
**Information technology — Multimedia
content description interface —**

**Part 3:
Visual**

*Technologies de l'information — Interface de description du contenu
multimédia —*

Partie 3: Visuel

Reference number
ISO/IEC 15938-3:2002(E)



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2002

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Foreword.....	v
Introduction	vi
1 Scope	1
1.1 Organization of the document.....	1
1.2 Overview of Visual Description Tools	1
2 Terms and Definitions	2
2.1 Default reference axis	2
2.2 DCT coefficients	2
2.3 Data element	3
3 Abbreviations and Symbols	3
3.1 General.....	3
3.2 Abbreviations.....	3
3.3 Arithmetic operators	3
3.4 Logical operators.....	3
3.5 Relational operators	3
3.6 Bitwise operators.....	4
3.7 Conditional operator	4
3.8 Assignment	4
3.9 Mnemonics	4
3.10 Constants	4
3.11 Functions.....	4
4 Conventions	5
4.1 Method of describing the DDL representation syntax.....	5
4.2 Method of describing the binary representation syntax	5
4.3 Method of describing the descriptor semantics	8
5 Basic structures.....	8
5.1 Introduction	8
5.2 Grid layout	8
5.3 Time series	11
5.4 Multiple view	15
5.5 Spatial 2D coordinates	16
5.6 Temporal interpolation.....	23
6 Color.....	29
6.1 Introduction	29
6.2 Color space	29
6.3 Color quantization	33
6.4 Dominant color	35
6.5 Scalable color	37
6.6 Color layout.....	42
6.7 Color structure	50
6.8 GoF/GoP Color.....	56
7 Texture	57
7.1 Introduction	57
7.2 Homogeneous texture.....	57
7.3 Texture browsing.....	61
7.4 Edge histogram.....	63
8 Shape	66
8.1 Introduction	66

8.2	Region shape	66
8.3	Contour shape.....	68
8.4	Shape 3D.....	71
9	Motion	73
9.1	Introduction	73
9.2	Camera motion.....	73
9.3	Motion trajectory.....	81
9.4	Parametric motion	84
9.5	Motion activity.....	87
10	Localization	92
10.1	Introduction	92
10.2	Region locator.....	92
10.3	Spatio-temporal locator	96
11	Others	103
11.1	Introduction	103
11.2	Face recognition	103
Annex A (normative) Basis functions for FaceRecognition		105
A.1 Basis matrix		105
A.2 Mean face		169
Annex B (normative) Binary representation of media time tools.....		171
B.1 Introduction		171
B.2 Binary representation syntax.....		172
B.3 Descriptor components semantics		173
Annex C (informative) Patent statements		174

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 15938 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 15938-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 15938 consists of the following parts, under the general title *Information technology — Multimedia content description interface*:

- *Part 1: Systems*
- *Part 2: Description definition language*
- *Part 3: Visual*
- *Part 4: Audio*
- *Part 5: Multimedia description schemes*
- *Part 6: Reference software*
- *Part 7: Conformance testing*
- *Part 8: Extraction and use of MPEG-7 descriptions*

Annexes A and B form a normative part of this part of ISO/IEC 15938. Annex C is for information only.

Introduction

This standard, also known as "Multimedia Content Description Interface," provides a standardized set of technologies for describing multimedia content. The standard addresses a broad spectrum of multimedia applications and requirements by providing a metadata system for describing the features of multimedia content.

The following are specified in this standard:

- **Description Schemes (DS)** describe entities or relationships pertaining to multimedia content. Description Schemes specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.
- **Descriptors (D)** describe features, attributes, or groups of attributes of multimedia content.
- **Datatypes** are the basic reusable datatypes employed by Description Schemes and Descriptors
- **Description Definition Language (DDL)** defines Description Schemes, Descriptors, and Datatypes by specifying their syntax, and allows their extension.
- **Systems tools** support delivery of descriptions, multiplexing of descriptions with multimedia content, synchronization, file format, and so forth.

This standard is subdivided into eight parts:

Part 1 – Systems: specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and descriptions.

Part 2 – Description definition language: specifies the language for defining the standard set of description tools (DSs, Ds, and datatypes) and for defining new description tools.

Part 3 – Visual: specifies the description tools pertaining to visual content.

Part 4 – Audio: specifies the description tools pertaining to audio content.

Part 5 – Multimedia description schemes: specifies the generic description tools pertaining to multimedia including audio and visual content.

Part 6 – Reference software: provides a software implementation of the standard.

Part 7 – Conformance testing: specifies the guidelines and procedures for testing conformance of implementations of the standard.

Part 8 – Extraction and use of MPEG-7 descriptions: provides guidelines and examples of the extraction and use of descriptions.

This document contains the visual elements (Descriptors and Description Schemes) that are considered for being part of the standard. All these Descriptive Structures are classified according to the types of visual features they describe. For each Descriptive Structure, there is one corresponding section in this document. The section specifies textual and binary syntax and semantics of the structures.

Information technology — Multimedia content description interface —

Part 3: Visual

1 Scope

1.1 Organization of the document

The structure of this document is as follows. Clauses 2-4 specify the terms, abbreviations, symbols and conventions used throughout the document. Clauses 5-11 contain definitions of the description tools standardized by 15938-3 grouped by the visual features they are associated with, starting with basic structures and containers in Clause 5, through color, texture, shape, motion, localization in Clause 10. Clause 11 contains the remaining, unclassified items.

Each description tool is described by the following subclauses:

- Syntax: Normative DDL specification of the Ds or DSs.
- Binary Syntax: Normative binary representation of the Ds or DSs.
- Semantic: Normative definition of the semantics of all the components of the corresponding D or DS.

1.2 Overview of Visual Description Tools

This part of ISO/IEC 15938 specifies tools for description of visual content, including still images, video and 3D models. These tools are defined by their syntax in DDL and binary representations and semantics associated with the syntactic elements. They enable description of the visual features of the visual material, such as color, texture, shape and motion, as well as localization of the described objects in the image or video sequence. An overview of the visual description tools is shown in Figure 1.

The basic structure description tools include five supporting tools of visual descriptions defined in clauses 6-11. They are categorized into two groups, descriptor containers and basic supporting tools. The former consists of three datatypes, GridLayout providing efficient representations of visual features on grids, TimeSeries representing temporal arrays of several descriptions, and MultipleView describing a 3D object using several pictures captured from different view angles. The latter contains two tools, Spatial2DCoordinateSystem used to specify the 2D coordinate system and TemporalInterpolation indicating the interpolation method between two samples on a time axis.

The remaining description tools, except for the FaceRecognition descriptor, are associated with visual features and are grouped into five feature categories: Color, Texture, Shape, Motion and Localization.

The color description tools include four color descriptors to represent different aspects of color features: representative colors (DominantColor), color distribution (ScalableColor), spatial distribution of colors (ColorLayout and ColorStructure). It also contains two supporting tools, ColorSpace and ColorQuantization used in DominantColor and an extension of ScalableColor to a group of frames or pictures (GoFGoPColor). All the color descriptors can be extracted from arbitrarily shaped regions.

The texture description tools facilitate browsing (TextureBrowsing) and similarity retrieval (HomogeneousTexture and EdgeHistogram) using the texture of a still or moving image region. All the texture descriptors can be extracted from arbitrarily shaped regions.

The shape description tools include two descriptors that characterize different shape features of a 2D object or region. The RegionShape descriptor captures the distribution of all pixels within a region and the Contour Shape descriptor characterizes the shape properties of the contour of an object. The Shape3D descriptor provides an intrinsic shape characterization of 3D mesh models.

The motion description tools include four descriptors that characterize various aspects of motion. The CameraMotion descriptor specifies a set of basic camera operations such as, for example, panning and tilting. The motion of a key point (pixel) from a moving object or region can be characterized by the MotionTrajectory descriptor. The ParametricMotion descriptor characterizes an evolution of an arbitrarily shaped region over time in terms of a 2D geometric transformation. Finally, the MotionActivity descriptor captures the pace of the motion in the sequence, as perceived by the viewer. All motion descriptors except for CameraMotion can be extracted from arbitrarily shaped regions.

The localization description tools can be used to indicate regions of interest in the spatial (RegionLocator) and spatio-temporal (SpatioTemporalLocator) domains.

The FaceRecognition descriptor is not associated with any particular visual feature and can be used to describe a human face for applications requiring the matching and retrieval of face images.

