INTERNATIONAL STANDARD ISO/IEC 23003-2:2010
TECHNICAL CORRIGENDUM 1

Published 2012-09-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — MPEG audio technologies —

Part 2:
Spatial Audio Object Coding (SAOC)

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Technologies audio MPEG —

Partie 2: Codage d'objet audio spatial (SAOC)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 23003-2:2010 was prepared by *Joint* Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

In Clause 2 "Normative references", add:

ISO/IEC 23000-12, *Information technology – Multimedia application format (MPEG-A) – Part 12: Interactive music application format*

In all tables, replace:

"reserved"

with:

"N/A"

ICS 35.040 **Ref. No. ISO/IEC 23003-2:2010/Cor.1:2012(E)**

In 5.1 Introduction, replace:

The number of objects that can be handled is in priniciple not limited.

with:

The number of objects that can be handled is in principle not limited.

In 5.5.2 Baseline Profile, replace:

Note that ISO/IEC 23000-12 (Information technology — Multimedia application format (MPEG-A) — Part 12: Interactive music spplication format) defines several brands that refer to the SAOC Baseline Profile.

with:

Note that ISO/IEC 23000-12 defines several brands that refer to the SAOC Baseline Profile.

In 6.1 Payloads for SAOC, replace:

Table 8 – Syntax of ResidualConfig()

Syntax	No. of bits	Mnemonic
ResidualConfig()		
{		
bsResidualSamplingFrequencyIndex;	**4**	**uimsbf**
bsResidualFramesPerSAOCFrame;	**2**	**uimsbf**
bsNumGroupsFGO;	**2**	**uimsbf**
for (i=0; i<bsNumGroupsFGO + 1; i++) {		
bsResidualPresent[i];	**1**	**uimsbf**
if (bsResidualPresent[i]) {		

with:

Table 8 – Syntax of ResidualConfig()

Syntax	No. of bits	Mnemonic
ResidualConfig()		
{		
bsResidualSamplingFrequencyIndex;	**4**	**uimsbf**
bsResidualFramesPerSAOCFrame;	**2**	**uimsbf**
bsNumEAO;	**2**	**uimsbf**
for (i=0; i<bsNumEAO + 1; i++) {		
bsResidualPresent[i];	**1**	**uimsbf**
if (bsResidualPresent[i]) {		

In 6.1 Payloads for SAOC, replace:

Table 15 – Syntax of PresetConfig()

Syntax	No. of bits	Mnemonic
PresetConfig()		
{		
bsNumPresets;	**4**	**uimsbf**
for (i=0; i<bsNumPresets+1; i++) {		
bsNumBytePresetLabel[i];	**8**	**uimsbf**
for (j=0; j<bsNumBytePresetLabel[i]; j++) {		
bsPresetLabel[i][j];	**8**	**bslbf**
}		
bsPresetMatrix;	**1**	**uimsbf**
if (bsPresetMatrix) {		
PresetMatrixData();		
} else {		

with:

Table 15 – Syntax of PresetConfig()

Syntax	No. of bits	Mnemonic
PresetConfig()		
{		
bsNumPresets;	**4**	**uimsbf**
for (i=0; i<bsNumPresets+1; i++) {		
bsNumBytePresetLabel[i];	**8**	**uimsbf**
for (j=0; j<bsNumBytePresetLabel[i]; j++) {		
bsPresetLabel[i][j];	**8**	**bslbf**
}		
bsPresetMatrix[i];	**1**	**uimsbf**
if (bsPresetMatrix[i]) {		
PresetMatrixData();		
} else {		

In 6.1 Payloads for SAOC, replace:

Table 21 – Syntax of SAOCFramingInfo()

Syntax	No. of bits	Mnemonic
SAOCFramingInfo()		
{		
bsFramingType;	**1**	**uimsbf**
If (bsLowDelayMode == 0) {		
bsNumParamSets;	**3**	**uimsbf**
} else {		
bsNumParamSets;	**1**	**uimsbf**
}		
if (bsFramingType) {		
for (ps=0; ps<numParamSets; ps++) {		Note 1
bsParamSlot[ps];	**nBitsParamSlot**	**uimsbf** Note 2
}		
}		
}		
Note 1: numParamSets is defined by numParamSets = bsNumParamSets + 1.		
Note 2: nBitsParamSlot is defined according to nBitsParamSlot = ceil(log2(numSlots)).		

