps to differentiate ‘real-world’ images from ‘computer-generated’ material, which are commonly created with no film grain. Furthermore, it is possible that film grain provides some visual cues that facilitate the correct perception of depth in two-dimensional pictures [7]. Even when movies are captured with digital cameras, artificial film grain is added at a post-processing stage to create a specific look, which artists qualify as “soft”, “organic”, or “living”.

This report focuses on film grain technology from the video compression and distribution point of view. It includes encoder-side and decoder-side aspects. On the encoder side, film grain technology provides means to denoise and/or analyse source video to improve compression and to determine statistical characteristics of the film grain to be synthesized at the decoder. At the decoder side, film grain technology provides the means to synthesize and blend film grain with the decoded video.

Film grain preservation during video distribution, and especially when targeting low bitrate applications can be a challenging for two reasons. First, compression gains related to temporal prediction cannot be fully exploited because of the random nature of the grain. Film grain noise is temporally independent, and as a result, motion compensation cannot be efficiently used for its prediction. Second, the grain commonly appears at high frequencies in the DCT domain, and it is typically filtered with other noise by in-loop filters, such as deblocking filters, or due to the quantization process [7]. This challenge is more severe with recent codecs, as bitrates gains have come along with noise elimination. In addition, introduction of pre-filtering in the video distribution chain can potentially remove film grain prior to compression. The use of quantization matrices ‎[8],‎[9] could potentially assist in the preservation of some of the film grain within the video content, however this also can have severe limitations, especially at lower bitrates, and for streaming applications.

## Use cases and applications

The use of film grain modelling technologies can be beneficial for image and video compression by providing improved subjective quality at a lower bitrate for certain types of video content. For example, these technologies can potentially provide benefits to video content that contains noise, such as film grain or image sensor noise.

The first use case of film grain synthesis is artistic intent: recreate the film grain at the decoder side, which was unavoidably lost by compression involved in content distribution at practical bitrates. In this case, the film grain is considered to be a significant aspect of the video, and the content provider wants it to be part of the user experience.

Preserving film grain through video compression would require high bit rates for several applications, such as adaptive streaming and broadcasting. On the other hand, removing film grain allows using the full potential of video compression technologies while requiring film grain synthesis after decoding.

The second use case, which can also complement the first one, is the masking of compression impairments, like blocking, banding, mosquito noise, etc. as well as impairments caused due to quantization. If there is no artistic intent, then the constraints on film grain model accuracy may be relaxed. For this use case, the encoder can adjust film grain parameters to fit coding parameters, so that defect masking is effective (e.g., adjust noise amplitude based on quantization step).

## Film grain modelling

Although some compression can be achieved by coding a video sequence directly, it was determined that filtering the content, compressing, and providing information that enables the regeneration of the film grain, even if that is just an approximation, can result in more efficient coding performance and a better visual outcome. This is called film grain modelling. Thus, film grain modelling technologies provide a means of optionally removing noise prior to or during the encoding process to improve compression efficiency and, subsequently, to reconstruct an approximation of the film grain during or after the decoding process. Film grain modelling technologies may also be used to add visual noise to decoded video to mask or attenuate the visibility of compression artefacts, such as blockiness or bit-depth-related artefacts, such as banding or contouring. Note here that visually pleasant noise can be added to the decoded video even if source video had no visible noise/film grain to fulfil the masking task mentioned above.

Use of light-dependent film grain model parameters may be useful, particularly for the simulation of photographic film grain, as photographic film grain is intrinsically intensity dependent. First, variation of film opacity is the result of a variation of grain density, as seen in Figure 1, which has an impact on perceived grain. Also, film is organized in several layers (usually 3) for each colour component, with various light sensitivities to reach to its full dynamic range. Light-sensitive crystals have a distribution of sizes. Larger crystals capture more photons and are more likely to be exposed than smaller crystals, particularly in darker regions. As a result, darker regions tend to have larger and more noticeable film grain. This results in a dependency of the noise characteristics on brightness. An example with both grain size and strength variation is shown in Figure 3.

A group of people running

Description automatically generated with low confidence A large group of people outside

Description automatically generated with medium confidence

Figure 3. Kodak Vision 250D film, 1500 dpi scan of 5.5x5.5mm areas of the same picture

Use of film grain modelling technologies to denoise a source video by using a pre-processor is illustrated in Figure 4. Source video is input to a denoising process that outputs a video sequence from which noise or film grain is attenuated or removed. A film grain parameterization process then compares the source and denoised videos to determine film grain model parameter values, which relate to the variance, spatial frequency characteristics, colour correlation, and other statistical characteristics of the film grain. The process of denoising followed by the film grain model parameter estimation is commonly referred to as the film grain analysis process. Such process will be further discussed in clause 7.5.4.

After these processes are performed, the denoised video is then encoded and the film grain model parameter values are either signalled in the coded bitstream or provided to the decoder by some external means.

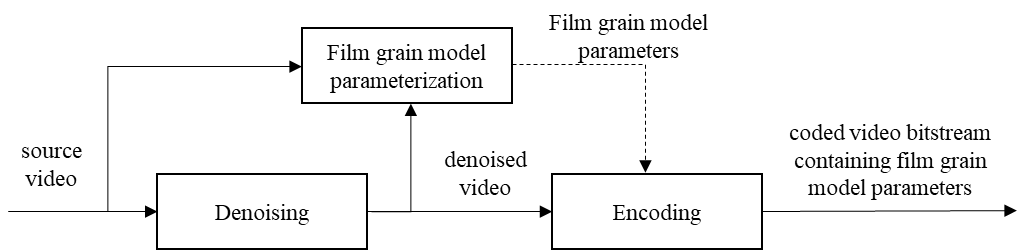


Figure 4. Use of film grain modelling technologies with a denoising pre-processing stage.

An alternative implementation of a film grain denoising and modelling system is also illustrated in Figure 5. In this case, the encoder itself acts as the denoising process.

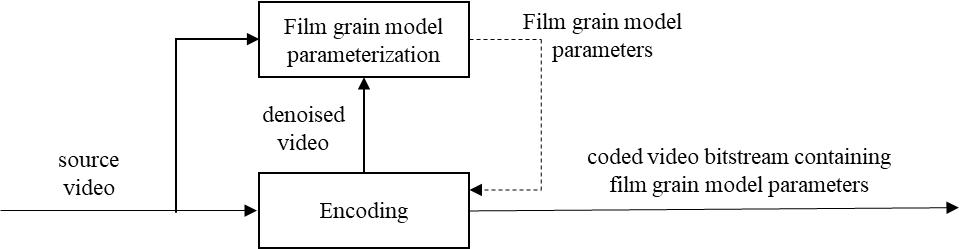


Figure 5. Use of film grain modelling technologies with an encoder to denoise source video.

The use of film grain modelling technologies with pre-determined film grain parameter values is illustrated in Figure 6. Note that film grain parameter values may, for example, have been pre-determined during post-production or when the statistical characteristics of the film grain are otherwise known *a priori*. The film grain parameters could be adjusted by the encoder depending on coding parameters or a default film grain configuration could be selected to mask coding artifacts. In such case, film grain can be added at the decoder side even if it was not present in the source video.

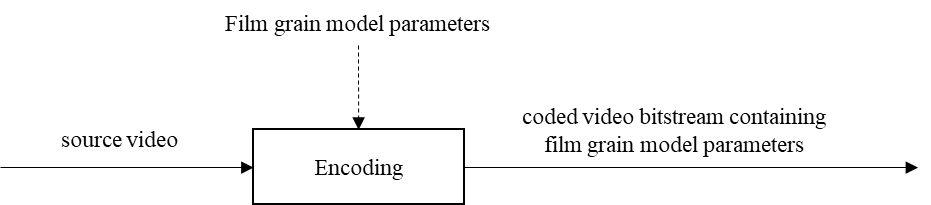


Figure 6. Use of film grain modelling technologies with pre-determined film grain model parameter values.

In any case, model parameters can be manually tuned/fine-tuned by a skilled person and provided to the encoder, for example, to be used as illustrated on Figure 6.

Figure 7 presents a decoding process, along with the film grain synthesis post-processing, for each of the examples provided in Figure 4 to Figure 6. At the decoder, the film grain model parameter values are parsed and input to a film grain synthesis process that generates simulated film grain and blends the grain with the decoded video to output decoded video with simulated film grain.

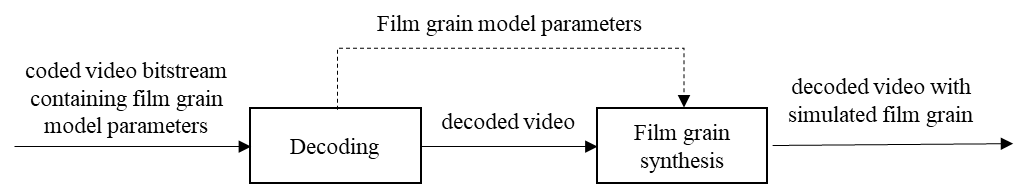


Figure 7. Decoding process along with the film grain synthesis post-processing.

# Film grain synthesis

## General

Film grain synthesis is commonly based on gaussian noise generation, with spatial correlation modelled either by frequency limits or autoregressive parameters, and local adaptation, that consists of adjusting grain amplitude and potentially also of adjusting spatial correlation, to target image intensity levels.

Gaussian noise can be generated with a random generator and a gaussian distribution table. The gaussian noise generator can be run for every pixel in the picture, along with spatial correlation methods, or can be run for a limited area (e.g., 64x64 pixels), further called “template”, which is then randomized to generate the full picture. Several such templates may need to be generated when spatial correlation varies across intensity intervals.

Template pattern randomization (i.e., extension to the full picture) can be performed by dividing the picture into blocks smaller than the template (e.g., 16x16 or 32x32 compared to 64x64 template), and by choosing a pseudo-random offset within the template space for each of those smaller blocks. A pseudo-random sign inversion can be added to the random offset to improve randomization. When such a process is performed, deblocking may be needed across randomization blocks, especially when spatial correlation within the grain pattern is significant (in other words: when the grain is large).

Working with templates avoids running the gaussian generator and correlation process for the full picture, which can be costly, not only because of more computations, but also because storing neighbouring lines (for spatial correlation) can be problematic in hardware. In contrast, using random offsets does not involve line storage, but just reading pre-computed templates at specific locations.

Local adaptation may be based on pixel intensity or a local average: a scaling factor and optionally a specific template is selected depending on underlying image intensity.

## General description of film grain synthesis

### General

This section describes the general process of template-based film grain synthesis, including grain template generation (see 7.2.2), block-based randomization (see 7.2.3), and local adaptation (see 7.2.4), as illustrated in Figure 8.