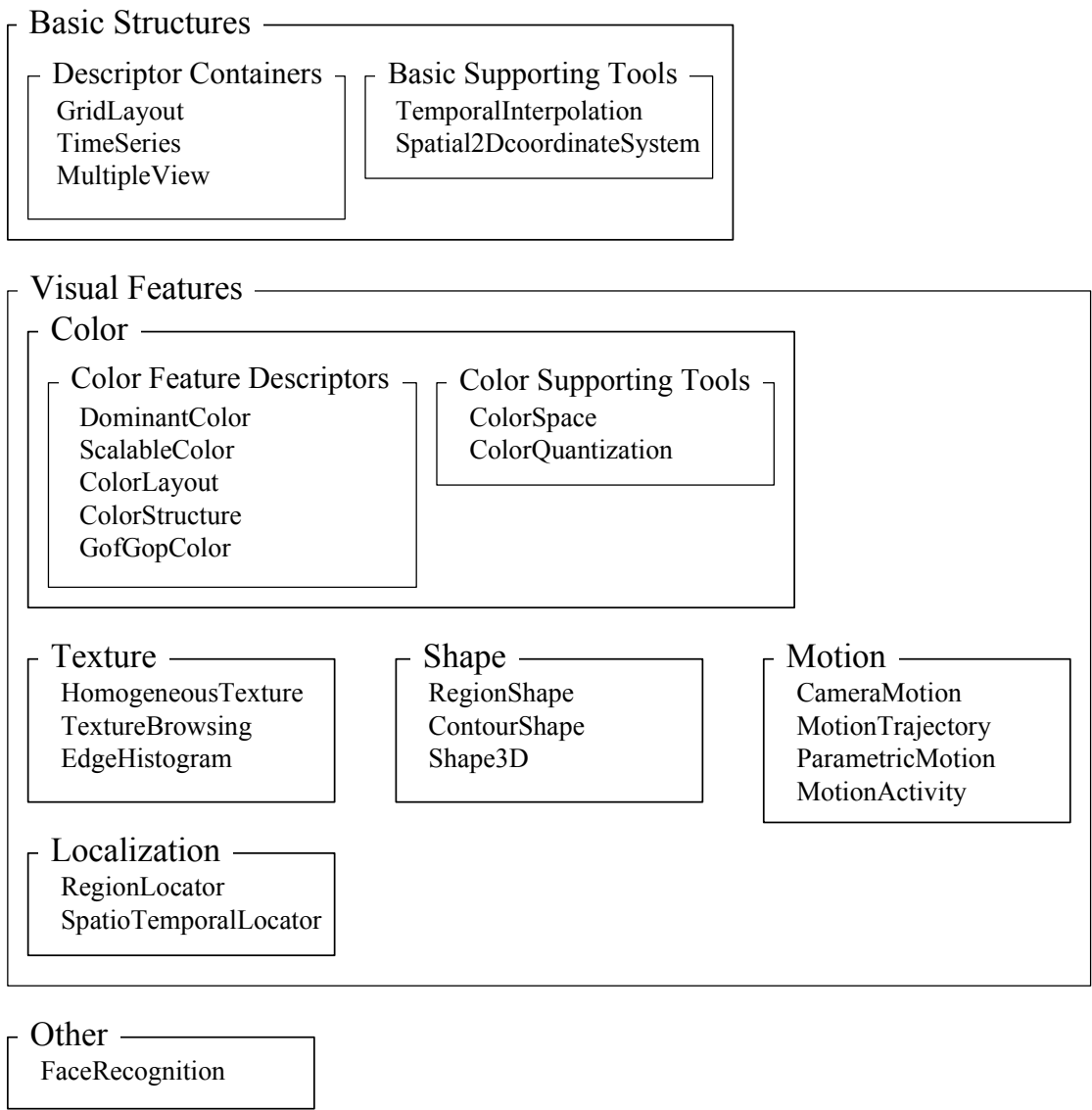


Figure 1 — Overview of Visual Description Tools

2 Terms and Definitions

2.1 Default reference axis

The default reference axis for angle calculation is the positive x (horizontal) axis. Positive angle is calculated anti-clockwise.

2.2 DCT coefficients

DCT coefficient

The signed amplitude of a specific cosine basis function.

AC coefficient

Any DCT coefficient for which the frequency in one or both dimensions is non-zero.

DC coefficient

The DCT coefficient for which the frequency in both dimensions is zero.

2.3 Data element

An item of data as represented before encoding and after decoding.

3 Abbreviations and Symbols

3.1 General

The mathematical symbols used to describe ISO/IEC 15938-3 are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin with zero.

3.2 Abbreviations

ART	Angular-Radial Transform
CSS	Curvature Scale Space
DDL	Description Definition Language
DS	Description Scheme
D	Descriptor
DCT	Discrete Cosine Transform
FOC	Focus of Contraction
FOE	Focus of Expansion
GoF	Group of Frames
GoP	Group of Pictures
HMMD	Hue-Min-Max-Difference
HSV	Hue-Saturation-Value
RGB	Red-Green-Blue

3.3 Arithmetic operators

+	Addition
-	Subtraction (as a binary operator) or negation (as a unary operator)
++	Increment, i.e. $x++$ is equivalent to $x=x+1$
--	Decrement, i.e. $x--$ is equivalent to $x=x-1$
*	Multiplication
x	Multiplication
^	Power
/	Integer division with truncation of the result towards zero. For example, $7/4$ and $-7/4$ are truncated to 1 and -2 respectively.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example, $3//2$ is rounded to 2, and $-3//2$ is rounded to -2.
÷	Used to indicate division in mathematical equations where no rounding is intended
%	Modulus operator, defined only for positive numbers
ld	Logarithm base 2
ceil	Minimum integer number greater or equal than the given floating point number

3.4 Logical operators

	Logical OR
&&	Logical AND
!	Logical NOT

3.5 Relational operators

>	Greater than
>=	Greater than or equal to
≥	Greater than or equal to
<	Less than
<=	Less than or equal to
≤	Less than or equal to

== Equal to

!= Not equal to

3.6 Bitwise operators

| OR

& AND

>> Shift right with sign extension

<< Shift left with zero fill

3.7 Conditional operator

$?: \text{condition} ? a : b = \begin{cases} a & \text{if}(\text{condition}) \\ b & \text{otherwise} \end{cases}$

3.8 Assignment

= Assignment operator

3.9 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

- bslbf Bit string, left bit first, where “left” is the order in which bits are written in ISO/IEC 15938-3. Bit strings are generally written as a string of 1s and 0s within single quote marks, e.g. ‘1000 0001’. Blanks within a bit string are for ease of reading and have no significance. For convenience, large strings are occasionally written in hexadecimal, in which case conversion to a binary in the conventional manner will yield the value of the bit string. Thus, the left-most hexadecimal digit is first and in each hexadecimal digit the most significant of the four digits is first.
- vluimsbf5 Variable length unsigned integer most significant bit first representation consisting of two parts. The first part defines the number n of 4-bit bit fields used for the value representation, encoded by a sequence of $n-1$ “1” bits, followed by a “0” bit signaling its end. The second part contains the value of the integer encoded using the number of bit fields specified in the first part.
- uimsbf Unsigned integer, most significant bit first.
- simsbf Signed integer, in two’s complement format, most significant bit (sign) first.
- vlclbf Variable length code, left bit first, where “left” refers to the order in which the VLC codes are written in ISO/IEC 15938-3. The byte order of multibyte words is most significant byte first.
- fsbf Float (32 bit), sign bit first. The semantics of the bits within a float are specified in the IEEE Standard for Binary Floating Point Arithmetic (ANSI/IEEE Std 754-1985).
- UTF-8 Binary string encoding defined in ISO 10646/IETF RFC 2279.

3.10 Constants

π 3.141 592 653 58...

e 2.718 281 828 45...

3.11 Functions

max() Maximum value in argument list

min() Minimum value in argument list

Sign() $Sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$

Abs() $Abs(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$

$\sum_{i=a}^{i<b} f(i)$ Summation of $f(i)$ with i taking integer values from a up to, but not including b .

$$\text{L1 norm} \quad L1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$$

$$\text{L2 norm} \quad L2(\mathbf{x}, \mathbf{y}) = \sum_i (x_i - y_i)^2$$

$$\text{Euclidean distance} \quad D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

4 Conventions

4.1 Method of describing the DDL representation syntax

The method of describing the DDL representation syntax is defined in ISO/IEC 15938-2 (MPEG-7 Description Definition Language).

4.2 Method of describing the binary representation syntax

4.2.1 Introduction

The video description elements can be encoded using the generic encoding mechanism defined in ISO/IEC 15938-1 or the video-specific binary representation syntax defined in the "Binary representation syntax" subclauses below.

4.2.2 Generic binary representation

The use of the video-specific syntax is signaled using the codec configuration mechanism defined in ISO/IEC 15938-1 and the following classification scheme is defined for this purpose.

```
<ClassificationScheme uri="urn:mpeg:mpeg7:cs:VisualDescriptorCodecCS:2001">
  <Term termID="1">
    <Name xml:lang="en">MPEG7CameraMotion</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Camera Motion
      Codec</Definition>
  </Term>
  <Term termID="2">
    <Name xml:lang="en">MPEG7ColorLayout</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Color Layout
      Codec</Definition>
  </Term>
  <Term termID="3">
    <Name xml:lang="en">MPEG7ColorQuantization</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Color Quantization
      Codec</Definition>
  </Term>
  <Term termID="4">
    <Name xml:lang="en">MPEG7ColorSpace</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Color Space
      Codec</Definition>
  </Term>
  <Term termID="5">
    <Name xml:lang="en">MPEG7ColorStructure</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Color Structure
      Codec</Definition>
  </Term>
  <Term termID="6">
    <Name xml:lang="en">MPEG7ContourShape</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Contour Shape
      Codec</Definition>
  </Term>
  <Term termID="7">
```

```

    <Name xml:lang="en">MPEG7DominantColor</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Dominant Color
    Codec</Definition>
</Term>
<Term termID="8">
    <Name xml:lang="en">MPEG7EdgeHistogram</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Edge Histogram
    Codec</Definition>
</Term>
<Term termID="9">
    <Name xml:lang="en">MPEG7FaceRecognition</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Face Recognition
    Codec</Definition>
</Term>
<Term termID="10">
    <Name xml:lang="en">MPEG7FoFGoPColor</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary GoFGoP Color
    Codec</Definition>
</Term>
<Term termID="11">
    <Name xml:lang="en">MPEG7GridLayout</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Grid Layout
    Codec</Definition>
</Term>
<Term termID="12">
    <Name xml:lang="en">MPEG7HomogeneousTexture</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Homogeneous Texture
    Codec</Definition>
</Term>
<Term termID="13">
    <Name xml:lang="en">MPEG7IrregularVisualTimeSeries</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Irregular Time Series
    Codec</Definition>
</Term>
<Term termID="14">
    <Name xml:lang="en">MPEG7MotionActivity</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Motion Activity
    Codec</Definition>
</Term>
<Term termID="15">
    <Name xml:lang="en">MPEG7MotionTrajectory</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Motion Trajectory
    Codec</Definition>
</Term>
<Term termID="16">
    <Name xml:lang="en">MPEG7MultipleView</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Multiple View
    Codec</Definition>
</Term>
<Term termID="17">
    <Name xml:lang="en">MPEG7ParametricMotion</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Parametric Motion
    Codec</Definition>
</Term>
<Term termID="18">
    <Name xml:lang="en">MPEG7RegionLocator</Name>
    <Definition xml:lang="en">ISO/IEC 15938-3 Binary Region Locator
    Codec</Definition>
</Term>
<Term termID="19">
    <Name xml:lang="en">MPEG7RegionShape</Name>

```

```

        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Region Shape
        Codec</Definition>
    </Term>
    <Term termID="20">
        <Name xml:lang="en">MPEG7RegularVisualTimeSeries</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Regular Time Series
        Codec</Definition>
    </Term>
    <Term termID="21">
        <Name xml:lang="en">MPEG7ScalableColor</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Scalable Color
        Codec</Definition>
    </Term>
    <Term termID="22">
        <Name xml:lang="en">MPEG7Shape3D</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Shape 3D
        Codec</Definition>
    </Term>
    <Term termID="23">
        <Name xml:lang="en">MPEG7Spatial2DCoordinateSystem</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Spatial 2D Coordinate
        System Codec</Definition>
    </Term>
    <Term termID="24">
        <Name xml:lang="en">MPEG7SpatioTemporalLocator</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary SpatioTemporal Locator
        Codec</Definition>
    </Term>
    <Term termID="25">
        <Name xml:lang="en">MPEG7TemporalInterpolation</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Temporal Interpolation
        Codec</Definition>
    </Term>
    <Term termID="26">
        <Name xml:lang="en">MPEG7TextureBrowsing</Name>
        <Definition xml:lang="en">ISO/IEC 15938-3 Binary Texture Browsing
        Codec</Definition>
    </Term>
</ClassificationScheme>

```

4.2.3 Video binary representation

The video-specific bitstream retrieved is described in subclauses entitled “Binary syntax representation” in clauses 5—11. Each data item in the bitstream is in bold type. It is described with its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The following constructs are used to express the conditions when data elements are present and are in normal type.