with:

Table 21 – Syntax of SAOCFramingInfo()

Syntax	No. of bits	Mnemonic
SAOCFramingInfo()		
{		
bsFramingType;	**1**	**uimsbf**
If (bsLowDelayMode == 0) {		
bsNumParamSets;	**3**	**uimsbf**
} else {		
bsNumParamSets;	**1**	**uimsbf**
}		
for (ps=0; ps<numParamSets; ps++) {		
if (bsFramingType) {		Note 1
bsParamSlot[ps];	**nBitsParamSlot**	**uimsbf** Note 2
} else {		
bsParamSlot[ps] =ceil(numSlots*(ps+1)/numParamSets)-1;		Note 1, 3
}		
}		
}		
Note 1: numParamSets is defined by numParamSets = bsNumParamSets + 1. Note 2: nBitsParamSlot is defined according to nBitsParamSlot = ceil(log2(numSlots)). Note 3: numSlots is defined by numSlots = bsFrameLength + 1.		

In 6.1 Payloads for SAOC, replace:

bsDcuParam Defines the parameter value for the DCU algorithm according to Table 41.

with:

bsDcuParam Defines the parameter value for the DCU algorithm according to Table 39.

In 6.1 Payloads for SAOC; replace:

Table 42 — numQuantSteps

XXX (dataType)	numQuantStepsXXXCoarse	numQuantStepsXXXFine
DCLD, DMG, PDG	15	31
IOC	4	8
OLD	8	16
NRG	32	64

with:

Table 42 – numQuantSteps

XXX (dataType)	numQuantStepsXXXCoarse	numQuantStepsXXXFine
DCLD, DMG, PDG	15	31
IOC	4	8
OLD	8	16
NRG	32	64

In 6.1 Payloads for SAOC, replace:

PresetUserDataContainer()

Syntactic element that contains preset rendering data in the user-defined preset representation format and has a length of exactly **bsPresetUserDataLen** bytes.

with:

PresetUserDataContainer()

Syntactic element that contains preset rendering data in the user-defined preset representation format and has a length of exactly **bsPresetUserDataLen** bytes.

All bitstream variables which are not explicitly described here are defined in ISO/IEC 23003-1:2007.

In 6.1 Payloads for SAOC, add:

bsResidualFramesPerSAOCFrame

Indicates the number of residual frames per SAOC frame, ranging from one to four according to Table 56 defined in ISO/IEC 23003-1:2007.

In 6.1 Payloads for SAOC, add:

SAOCDiffHuffData() Syntactic element that contains one or two temporally subsequent parameter subsets of a given parameter in the SAOC frame, where the quantized values are coded using a combination of differential coding and Huffman coding.

In Clause 7 SAOC processing, omit the time/band indices for all signals and parameters.

In 7.1.2 Dequantization of the SAOC parameters, replace:

Table 47 – OLD parameter quantization table

idx	0	1	2	3	4	5	6	7
OLD[idx]	$10^{-15.00}$	$10^{-4.50}$	$10^{-4.00}$	$10^{-3.50}$	$10^{-3.00}$	$10^{-2.50}$	$10^{-2.20}$	$10^{-1.90}$
idx	8	9	10	11	12	13	14	15
OLD[idx]	$10^{-1.60}$	$10^{-1.30}$	$10^{-1.00}$	$10^{-0.80}$	$10^{-0.60}$	$10^{-0.40}$	$10^{-0.20}$	1

with:

Table 47 – OLD parameter quantization table

idx	0	1	2	3	4	5	6	7
OLD[idx]	$10^{-15.0}$	$10^{-4.5}$	$10^{-4.0}$	$10^{-3.5}$	$10^{-3.0}$	$10^{-2.5}$	$10^{-2.2}$	$10^{-1.9}$
idx	8	9	10	11	12	13	14	15
OLD[idx]	$10^{-1.6}$	$10^{-1.3}$	$10^{-1.0}$	$10^{-0.8}$	$10^{-0.6}$	$10^{-0.4}$	$10^{-0.2}$	1