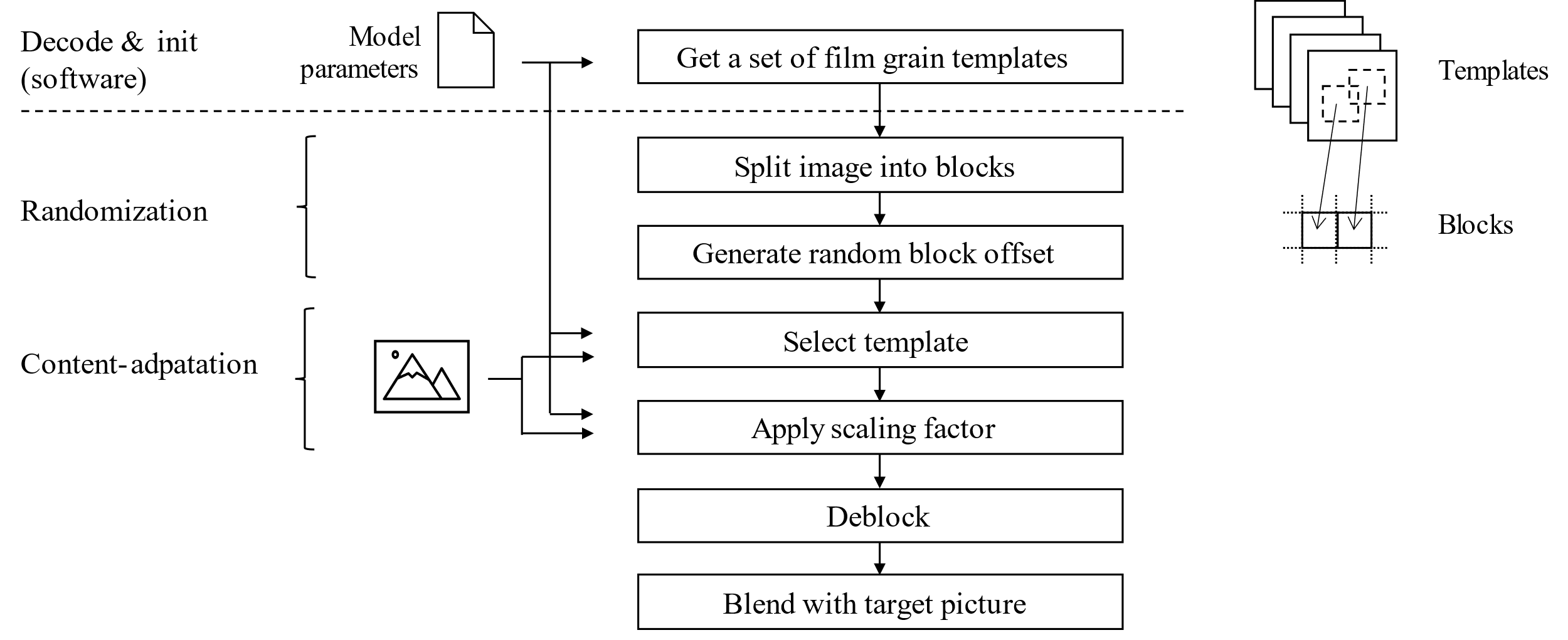


Figure 8. Template-based film grain synthesis workflow

### Grain template generation

#### General

The first step in the synthesis process is to generate film grain pattern templates (small patches, typically 64x64 pixels) according to the model parameters that are received by the decoder. When the model parameters specify different grain characteristics for different pixel intensities, then as many templates should be generated. Some implementations may support a limited number of templates, by potentially merging the characteristics for different intensity intervals. Depending on the implementation, templates can be precomputed and stored for further use, e.g., in the initialization process, or they can be created on-fly. The key model parameters are the variance and spatial correlation of the film grain as a function of the intensity (e.g., luma value) of the source video. Other model parameters relate to the bit depth and colour characteristics of the film grain compared to the source video, the way in which film grain is blended with the source video (additive or multiplicative), the persistence of model parameters from frame to frame, and the type of spatial correlation model (frequency-filtering or auto-regressive, as explained below).

#### Frequency filtering model

In a frequency filtering model, the film grain characteristics are specified by horizontal and vertical spatial cut-off frequencies. The grain template for a given set of cut-off frequencies can be generated as follows:

1. Generate a 2-dimensional array of random-value elements having a normalized Gaussian distribution. The 2-dimensional array represents discrete cosine transform (DCT2) coefficients. The column and row indices of the array represent horizontal and vertical frequencies, respectively. The array size may be implementation dependent.
2. Set the value of all elements of the random-value DCT2 array to 0 when the corresponding column and row indices are not within the corresponding high and low horizontal and vertical cut-off frequencies, also set the DC element to zero.
3. Calculate the inverse discrete cosine transform (IDCT2) of the array produced in step2.

#### Autoregressive model

The film grain characteristics could be specified by autoregressive filter coefficients instead of spatial cut-off frequencies. In this case, a grain template can be generated using an autoregressive filter as follows:

7‑1

Where *n* is a zero-mean Gaussian variable, *a*i,j are autoregressive coefficients, *b*k are colour correlation coefficients, *L* is a lag parameter, *c* is the colour component index, and *G*(*x*,*y,c*) is the grain value for colour component *c*, at the sample location ( *x*, *y )*. An autoregressive filter with lag parameter value equal to 2 is illustrated in Figure 9.

Shape

Description automatically generated with medium confidence

L=2

Figure 9. Sample grid of autoregressive filter with lag equal to 2

This is a generic description of an autoregressive film grain model, and depending on the specific syntax in use to transmit this information, some parameters could be fixed or generated from others.

### Randomization

#### General

Randomization consists of extending a film grain template (or a set of them), generated according to the model parameters, to the full picture.

It essentially consists in dividing the picture into blocks smaller than the template, and for each block, selecting a random region of the template, as illustrated in Figure 10. Methods considered here can bring additional diversity by randomly inverting the sign of the template for each block, which does not change the statistical properties since the gaussian noise is centred around zero. Other known methods of randomization are horizontal and vertical flip, which could be used if the film grain model is symmetrical, and rotations if the model is isotropic, but are not further described here.

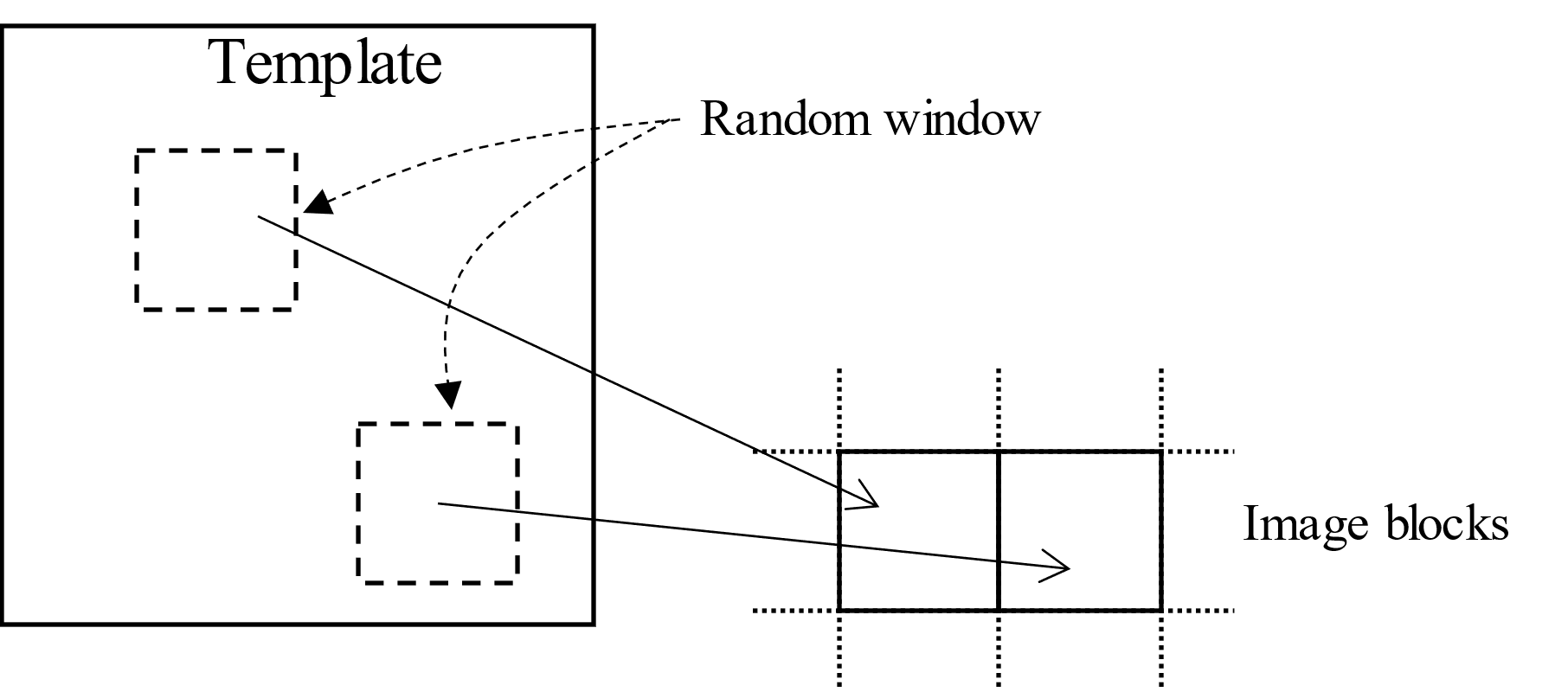


Figure 10. image blocks each taking a random window from template.

Selecting a random region (or window) within the template(s) for a given block can be performed by generating a (pseudo) random x/y offset with the correct range, considering the size of the template and the size of the window.

The challenge of randomization is to extend the template(s) to the full picture without the human eye noticing repeated elements. In fact, even though the template samples are pseudo-random, the resulting texture, or part of it, may be recognized by the human visual system as a pattern. If such a pattern is repeated in a predictable manner (see Figure 12), it may draw attention of the observer. Typically, such repetitions would be masked by the video/picture content and changes in the grain between the consecutive video frames. However, these patterns may become visible in smooth areas and in still pictures. The visibility of such patterns may also depend on the picture resolution. Several approaches described further can be used to reduce visible repetitions.

#### Choice of initialization parameters for the pseudo-random generator.

Certain pseudo-random gaussian sequences may form somewhat prominent visual features when arranged in a rectangular template. Since samples selection is done with a pseudo-random process, characteristics of the template and the prominence of the pattern depend on the initialization value of the pseudo-random generator used to draw samples from a gaussian distribution. Well-chosen initialization of the pseudo-random process helps avoiding the generation of grain templates with such prominent features.

One approach can be to transmit initialization values along with the model parameters, which has certain signalling cost but can guarantee that a desired pattern has been generated. Initialization can also be performed based on a set of known well-chosen initial values.

#### Block size

Another consideration to reduce visibility of repeated patterns is to choose a random window (matching block size) that is smaller than the template, The probability of template samples to be part of the random window for different sizes is shown in Figure 11. As an example, a window ¼ the size of the template in each dimension barely exhibits visible repetitions, while a window ½ the size of the template in each dimension may: the eye can recognize the centre of the pattern and identify displacements, as illustrated in Figure 12.

However, the window should be large enough compared to the size of the grains, so that the statistics within the window are still representative of the grain pattern, and ultimately, the model parameters. The compromise between window size and template size likely depends on implementation considerations.



Figure 11. Probability of (N x N) template samples to be part of a random (n x n) window,  
with n/N equal to 1/8 (left), 1/4 (centre), 1/2 (right)



Figure 12. Potential visual effect of randomization window size

#### Offset randomness

In addition to block size, randomization quality also depends on x/y offset randomness: adjacent blocks should avoid selecting similar offsets, which would mean a similar region of the template, thus visible repetitions. Similarly, repeating sequences of offsets would cause repetition of groups of blocks, which would be visible and should be avoided.

To reach these goals:

* The repetition period of the pseudo-random generator used to derive x/y offsets has a direct impact on the potential repetition of offsets. In the case of an LFSR (linear feedback shift register), the repetition period is related to the order of the polynomial used. The larger it is, the better the randomness. In addition to repetition period, a longer size allows extracting longer or more bit fields to generate different offsets and sign flips at the same time. Figure 13 illustrates an example of a 32-bit LFSR using a 31-order polynomial.
* Reducing the choice to a limited number of evenly spaced positions (e.g., one out of 4), as illustrated in Figure 14, both ensures a minimal spacing and better scrambling, because the (finite) pseudo-random possibilities are spread over a smaller number of larger displacements instead of many small (and potentially close) displacements. This has the additional benefit of enabling aligned reads of template memory in a practical implementation. On the other and, the number of positions should be kept large enough to allow sufficient randomization diversity.
* As the number of random values needed for the x and y offsets depends on the block size and offset alignment, this number may not be a power of two. In that case, care should be taken in the derivation of offsets from the random generator (or a bitfield extracted from it) so that the random distribution of offsets is uniform (as assumed in Figure 11). This is discussed in Annex A, including hardware cost considerations.
* The successive offsets should be as far as possible from each other. Depending on how offsets are derived from the pseudo-random generator, the method can differ (examples are given in A.2).