<pre> while (condition) { data_element ... } </pre>	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.
<pre> do { data_element ... } while (condition) </pre>	The data element always occurs at least once. The data element is repeated until the condition is not true.

<pre>if (condition) { data_element ... } else { data_element ... }</pre>	If the condition is true, then the first group of data elements occurs next in the data stream. If the condition is not true, then the second group of data elements occurs next in the data stream.
<pre>for (i=m ; i<n ; i++) { data_element ... }</pre>	The group of data elements occurs (n-m) times. Conditional construct within the group of data elements may depend on the value of the loop control variable i, which is set to m for the first occurrence, incremented by one for the second occurrence, and so forth.
<pre>/* comments */</pre>	Explanatory comments that may be deleted entirely without in any way altering the syntax.

The syntax uses a ‘C-code’ convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true and a variable or expression evaluating to zero is equivalent to a condition that is false. In many cases a literal string is used in a condition. In such cases a literal string is used to describe the value of a bitstream element.

As noted, a group of data element may contain nested conditional constructs. For compactness, the brackets {} are omitted when only one data element follows. The elements of a multidimensional table are represented as follows.

- data_element[n]** data_element[n] is the n+1st element in an array of data
- data_element[m][n]** data_element[m][n] is the m+1st, n+1st element in a two-dimensional array of data
- data_element[l][m][n]** data_element[l][m][n] is the l+1st, m+1st, n+1st element in a three-dimensional array

The elements of a multidimensional array are transmitted in the bitstream starting with data_element[0][0] and with the outermost elements incremented first, i.e. data_element[0][1] is sent second, data_element[0][2] third, etc.

4.3 Method of describing the descriptor semantics

The general semantics of the descriptors are defined in the introductory sections of respective subclauses. The semantics of the syntax components is defined in sections “Descriptor components semantics”. The ordering in the semantics sections normally follows the order in which the items appear in the binary representation syntax, which is typically equivalent to the order of items in the DDL instantiation (not schema specification).

5 Basic structures

5.1 Introduction

This clause introduces five supporting tools for the visual descriptions defined in clauses 6–11. They are categorized into two groups, descriptor containers and basic supporting tools. The former consists of three datatypes, GridLayout providing efficient representations of visual features on grids, VisualTimeSeries representing temporal arrays of several descriptions, and MultipleView describing a 3D object using several pictures captured from different view angles. The latter contains two tools, Spatial2DCoordinateSystem used to specify the 2D coordinate system and TemporalInterpolation indicating the interpolation method between two samples on a time axis.

5.2 Grid layout

5.2.1 Introduction

This datatype specifies a structure that allows an image to be split into a set of rectangular regions, so that each region can be described separately. Each region of the grid can be described in terms of other descriptors such as color or texture.

5.2.2 DDL representation syntax

```

<complexType name="GridLayoutType" final="#all">
  <sequence>
    <element name="Descriptor" type="mpeg7:VisualDType"
      minOccurs="1" maxOccurs="65025"/>
  </sequence>
  <attribute name="numOfPartX" type="mpeg7:unsigned8" use="required"/>
  <attribute name="numOfPartY" type="mpeg7:unsigned8" use="required"/>
  <attribute name="descriptorMask" use="optional">
    <simpleType>
      <restriction base="string">
        <pattern value="(0|1)*"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>

```

5.2.3 Binary representation syntax

GridLayout {	Number of bits	Mnemonic
DescriptorID	8	uimsbf
numOfPartX	8	uimsbf
numOfPartY	8	uimsbf
DescriptorMaskPresent	1	bslbf
if(DescriptorMaskPresent) {		
descriptorMask	partNumX*partNumY	bslbf
}		
for(k=0;k<partNumX*partNumY; k++) {		
if(DescriptorMaskPresent) {		
if(descriptorMask[k]) {		
Descriptor[k]		Descriptor instance specified by descriptorID
}		
} else {		
Descriptor[k]		Descriptor instance specified by descriptorID
}		
}		
}		

5.2.4 Descriptor components semantics

DescriptorID

This field, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier indicates the descriptor type accommodated in the grid layout.

The assignment of IDs to the descriptors is specified in Table 1.

Table 1 — Assignment of IDs to descriptors

ID	Descriptor
0	Forbidden
1	CameraMotion
2	ColorLayout
3	ColorSpace
4	ColorStructure
5	ColorQuantization
6	ContourShape
7	DominantColor
8	EdgeHistogram
9	FaceRecognition
10	GoFGoPColor
11	GridLayout
12	HomogeneousTexture
13	IrregularVisualTimeSeries
14	MotionActivity
15	MotionTrajectory
16	MultipleView
17	ParametricMotion
18	RegionLocator
19	RegionShape
20	RegularVisualTimeSeries
21	ScalableColor
22	Shape3D
23	Spatial2DCoordinateSystem
24	SpatioTemporalLocator
25	TemporalInterpolation
26	TextureBrowsing
27-255	Reserved

numOfPartX

This attribute specifies the number of horizontal partitions in the grid over the image.

numOfPartY

This attribute specifies the number of vertical partitions in the grid over the image.

DescriptorMaskPresent

This field, which is only present in the binary syntax, indicates whether all partitions of the image contain the descriptors. If DescriptorMaskPresent is set to 0 then all partitions contain the descriptor. If DescriptorMaskPresent is set to 1 then the DescriptorMask attribute indicates which partitions contain descriptors.

descriptorMask

This attribute specifies a bit-field that indicates whether a descriptor is assigned to the corresponding partition. The partitioned image is indexed from left to right and top to bottom. For example, if a descriptorMask of "0110" is given for a 2x2 partitioned image then the upper right and lower left quarter of the image contain a descriptor.

Descriptor

This element specifies the visual descriptors that have been assigned to the cells. When a visual descriptor is assigned to a cell within grid layout, it defines the properties of the particular cell according to the semantic definition of the descriptor used. In other words, each cell is treated as an individual image and the descriptor values are computed accordingly. For example, a dominant color descriptor assigned to a cell specifies the dominant colors of the pixels within that cell. All the restrictions on the image size, etc. are now applicable to the size of the cell within the grid. All the descriptor instances in a single GridLayout instance must be of the same type.

The following visual description tools cannot appear in the GridLayout: VisualTimeSeries, MultipleView, Spatial2DCoordinateSystem, TemporalInterpolation, ColorSpace, ColorQuantization, Shape3D, CameraMotion, MotionTrajectory, ParametricMotion, SpatioTemporalLocator.

Grid layout can be applied to a video segment, in which case the geometry of the grid layout is constant over time. Each frame in the segment is divided into the same number of cells and the cells at corresponding locations form sequences that can be viewed as a group of frames. The GoFGoPColor and MotionActivity descriptors can then be applied to each sequence defined by the grid layout.

5.3 Time series

5.3.1 Introduction

This datatype specifies a temporal series of descriptors in a video segment. Two types of time series datatypes are defined: *RegularVisualTimeSeries* and *IrregularVisualTimeSeries*. In the *RegularVisualTimeSeries*, descriptors are located regularly (with constant intervals) within a given time span. Alternatively, descriptors are located irregularly in the *IrregularVisualTimeSeries*. Both structures consist of a series of descriptors and temporal intervals between them as illustrated in Figure 2.

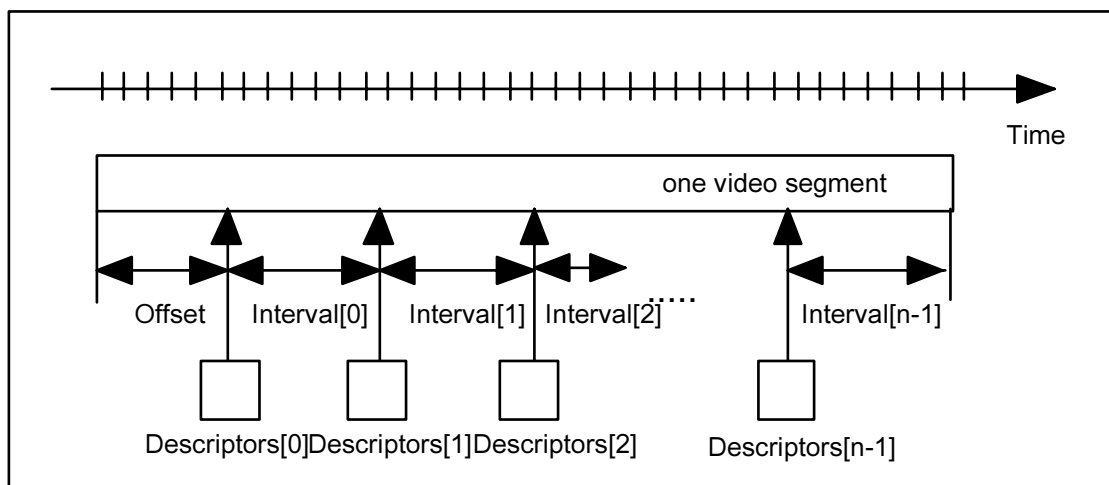


Figure 2 — Overview of the VisualTimeSeries

5.3.2 VisualTimeSeries

5.3.2.1 Introduction

The *VisualTimeSeries* serves as the base type for *RegularVisualTimeSeries* and *IrregularVisualTimeSeries*. As it is an abstract type, only the DDL representation is defined.