In 7.1.2 Dequantization of the SAOC parameters, replace:

```
while (ps=0; ps<numParamSet; ps++) {
    switch (bsXXXdataMode[pi][ps]) {
    case 0: /* default */
          for (pb=0; pb<numBands, pb++) {
                 switch (XXX) {
                 case OLD, NRG, IOC, DCLD, DMG, PDG:
                        idxXXX[pi][ps][pb] = 0;
                        break;
                 }
          }
          break;
```

with:

```
while (ps=0; ps<numParamSet; ps++) {
    switch (bsXXXdataMode[pi][ps]) {
    case 0: /* default */
          for (pb=0; pb<numBands, pb++) {
                 switch (XXX) {
                 case NRG, DCLD, DMG, PDG:
                        idxXXX[pi][ps][pb] = 0;
                        break;
                 }
                 case OLD:
                        idxXXX[pi][ps][pb] = 15;
                        break;
                 }
                 case IOC:
                        idxXXX[pi][ps][pb] = 5;
                        break;
                 }
          }
          break;
```


In 7.2.3 Unquantized interface for the MPS parameters, replace:

For an efficient practical implementation and to prevent a loss in precision, the parameter interface to the MPS decoder may alternatively be established in a direct, unquantized way. Rather than writing an actual MPS bitstream, the relevant parameters may be passed directly to the MPS decoder.

with:

For an efficient practical implementation and to prevent a loss in precision, the parameter interface to the MPS decoder may alternatively be established in a direct, unquantized way. The required range of all relevant parameters is determined by the minimal and maximal values of the corresponding dequantization scheme. Rather than writing an actual MPS bitstream, the relevant parameters may be passed directly using binary32 (single) floating point format (IEEE 754-2008) to the MPS decoder.

In 7.4 Post(processing) downmix compensation, replace and move the corresponding text to "7.5 Signals and parameters":

If the post(processed) downmix $\mathbf{X}^{n,k}_{\text{post(processed)}}$ is used, the following modification should be taken prior to SAOC decoding/transcoding:

$$\mathbf{X}^{n,k} = \mathbf{W}^{n,k}_{\text{PDG}} \mathbf{X}^{n,k}_{\text{post(processed)}},$$

where $\mathbf{X}^{n,k}$ represents the input signal to the SAOC decoder/transcoder.

The matrix $\mathbf{W}_{\text{PDG}}^{n,k}$ is defined for every time-slot n and every hybrid subband k. Its elements are obtained from the transmitted PDG parameters which are defined for a given parameter time-slot l and a given processing band m. The mapping to the hybrid domain is done according to Table A.31, ISO/IEC 23003-1:2007. If post(processed) downmix compensation is applied (bsPdgFlag = 1), the matrix $\mathbf{W}_{\text{PDG}}^{l,m}$ is defined as:

$$\mathbf{W}_{\text{PDG}}^{l,m} = \left(PDG_0^{l,m} \right), \qquad \text{for mono downmix,}$$

$$\mathbf{W}_{\text{PDG}}^{l,m} = \begin{pmatrix} PDG_0^{l,m} & 0 \\ 0 & PDG_1^{l,m} \end{pmatrix}, \qquad \text{for stereo downmix,}$$

where $PDG_j^{l,m} = \mathbf{D}_{\text{PDG}}\left(j,l,m\right)$.

with:

If the post(processed) downmix $\mathbf{X}_{\text{post(processed)}}$ compensation is applied (bsPdgFlag = 1), the following modification should be taken prior to the SAOC decoding/transcoding

$$\mathbf{X} = \mathbf{W}_{\text{PDG}} \mathbf{X}_{\text{post(processed)}} .$$

The matrix $\mathbf{W}_{\text{PDG}}$ is obtained from the transmitted PDG parameters as

$$\mathbf{W}_{\text{PDG}} = \begin{pmatrix} PDG_0 & 0 \\ 0 & PDG_1 \end{pmatrix}, \qquad \text{for stereo downmix,}$$

$$\mathbf{W}_{\text{PDG}} = \begin{pmatrix} PDG_0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \text{for mono downmix.}$$

Here, the dequantized post(processed) downmix gains are obtained according to 7.1.2 as

$$PDG_j = \mathbf{D}_{\text{PDG}}\left(j,l,m\right).$$

In 7.5.2 Signals and parameters, replace:

$$\mathbf{X} = \mathbf{x}^{n,k} = \begin{pmatrix} l_0 \\ r_0 \end{pmatrix}, \qquad \text{for stereo downmix,}$$

$$\mathbf{X} = \mathbf{x}^{n,k} = \begin{pmatrix} d_0 \\ 0 \end{pmatrix}, \qquad \text{for monoo downmix.}$$

with:

$$\mathbf{X} = \begin{pmatrix} l_0 \\ r_0 \end{pmatrix}, \qquad \text{for stereo downmix,}$$

$$\mathbf{X} = \begin{pmatrix} d_0 \\ 0 \end{pmatrix}, \qquad \text{for mono downmix.}$$