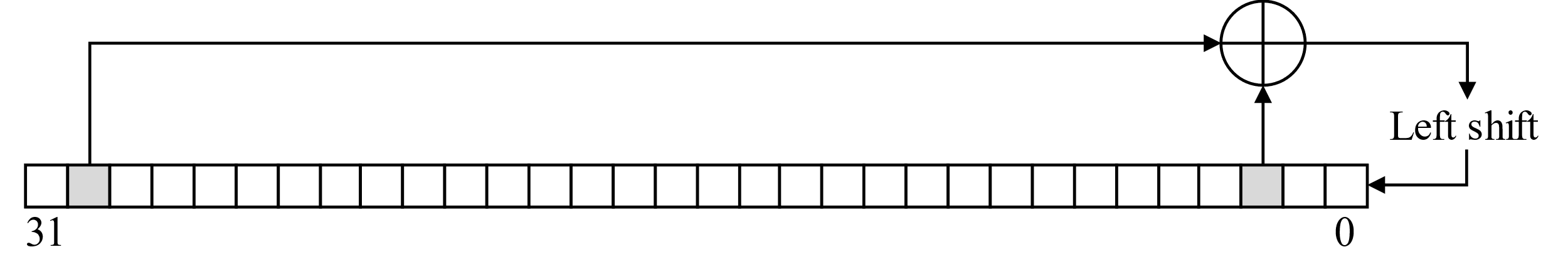


Figure 13 LFSR with 31st order polynomial (*x*31 + *x*3 + 1)

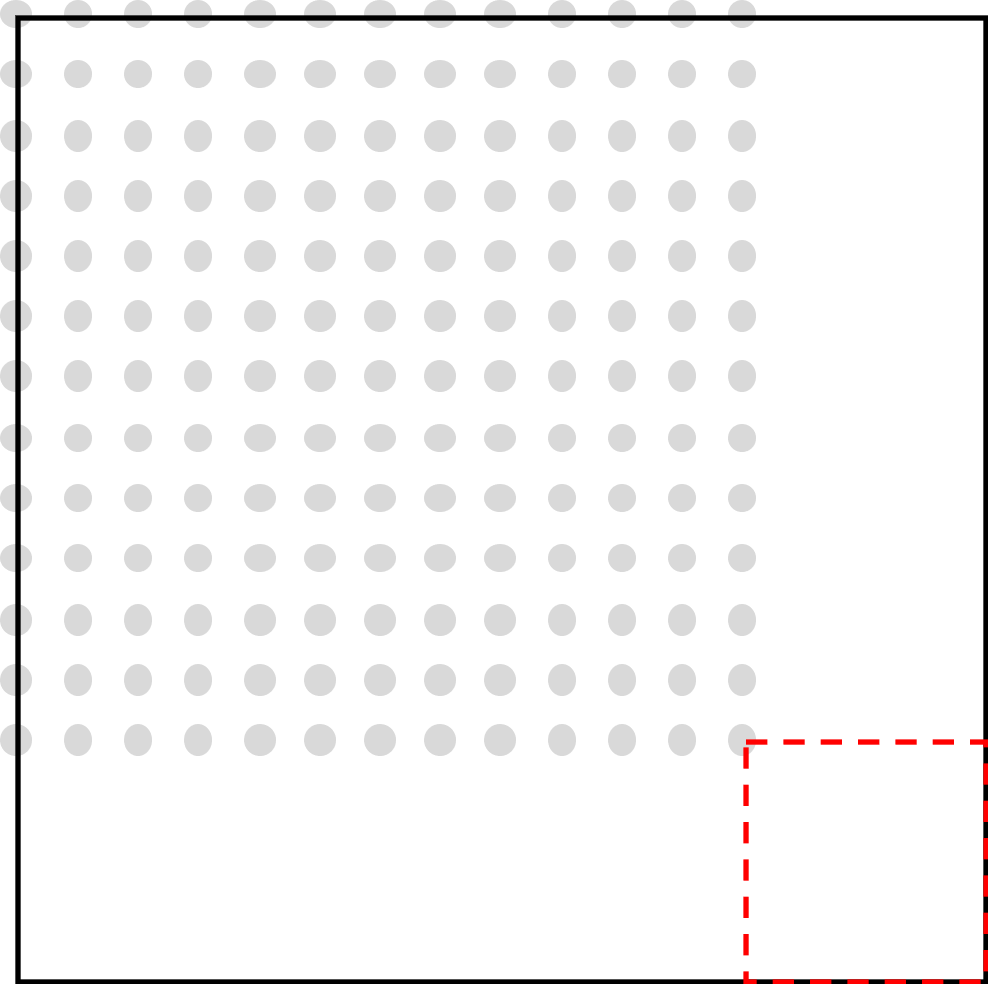


Figure 14 Random window offsets limited to a few, evenly spaced positions (gray dots)

### Local adaptation

#### General

As explained in the introduction, film grain characteristics vary significantly depending on exposure, both in terms of amplitude and spatial correlation, with the amplitude variation being the most obvious. These variations are inherent to the image formation process at the physical film level. Replicating them on synthetic grain makes it look natural by anchoring it into the image, while a fixed overlay would likely not be visually pleasing.

Here, local adaptation consists of selecting the relevant grain template and grain amplitude according to pixel values and model parameters. For example, a dependency of grain strength/amplitude on the signal intensity can be defined for each colour component. In addition, specific grain correlation parameters may depend on the signal intensity.

Local adaptation can be performed per-pixel or on a sub-block basis (based on local intensity average), with different implications, which will be detailed in the following sections.

#### Adaptation of grain shape

Adaptation of grain shape consists of selecting a specific grain template depending on local intensity, according to the model. This requires the initial stage to generate as many grain templates as required to fit the model within practical implementation limits, as described in 7.2.1.

Template selection can be applied either per-pixel or on a sub-block basis. This question arises because, since grain is spatially correlated, changing template within a block may destroy this spatial correlation, even though the random window and potential sign flip (see 7.2.2) is kept constant for the whole block whatever the template selected:

* Selecting template on a sub-block basis guarantees that the grain characteristics are preserved within the sub-block as long as the grain size is significantly smaller than the sub-block. However, that requires computing a local intensity average to select the relevant template. Computing an average over a sub-block requires line buffers, which may be too costly in hardware implementations. Also, changing template on sub-block boundaries may require deblocking so that the transition is not visible, as illustrated in Figure 15.
* Selecting a template on a pixel basis does not require line buffers nor deblocking, but requires solving the problem of spatial correlation preservation. This can be done at template generation time, by ensuring template patterns are correlated with each other, making transitions smooth, as illustrated in Figure 16. For example, templates can be the result of filtering the same underlying gaussian noise pattern with different frequency cut-offs or autoregressive coefficients. In implementations supporting a limited number of templates for storage cost reasons, transition smoothness could be further improved by pattern interpolation.



Figure 15 Sub-block-based template selection



Figure 16 Pixel-based template selection

#### Adaptation of grain amplitude

Adaptation of grain amplitude consists of scaling the amplitude of the grain template selected in the previous step by a scaling factor that depends on local intensity, according to the model.

There is no benefit in performing this adaptation on a sub-block basis: a pixel-based approach is more similar to the physical process, more closely follows the image, and avoids transitions on a fixed grid, which could potentially be visible in a video sequence.

### Block boundary treatment

Deblocking is required because of block-based randomization (see 7.2.2), especially when grain is large.

When local adaptation is performed on a sub-block basis, deblocking may be needed on sub-block borders too.

Deblocking traditionally involves low-pass filtering block borders. However, for horizontal borders, this means vertical filtering, requiring line buffers (typically two, when using 3 coefficients).

To avoid the cost of such line buffers in hardware implementation, it may be preferable to use an overlapping process instead of a filter. For example, on the first two lines, the samples of the current template matching two different random windows are blended together, the window of the current block and the block above. No line storage is required, just reading two places in the template (the address of these places can be computed on the fly).

Another option is to attenuate grain amplitude on horizontal borders.

### Blending

The final stage is to blend the synthesized film grain with the picture. Though both multiplicative and additive blending may be specified, practical implementations commonly support additive blending only, followed by appropriate clipping.

## Examples of film grain synthesis using the frequency filtering model

### SMPTE RDD-5

#### General

SMPTE RDD-5 [6] specifies a fixed point, bit-accurately reproductible process for film grain synthesis that makes use of FGS SEI message, with some restrictions:

* Frequency filtering mode (model\_id = 0)
* Additive blending mode (blending\_mode\_id = 0)
* Limitation of the range of numbers, so that computation bit depth is limited and practical (log2\_scale\_factor from 2 to 7)
* No overlaps between intensity intervals
* The number of model parameters is limited to 3, limiting it to scale and high frequency cut-offs (no low frequency cut-offs, and no cross-component correlation)
* Scale parameter is limit to 8-bit, and frequency cut-offs limited to the [2, 14] range (full band being 15)
* 4:2:0

The random generator is defined as an LFSR (the one illustrated in Figure 13), and an array of initial values is specified. A table of gaussian values is also specified to enable conversion of LSFR values, which have a uniform distribution, into a gaussian distribution.

When picture colour format is YUV 4:2:0, a specific rule is defined to convert the scaling factors and frequency cut-offs for chroma components, before any further processing. The process is then the same for all colour components (template size, randomization block size, etc, is the same).

#### Grain template generation

The grain template generation process is similar to what is described in 7.2.2.2, with 64x64 template size. It is made reproductible by making use of specific LFSR initialisation values, the gaussian table, and a specific DCT2 transform matrix.

In addition, vertical pre-deblocking is baked into the grain templates, using an attenuation technique as described at the end of section 7.2.5: an attenuation factor is applied on two lines every 8 lines, that factor depending on the vertical frequency cut-off. This can work because template selection is made constant within every 8x8 block in the current image, and vertical random offset is a multiple of 8.

Note that since initialization values are always the same, depending on frequency cut-offs only, the template generation process does not have to be repeated for every frame. Also, as frequency cut-off range is limited to 13 values in each direction, only 169 different templates exist, that could potentially be stored in a fixed database. However, only 10 different templates are allowed to be referenced by a given FGC SEI message (in other words: within a picture).

#### Randomization

Randomization block size is 16x16. Random offsets are derived from the LFSR, which is initialized with a value that depends on colour component and picture identifiers read from the bitstream (poc and idr\_pic\_id). It is required that pictures 32 of fewer frames apart in decoding order do not have the same identifier.

Random offsets are extracted from 16-bit fields of the LFSR (16 MSBs for x, 16 LSBs for y), and mapped to the required range with a modulo operation as described in annex A.1, with R equal to 16 and N equal to 52 or 56. Horizontal offset is a multiple of 4, with 13 possible values (0 to 48 in steps of 4). The modulo is then 13x4=52, applied on the 16-bit field, and the two lower bits of the result are set to zero; this is equivalent to integer division by 4, modulo 13, then multiplication by 4, which also means that only 14 bits of the LFSR are actually used in the end. Similarly, vertical offset is a multiple of 8, with 7 possible values (0 to 48 in steps of 8), giving modulo 56 (or 7 after integer division by 8), and actually using 13 bits of the LFSR.

Additionally, a random template sign inversion is derived from the LSB of the LFSR.

#### Local adaptation

Local adaptation of both grain amplitude and spatial frequency limits is supported by selecting template and scaling factor based on the average sample value over 8x8 blocks (non-overlapping). As explained in 7.2.4.3, this has some hardware cost implication, needing a 7-line buffer to compute the 8x8 average.

#### Deblocking

As discussed in section 7.2.4.2, since local adaptation can change template for each 8x8 block and template patterns are independent (uncorrelated with each other), deblocking is needed on 8x8 blocks, rather than (or in addition to) the 16x16 randomization blocks.

As exposed in the template generation section, vertical deblocking is handled at template generation stage. Horizontal deblocking is performed by smoothing both sides of the block borders, using [0.25 0.5 0.25] filter coefficients.

### Variants based on of RDD-5

#### Implementation in VTM and HM

An example film grain synthesis implementation based on SMPTE RDD-5 is present in both VTM [20] and HM [21]. The differences from RDD-5 are the following:

* Bit depth higher than 8 is supported, by scaling the film grain appropriately; for example, for 10-bit video, film grain is shifted left 2 bits before blending.
* The VTM 64x64 inverse DCT2 transform is used for template generation

#### Resolution-adaptive block size

In addition to what is described in 7.3.2.1, this example implementation uses resolution-adaptive local adaptation blocks:

* 8x8 up to HD resolution;
* 16x16 up to UHD resolution;
* 32x32 above UHD resolution. In this case, the randomization blocks are also enlarged to 32x32, with adjustment of modulo values for random offset derivation: 36 for horizontal offset and 40 for vertical offset.

#### Single-line block

In addition to what is described in 7.3.2.1, this example uses single-line blocks to avoid the usage of line buffer in the average block intensity computation used for local adaptation. Example of block size may be 8x1 up to HD resolution, 16x1 up to UHD resolution and 32x1 above UHD resolution.

This example also restricts the number of different templates within a picture to a single one (disabling local adaptation of pattern), to avoid the problem of template switching every line potentially destroying the spatial consistency of film grain, as explained in 7.2.4.2.