5.3.2.2 DDL representation syntax

```
<complexType name="VisualTimeSeriesType" abstract="true">
  <sequence>
    <element name="TimeIncr" type="mpeg7:mediaDurationType"/>
  </sequence>
  <attribute name="offset" type="mpeg7:mediaDurationType"
    use="optional" default="PT0S"/>
</complexType>
```

5.3.3 RegularVisualTimeSeries

5.3.3.1 Introduction

This datatype specifies a time series in which descriptors are located regularly within a given time span. A default value for the temporal interval can be specified thus enabling a simple representation for applications that require low complexity.

5.3.3.2 DDL representation syntax

```
<complexType name="RegularVisualTimeSeriesType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualTimeSeriesType">
      <sequence>
        <element name="Descriptor" type="mpeg7:VisualDType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

        minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
</extension>
</complexContent>
</complexType>

```

5.3.3.3 Binary representation syntax

RegularVisualTimeSeries {	Number of bits	Mnemonic
DescriptorID	8	bslbf
NumOfDescriptors	32	uimsbf
IsRandomAccess	1	bslbf
if(IsRandomAccess) {		
DescriptorLength	16	uimsbf
}		
TimeIncr	See annex B	mediaDurationType
IsOffset	1	bslbf
if(IsOffset) {		
offset	See annex B	mediaDurationType
}		
if(IsRandomAccess) {		
BitStuffing	0-7	vlclbf
}		
for(k=0; k<NumOfDescriptors; k++) {		
Descriptor[k]		Descriptor instance specified by DescriptorID
if(IsRandomAccess) {		
BitStuffing	0-8*DescriptorLength-1	vlclbf
}		
}		
}		

5.3.3.4 Descriptor components semantics

DescriptorID

This field, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier specifies the descriptor type accommodated in the time series. The assignment of IDs to the descriptors is specified in Table 1.

NumOfDescriptors

This field, which is only present in the binary representation, specifies the number of descriptor instances accommodated in the time series.

IsRandomAccess

This field, which is only present in the binary representation, specifies the access mode, which is either:

- random access if the flag is set to 1; in this case DescriptorLength and BitStuffing elements are present in the binary representation
- no random access if the flag is set to 0; in this case no bit stuffing is allowed and descriptor instances are not padded, which means they may have different lengths

DescriptorLength

This field, which is only present in the binary representation, specifies the length of each descriptor instance in bytes. The value of this element is the size of the largest descriptor instance, aligned to a byte boundary by bit stuffing using 0-7 '1' bits.

TimeIncr

This element specifies the default time interval. The time interval is defined as an interval between descriptor locations. An interval that follows a descriptor is associated with the descriptor. The type of this element "mediaDurationType" is specified in ISO/IEC 15938-5.

IsOffset

This field, which is only present in the binary representation, signals the presence of the offset attribute. If it is equal to 1 (true) offset is present, if 0 (false) offset is not specified (i.e. default value should be used).

offset

This attribute specifies the offset, i.e., the interval between the starting time point of a given time span and the location of the first descriptor. The default value is zero (represented as "PT0S" in DDL). This attribute is illustrated as "Offset" in Figure 2.

BitStuffing

This field, which is only present in the binary representation, specifies stuffing bits (a sequence of '1's) to align the descriptor to a byte boundary.

Descriptor

This element specifies the visual descriptor accommodated in this time series. Only one type of child descriptor is allowed to be instantiated. Its binary syntax and semantics follow those of the assigned descriptor. In random access mode, if the size of a particular descriptor instance is smaller than DescriptorLength, it is padded with the required number of '1' bits.

5.3.4 IrregularVisualTimeSeries**5.3.4.1 Introduction**

This datatype is used to describe a time series in which descriptors are located irregularly within a given time span. The temporal interval can be associated with a descriptor by means of describing the series of (descriptor, time info.) pairs. This enables an efficient representation for an application that has the requirement of narrow transmission bandwidth or low storage capability.

5.3.4.2 DDL representation syntax

```
<complexType name="IrregularVisualTimeSeriesType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualTimeSeriesType">
      <sequence minOccurs="1" maxOccurs="unbounded">
        <element name="Descriptor" type="mpeg7:VisualDType"/>
        <element name="Interval" type="mpeg7:unsigned32"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.4.3 Binary representation syntax

IrregularVisualTimeSeries {	Number of bits	Mnemonic
DescriptorID	8	bslbf
NumOfDescriptors	32	uimsbf
IsRandomAccess	1	bslbf
if(IsRandomAccess) {		
DescriptorLength	16	uimsbf
}		
TimeIncr	See annex B	mediaDurationType
IsOffset	1	bslbf
if(IsOffset) {		
offset	See annex B	mediaDurationType
}		

IsShortInterval	1	bslbf
if(IsRandomAccess) {		
BitStuffing	0-7	vlclbf
}		
for(i=0; i<NumOfDescriptors; i++) {		
Descriptor[i]		descriptor instance specified by DescriptorID
if(IsRandomAccess) {		
BitStuffing	0-8*DescriptorLength-1	vlclbf
}		
if (IsShortInterval) {		
ShortInterval[i]	8	uimsbf
} else {		
LongInterval[i]	32	uimsbf
}		
}		
}		

5.3.4.4 Descriptor components semantics

DescriptorID

This field, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier specifies the descriptor type accommodated in the time series. The assignment of IDs to the descriptors is specified in Table 1.

NumOfDescriptors

This field, which is only present in the binary representation, specifies the number of descriptor instances accommodated in the time series.

IsRandomAccess

This field, which is only present in the binary representation, specifies the access mode, which is either:

- random access if the flag is set to 1; in this case DescriptorLength and BitStuffing elements are present in the binary representation
- no random access if the flag is set to 0; in this case no bit stuffing is allowed and descriptor instances are not padded, which means they may have different lengths

DescriptorLength

This field, which is only present in the binary representation, specifies the length of each descriptor instance in bytes. The value of this element is the size of the largest descriptor instance, aligned to a byte boundary by bit stuffing using 0-7 '1' bits.

IsShortInterval

This field, which is only present in the binary representation, indicates the size of the ShortInterval/LongInterval field. If IsShortInterval is set to 1, then 8-bit unsigned integer ("unsigned8") is used. If IsShortInterval is set to 0, then a 32-bit unsigned integer ("unsigned32") is used.

TimeIncr

This element specifies the base unit of the time interval. The time interval between descriptor locations is specified as a multiple of this base unit. The type of this element, MediaDurationType, is specified in ISO/IEC 15938-5.

IsOffset

This field, which is only present in the binary representation, signals the presence of the offset attribute. If IsOffset is set to 1 then the offset attribute is present. If IsOffset is set to 0 then the offset attribute is not specified (i.e. the default value should be used).

offset

This attribute specifies the offset, i.e., the interval between the starting time point of a given time span and the location of the first descriptor. The default value is zero (represented as "PT0S" in DDL). This element is illustrated as "Offset" in Figure 2.

BitStuffing

This field, which is only present in the binary representation, specifies stuffing bits (a sequence of '1's) to align the descriptor to a byte boundary.

Descriptor

This element specifies the visual descriptor accommodated in this time series. Only one type of child descriptor is allowed to be instantiated. Its binary syntax and semantics follow those of the assigned descriptor. In random access, if the size of a particular descriptor instance is smaller than DescriptorLength, it is padded with the required number of '1' bits.

Interval/ShortInterval/LongInterval

This element specifies the time interval between the current and the preceding descriptor. The value of the element is specified in units defined by TimeIncr.

5.4 Multiple view**5.4.1 Introduction**

The MultipleView descriptor specifies a structure that combines 2D Descriptors representing a visual feature of a 3D object seen from different view angles. The descriptor forms a 3D view-based representation of the object. Any 2D visual descriptor, such as for example contour-shape, region-shape, color or texture can be used. The Multiple View descriptor supports integration of the 2D descriptors used in the image plane to describe features of 3D (real world) objects.

The MultipleView representation of a 3D object captures the visual properties of an object at a specific time instance or during a time period. When associated with a video segment, MultipleView contains a set of visual descriptors, each describing a visual feature of an object's view at some time instance during the persistence of the video segment. The IsVisible flag indicates for each component descriptor whether the view that it describes is actually visible in the video segment.

5.4.2 DDL representation syntax