In 7.5 Signals and parameters, add:

Output covariance

The output covariance matrix $\mathbf{F}$ with elements $f_{i,j}$ is given as

$$\mathbf{F} = \mathbf{AEA}^*, \qquad \text{for binaural rendering,}$$
$$\mathbf{F} = \mathbf{M}_{\text{ren}}\mathbf{EM}_{\text{ren}}^*, \qquad \text{otherwise.}$$

In 7.5 Signals and parameters, add:

Input covariance

The input covariance v is given as

$$v = \mathbf{DED}^* + \varepsilon^2 .$$

In 7.6 SAOC transcoding/decoding modes, use the following structure:

7.6 SAOC transcoding/decoding modes
- 7.6.1 Overview
- 7.6.2 Decorrelated signal
- 7.6.3 Transcoding modes
  - 7.6.3.1 Introduction
  - 7.6.3.2 Mono downmix (“x-1-5”) processing mode
    - 7.6.3.2.1 Introduction
    - 7.6.3.2.2 SAOC downmix preprocessor unit
    - 7.6.3.2.3 SAOC parameter processing unit
  - 7.6.3.3 Stereo downmix (“x-2-5”) processing mode
    - 7.6.3.3.1 Introduction
    - 7.6.3.3.2 SAOC downmix preprocessor unit
    - 7.6.3.3.3 SAOC parameter processing unit
- 7.6.4 Decoding modes
  - 7.6.4.1 Introduction
  - 7.6.4.2 Mono to binaural "x-1-b" processing mode
  - 7.6.4.3 Mono to stereo "x-1-2" processing mode
  - 7.6.4.4 Mono to mono "x-1-1" processing mode
  - 7.6.4.5 Stereo to binaural "x-2-b" processing mode
  - 7.6.4.6 Stereo to stereo "x-2-2" processing mode
  - 7.6.4.7 Stereo to mono "x-2-1" processing mode

In 7.6.2 Mono downmix (“x-1-5”) processing mode, replace:

Estimation of power and cross power terms

Incorporating index h denoting the OTT element, the power and cross power terms can be estimated by:

$$p_{h,0}^2 = \sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1} w_{0,i}^h w_{0,j}^h e_{i,j}\right), \qquad p_{h,1}^2 = \sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1} w_{1,i}^h w_{1,j}^h \; e_{i,j}\right), \qquad R_h = \sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1} w_{0,i}^h w_{1,j}^h \; e_{i,j}\right).$$

Derivation of the MPS parameters

Finally, the corresponding CLD and ICC parameters are derived as:

$$CLD_h^{l,m} = 10\log_{10}\left(\max\left(\frac{p_{h,0}^2}{p_{h,1}^2}, \varepsilon^2\right)\right),$$

$$ICC_h^{l,m} = \frac{\max\left(R_h, \varepsilon^2\right)}{\sqrt{\max\left(p_{h,0}, \varepsilon^2\right)\max\left(p_{h,1}, \varepsilon^2\right)}},$$

with:

The index h refers to the OTT_h element

$$CLD_h = 10\log_{10}\left(\frac{r_{0,0}^h}{r_{1,1}^h}\right), \qquad ICC_h = \frac{r_{0,1}^h}{\sqrt{r_{0,0}^h r_{1,1}^h}}.$$

The terms $r_{i,j}^h$ can be estimated as

$$r_{i,j}^h = \max\left(\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} w_{i,n}^h w_{j,m}^h \; e_{n,m}, \varepsilon^2\right).$$

In 7.6.2 Mono downmix ("x-1-5") processing mode, replace:

$$ADG^{l,m} = 10\log_{10}\left(\max\left(\frac{f^{l,m}}{v^{l,m}}, \varepsilon^2\right)\right).$$

The scalar $f^{l,m}$ is computed as

$$f^{l,m} = \sum_{i=0}^{N-1} f_{i,i}^{l,m}.$$

with:

$$ADG = 10\log_{10}\left(\max\left(\frac{trace\left(\mathbf{F}\right)}{v}, \varepsilon^2\right)\right).$$

In 7.6.2 Mono downmix ("x-1-5") processing mode, replace:

The following subclauses give a description of the SAOC transcoding mode for the mono downmix case. The object parameters (OLD, IOC, DMG, DCLD) from the SAOC bitstream are transcoded into spatial parameters (CLD, ICC, CPC, ADG) for the MPS bitstream according to the rendering information. The downmix is not modified.

with:

The following subclauses describe the processing steps dedicated to the transformation of SAOC parameters (OLD, IOC, DMG) into MPS data (CLD, ICC, ADG) according to the rendering information for the mono downmix case, see Figure 13 (left). The downmix signal is not modified.