## Examples of film grain synthesis using the autoregressive models

### FGC SEI message based autoregressive model

In the FGC SEI message, when using the autoregressive model, the lag parameter is not signalled explicitly. Instead it is implicitly specified to be in the range of 0 to 2, inclusive, depending on the number of model values present for each intensity interval in which the film grain has been modelled. The coefficients allowed for the FGC autoregressive model are constrained as illustrated in Figure 17 where the variable *A* is the pixel aspect ratio of the film grain. The film grain pixel aspect ratio could be useful in cases in which a picture is stretched in resizing or was captured with isomorphic optics [6].

Diagram

Description automatically generated with low confidence

Figure 17. Autoregressive model parameters in FGC SEI message

The following describes an example of an autoregressive model for film grain synthesis when the following properties apply:

– 4:2:0 chroma format

– Auto-regression model with additive blending mode

– The colour space for film grain simulation is the same as for encoding.

– lag value is less than or equal to 2

– all colour correlation coefficients are equal to 0

– pixel aspect ratio for film grain is equal to 1

FGC SEI message provides the following AR coefficients per each intensity interval:

1. The standard deviation of the Gaussian noise,
2. The first order spatial correlation for neighbouring samples ( x − 1, y ) and ( x, y − 1 ),
3. The colour correlation between consecutive colour components,
4. The first order spatial correlation for neighbouring samples ( x − 1, y − 1 ) and ( x + 1, y − 1 ),
5. The aspect ratio of the modelled grain,
6. The second order spatial correlation for neighbouring samples ( x, y – 2 ) and ( x – 2, y )

Depending on the intensity interval the luma or chroma sample belongs to, the film grain sample is generated by filtering a random value with normalized Gaussian distribution and the neighbouring scaled film grain samples, as shown in Figure 17, with the corresponding AR coefficients. The output luma or chroma sample is generated by blending the decoded luma or chroma sample with the film grain sample.

### AFGS1 model

#### General

AFGS1 [23] is the film grain synthesis specification that uses ITU-T T.35 user generated metadata to signal the film grain synthesis metadata. The specification uses an autoregressive model for the grain template generation and a piece-wise linear function to model the dependency between the grain strength and the signal intensity.

#### Grain template generation

AFGS1 uses an autoregressive model for representing the film grain pattern. This model is used to synthesize the template of the film grain. Let *G*(*x*, *y*) be a zero-mean film grain sample at the position with coordinates (*x*, *y*). For the parameters lag *L* = 2, the grain sample G is calculated as follows (see Figure 18):

*G*(*x*, *y*) = *a*0 *G*(*x* − 2, *y* − 2) + *a*1 *G*(*x* − 1, *y* − 2) + *a*2 *G*(*x*, *y* − 2) +… + *z*(*x*, *y*) 7‑2

where parameters *ai* are autoregressive coefficients and *z*(*x*, *y*) is a sample of gaussian noise at position (*x*, *y*).

In AFGS1, lags of size 0 to 3 are supported.

Graphical user interface, application

Description automatically generated

Figure 18 Sample of film grain G(x, y) with AR coefficients in the luma component [26].

The description above applies to modelling of the grain for the luma component. Since film grain in YCbCr video components may be correlated, the AR models for chroma components have an additional AR coefficient to capture the correlation between the chroma component grain and a collocated luma sample grain.

When the parameters for the film grain frame are received, a 64x64 film grain template is generated for the luma component. For YCbCr 4:2:0 video, the chroma grain template of size 32x32 is generated for each chroma component.

#### Randomization

Randomization block size is 32x32 for luma, and 16x16 for chroma (when in 4:2:0 chroma format). The random offsets for chroma and luma blocks are synchronized, to keep the correlation between luma and chroma grain.

The pseudo-random number generator used in the algorithm is a shift-back linear-feedback shift register (LFSR) based on XOR, with a length of 16 bits. The corresponding feedback polynomial is *x*16 + *x*15 + *x*13 + *x*4 + 1. Two offsets for the film grain blocks are generated using eight most significant bits on the register. The chroma offsets are from 0 to 15, while luma offsets are equal to chroma offsets multiplied by two. The pseudo-random number generator is initialized in the beginning of each 32×32 luma block row to enable parallel processing.

#### Local adaptation

The model used does not allow changing the grain shape within a picture, thus only local adaptation of grain strength is needed. When adding film grain to the Y component, the following model is used:

Y´= Y + *f* (Y) *G*L  7‑3

where Y´ is the luma re-noised with film grain, Y is the reconstructed value of luma (before adding film grain), and *GL* is the luma film grain sample. *f*(Y) is a function that scales the film grain based on the value of the luma (since the film grain typically depends on the intensity of the signal, the luma film grain is modelled as a function of the signal intensity). *f*(Y) is represented with the piecewise-linear function that can be implemented as a pre-computed look-up table (LUT). For all bit depths, LUT takes 256 values, and for bit-depths higher than 8, the values between the LUT entries are obtained with linear interpolation. Up to 14 *f*(Y) pairs can be signaled to represent the scaling function for the Y component.

For a chroma component (e.g., Cb), the film grain is scaled using the following equations:

Cb´ = Cb + *f* (*u*) *G*Cb, 7‑4

*u* = *b*Cb Cb + *d*Cb Yav  + *h*, 7‑5

where u is the index corresponding to a Cb component scaling function. The index depends on both the Cb and luma components for the pixel. *Y*av is the average luma corresponding to the chroma sample, taken from one line of samples. For 4:2:0 YCbCr format, *Y*av = (*Y*1 + *Y*2 + 1) >> 1, where *Y*1, *Y*2 are neighbouring (co-located) luma samples located on the even line (numbering starts from 0).

#### Block overlap

There is an option to use overlap between the film grain values at the 32x32 film grain block boundaries. The overlap can be used to ensure that possible artifacts at the film grain block boundaries are attenuated. The overlap is applied before scaling the grain samples and adding them to the reconstructed blocks. As shown in Figure 19, current block samples overlap only with grain blocks to the right and below [26].

A picture containing crossword puzzle, shoji

Description automatically generated

Figure 19 Overlapped blocks

The overlap between the luma blocks is two samples, and between the chroma blocks (in YCbCr 4:2:0) is one sample. To enable the overlap, the random offsets or overlapped grain values for the above row need to be saved before the current row is processed. The operations used in the grain overlap are as follows:

*G*cur(x, 0) = (27 \* *G*up(*x*, 32) + 17 \* *G*cur(*x*, 0) + 16) >> 5 7‑6

*G*cur(x, 1) = (17 \* *G*up(*x*, 33) + 27 \* *G*cur(*x*, 1) + 16) >> 5 7‑7

where *G*cur(*x*, 0) are samples of row 0 of the current block and *G*up(*x*, 32) are samples of row 32 of the upper block.

The overlap operation between chroma blocks is done as follows:

*G*cur(x, 0) = (23\* *G*top(*x*, 16) + 22 \* *G*cur(*x*, 0) + 16) >> 5 7‑8

#### Blending

The generated film grain is consecutively applied for each 32x32 luma block (16x16 chroma block) of reconstructed video samples, in raster scan order, using additive blending.

The following operation for adding grain to the samples of a luma block is used:

Y´(*x*, *y*) = clip3( Y(*x*, *y*) + ((GL(*x + sx*, *y + sy*) \* *f* (Y) + 2shift−1 ) >> shift ), *a*, *b*),   7‑9

where *a* and *b* define the legal range, *x* and *y* are coordinates inside the block, and parameter shift controls scaling of the film grain.

## Example of film grain synthesis supporting both the frequency filtering and autoregressive models

### General

The “Versatile Film Grain” synthesis software [xx] supports both frequency filtering and autoregressive models, with local adaptation of both grain shape and strength, while not requiring line buffers.

More details about software usage and organization are given in annex B.2.2.

### Grain template generation

Depending on frequency or auto-regressive model, 64x64 templates are generated according to the process described in section 7.2.2.2 or 7.2.2.3. For chroma in 4:2:0 colour format, template size is 32x32.

For the autoregressive model, a larger template is first created then cropped to 64x64 (or potentially 32x32 for chroma) to leave space for the convergence of the autoregressive filter.

The same underlying gaussian noise (same random generator initialization) is used to generate all grain templates to be used in a given picture, so that transition from one to another is smooth if change happens at pixel level, as discussed in section 7.2.4.2.

The maximal number of simultaneous templates supported is configurable, and when more than supported are requested by received parameters, graceful degradation is implemented by restricting to the ones listed first. This requires transmitting the most important film grain characteristics first in an FGC SEI message, which is possible because the ordering of intensity intervals is flexible.

### Randomization

The random generator is a 32-bit LFSR, the same as the SMPTE RDD5 example but bit-reversed (the right-shifting variant shown in Figure 28 and discussed in annex A.2). Similarly to RDD5, an array of initialization values is defined.

Randomization block size is 16x16 (for luma). Random offsets and template sign flips for all colour components are derived from different 10-bit bitfields of the 32-bit LFSR register; those bitfields partially overlap.

For a luma 16x16 block, 13 horizontal and 12 vertical positions are allowed. The transition from a 10-bit field to 12 or 13 positions uses the multiplication and shift technique described in annex A.1, which translates into one (12 = 8 + 4) or two (13 = 8 + 4 + 1) adders in hardware.

### Local adaptation

Local adaptation is sample-based. Based on the intensity of the current sample, the appropriate scaling factor and template index is selected. Pattern interpolation is supported.

Performing sample-based adaptation avoids the potential cost of line buffers in an hardware implementation.

### Deblocking

Horizontal deblocking uses a 3-tap filter, while vertical deblocking uses overlapping, to avoid line buffers.

### Blending

Only additive blending is supported, with clipping to allowable range.

# Film grain analysis

## General

The film grain analysis process is applied on the encoder side and is a non-normative process regardless of the synthesis method and standard in use. It can be implemented as a pre-processing step or as a part of the encoding process. Ultimately, it provides indicative features of the film grain in accordance with the selected parameterized model and supported metadata. It determines film grain model parameters to be sent in the appropriate metadata mechanism, e.g., in an SEI message, to enable content-aware decoder-side film grain synthesis. The analysis process is not mandatory, and manually tuned parameters can be provided to the encoder based on an input configuration file.

Figure 20 depicts a simplified framework for film grain analysis. As an input to the process, besides the source video, the denoised representation is also required. Edge and complex texture analysis is performed in order to determine a map of flat and non-flat regions in the scene. This is done since high-frequency components, such as edges and texture, can interfere with the analysis process of the film grain, which also resides in high-frequencies.

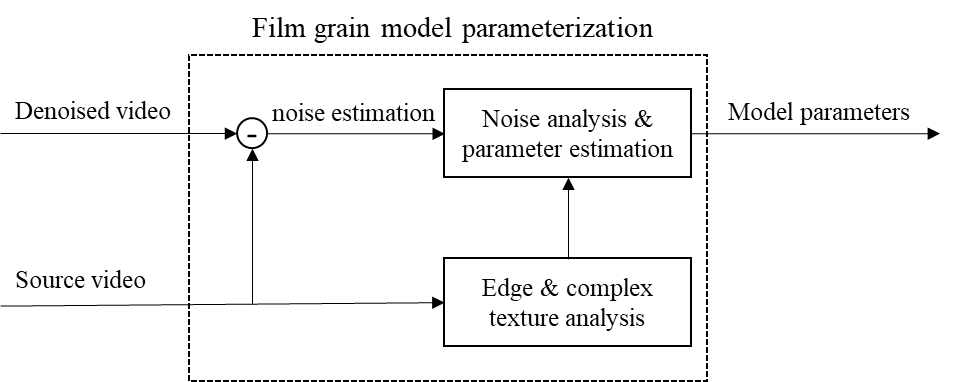


Figure 20. Film grain analysis and parameter estimation general steps.

The presented generalized model is one common approach to estimate noise and noise parameters. Different approaches are possible, for example, by using appropriate technics noise can be estimated without the need for denoised version of the source video. Anyhow, the most commonly used film grain analysis workflow consists of the following commonly implemented steps, divided into two groups.

Film grain model independent steps (independent from film grain parameter format):

1. a pre-processing step to produce a noise estimate (a.k.a. film grain image). It consists of:
   1. denoising the video and
   2. finding the difference between source and denoised pairs to get noise estimate (film grain image).
2. a pre-processing step to produce other information about the characteristics of the input video such as edge and texture analysis.