```
<complexType name="MultipleViewType" final="#all">
  <sequence minOccurs="1" maxOccurs="16">
    <element name="IsVisible" type="boolean"/>
    <element name="Descriptor" type="mpeg7:VisualDType"/>
  </sequence>
  <attribute name="fixedViewsFlag" type="boolean" use="required"/>
</complexType>
```

5.4.3 Binary representation syntax

MultipleView{	Number of bits	Mnemonic
DescriptorID	8	uimsbf
fixedViewsFlag	1	bslbf
NumOfViews	4	uimsbf
for(k=0;k<NumOfViews;k++) {		
IsVisible[k]	1	bslbf
Descriptor[k]		Description instance specified by DescriptorID
}		
}		

5.4.4 Descriptor components semantics

DescriptorID

This field, which is only present in the binary representation, specifies a descriptor identifier. The descriptor identifier specifies the descriptor type accommodated in the multiple views. The assignment of IDs to the descriptors is specified in Table 1.

fixedViewsFlag

This attribute indicates whether the viewing parameters are fixed or arbitrary/unknown. If fixedViewsFlag is set to 0, the viewing parameters are arbitrary. If fixedViewsFlag is set to 1, the viewing parameters are defined as follows.

For the ContourShape descriptor, the primary, secondary and tertiary viewing directions are determined by the analysis of the covariance matrix of the 3D object. The terms primary, secondary and tertiary eigenvector are used for the eigenvector with the largest, medium and smallest eigenvalue, respectively.

View 1 is spanned by the primary and secondary eigenvector of the covariance matrix. Its direction is therefore the direction of the tertiary eigenvector. Similarly, View 2 is in the direction of the secondary eigenvector and View 3 in the direction of the primary eigenvector.

In addition, another four views are added. The position of View 4 – View 7 can best be described with reference to a coordinate system, where View 1, View 2 and View 3 define axis (1,0,0), (0,1,0) and (0,0,1):

- View 4: (1,1,1)
- View 5: (-1,1,1)
- View 6: (-1,1,-1)
- View 7: (1,1,-1)

This flag can only be set to 1 if ContourShape is used.

NumOfViews

This field, which is only present in the binary representation, specifies the number of views used. In the binary representation, the following mapping of the bit patterns is used: 0000->1, ..., 1111->16.

IsVisible

This element specifies a 1-D array that contains a flag for each view, specifying whether the view is visible within the associated visual material.

Descriptor

This element specifies the visual descriptor instance accommodated in this descriptor. Only one type of child descriptor is allowed to be instantiated.

5.5 Spatial 2D coordinates

5.5.1 Introduction

This descriptor specifies a 2-D spatial coordinate system to be used by reference in other Ds/DSs (denoted by D1). The coordinate system is defined by mapping from the points in the descriptor coordinate system (used within D1) to the points in the current image. Referring to this descriptor, the coordinates in the visual descriptor can be mapped to the current image. The *current image* is the image that is currently referenced by the description, while the *descriptor coordinate system* is the coordinate system in which the described value is valid. The descriptor coordinate system can be the source image from which the current image is extracted or the real world. The term *source image* is used to express the image from which the description is extracted. Therefore, the source image can be a frame in a video, an image from which the current image is extracted, or a mosaic. When the current image is extracted from the source image, this descriptor specifies the mapping from the source image to the current image. Thus, without modifying the descriptor instantiations, the description can be adapted to the current image by modifying the mapping.

The Spatial2DCoordinateSystem supports two kinds of coordinate systems: *local* and *integrated* (see Figure 3). In a local coordinate system, the coordinates in the descriptor coordinate system are mapped to the current image. For a video sequence, the same mapping is applied to all frames. In an integrated coordinate system, each image (frame) of e.g. a video sequence may be mapped to different areas with respect to the first frame of a shot or video. In this case, the local coordinate system is used to specify the coordinate system of the first frame (i.e. the first frame is the current frame). Thus, the integrated coordinate system can be used to represent coordinates on a mosaic of a video shot.

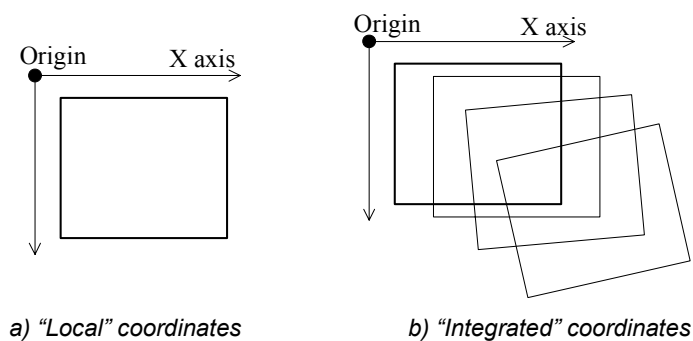


Figure 3 — "Local" and "Integrated" coordinate system

In the syntax, the IntegratedCoordinateSystem element allows specification of an integrated coordinate system, and the LocalCoordinateSystem element allows specification of a local one, when it is different from the default coordinate system of the current image. In the default local coordinate system, the origin is placed at the top left corner of the image and the first and the second axis are aligned with the horizontal and vertical lines of the image and directed to the right and to the bottom of the image, respectively. If additionally an integrated coordinate system is specified, the mapping applies to the first coordinate space of the first frame.

5.5.2 DDL representation syntax

```
<complexType name="Spatial2DCoordinateSystemType" final="#all">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <sequence>
        <element name="Unit" minOccurs="0">
          <simpleType>
            <restriction base="string">
              <enumeration value="pixel"/>
              <enumeration value="meter"/>
              <enumeration value="pictureHeight"/>
              <enumeration value="pictureWidth"/>
              <enumeration value="pictureWidthAndHeight"/>
            </restriction>
          </simpleType>
        </element>
        <element name="LocalCoordinateSystem" minOccurs="0">
          <complexType>
            <sequence>
              <choice>
                <sequence minOccurs="1" maxOccurs="3">
                  <element name="Pixel">
                    <simpleType>
                      <restriction base="mpeg7:integerVector">
                        <length value="2"/>
                      </restriction>
                    </simpleType>
                  </element>
                  <element name="CoordPoint">
                    <simpleType>
                      <restriction base="mpeg7:floatVector">
                        <length value="2"/>
                      </restriction>
                    </simpleType>
                  </element>
                </sequence>
                <sequence minOccurs="1" maxOccurs="3">
                  <element name="CurrPixel">
                    <simpleType>
                      <restriction base="mpeg7:integerVector">
```