In 7.6.2 Mono downmix ("x-1-5") processing mode, replace:

The respective contribution of each object to the two outputs of OTT element 0 is obtained by summation of the corresponding elements in $\mathbf{M}_{\text{ren}}^{l,m}$. This summation gives a sub-rendering matrix $\mathbf{W}_0^{l,m}$ of OTT element 0:

$$\mathbf{W}_0^{l,m} = \begin{pmatrix} w_{0,0}^0 & \cdots & w_{0,N-1}^0 \\ w_{1,0}^0 & \cdots & w_{1,N-1}^0 \end{pmatrix} = $$

$$= \begin{pmatrix} m_{0,Lf}^{l,m} + m_{0,Rf}^{l,m} + m_{0,C}^{l,m} + m_{0,Lfe}^{l,m} & \cdots & m_{N-1,Lf}^{l,m} + m_{N-1,Rf}^{l,m} + m_{N-1,C}^{l,m} + m_{N-1,Lfe}^{l,m} \\ m_{0,Ls}^{l,m} + m_{0,Rs}^{l,m} & \cdots & m_{N-1,Ls}^{l,m} + m_{N-1,Rs}^{l,m} \end{pmatrix}.$$

The CLDs and ICCs of the subsequent OTT boxes ($CLD_h^{l,m}$, $ICC_h^{l,m}$, $h = 0,\ldots,4$) are calculated using the sub-rendering matrices defined as:

$$\mathbf{W}_1^{l,m} = \begin{pmatrix} w_{0,0}^1 & \cdots & w_{0,N-1}^1 \\ w_{1,0}^1 & \cdots & w_{1,N-1}^1 \end{pmatrix} = \begin{pmatrix} m_{0,Lf}^{l,m} + m_{0,Rf}^{l,m} & \cdots & m_{N-1,Lf}^{l,m} + m_{N-1,Rf}^{l,m} \\ m_{0,C}^{l,m} + m_{0,Lfe}^{l,m} & \cdots & m_{N-1,C}^{l,m} + m_{N-1,Lfe}^{l,m} \end{pmatrix},$$

with:

The respective contribution of each object to the two outputs of OTT$_h$ element is obtained by summation of the corresponding elements in the rendering matrix $\mathbf{M}_{\text{ren}}$. The subsequent sub-rendering matrices $\mathbf{W}_h$ with elements $w_{i,j}^h$ are defined as

$$\mathbf{W}_0^{l,m} = \begin{pmatrix} w_{0,0}^0 & \cdots & w_{0,N-1}^0 \\ w_{1,0}^0 & \cdots & w_{1,N-1}^0 \end{pmatrix} = $$

$$= \begin{pmatrix} \sqrt{m_{0,Lf}^{l,m\,2} + m_{0,Rf}^{l,m\,2} + m_{0,C}^{l,m\,2} + m_{0,Lfe}^{l,m\,2}} & \cdots & \sqrt{m_{N-1,Lf}^{l,m\,2} + m_{N-1,Rf}^{l,m\,2} + m_{N-1,C}^{l,m\,2} + m_{N-1,Lfe}^{l,m\,2}} \\ \sqrt{m_{0,Ls}^{l,m\,2} + m_{0,Rs}^{l,m\,2}} & \cdots & \sqrt{m_{N-1,Ls}^{l,m\,2} + m_{N-1,Rs}^{l,m\,2}} \end{pmatrix}.$$

$$\mathbf{W}_1^{l,m} = \begin{pmatrix} w_{0,0}^1 & \cdots & w_{0,N-1}^1 \\ w_{1,0}^1 & \cdots & w_{1,N-1}^1 \end{pmatrix} = \begin{pmatrix} \sqrt{m_{0,Lf}^{l,m\,2} + m_{0,Rf}^{l,m\,2}} & \cdots & \sqrt{m_{N-1,Lf}^{l,m\,2} + m_{N-1,Rf}^{l,m\,2}} \\ \sqrt{m_{0,C}^{l,m\,2} + m_{0,Lfe}^{l,m\,2}} & \cdots & \sqrt{m_{N-1,C}^{l,m\,2} + m_{N-1,Lfe}^{l,m\,2}} \end{pmatrix},$$