Model specific step:

1. film grain analysis and parameter estimation step to determine model parameters based on the outputs from 1) and 2).
   1. estimate grain amplitude based on underlying picture intensity
   2. estimated frequency limits (cut-off frequencies) for frequency based model or auto-regressive parameters on relevant picture intensity levels

Film grain analysis and parameter estimation depend on the selected parameterized model. Some examples of a possible implementation are provided in the following subclauses.

## Denoising and image analysis

### Denoising

In existing video distribution systems, denoising is a commonly available and widely used process. For example, denoised sequence coding can significantly improve compression efficiency since uncorrelated noise is removed [16]. Possible solutions for denoising the input video source include the use of motion adaptive, or motion compensated temporal filtering (MCTF) methods. MCTF methods utilize motion search and motion compensation algorithms to exploit temporal redundancies and isolate the video signal from the noise signal, and therefore achieve denoising. Motion adaptive methods perform simple decisions using temporally neighbouring samples to determine noise. Although they are commonly less performing, they usually also have lower complexity, e.g., one variant is given in [13] . Other approaches can be used instead of MCTF. Block-matching and 3D filtering (BM3D) is one of the most popular hand-crafted image denoising algorithm [12]. In some implementations, reconstructed video sequence can be used, reducing the need for denoised sequence in film grain estimation (depending on the quantization parameter, reconstructed sequence can highly approximate denoised sequence with most of the details preserved, but with low frequencies (noise) removed).

In contrast to traditional analytical denoising approaches, recent advances utilize data driven deep learning techniques to provide state of the art performance [13][14][15] . However, deep learning approaches usually suffer from excessive computational complexity.

After denoising, film grain images that represent noise estimates are generated by computing the difference between the original input frames and the filtered/denoised frames that correspond to the same time instance.

### Edge and texture analysis

Since denoising often alters actual picture details in addition to noise, the film grain image resulting from the difference of source picture and denoised picture can contain more than film grain in the textured or edge areas. It is better to avoid those areas, and estimate film grain parameters on flat (textureless) areas.

One possibility could be to detect flat blocks using gradient-based features with thresholds that are more lenient to allow for correct grain modelling in extreme cases, for example, as proposed in [17] . In this method, for each block in the denoised image, horizontal and vertical gradients are computed and stored in gx and gy vectors, then their 2x2 covariance matrix C is computed, and its features are compared with thresholds for the block to be accepted as flat enough: the ratio of eigenvalues (indicative of texture directionality), their sum (indicative of texture strength), and the norm of C.

Another approach could involve well established edge detection algorithms such as Canny edge detector in conjunction with morphological operations. To create a map of flat regions, one approach can be to perform analysis on three scales using the source sequence: the original and two subsampled resolutions, one by a factor of 2 and another by a factor of 4 horizontally and vertically. At each scale, a Canny edge detector is applied (typically without the classical Canny detector blurring step) to determine edges. Morphological operations are applied afterwards at each scale to extend the influence of the edges and to have higher confidence that the edges do not jeopardize the process of estimating the film grain. The obtained maps at the subsampled scales can then be upsampled to the original resolution and combined to form final map. Additional morphological operations can be subsequently applied to the final map. As an illustration, VTM implementation applies 4-pass dilations at original scale, 3-pass dilatation on subsampled scale by factor 2, and 2-pass dilations on subsampled scale by factor 4. After upsampling and combining all scales in one flat region map at full resolution, 2-pass dilations followed by 1-pass erosion are applied.

The film grain image and map of flat regions are then provided to the noise analysis & parameter estimation module.

Additionally, a planar trend removal can be applied [23]. In this implementation, planar trends in the selected flat blocks are removed by fitting a plane, *h*(*x*, *y*) = *ax* + *by* + *c*, to the block, and by computing the trend-removed noise patch as *N’*(*i*, *j*) = *N*(*i*, *j*) - *h*(*i*, *j*).

## Determination of grain scaling function

### General

Since film grain is dependent on the local characteristics of an image, different amplitude of film grain can be applied for different intensities or intensity intervals of an input image in order to get film grain of appropriate strength before finally blending it to the input image. Thus, initial film grain pattern is scaled to the proper intensity based on the scaling factor. Multiplication of a pattern by the scaling factor determines the level at which the film grain will be perceived at the final image, and by doing that it is ensured that the film grain is simulated at the correct scale.

The relationship between local image intensity and grain amplitude, called here scaling function, can be determined by analyzing the film grain image, the filtered image, and the flat-region map produced during pre-processing. Scaling function, for example, can be represented as polynomial function, or lower-degree approximation of a (polynomial) scaling function can be considered depending on the implementation. For example, piece-wise linear function or stepwise constant scaling function as illustrated in Figure 21 can be considered implementation.

Chart

Description automatically generated

Figure 21. An example of film grain scaling function defined with the intensity interval and scaling factors. This approach is used in FGC SEI message defined in H.264, HEVC and VVC.

The analysis is typically performed on a grid of non-overlapping bocks, that belong to a flat region, as determined by the edge and texture analysis stage. For each block, two features are computed: the first one is the average intensity of the block in the denoised image, and the second one is the noise level of the block in the film grain image. They define observation points that are used in further analysis.

Scaling function can be obtained by fitting a polynomial function to the observations. Some additional processing of the observations can be required before fitting function, for example, to remove outlier and to improve estimated function precision.

### An example of FGC SEI message scaling factor estimation

#### General

This example is taken from the VTM reference code, where image is analyzed by blocks of different size depending on resolution: 8x8 for HD and below, 16x16 up to UHD, and 32x32 above. The further process, that operates on observation points collected in previous step, is described as follows.

#### Data points regularization and discarding potential outliers

Data points regularization is used to regularize high variations and to control the excessively fluctuating points. The following regularization function can be used:

k=3.0;

m=0.5;

tmp = k \* pow((double)(var), m) + .5;

noise\_strength = Floor( tmp );

Coefficient m defines regularization of dispersity/heteroscedasticity of observation points.

Coefficient k defines the level of film grain to be added back, it should be aligned to the level of the film grain being removed from the source (during the filtering and compression).

After regularization is applied, the observation points with the noise\_strenght higher than 16 << (bitDepth – 8) are discarded from the calculation to limit observation points to meaningful values. If noise\_strength is higher than the maximum defined value, there is a possibility that noise estimation is biased by edges and other high frequency components that are removed during the filtering.

#### Curve fitting and curve quantization

##### General

The observation points are used to determine scaling factors (grain strength) in two steps:

* derive a scaling function by two-pass curve fitting
* quantize the fitted curve to produce a stepwise function

The process is also illustrated in Figure 22.

|  |  |
| --- | --- |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| a | e |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| b | f |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| c | g |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |
| d | h |

Figure 22. 2-pass curve fitting process

##### Sub-range averaging

On Figure 22a observation points are already regularized and limited to maximum value. Complete intensity dynamic range (horizontal axis) is divided (uniformly quantized) into non-overlapping intervals of size 16. Observation points are then grouped per intervals. For each intensity sub-range, the average value of the variance is calculated only if minimum number of points within the sub-interval is 8. It is illustrated on Figure 22b.

##### Discarding potential outliers (1st pass)

If there is single point without any neighbouring points, it is considered as an estimation error and it is removed from further calculations. Single points are filtered out, see the missing point in Figure 22c compared to Figure 22b. This step is highly useful in the corner cases, for example, if singe point appear at the intensity range where film grain should not be present or at least its strength should be low.

##### Extreme points extrapolation

A step of extension of points to the left and to the right towards the zero is then applied to smooth transition from intensities with film grain to the intensities without film grain. The process finds the most left and most right point and extends the data-point range, e.g., see added points in Figure 22c. At most 4 new points are added to the left and 4 new points to the right.

##### Discarding potential outliers (2nd pass)

Also remove points if intensity (horizontal axis) is less than 40 or larger than 950 (for 10-bit signal; these values are shifted for bitdepths other than 10). Indeed, film grain is usually not added to the very dark and very bright regions.

##### Curve fitting (1st pass)

Next step is to fit the parametrized curve by using 4rd order polynomial fitting, Figure 22d.

##### Discarding potential outliers (3rd pass)

+0.6\*std and -1.2\*std around the first fitted curve is used to filter out the points outside the given range. This step filters out remaining outliers and biased variance points.

##### Curve fitting (2nd pass)

The remaining points are entering second pass of curve fitting. Previous processes are repeated (8.3.2.3.2 to 8.3.2.3.7 illustrated on Figure 22f to Figure 22h) and again the polynomial curve is fitted, but this time using reduced set of observation points. The final scaling function is illustrated on Figure 22h.

#### Final scaling function approximation

Final step approximates the scaling function represented by the 4rd order polynomial curve by a simplified stepwise scaling function, see Figure 23. The stepwise function is derived by using Lloyd-max non-uniform quantization with 4 quantization levels. Quantizer is adapted (trained) for each new set of points on-fly.

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated | Chart, box and whisker chart  Description automatically generated |
| a | b |

Figure 23. Fitted curve (left) and quantized curve (right).

The use of stepwise scaling function leads to several intensity intervals and scaling factors as it is illustrated Figure 22b. Note, stepwise scaling function is requirement of the FGC SEI design. Currently, film grain SEI message as defined in VSEI and implemented in VTM comprises the following parameters that define film grain scaling function which is applied at the decoder/synthesis side:

* fg\_intensity\_interval\_upper\_bound[ c ][ i ] less than fg\_intensity\_interval\_lower\_bound[ c ][ i+1 ]
* fg\_comp\_model\_value[ c ][ i ][ 0 ],

where fg\_comp\_model\_value[ c ][ i ][ 0 ] represents the scaling factor, i is index of the interval, and c is colour component. One example of such scaling function is also given in Figure 21. Note that each step of a given stepwise function represents one intensity interval, defined with its bounds (lower and upper bound) and scaling factor value.

Analysis is performed in the same way for all colour components.

### An example of AFGS1 scaling factor estimation

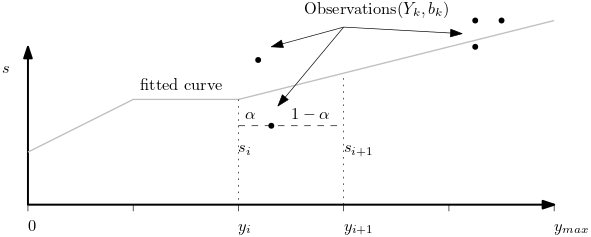


Figure 24. Illustration of the observation points (Yk , *b*k) containing average block intensity

Figure 24 is an illustration of the how observation points (Yk , *b*k) containing average block intensity, Yk, and the measured noise scale, *b*k, are used as constraints when solving for the noise strength values si which are a function of block intensity [24][26].

The result of the flat block estimation gives a number of tuples relating the block intensity, *Y,* to the noise level within the block *bk*. A piecewise representation of the noise-strength mapping is assumed. Let ***s =*** [*s1,* *s2, …, sn*]be the values representing the piecewise linear function, equally sampled on the [0, *Y*max] domain. Each of the observations, (*Yk* , *bk*), provides a constraint on the noise strength table. In practice, n << 28, so the constraints are applied on interpolated values between adjacent points (Figure 25). Letting ik denote the largest index of the lookup table below *Yk*, we can represent the piecewise constraint on the two adjacent points as

(1 – )*sik*+*sik*+1 = *bk* 8‑1

These equations for all (*Yk* , *bk*) are stacked into a linear system, which can be solved for ***s***. In order to account for some variables having no or little constraints, a regularization term is added to smooth across lookup table bins that are lacking data. This can happen when all the flat-blocks are identified in low-intensity regions, as there will be no constraints on the higher intensity range. With regularization, the following objective is minimized:

mins |A s – b|2 + | s |2 8‑2

The parameter β is scaled proportionally to the number of observations used to construct A. The solution to the regularized problem is the solution to the following system of equations:

*ATA s* – β ∇2*s* = *ATb* 8‑3

## Determination of cut-off frequencies for frequency filtering model

### General

Cut-off frequencies can be estimated as follows. The film grain image (noise estimate) is scanned block by block, using non-overlapping blocks grid, where block size (denoted as N) depends on the specific implementation. Only blocks within the flat part of film grain image are processed further. For each block within a flat region, a forward transform (usually DCT-2) is applied. Then, each coefficient resulting from the transform is squared. Afterwards, an average squared transformed block Bavg is computed over all available squared transformed blocks *Bi* (pixel-wise) as:

8‑4

where *K* is the total number of blocks that is used for the calculations. Thereafter, the average columns Bc and by rows Br vectors are calculated (the DC component for the first row and the first column is discarded from the computation) as follows:

8‑5

8‑6

The average vectors Bc and Br are regularized to suppress peaks. The average vectors are represented as a curve and its intersection points with a total average value avg of a block Bavg is computed (). The value of avg is therefore subsequently used as a threshold for other computations.