```

        <length value="2"/>
      </restriction>
    </simpleType>
  </element>
  <element name="SrcPixel">
    <simpleType>
      <restriction base="mpeg7:integerVector">
        <length value="2"/>
      </restriction>
    </simpleType>
  </element>
</sequence>
</choice>
  <element name="MappingFunct" type="string"
    minOccurs="0" maxOccurs="2"/>
</sequence>
  <attribute name="name" type="string" use="required"/>
  <attribute name="dataSet" type="anyURI" use="optional"/>
</complexType>
</element>
<element name="IntegratedCoordinateSystem" minOccurs="0">
  <complexType>
    <sequence minOccurs="0" maxOccurs="65535">
      <element name="TimeIncr" type="mpeg7:MediaIncrDurationType"/>
      <element name="MotionParams" type="float"
        minOccurs="2" maxOccurs="12"/>
    </sequence>
    <attribute name="modelType" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="translational"/>
          <enumeration value="rotationAndScaling"/>
          <enumeration value="affine"/>
          <enumeration value="perspective"/>
          <enumeration value="quadratic"/>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="xOrigin" type="float" use="required"/>
    <attribute name="yOrigin" type="float" use="required"/>
  </complexType>
</element>
</sequence>
  <attribute name="xRepr" type="mpeg7:unsigned8" use="required"/>
  <attribute name="yRepr" type="mpeg7:unsigned8" use="required"/>
  <attribute name="xSrcSize" type="positiveInteger" use="optional"/>
  <attribute name="ySrcSize" type="positiveInteger" use="optional"/>
</extension>
</complexContent>
</complexType>

```

5.5.3 Binary representation syntax

Spatial2DCoordinateSystem {	Number of bits	Mnemonic
id	See ISO 10646	UTF-8
xRepr	8	uimsbf
yRepr	8	uimsbf
XSrcSizeDefined	1	bslbf
if (XSrcSizeDefined) {		
xSrcSize		vluimsbf5
}		
YSrcSizeDefined	1	bslbf
if (YSrcSizeDefined) {		
ySrcSize		vluimsbf5
}		
UnitDefined	1	bslbf
LocalCoordinatesDefined	1	bslbf
IntegratedCoordinatesDefined	1	bslbf
if (UnitDefined) {		
Unit	3	bslbf
}		
if(LocalCoordinatesDefined) {		
NameLength		vluimsbf5
name	8*NameLength	bslbf
DataSetDefined	1	bslbf
if(DataSetDefined) {		
DataSetLength		vluimsbf5
dataSet	8*DataSetLength	bslbf
}		
Coord	1	bslbf
NumOfPoints	2	uimsbf
for(k=0; k<NumOfPoints; k++) {		
if(!Coord) {		
CurrPixelX	xRepr	usimsbf
CurrPixelY	yRepr	usimsbf
SrcPixelX	16	simsbf
SrcPixelY	16	simsbf
} else {		
PixelX	xRepr	usimsbf
PixelY	yRepr	usimsbf
CoordPointX	32	fsbf
CoordPointY	32	fsbf
}		
}		
NumOfMappingFuncs	2	uimsbf
for(l=0; l<NumOfMappingFuncs; l++) {		
MappingFuncLength		vluimsbf5
MappingFunc	8*MappingFuncLength	bslbf

}		
}		
if(IntegratedCoordinatesDefined) {		
modelType	3	uimsbf
xOrigin	32	fsbf
yOrigin	32	fsbf
NumOfMotionParamSets	16	uimsbf
for(k=0; k<NumOfMotionParamSets; k++) {		
TimeIncr	See annex B	MediaIncrDurationType
for(l=0; l<NumOfParams; l++) {		
MotionParams	32	fsbf
}		
}		
}		
}		

5.5.4 Descriptor components semantics

id

This attribute, which is inherited from HeaderType, allows referencing of the coordinate specification from other descriptors. The value is binary represented as a string in UTF-8 format.

xRepr, yRepr

These attributes specify the number of bits used for the representation of X and Y coordinates in the coordinate system of the source image. If the unit is pixel, the coordinates are expressed by the unsigned integer of xRepr bits for the horizontal axis and yRepr bits for the vertical axis respectively.

XSrcSizeDefined, YSrcSizeDefined

These fields, which are only present in the binary representation, indicate if the xSrcSize and ySrcSize attributes are defined. If the field is set to 1, the corresponding attribute follows. If the field is set to 0 then the attribute is not present.

xSrcSize, ySrcSize

These attributes specify the image size of the source image expressed as the number of pixels along the x (xSrcSize) and y (ySrcSize) axis. The value can be used to convert between pixel-based coordinates and coordinates normalized with respect to the image size.

UnitDefined

This field, which is only present in the binary representation, indicates if the units for the coordinate system are defined. If UnitDefined is set to 1 then the definition of units follows. If UnitDefined is set to 0 then the default units are used.

LocalCoordinatesDefined

This field, which is only present in the binary representation, indicates if a local coordinate system is defined. If LocalCoordinatesDefined is set to 1 then the definition of a local coordinate system follows. Otherwise, the default local definition is used. In the default local coordinate system the origin is placed at the top left corner of the image and the first and the second axis are aligned to the horizontal and vertical axis of the image and directed to the right and to the bottom of the image, respectively.

IntegratedCoordinatesDefined

This field, which is only present in the binary representation, indicates if an integrated coordinate system is defined. If IntegratedCoordinatesSystem is set to 1 then an integrated coordinate system is used and mappings between local images (except for the first frame) and the coordinate system follow. Otherwise, a local coordinate system is used.

Unit

This element specifies the units of the descriptor coordinate system. It is only applicable to descriptors that do not have a fixed unit specification of the descriptor coordinate values. The default unit is pixel. Other possible units are meters (only applicable to the CoordPoint element specification), or pixel coordinates normalized by pictureWidth in each dimension, or by pictureHeight in each dimension, or by pictureWidth and pictureHeight on horizontal and vertical axes respectively.

The meaning of the binary codes is specified in Table 2.

Table 2 — The meaning of Unit

Unit	Meaning
"000"	pixel
"001"	meter
"100"	pictureHeight
"101"	pictureWidth
"110"	pictureHeightAndWidth
"010-011" and "111"	Reserved

LocalCoordinateSystem

This element specifies the mapping from the descriptor coordinate system to the current image. The descriptor coordinate system can be a source image (an image, a frame of video, or a mosaic), or a real world coordinate system. When the descriptor coordinate system is the source image, the mapping from the points in the source image (SrcPixel) to the points in the current image (Pixel) is specified. When the descriptor coordinate system is a real world coordinate system, the origin is located in the top left corner of the image, the x and y axes are aligned to the horizontal and vertical axes of the image and the pixel unit is used.

If a mapping of one point is described, the mapping is translational. If mappings of two points are described, a rotation/scaling model is used and if mappings of three points are described, a rotation/asymmetric scaling (affine) model is used (see Figure 4). If two or more mappings are described, they should be linearly independent.

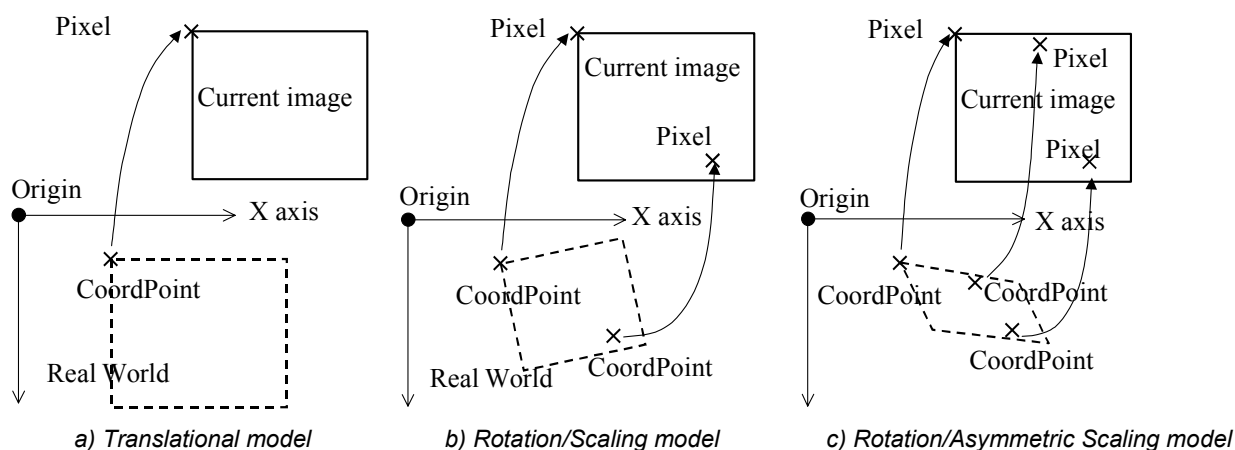


Figure 4 — Motion models described by CoordinateMapping ('SrcPixel' and 'CoordPoint')

NameLength

This field, which is only present in the binary representation, specifies the length of the following coordinate mapping name.

name

This attribute specifies the coordinate mapping name.

DataSetDefined

This field, which is only present in the binary representation, indicates the presence of a dataset definition. If DataSetDefined is set to 1 then a definition of a data set follows.

DataSetLength

This field, which is only present in the binary representation, specifies the length of the following dataSet attribute.

dataSet

This attribute specifies a reference to a data set, of which the SrcPixel based coordinate system is used for mapping to the local one. E.g. after image scaling for UMA here a reference can be specified to the data used to generate the visual descriptors referencing this coordinate system.

NumOfPoints

This field, which is only present in the binary representation, specifies the number of Pixels and SrcPixels or CoordPoints used to specify the mapping between the corresponding coordinate systems.

CurrPixelX, CurrPixelY, PixelX, PixelY, SrcPixelX, SrcPixelY, CoordPointX, CoordPointY

These elements specify a mapping between a descriptor coordinate system and the current image. A mapping is defined by points in the current image (CurrPixelX, CurrPixelY) and the corresponding points in the source image (SrcPixelX, SrcPixelY) (see Figure 5). Another possibility is to map the points of the current image (PixelX, PixelY) to points in a real world coordinate system (CoordPointX, CoordPointY). If a mapping of one point is described, the mapping is translational. If mappings of two points are described, a rotation/scaling model is used and if mappings of three points are described, a rotation/asymmetric scaling (affine) model is used (see Figure 4). Pixel positions in the current image and the source image are described assuming that the origin is located at the top-left corner of the image and the unit is pixel.

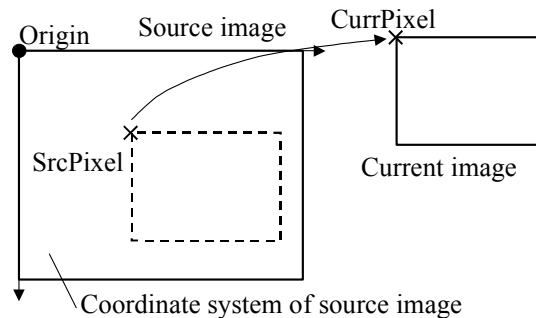


Figure 5 — CoordinateMapping (CurrPixel and SrcPixel)

Coord

This field, which is only present in the binary representation, indicates if SrcPixel or CoordPoint elements are used in the mapping. If Coord is set to 0 then the mapping between the source image and the current image is defined by SrcPixel elements. Otherwise, the mapping is defined by CoordPoint elements.

NumOfMappingFuncs

This field, which is only present in the binary representation, specifies the number of the following MappingFunc descriptions. MappingFunc's are described if the bilinear interpolation is insufficient.

MappingFuncLength

This field, which is only present in the binary representation, specifies the length of the MappingFunc element in bytes.

MappingFunc

This element is only applicable if CoordPoint elements are specified. It is needed if the specified mapping to the CoordPoint elements is not precise but only a rough approximation as it might be the case for real world coordinates used e.g. in maps. To allow a higher precision the MappingFunc specifies a functional expression in an ISO/IEC 9899 C format, using x and y according to the Pixel coordinates. Accordingly, the descriptor specifies how to map coordinate values between the Pixel and the CoordPoint elements:

- a mapping to the "x"/"y" coordinate of the CoordPoints is specified by a string starting with "x="/"y="
- the remaining part of the string consists of a functional expression in which "x" and "y" represent the x and y coordinates in the "SrcPixel" coordinates.

According to this rule, "x=" and "y=y+0.0001*x*100" represents a mapping from Pixels to CoordPoints by (a) shift of 100 and (b) by correcting the linear interpolation by "0.0001*x*x".

If MappingFunc is not specified, bilinear interpolation between the mapped points has to be considered as a precise mapping.

IntegratedCoordinateSystem

This element specifies an integrated coordinate system using the mappings (motions) from the first image (current image) to other images. The mapping from the descriptor coordinate system to the current image is specified by the LocalCoordinateSystem element (if this element is absent, the default coordinate system is used for the first image).

modelType

This attribute specifies the number of parameters the applied motion model consists of and with this it specifies the motion model using 3 bits. The possible values are: 2 (translational model), 4 (rotation/scaling model), 6 (affine model), 8 (planar perspective model), 12 (quadratic model). These motion models are already included in the Visual descriptors as part of the ParametricMotion descriptor. The bit codes for this field and the associated NumOfParams values are specified in Table 43.

xOrigin, yOrigin

These attributes specify the X,Y coordinates in the local coordinate system specified by the LocalCoordinateSystem mapping. They specify the origin of the motion model.

numOfMotionParamSets

This attribute specifies the total number of motion parameter sets (frames). The IntegratedCoordinateSystem captures information over a period of time. Therefore, also temporal reference information has to be provided. The time frame (a shot) of the video sequence is assumed to be provided by a higher order DS. As illustrated in Figure 6, sets (vectors) of motion parameters (MotionParams) are spread over this period, the total number is specified by the value of the attribute numOfMotionParamSets. Each set of motion parameters carries its own temporal information in TimeIncr, which specifies the frame it is related to. This provides high flexibility for temporal spacing of the frames used. It is possible to use for instance every frame, every fifth or even an irregular spacing.

TimeIncr

This element specifies the time between the first frame of the considered video sequence and the frame that the set of motion parameters refers to (see temporal reference system in Figure 6).

MotionParams

This element specifies a motion parameter value as defined for the ParametricMotion descriptor. The motion parameters describe motion at a certain time (as indicated by TimeIncr) with respect to the first frame of the considered video sequence.

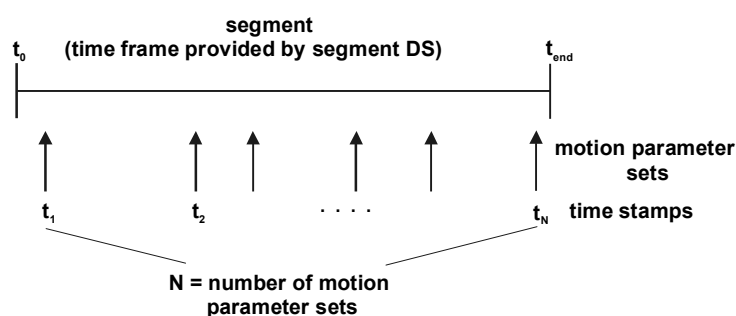


Figure 6 — Temporal reference system in relation to frame 0

5.6 Temporal interpolation

5.6.1 Introduction

This datatype specifies temporal interpolation using connected polynomials. This can be used to approximate multi-dimensional variable values that change with time, such as an object position in a video sequence. The descriptor size is usually much smaller than would be required to describe all the real values. In Figure 7, 25 real values are represented by five linear interpolation functions and two quadratic interpolation functions. The beginning of the temporal interpolation is always aligned to time 0.

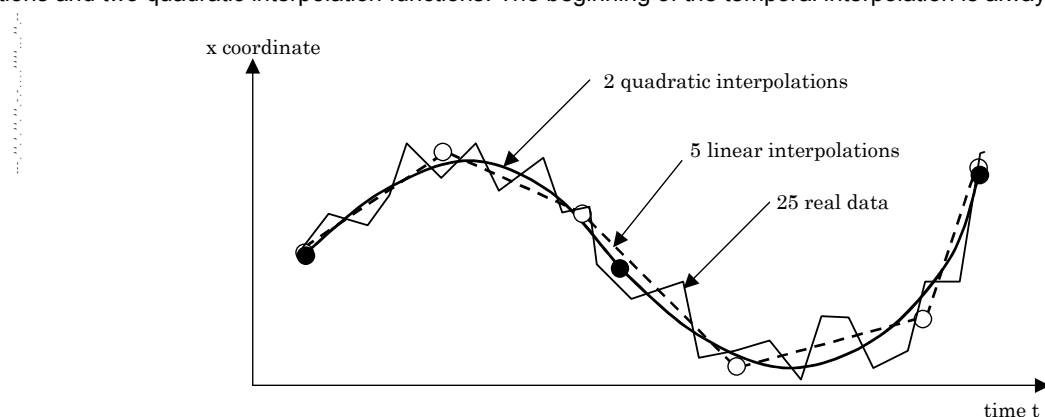


Figure 7 — Real data and interpolation functions

The dimension of the interpolated variable is specified by the Dimension field (except for the temporal element) and some examples are shown in Table 3. The connection point is called a "key point". The number of key points is described by the NumOfKeyPoints field. The time of key points is described by the WholeInterval element or an array of KeyTimePoint elements. WholeInterval is used only for constant intervals and the length of the interval is derived by dividing the specified duration by NumOfKeyPoints-1. The values of the key points are described by an array of KeyValue element, with NumOfKeyPoints values for each dimension.

Table 3 — Examples of the value of dimension

D or DS using TemporalInterpolation	Value of Dimension
2D MotionTrajectory	2
3D MotionTrajectory	3
ParameterTrajectory (Translational Model)	2
ParameterTrajectory (Affine Transformation Model)	6
ParameterTrajectory (Parabolic Model)	12

TemporalInterpolation can use two types of interpolation functions: first order and second order polynomials. The type of the interpolation function is indicated by the type attribute. When a first order polynomial is used (type="firstOrder"), the function can be calculated from the key points. When a second order polynomial is used (type="secondOrder"), the second order coefficient is described by the param attribute. The relation between type, interpolation functions, key points and param is shown in Table 4. In the table, t_a and t_b ($t_a < t_b$) represent the times of two successive key points, and f_a and f_b indicate values of key points at t_a and t_b . In the calculation of interpolation functions, the values of time are expressed in seconds. c_2 is the value of param and c_1 is the value derived from the constraints and other parameter values. The constraint ensures that all interpolation function pass through the key points at both ends. If reliable interpolation functions are not obtained in the extraction process, type is set to "notDetermined". In this case, the interpolation functions are not specified and reliable values between successive key points cannot be determined from the description.

Table 4 — Interpolation function specified by type and param

type	Interpolation Function Form	param	Constraint
notDetermined	(none)	(none)	(not applicable)
startPoint	(none)	(none)	(not applicable)
firstOrder	$f(t) = f_a + c_1(t - t_a)$	(none)	$c_1 = \frac{f_b - f_a}{t_b - t_a}$
secondOrder	$f(t) = f_a + c_1(t - t_a) + c_2(t - t_a)^2$	c_2	$c_1 = \frac{f_b - f_a}{t_b - t_a} - c_2(t_b - t_a)$

5.6.2 DDL representation syntax