In 7.6.2.2 Sub-rendering matrices for each OTT element, remove:

Additional information is provided by the rendering matrix $\mathbf{M}_{\text{ren}}^{l,m}$ with elements $m_{i,j}^{l,m}$, yielding the mapping of all audio input channels i to the desired output channels j. The rendering matrix $\mathbf{M}_{\text{ren}}^{l,m}$ for the 5.1 output configuration is given by:

$$\mathbf{M}_{\text{ren}}^{l,m} = \begin{pmatrix} m_{0,Lf}^{l,m} & \cdots & m_{N-1,Lf}^{l,m} \\ m_{0,Rf}^{l,m} & \cdots & m_{N-1,Rf}^{l,m} \\ m_{0,C}^{l,m} & \cdots & m_{N-1,C}^{l,m} \\ m_{0,Lfe}^{l,m} & \cdots & m_{N-1,Lfe}^{l,m} \\ m_{0,Ls}^{l,m} & \cdots & m_{N-1,Ls}^{l,m} \\ m_{0,Rs}^{l,m} & \cdots & m_{N-1,Rs}^{l,m} \end{pmatrix}.$$

In 7.6.2.4 Derivation of the MPS parameters, remove:

Matrix $\mathbf{F}^{l,m}$ of size $N_{\text{MPS}} \times N_{\text{MPS}}$ with elements $f_{i,j}^{l,m}$ is given as

$$\mathbf{F}^{l,m} = \mathbf{A}^{l,m} \mathbf{E}^{l,m} \left(\mathbf{A}^{l,m}\right)^*.$$

In 7.6.2.4 Derivation of the MPS parameters, remove:

The scalar $v^{l,m}$ is computed as

$$v^{l,m} = \mathbf{D}^l \mathbf{E}^{l,m} \left(\mathbf{D}^l\right)^* + \varepsilon^2 .$$

In 7.6.3 Stereo downmix ("x-2-5") processing mode, replace:

$$\begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - 3\mathbf{y} \text{ with } \mathbf{y} = \frac{b_{i,3}}{\left(\sum_{j=1,2} \left(\gamma_{i,j}\right)^2\right) + \varepsilon} \boldsymbol{\gamma}^T .$$

with:

$$\begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - 3\mathbf{y} \text{ with } \mathbf{y} = \frac{b_{i,3}}{\max\left(\sum_{j=1,2} \left(\gamma_{i,j}\right)^2, \varepsilon^2\right)} \boldsymbol{\gamma}^T .$$

In 7.6.3 Stereo downmix ("x-2-5") processing mode, replace:

$$\mathbf{P}_2 = \begin{cases} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, & r_{1,2} > 0, \\ \mathbf{v}_\mathrm{R} \cdot diag\left(\mathbf{w}_\mathrm{d}\right), & otherwise. \end{cases}$$

with:

$$\mathbf{P}_2 = \begin{cases} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, & r_{1,2} > \varepsilon^2, \\ \mathbf{v}_\mathrm{R} \cdot diag\left(\mathbf{w}_\mathrm{d}\right), & otherwise. \end{cases}$$

In 7.6.3 Stereo downmix ("x-2-5") processing mode, replace:

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 1 \end{pmatrix} \hat{\mathbf{G}}$$

with:

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \hat{\mathbf{G}} .$$

In 7.6.3 Stereo downmix ("x-2-5") processing mode, replace:

$$\mathbf{G} = \begin{cases} diag\left(\mathbf{g}_\mathrm{vec}\right) \cdot \hat{\mathbf{G}}, & r_{1,2} > 0, \\ \hat{\mathbf{G}}, & otherwise. \end{cases}$$

with:

$$\mathbf{G} = \begin{cases} diag\left(\mathbf{g}_\mathrm{vec}\right) \cdot \hat{\mathbf{G}}, & r_{1,2} > \varepsilon^2, \\ \hat{\mathbf{G}}, & otherwise. \end{cases}$$