Based on the analysis of the intersection point(s), cut-off frequencies are obtained. The appropriate intersection points are chosen to represent the cut-off frequencies of the frequency-filtered film grain model.

This process can be repeated to estimate frequency limits for different intensity intervals, as is allowed by FGC SEI and could be required to accurately model the film grain, since in a photographic process, its spatial frequency highly depends on exposure.

The scaling parameters signalled for each intensity interval depend on the noise variance estimated in the previous step, but also need to factor in the frequency limits, since on synthesis side, different frequency limits lead to different grain amplitudes.

If no intersection points are found, film grain is not present in the input frame. If the analysis process determines that no grain is present in the source video, the appropriate syntax element values could be set to indicate that synthesis will not be performed at the decoder side.

### An example of FGC SEI message cut-off frequency estimation

The following method provides an example according to the VTM implementation which is based on SMPTE RDD-5 specification with some additional modifications, as described in 7.3.2.1 and 7.3.2.2.

The illustrated implementation limits filtering to low pass filtering as defined by SMPTE RDD-5, even that FGC SEI supports band pass filtering.

An implementation of the FGC SEI message parameter estimation that conforms to the method described in the FGC SEI message [2] uses 16x16 arrays to define film grain patches. However, the methods specified in [6] and subclause 7.2.2 of this document use 64x64 arrays. Thus, 64x64 DCT-2 as defined in [1] can be used within analysis process. Some implementations can choose to use 64x64 DCT-2 as described in [6] instead.

Chart

Description automatically generated

Figure 25. Cut-off frequency estimation (intersection points x1 and x2 represent estimated frequencies).

Figure 25 illustrates the process of horizontal and vertical cut-off frequency estimation on a 64x64 arrays. For a give intersection points x1 and x2, the model values are set to (according to the SMPTE RDD 5 specification scaling to 16x16 arrays is needed):

comp\_model\_value[ c ][ I ][ 1 ] = Clip3(2,14,(x1-1)>>2) and

comp\_model\_value[ c ][ i ][ 2 ] = Clip3(2,14,(x2-1)>>2),

where *i* is the index of the interval, and *c* is the colour component index. If no intersection points are found, film grain is not present in the input frame.

## Determination of autoregressive model coefficients

To estimate the autoregressive coefficients, the model described in 7.4.2 is used, and the coefficients *ai,j* that best approximate the grain image in the flat regions are chosen.

Either they are determined by least squares error minimization using all pixels from grain image in flat regions, or by estimating the auto-correlations necessary to build the Yule-Walker equations .

More details on the AFGS1 and AV1 estimation of the autoregressive model parameters can be obtained from [24].

# Film grain metadata

## General

The VVC, HEVC, and AVC standards natively support signalling of film grain parameter values using well defined supplemental enhancement information message supplement. AV1 standard signal the film grain synthesis parameters as part of the bitstream since film grain synthesis is mandatory in AV1. In addition, the AGFS1 standard provides a mechanism for signaling film grain parameters as the ITU-T T.35 metadata that is compatible with most video codec standards.

The VVC, HEVC, AFGS1, and AVC standards currently directly specify a supplemental enhancement information (SEI) message for indicating the usage of film grain synthesis when decoding and rendering video or image data. In particular, VVC, HEVC, and AVC support the film grain characteristics (FGC) SEI message, which is capable of indicating two different film grain models and different blending modes. More specifically, this SEI message supports a frequency filtering and an autoregressive film grain model, and additive or multiplicative blending modes. It can also indicate one or more intensity intervals, the film grain variance for each intensity interval, and the spatial frequency characteristics of the film grain for each intensity interval, providing considerable flexibility to content creators and encoding manufacturers on signaling different types and levels of film grain noise.

An alternative film grain synthesis scheme, such as a different use of an autoregressive model, can also be indicated through the use of the ITU-T T.35 registered or unregistered user data SEI messages that are also supported in these standards. The VVC, HEVC, and AVC standards intentionally do not specify any constraints or limitations on the postprocessing techniques that can be used with decoded data from such user data SEI messages, therefore enabling more applications and implementations.

## Film grain characteristics SEI message

### General

Film grain metadata for use with VVC, HEVC, and AVC can be signalled via an FGC SEI message specified in the corresponding coding standard. For VVC, the SEI payload Type is specified in [1] Annex D and the syntax and semantics are specified in [2] clause 8.5. For HEVC, the FGC SEI message is specified in [3] clauses D.2.13 and D.3.13. For AVC, the FGC SEI message is specified in [4] clauses D.1.21 and D.2.21. The different FGC SEI message versions are similar but have some standard-specific differences. For example, persistence of the FGC SEI message is specified differently for AVC than it is for HEVC and VVC.

### Interpretation of FGC SEI message syntax

A guide to interpretation of the syntax in the FGC SEI message [2] is shown in Table 2. Similar interpretations apply to the FGC SEI messages specified in AVC [4] and HEVC [3].

Table 2. Guide of the FGC SEI message syntax interpretation

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Significance** |
| **fg\_model\_id** | 0,1 | Model to be used in grain synthesis. 0: Frequency filtering, 1: Auto-regression |
| **fg\_separate\_colour\_description\_present\_flag** | 0,1 | Defines whether the colour description for the film grain specified is same as for the coded video sequence. |
| **fg\_blending\_mode\_id** | 0,1 | Blending mode used to combine grain and decoded samples.0: Additive, 1: Multiplicative |
| **fg\_comp\_model\_present\_flag** | 0,1 | Defines the presence of film grain model parameters for each colour component |
| **fg\_num\_intensity\_intervals\_minus1** | 0-255 | Defines the number of intensity intervals for each colour component |
| **fg\_num\_model\_values\_minus1** | 0-5 | Specifies the number of component model values available in the SEI ( default values will be used for the remaining component model values) |
| **fg\_intensity\_interval\_lower\_bound** | 0-255 | Lower bound for each intensity intervals for which the model is applicable |
| **fg\_intensity\_interval\_upper\_bound** | 0-255 | Upper bound for each intensity intervals for which the model is applicable |
| **fg\_comp\_model\_value** |  | Component model values have different meaning depending on the value of fg\_film\_grain\_model\_id |
| **fg\_characteristics\_persistence\_flag** | 0,1 | Indicates the persistence of the FGC SEI message. |

The component model values specify the strength, shape, density, and other characteristics of the film grain. A unique set of component model values can be signalled for each intensity interval for each colour component to be processed. The value range of each syntax element may be constrained by specific practice or implementation.

When the frequency-filtering film grain model is signalled (film\_grain\_model\_id = 0), comp\_model\_values are interpreted as follows for each colour component, c, and intensity interval, i.

* comp\_model\_value[ c ][ i ][ 0 ] : the standard deviation of Gaussian noise
* comp\_model\_value[ c ][ i ][ 1 ] : horizontal high cutoff frequency
* comp\_model\_value[ c ][ i ][ 2 ] : vertical high cutoff frequency
* comp\_model\_value[ c ][ i ][ 3 ] : horizontal low cutoff frequency
* comp\_model\_value[ c ][ i ][ 4 ] : vertical low cutoff frequency
* comp\_model\_value[ c ][ i ][ 5 ] : correlation between consecutive colour components

fg\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of 0 to 2fgBitDepth[ c ] − 1, inclusive. The derivation of fgBitDepth[ c ] value is specified in [2].

When the autoregressive film grain model is signalled (film\_grain\_model\_id = 1), comp\_model\_values are interpreted as follows.

* comp\_model\_value[ c ][ i ][ 0 ] : the standard deviation of Gaussian noise
* comp\_model\_value[ c ][ i ][ 1 ] : first order correlation for neighbouring samples ( x − 1, y ) and ( x, y − 1 )
* comp\_model\_value[ c ][ i ][ 2 ] : correlation between consecutive colour components
* comp\_model\_value[ c ][ i ][ 3 ] : first order correlation for neighbouring samples ( x − 1, y − 1 ) and ( x + 1, y – 1 )
* comp\_model\_value[ c ][ i ][ 4 ] : aspect ratio of the modelled grain
* comp\_model\_value[ c ][ i ][ 5 ] : second order correlation for neighbouring samples ( x - 2, y ) and ( x, y - 2 )

fg\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of −2( fgBitDepth[ c ] − 1 ) to 2( fgBitDepth[ c ] − 1 ) − 1, inclusive. The derivation of fgBitDepth[ c ] value is specified in [2].

The frequency-filtering and autoregressive models for film grain synthesis share the following processing steps:

* Determination of applicable intensity intervals for each sample for each applicable colour component. A different set of FGS model parameters may be signalled in the FGC SEI message for each intensity interval.
* Generation of synthesized grain.
* Blending of synthesized grain and decoded image.

The methods for determining intensity intervals and generating synthesized grain depend on the signalled FGS model. Methods for the frequency-filtering model are described in subclause 7.3. Methods for the autoregressive model are described in subclause 7.4.

fg\_comp\_model\_value[ c ][ i ][ 0 ] together with intensity interval boundaries is used to define a scaling function (piece-wise constant scaling function). The scaling function indicates the level at which the film grain will be perceived in the final output frame.

Additive (fg\_blending\_mode\_id = 0) and multiplicative (fg\_blending\_mode\_id = 1) grain blending methods are specified in [2].

## AFGS1 metadata (SEI message)

AOMedia Film grain synthesis model (AFGS1) [23] is an autoregressive film grain synthesis model that can be indicated in the VVC, HEVC, and AVC standards using the ITU-T T.35 registered or unregistered user data SEI messages. The AFGS1 model is equivalent to the AV1 [25] film grain synthesis algorithm on the frame level.

The following sub-sections provide interpretation of the syntax for this film grain synthesis model.

### Interpretation of AFGS1 metadata syntax

A guide to interpretation of the syntax in the AFGS1 metadata is shown in 3.