```
<complexType name="TemporalInterpolationType" final="#all">
  <sequence>
    <choice>
      <element name="WholeInterval">
        <complexType>
          <choice>
            <element name="MediaDuration"
              type="mpeg7:mediaDurationType"/>
            <element name="MediaIncrDuration"
              type="mpeg7:MediaIncrDurationType"/>
          </choice>
        </complexType>
      </element>
      <element name="KeyTimePoint">
        <complexType>
          <choice>
            <element name="MediaTimePoint"
```

```

        type="mpeg7:mediaTimePointType"
        minOccurs="2" maxOccurs="65535"/>
      <element name="MediaRelTimePoint"
        type="mpeg7:MediaRelTimePointType"
        minOccurs="2" maxOccurs="65535"/>
      <element name="MediaRelIncrTimePoint"
        type="mpeg7:MediaRelIncrTimePointType"
        minOccurs="2" maxOccurs="65535"/>
    </choice>
  </complexType>
</element>
</choice>
<element name="InterpolationFunctions" minOccurs="1" maxOccurs="15">
  <complexType>
    <sequence>
      <element name="KeyValue" minOccurs="2" maxOccurs="65535">
        <complexType>
          <simpleContent>
            <extension base="float">
              <attribute name="type" use="optional"
                default="firstOrder">
                <simpleType>
                  <restriction base="string">
                    <enumeration value="startPoint"/>
                    <enumeration value="firstOrder"/>
                    <enumeration value="secondOrder"/>
                    <enumeration
                      value="notDetermined"/>
                  </restriction>
                </simpleType>
              </attribute>
              <attribute name="param" type="float"
                use="optional" default="0.0"/>
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
</sequence>
</complexType>

```

5.6.3 Binary representation syntax

TemporalInterpolation {	Number of bits	Mnemonic
NumOfKeyPoints	16	uimsbf
ConstantTimeInterval	1	bslbf
QuantizationFlag	1	bslbf
if (ConstantTimeInterval) {		
WholeIntervaleDataType	1	bslbf
if (!WholeIntervalDataType) {		
MediaDuration	See annex B	mediaDurationType
} else {		
MediaIncrDuration	See annex B	MediaIncrDurationType
} else {		
KeyTimePointDataType	2	bslbf
if(KeyTimePointDataType==00) {		
for(j=0; j<NumOfKeyPoints; j++)		
MediaTimePoint	See annex B	mediaTimePointType
} else if(KeyTimePointDataType==01) {		
for(j=0; j<NumOfKeyPoints; j++)		
MediaRelTimePoint	See annex B	MediaRelTimePointType
} else if (KeyTimePointDataType==10) {		
for(j=0; j<NumOfKeyPoints; j++)		
MediaRelIncrTimePoint	See annex B	MediaRelIncrTimePointType
}		
}		
Dimension	4	uimsbf
for(j=0; j<Dimension; j++) {		
DefaultFunction	1	bslbf
for(k=0; k<NumOfKeyPoints; k++) {		
if(!DefaultFunction) {		
type	2	bslbf
if(type==10) {		
param	32	fsbf
}		
}		
if(!QuantizationFlag) {		
KeyValue	32	fsbf
} else {		
QuantizedKeyValue	if (j==0) XRepr else if (j==1) YRepr	uimsbf
}		
}		
}		
}		

5.6.4 Descriptor components semantics

NumOfKeyPoints

This field, which is only present in the binary representation, specifies the number of sampled positions, denoted as key points, used for the knots of interpolation functions: from 2 to 65535. In the DDL representation the corresponding value can be obtained by counting the number of KeyValue elements in any InterpolationFunctions instance. The size of InterpolationFunctions must be the same for all its instances in a single descriptor. When the time points are described by the KeyTimePoint element, the number of MediaTimePoint (or MediaRelTimePoint or MediaRelIncrTimePoint) elements must also be equal to NumOfKeyPoints.

ConstantTimeInterval

This field, which is only present in the binary representation, specifies the type of key time point description. If it is set to "0", the intervals between key points are not constant and each time point is described by KeyTimePoints. Otherwise, the intervals are constant and specified by WholeInterval. In this case, the length of these intervals is calculated as $\text{WholeInterval}/(\text{NumOfKeyPoints}-1)$.

QuantizationFlag

This field, which is only present in the binary representation, specifies if KeyValue is encoded as a floating-point number or quantized. If it is set to "0", floating-point representation is used, otherwise quantization to integer representation is performed. When TemporalInterpolation is used in the SpatioTemporalLocator, quantization of key value is not allowed and this field must be set to "0".

WholeIntervalDataType

This field, which is only present in the binary representation, specifies the type of WholeInterval. The meaning of the values is specified in Table 5.

Table 5 — The meaning of WholeIntervalDataType

WholeIntervalDataType	Meaning
"0"	mediaDurationType
"1"	MediaIncrDurationType

WholeInterval

This element specifies the whole temporal interval of interest. If this field is defined, the time interval between each successive pair of key points is constant. The beginning of the whole interval is always fixed to 0, and the length of these intervals is calculated by $\text{WholeInterval}/(\text{NumOfKeyPoint}-1)$. If the time intervals are not constant, all key time should be specified explicitly using the KeyTimePoint element. WholeInterval supports two types, MediaDurationType and MediaIncrDurationType. In the binary representation, WholeIntervalDataType specifies the type.

MediaDuration

This element specifies the value of WholeInterval using mediaDurationType (see ISO/IEC 15938-5).

MediaIncrDuration

This element specifies the value of WholeInterval using MediaIncrDurationType (see ISO/IEC 15938-5).

KeyTimePointDataType

This field, which is only present in the binary representation, specifies the type of KeyTimePoint. The meaning of the values of this field is specified in Table 6.

Table 6 — The meaning of KeyTimePointDataType

KeyTimePointDataType	Meaning
"00"	mediaTimePointType
"01"	MediaRelTimePointType
"10"	MediaRelIncrTimePointType
"11"	prohibited

KeyTimePoint

This element specifies the series of time points for each key point. The number of time points in the series must be equal to the value of NumOfKeyPoints. Time points are always sorted in increasing order. KeyTimePoint supports three types: mediaTimePointType, MediaRelTimePointType and MediaRelIncrTimePointType. In the binary representation, KeyTimePointDataType specifies the type.

MediaTimePoint

This element specifies the KeyTimePoint using mediaTimePointType.

MediaRelTimePoint

This element specifies the KeyTimePoint using MediaRelTimePointType.

MediaRelIncrTimePoint

This element specifies the KeyTimePoint using MediaRelIncrTimePointType.

InterpolationFunctions

This element specifies interpolation functions in each dimension. The size of this element must be equal to the value of the Dimension field.

Dimension

This field, which is only present in the binary representation, specifies the dimension (number of components) of interpolated values. The allowed range is from 1 to 15. Table 3 shows examples of the value of dimension for some descriptors and tools defined in ISO/IEC 15938-3. In the DDL representation, the value of dimension is obtained by counting the number of InterpolationFunctions.

KeyValue

This element specifies the value of each key point. The number of occurrences of KeyValue in a single InterpolationFunctions must be equal to the value of NumOfKeyPoints. The KeyValue elements must appear in increasing time order.

QuantizedKeyValue

This field, which is only present in the binary representation, specifies the value of each key point using integer encoding obtained by quantization of the floating-point key values. It can only be used for 2-dimensional values. The number of bits used is specified by XRepr for the first dimension ($j=0$) and YRepr for the second dimension. The values of XRepr and YRepr are obtained from the description tool that uses TemporalInterpolation. In case of MotionTrajectory descriptor, for example, these are the values of xRepr/yRepr attributes of the descriptor itself or the referenced Spatial2DCoordinateSystem. The integer value is obtained by uniform quantization of the definition interval. For example, if Units is "pictureWidth" and xRepr is 3, then key value for $j=0$ takes values between 0 and 1 which should be quantized to 3 bits by an 8-bin uniform quantization (e.g. 0.2 would be encoded as "001"). The number of occurrences of QuantizedKeyValue in a single InterpolationFunctions must be equal to the value of NumOfKeyPoints. The QuantizedKeyValue elements must appear in increasing time order.

DefaultFunction

This field, which is only present in the binary representation, indicates if the default interpolation function is used. If DefaultFunction is set to 1 then the default interpolation function (linear interpolation) is used in the current dimension.

type

This attribute specifies the type of the key point. If this is "startPoint", the key point is the starting point of the interpolation. If it is "firstOrder" or "secondOrder", the key point follows the linear or quadratic interpolation function. If it is "notDetermined", the key point is not the starting point of the interval but no interpolation function is defined between the key point and the previous key point. If it is absent, the value of firstOrder (linear interpolation) is used as a default value. In the binary representation, the relation between the binary encoding and the meaning of type is given in Table 7.

Table 7 — The meaning of type

type	Meaning
"00"	startPoint
"01"	firstOrder
"10"	secondOrder
"11"	notDetermined

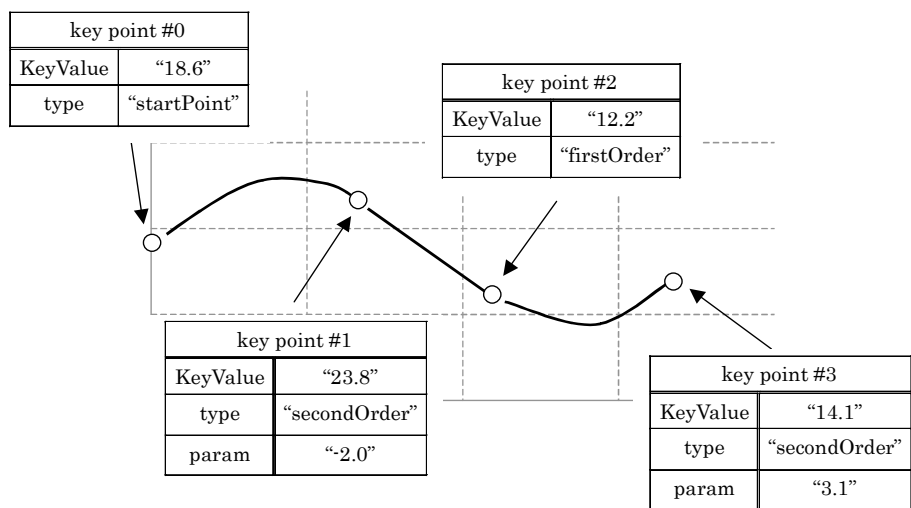


Figure 8 — Example of Key, Value, type and param

param

This attribute specifies the coefficient of the interpolation function when relevant, i.e. c_2 (see Table 4). This attribute is meaningful only when the function type is "secondOrder" and specifies the second-order coefficient of the interpolation function.

6 Color

6.1 Introduction

This clause provides four color descriptors to represent different aspects of color features: representative colors (DominantColor), color distribution (ScalableColor), global spatial distribution of colors (ColorLayout), and local spatial distribution (ColorStructure). It also contains the GoFGoPColor descriptor, which is an extension of ScalableColor to group of frames or pictures. Finally, two supporting tools are defined, ColorSpace and ColorQuantization, which are used in the DominantColor descriptor. All the color descriptors can be extracted from arbitrarily shape regions.

6.2 Color space

6.2.1 Introduction

This datatype specifies the color space in which the color descriptors are expressed. The following color spaces are supported:

- RGB
- YCbCr
- HSV
- HMMD
- Linear transformation matrix with reference to RGB
- Monochrome

A flag is also provided to allow indication of the availability of a color primaries reference.

6.2.2 DDL representation syntax