Table 3. Guide to interpretation of the AFGS1 metadata

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Significance** |
| **apply\_grain** | 0,1 | specifies whether film grain should be added to this frame:  0: not applied,  1: applied |
| **grain\_seed** | 0-65535 | specifies the starting value for the pseudo-random number register used during film grain synthesis. |
| **film\_grain\_param\_set\_idx** | 0-7 | an index of a film grain parameter set. Up to 8 parameter sets may be simultaneously stored. |
| **update\_grain** | 0,1 | 1: means that a new set of parameters is sent for the current film\_grain\_param\_set\_idx.  0: previous set of parameters in film\_grain\_param\_set\_idx should be used. |
| **num\_y\_points** | 0-14 | number of points for the piece-wise linear scaling function of the luma component. |
| **point\_y\_value[ i ]** | 0-255 | x (luma value) coordinate for the i-th point of the piecewise linear scaling function for luma component (In case of 10 bit video, these values correspond to luma values divided by 4). |
| **point\_y\_scaling[ i ]** | 0-255 | scaling (output) value for the i-th point of the piecewise linear scaling function for luma component. |
| **luma\_only\_grain** | 0-1 | 1: film grain synthesis process is only applied to the luma component.  0: film grain synthesis process can be applied to the chroma components. |
| **chroma\_scaling\_from\_luma** | 0-1 | 1: chroma scaling is inferred from the luma scaling.  0: chroma scaling is signaled independently. |
| **num\_cb\_points** | 0-10 | number of points for the piece-wise linear scaling function of the Cb component. |
| **point\_cb\_value[ i ]** | 0-255 | x coordinate for the i-th point of the piecewise linear scaling function for Cb component (In case of 10 bit video, these values correspond to luma values divided by 4). |
| **point\_cb\_scaling[ i ]** | 0-255 | scaling (output) value for the i-th point of the piecewise linear scaling function for Cb component. |
| **num\_cr\_points** | 0-10 | number of points for the piece-wise linear scaling function of the Cr component. |
| **point\_cr\_value[ i ]** | 0-255 | x coordinate for the i-th point of the piecewise linear scaling function for Cr component (In case of 10 bit video, these values correspond to luma values divided by 4). |
| **point\_cr\_scaling[ i ]** | 0,…, 255 | scaling (output) value for the i-th point of the piecewise linear scaling function for Cr component. |
| **grain\_scaling\_minus\_8** | 0-3 | represents shift – 8 applied to the values of the chroma component. The parameter determines the range and quantization step of the standard deviation of film grain. |
| **ar\_coeff\_lag** | 0-3 | determines the number of auto-regressive coefficients for luma and chroma. |
| **ar\_coeffs\_y\_plus\_128[ i ]** | 0-255 | specifies auto-regressive coefficients used for the Y plane. |
| **ar\_coeffs\_cb\_plus\_128[ i ]** | 0-255 | specifies auto-regressive coefficients used for the Cb component. |
| **ar\_coeffs\_cr\_plus\_128[ i ]** | 0-255 | specifies auto-regressive coefficients used for the Cb component. |
| **ar\_coeff\_shift\_minus\_6** | 0-3 | specifies the range of the auto-regressive coefficients. Values of 0, 1, 2, and 3 correspond to the ranges for auto-regressive coefficients of [-2, 2), [-1, 1), [-0.5, 0.5) and [-0.25, 0.25) respectively. |
| **grain\_scale\_shift** | 0-3 | specifies how much the Gaussian random numbers should be scaled down during the grain synthesis process. |
| **cb\_mult** | 0, …, 255 | a multiplier for the Cb component used in derivation of the input index to the Cb component scaling function. |
| **cb\_luma\_mult** | 0, …, 255 | a multiplier for the average luma component used in derivation of the input index to the cb component scaling function. |
| **cb\_offset** | 0- 511 | an offset used in derivation of the input index to the cb component scaling function. |
| **cr\_mult** | 0- 255 | a multiplier for the Cr component used in derivation of the input index to the Cr component scaling function. |
| **cr\_luma\_mult** | 0- 255 | a multiplier for the average luma component used in derivation of the input index to the Cr component scaling function. |
| **cr\_offset** | 0- 511 | an offset used in derivation of the input index to the Cr component scaling function. |
| **overlap\_flag** | 0, 1 | 1: the overlap between film grain blocks shall be applied.  0: the overlap between film grain blocks shall not be applied. |
| **clip\_to\_restricted\_range** | 0, 1 | 1: clipping to the restricted (studio) range shall be applied to the sample values after adding the film grain  0: clipping to the full range shall be applied to the sample values after adding the film grain. |

The number of luma AR coefficients is determined as

numPosLuma = 2 \* ar\_coeff\_lag \* ( ar\_coeff\_lag + 1 ).

The number of chroma AR coefficients is typically found as

numPosChroma = numPosLuma + 1.

The last chroma AR coefficient models correlation between the chroma grain sample and a co-located luma grain sample.

1. : Considerations on the derivation of x/y offsets from random generator

Annex A contains more detailed considerations on the process of deriving x/y offsets from a random number generator, as mentioned in section 7.2.3.4.

* 1. Preserving uniform distribution when offset range is not a power of two

The range of random values needed for the x and y offsets depends on the block size and offset alignment, as seen in section 7.2.3.

Going from values of *p* with a uniform distribution in the range of 0 .. 2R-1 (typical of a pseudo-random generator, or a bit field extracted from it), with R being the number of bits of p, to a variable *x* (random offsets) withuniform distribution in a different range 0 .. N-1, where N is not necessarily a power of two, can be achieved with different methods, which have different implications.

One method is to take a modulo: *x* = *p* % N. This splits the range of *p* into Ceil(2R ÷ N) bins of the same size, except the last one which can be smaller, and *x* is basically an offset within the current bin, as illustrated in Figure 26. Since the number of bins is typically high and they all have the same size except the last one, the resulting distribution uniformity is typically good. However, modulo is equivalent to a division, and when N is not a power of two, this can be costly for hardware implementations.

A picture containing graphical user interface

Description automatically generated

Figure 26. Conversion of range 0..15 to 0..2 using modulo

To avoid a non-power-of-two modulo, another method is to multiply by N, then take the integer division by a power of two (bit masking, or right shift): *x* = ( *p \** N ) / 2R. This divides the range of *p* into N bins, with *x* the zero-based index of the bin, as illustrated in 27. The bins do not have the same size: some can contain 2R/N values while others 2R/N + 1, which makes the distribution uniform up to N ÷ 2R (relative to the probability of one bin). The hardware cost of the multiplication by N is optimized by selecting N with a simple decomposition into powers of two (to minimize the number of adders for the multiplication), and a low number for R (to minimize the size of the adders), while keeping the uniformity error at an acceptable level. (relative to the probability of one bin). The hardware cost of the multiplication by N is optimized by selecting N with a simple decomposition into powers of two (to minimize the number of adders for the multiplication), and a low number for R (to minimize the size of the adders), while keeping the uniformity error at an acceptable level.

A picture containing schematic

Description automatically generated

Figure 27. Conversion of range 0..15 to 0..2 using multiplication and shift

* 1. Considerations on left of right-shifting LFSR

Another consideration to optimize spacing between offsets in addition to alignment (restricting to a few evenly spaced locations) relates to how the random generator is implemented and what bit field is used. The example given is an LFSR. Such generator could shift left and feed the new (pseudo-random) bit from the right (LSB), as in Figure 13; then two successive values of the LFSR would be related by a multiplication by 2, and potentially +1. The same holds for a continuous bitfield extracted from the LFSR (in the same order), for example the 16 lower bits. For low values, and more for the multiplication method described in section A.1, this could lead to successive offsets that are similar, with the eye potentially identifying the duplication/shift. Another option is that the LFSR shifts right and feeds the new bit from the left (MSB), as in Figure 28; that way, two successive LFSR values are related by a division by two, and potentially +half range, which in turn means that successive offsets are less likely to be close.

Table

Description automatically generated with medium confidence

Figure 28 Right-shifting LFSR with polynomial (*x*31 + *x*3 + 1)

* 1. Specific considerations when offset range is a power of two

When N is a power of two, both modulo and multiplication techniques simplify to a bitfield extraction, at no cost in hardware. Using a right or left-shifting LFSR is then an independent consideration.

1. : Example implementations of film grain synthesis technologies

Annex B describes several example implementations of FGC SEI message and film grain analysis and synthesis technologies.

3. 1. FGC SEI message insertion and manipulation
      1. Explicit insertion of FGC SEI message with VTM and HM encoder

VTM [18] and HM [19] reference software implement insertion of an FGC SEI message in the encoder. In order to do so, several encoder configuration parameters are provided and need to be specified by user. In fact, all the parameters defined and described in the 7.3 and 9.2 need to be provided by user to the encoder.

At first, SEIFGCEnabled is used to enable FGC SEI encoding. If other parameters are not provided, encoder will use default film grain parameters. Then, SEIFGCPerPictureSEI parameter is provided to control the frequency of inserting FGC SEI messages to the bitstream. For example, FGC SEI message can be inserted for each frame, or once per I period. In the latter case, FGC SEI message is applied to following frames until new FGC SEI message arrives or bitstream ends. Note that, in such case, when FGC SEI is inserted once per I period, SEIFGCPersistenceFlag needs to be set to 1, otherwise it is 0 (as it is described in the semantics of the FGC SEI message).

SEIFGCModelID indicates parametric model in use. For frequency model, which is implemented within VTM and HM decoder, this parameter is set to 0. Also, SEIFGCBlendingModeID equals to 0 indicates additive blending mode, which is the only blending mode that is supported by the film grain synthesis implementation in the current VTM and HM decoders.

SEIFGCLog2ScaleFactor indicates the scale of the scaling factors that are used in the definition of the film grain scaling function definition. Acting on this parameter is a quick way of changing the film grain strength.

Thereafter, user should provide three flags used to indicate if the model parameters are present for the Luma and two Chroma components. SEIFGCCompModelPresentComp0, SEIFGCCompModelPresentComp1 and SEIFGCCompModelPresentComp2 indicate if film grain synthesis is applied for Y, Cb and Cr component, respectively.

For each component on which film grain should be applied, user needs to specify scaling function and cut-off frequencies for low-pass filtering for each intensity interval. It is done in the following way. First, SEIFGCNumIntensityIntervalMinus1Comp0, SEIFGCNumIntensityIntervalMinus1Comp1 and SEIFGCNumIntensityIntervalMinus1Comp2, indicating number of intensity intervals minus one. For example, scaling function on Figure 23b uses seven intensity intervals (only intervals with non-zero scaling factor are counted). For each interval, lower and upper bound needs to be defined. Note that overlapping intervals are not supported by VTM and HM decoders. To indicate interval bounds, user needs to specify SEIFGCIntensityIntervalLowerBoundComp0 and SEIFGCIntensityIntervalUpperBoundComp0. The same applies for Comp1 and Comp2. The SEIFGCIntensityIntervalLowerBoundComp0 parameter is in fact array of lower bounds for all intervals within the scaling function arranged from left to the right in ascending order. In the same manner, SEIFGCIntensityIntervalUpperBoundComp0 is an array of all upper bounds of the scaling function. For example, if there is more than one intensity interval, the given syntax should look like:

SEIFGCIntensityIntervalLowerBoundComp0: 10 50 90 120 180 225

SEIFGCIntensityIntervalUpperBoundComp0: 49 89 119 179 224 250

Then, for each component, the number of model parameters needs to be defined by setting SEIFGCNumModelValuesMinus1Comp0, SEIFGCNumModelValuesMinus1Comp1 and SEIFGCNumModelValuesMinus1Comp2 (the parameter value indicates the number of model parameters minus one). At least one and maximum three parameters are defined for each intensity interval of each component. Finally, model parameters are given by SEIFGCCompModelValuesComp0, SEIFGCCompModelValuesComp1 and SEIFGCCompModelValuesComp2. For example, three model values for each intensity interval for Comp0 (scaling factor, horizontal cut-off frequency, vertical cut-off frequency) can be defined as:

SEIFGCCompModelValuesComp0: 16 8 8 25 8 8 30 12 12 30 12 10 20 10 8 16 8 8 12 7 8

In the example, 7 groups of parameters are defined, e.g., one for each intensity interval that are defined in the example above.

As described in the FGC SEI specification, horizontal cut-off frequency, vertical cut-off frequency might be inferred depending on the number of model values. For example, two model parameters can be defined instead of three, in which case the first parameter is scaling factor, the second parameter is horizontal cut-off frequency, and the third parameter (vertical cut-off frequency) is the same as the horizontal cut-off frequency. In another case, only one parameter is defined which indicates the scaling factor. Cut-off frequencies in that case take the default value of 8.

The number of intensity intervals per component or the total number of pairs of cut-off frequencies shared between all three components may be limited in some synthesis implementations, which can require careful parameter settings.

At the end, a typical encoder command line to run VTM/HM with FGC SEI insertion can look like:

EncoderApp.exe –c cfg/encoder\_randomaccess\_vtm.cfg –c cfg/per\_sequence/input\_sequnce.cfg –c cfg/sei\_vui/film\_grain\_characterstics.cfg [...]

* + 1. Manipulation of FGC SEI message (bitstream post-processor)

The FGC SEI message rewriter included in the VTM allows FGC SEI message parameter values to be modified in existing bitstreams without re-encoding. It also supports inserting FGC SEI messages in a VVC bitstream originally encoded without any FGC SEI messages. The bitstream with the inserted FGC SEI message has an identical MD5sum value to the bitstream having the FGC SEI message enabled during encoding. The tool supports three operational modes, controlled by SEIFilmGrainOption: 0-no change; 1-FGC SEI removal; 2-FGC SEI insertion; 3-FGC SEI rewriter.

1. FGC SEI removal (applied on bitstreams that include FGC SEI messages)
2. FGC SEI insertion (applied on bitstreams without FGC SEI message)
3. FGC SEI modification (applied on bitstreams with FGC SEI message)

The two typical use cases and examples are given below:

1. A bitstream (test\_fg1.bit) is encoded with a pre-configured FGC SEI message (fg1.cfg), but the film grain parameters are not producing satisfactory film grain effects. One can use the tool to modify the values (fg2.cfg) of the existing FGC SEI message. Example command line:   
   ./SEIFilmGrainAppStatic -c fg2.cfg -b test\_fg1.bit -o test\_fg2.bit --SEIFilmGrainOption=3
2. A bitstream (test.bit) is encoded without FGC SEI message, so it cannot take advantage of the film grain synthesis. One can use the tool to easily insert the FGC SEI (fg.cfg) into the bitstream. Example command line:

./SEIFilmGrainAppStatic -c fg.cfg -b test.bit -o test\_fg.bit --SEIFilmGrainOption=2

The syntax of configuration files used to define a new FGC SEI is the same as described in B.1.1.

The tool manipulates the VVC bitstream at NALU level, so processing time is extremely fast (e.g., to rewrite the FGC SEI message in a 4K/8K resolution bitstream, the entire processing time is less than 0.1 sec). The basic workflow is illustrated in Figure 29.

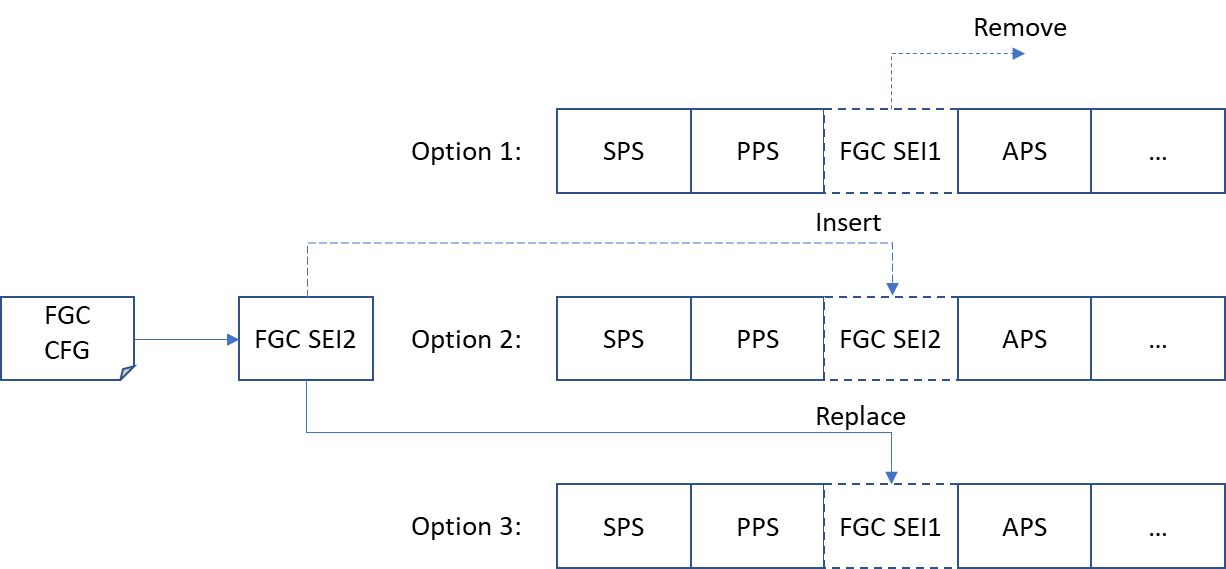


Figure 29. Basic workflow of the FGC SEI message rewriter

* 1. Film grain synthesis example implementations
     1. FGC SEI message implementation – frequency filtering model

One implementation for frequency model is in VTM [18] and HM [19] reference software.

To enable film grain synthesis on the decoder side, appropriate FGC SEI messages need to be embedded within the bitstream.

The synthesis implementation is based on RDD-5, with the following restrictions on supported parameter values: SEIFGCModelID and SEIFGCBlendingModeID should be equal to zero, meaning frequency model and additive blending mode. The number of intensity intervals per component is not limited, however the total number of pairs of cut-off frequencies shared between all three components must be less than or equal to 10.

Decoder can be run with:

DecoderApp.exe -b str.bin -o recon.yuv --SEIFGSFilename=fg\_recon.yuv

* + 1. Versatile film grain synthesis software

A standalone film grain synthesis software compatible with FGC SEI in both frequency filtering and auto-regressive mode is available at https://github.com/InterDigitalInc/VersatileFilmGrain. This is a command-line utility that takes as input a YUV file, a configuration file, and outputs a YUV file. Configuration syntax is either the same as described in B.1.1, or is an SEI dump produced by the SEI tool described in B.1.2.

This software uses the template-based approach described in 7.2, with 64x64 templates and 16x16 randomisation blocks (for luma, and half those sizes for chroma). Pixel-wise local adaptation of both grain size and amplitude is supported. Macros define the complexity limits (e.g. number of templates) to enable experimenting with them. When FGC SEI message contents goes beyond supported limits, graceful degradation is supported by e.g. generating only as many templates as supported, in the order they are defined in the SEI message, and re-using the closest ones if more are defined.

The software is organized in a low-level layer, which could be implemented in hardware, and higher layers dealing with parameter decoding and interpretation and file I/O. The focus of this software is to be as complete as possible in the support of FGC SEI message (e.g. local adaptation of grain size), while keeping the cost of a potential hardware implementation as low as possible (e.g. no line buffers).

* 1. Film grain analysis example implementations
     1. Denoising

The following options in VTM [18] and HM [19] provide a means to use any denoising algorithm and any mask (for flat blocks detection) calculation algorithm.

SEIFGCExternalDenoised: path/denoised\_sequence\_name.yuv

SEIFGCExternalMask: path/mask\_sequence\_name.yuv

If those parameters are provided, VTM/HM will use external sources for film grain analysis part, e.g., denoised sequence is obtained a priori. If the parameter is not provided (empty string by default) VTM/HM will use MCTF and Canny edge detector in conjunction with morphological operations (MCTF and Canny for film grain analysis are already part of VTM and HM).

For example, one possibility to find denoised equivalent of the input sequence is to use denoisers implemented in ffmpeg [20]. One such denoiser, which implementation can be found in ffmpeg, is called High Quality Denoiser 3D (hqdn3d). The command line to obtain denoised video is (for raw video, full hd resolution, 10-bit):

ffmpeg.exe -vcodec rawvideo -video\_size 1920x1080 -pix\_fmt yuv420p10le -i input.yuv -vf hqdn3d=7:7:7:7 -vcodec rawvideo -an -f rawvideo denoised.yuv

where parameters of the denoiser (7:7:7:7) represent spatial and temporal filter strengths for luma and chroma components (note strengths are selected arbitrarily for illustration purpose and can be adapted in addition from sequence to sequence).

Another denoiser implementation in ffmpeg [20] is image and video denoising by sparse 3D transform-domain collaborative filtering (Block-matching and 3D filtering BM3D) [21][22].

According to the ffmpeg documentation, to denoise frames using BM3D, filter accepts the following options wich are summarized in Table 4.

Table 4: FFmpeg's BM3D filter parameters

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Range** | **Description** |
| **sigma** | 0-999.9 | The strength of denoising. Default value is 1.  The denoising algorithm is very sensitive to sigma, so adjust it according to the source. Technically, sigma represents the standard deviation of i.i.d. zero mean additive white Gaussian noise. |
| **block** |  | Local patch size. This sets dimensions in 2D. A block is the basic processing unit of BM3D. Generally, larger block - denoising will be slower. |
| **Bstep** | 1-64 | Sliding step for processing blocks. Default value is 4. Smaller values allow processing more reference blocks and is slower |
| **Group** | 1-256 | Maximal number of similar blocks for 3rd dimension. Default value is 1. When set to 1, no block matching is done. Larger values allow more blocks in single group. If larger value is used the sparsity in a transformed group raises, leading to the stronger filtering, and also slower in the DCT/IDCT step. |
| **Range** | 1-231-1 | Radius for search block matching. Default is 9. Larger is slower, with higher probability to find similar patches. |
| **Mstep** | 1-64 | Set step between two search locations for block matching. Default is 1. Smaller is slower, with 1 being is equivalent to full-search block-matching. |
| **Thmse** | 0-231-1 | Threshold of mean square error for block matching. Larger the value, more blocks will be matched. Losses to fine structures and details can be observed if the value is too large. Should be increased if the noise is strong. The default value is automatically adjusted according to sigma. |
| **Hdthr** |  | Set thresholding parameter for hard thresholding in 3D transformed domain. Larger values result in stronger hard-thresholding filtering in frequency domain. However it is advise first to adjust sigma rather than Hdthr to obtain desired result. |
| **Estim** | basic or final | Set filtering estimation mode. Default is basic. |
| **Ref** |  | If enabled, filter will use 2nd stream for block matching. Default is disabled for basic value of estim option, and always enabled if value of estim is final. |
| **Planes** |  | Set planes to filter. Default is all available except alpha. |

Some of the possible filter options are given in the examples below.If any of options is not present in the command line, it will use default value.

ffmpeg -vcodec rawvideo -video\_size 1920x1080 -pix\_fmt yuv420p10le -i fg\_recon.yuv -filter\_complex bm3d=sigma=a:block=b:bstep=c:group=d:estim=basic -vcodec rawvideo -an -f rawvideo denoised\_bm3d.yuv

Where generic coefficients a,b,c,d,e represent the parameters of the filter and should be replaced with the numbers during real denoising process. There is possibility to filter only luma component by adding option planes=1.

In another scenario, SEIFGCExternalDenoised can be initialized with the reconstructed (noiseless) frames if quantization parameter is not very high to introduce very strong compression artifacts.

One possible encoder command line can look like:

EncoderApp.exe –c cfg/encoder\_randomaccess\_vtm.cfg –c cfg/per\_sequence/input\_sequnce.cfg –c cfg/sei\_vui/film\_grain\_characterstics\_analysis.cfg --SEIFGCExternalDenoised=denoised.yuv --SEIFGCExternalMask=mask.yuv

And decoder can be run with:

DecoderApp.exe -b str.bin -o recon.yuv --SEIFGSFilename=fg\_recon.yuv

* + 1. FGC SEI message implementation – frequency filtering model

Instead of manually tuned configuration parameters, a method to automatically estimate intensity intervals and associated scaling factors used in the film grain synthesis process can be applied. One possible implementation is provided within VTM [18] and HM [19] reference software. It implements analysis of the noise if present in the source video. Otherwise, if noise is not present, codec will set appropriate flags to zero to disable noise synthesis on the decoder side.

To enable film grain analysis module at the encoder side, SEIFGCAnalysisEnabled parameter in configuration file should be set to 1. A subset of FGC SEI parameters is sufficient to be provided. Hence, some configuration parameters need to be provided to the encoder based on the user preferences. However, there is no need to specify all FGC SEI configuration parameters since many of them are re-calculated (re-set) in the film grain analysis process. One example of configuration file when film grain analysis is used within VTM or HM solution is provided (see cfg/sei\_vui/*film\_grain\_characteristics\_test\_analysis.cfg* file). Note that SEIFGCCompModelPresentComp0, SEIFGCCompModelPresentComp1 and SEIFGCCompModelPresentComp2 parameters are still present. Reason behind is that manual control to disable the film grain synthesis on individual components is still highly desirable. For example, even if film grain may be detected in chroma components during analysis process, SEIFGCCompModelPresentComp flag from configuration file will have higher importance if it is set to 0. It means that even if analysis module detects film grain in chroma, it is not coded in SEI, and SEIFGCCompModelPresentComp flag remains 0. On the contrary, if SEIFGCCompModelPresentComp is 1 within the configuration file, the software uses analysis results on film grain. It means that if analysis part did not find film grain, for example in chroma component, it will reset the SEIFGCCompModelPresentComp flag to 0 even if it was 1 in configuration file. Rationale behind can be found in a fact that the purpose of the analysis part is to estimate film grain as close as possible to the original one. In this case SEIFGCCompModelPresentComp flags can be interpreted as a flag that indicate if analysis is performed on particular component or not. This approach is only implementation oriented, and do not require any changes in SEI message format.

* + 1. AFGS1 parameters estimation

Estimation of AFGS1 film grain parameters can be done with the help of the AV1 reference software package, libaom [27]. The software package contains a noise\_model executable. The executable can be used to estimate AFGS1 film grain model parameters as follows (please see an example command line below):

noise\_model --fps=25/1 --width=854 --height=480 --i420 --input-denoised=denoised.854\_480.yuv --input=original.854\_480.yuv --output-grain-table=film\_grain.tbl

Parameter --input-denoised represents a denoised version of the source video sequence, --input represents the original source video containing film grain, and --output-grain-table specified a text file name that contains a table with AFGS1 film grain model parameters. The film grain parameters table file can also be input in an AV1 bitstream by using the --film-grain-table=film\_grain.tbl parameter with an aomenc executable, while using the denoised source as the input video sequence.