```
<complexType name="ColorSpaceType" final="#all">
  <sequence>
    <element name="ColorTransMat" minOccurs="0">
      <simpleType>
        <restriction base="string"/>
      </restriction>
    </element>
  </sequence>
</complexType>
```

```

        <list itemType="mpeg7:unsignedl6"/>
      </simpleType>
      <length value="9"/>
    </restriction>
  </simpleType>
</element>
</sequence>
<attribute name="colorReferenceFlag" type="boolean"
  use="optional" default="false"/>
<attribute name="type" use="required">
  <simpleType>
    <restriction base="string">
      <enumeration value="RGB"/>
      <enumeration value="YCbCr"/>
      <enumeration value="HSV"/>
      <enumeration value="HMMD"/>
      <enumeration value="LinearMatrix"/>
      <enumeration value="Monochrome"/>
    </restriction>
  </simpleType>
</attribute>
</complexType>

```

6.2.3 Binary representation syntax

ColorSpace {	Number of bits	Mnemonic
colorReferenceFlag	1	bslbf
type	4	bslbf
if (type=='LinearMatrix') {		
for(j=0; j<3; j++) {		
for(k=0; k<3; k++) {		
ColorTransMat[j][k]	16	uimsbf
}		
}		
}		

6.2.4 Descriptor components semantics

colorReferenceFlag

This attribute indicates the presence of the RGB primaries and gamma specification. If colorReferenceFlag is set to 0, no reference RGB primaries or gamma value are available. Otherwise, the color values are normalized using the following specification.

The CIE tristimulus chromaticity values of the RGB primaries and the illuminant white point are specified in Table 8.

Table 8 — CIE chromaticities for reference RGB primaries and illuminant white

	Red	Green	Blue	White point (D65)
<i>x</i>	0.6400	0.3000	0.1500	0.3127
<i>y</i>	0.3300	0.6000	0.0600	0.3290
<i>z</i>	0.0300	0.1000	0.7900	0.3583

The gamma correction used to obtain the color values is specified using the following formula, where V_L is the linear input value of any captured component (R_L , G_L , B_L) and V denotes the nonlinear resulting value.

$$\text{if } V_L \leq 0.00304 \text{ then } V = 12.92 \times V_L$$

$$\text{otherwise } V = 1.055 \times V_L^{(1.0 \div 2.4)} - 0.055$$

type

This attribute specifies the color space.

- type “RGB” is an RGB color space with or without reference primaries (depending on colorReferenceFlag). As noted above the R, G and B values are in the range [0,1].
- type “YCbCr” is expressed by a linear transformation:

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\ Cb &= -0.169 \cdot R - 0.331 \cdot G + 0.500 \cdot B \\ Cr &= 0.500 \cdot R - 0.419 \cdot G - 0.081 \cdot B \end{aligned}$$

It follows that the Y component values are in the range [0,1] and Cb and Cr component values are in the range [-0.5,0.5].

- type “HSV” specifies a nonlinear transformation. It consists of the Value representing the lightness of color, the Saturation indicating the degree of colorfulness, and the Hue (H) representing the dominant spectral tone of the color, denoted by an angle from 0 to 360 degrees. Their values are derived from the normalized RGB values (ranging from 0 to 1) as follows:

```
Max = max(R, G, B);
Min = min(R, G, B);
Value = max(R, G, B);
if( Max == 0 ) then
    Saturation = 0;
else
    Saturation = (Max-Min)/Max;
if( Max == Min ) Hue=0; /* achromatic */
otherwise:
if( Max == R && G >= B )
    Hue = 60*(G-B) / (Max-Min)
else if( Max == R && G < B )
    Hue = 360 + 60*(G-B) / (Max-Min)
else if( G == Max )
    Hue = 60*(2.0 + (B-R) / (Max-Min))
else
    Hue = 60*(4.0 + (R-G) / (Max-Min))
```

The Saturation component takes values in the range [0,1], the Value component has values in the range [0,1] and the Hue component has values in the range [0,360].

The HSV color space can best be interpreted as a cylinder, where the Hue component approximately represents the angle in degrees, as shown in Figure 9.

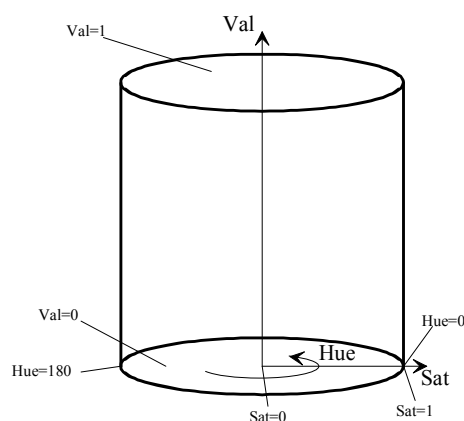


Figure 9 — Cylinder representing the limits of the HSV color space

- type “HMMD” is defined by a nonlinear, reversible transformation from the RGB color space. There are five distinct attributes (components) in the HMMD color space, however only three of them (Hue, Max, Min, or Hue, Diff, Sum) are sufficient to define the color space. The five attributes can be characterized as follows:

- Hue: the same as in HSV.
- Max: indicates how much black color it has, giving a flavor of shade or blackness.
- Min: indicates how much white color it has, giving a flavor of tint or whiteness.
- Diff: indicates how much gray it contains and how close to the pure color, giving a flavor of tone or colorfulness.
- Sum: simulates the brightness of the color.

The transformations for Max, Min and Hue are the same as the equations for Min, Max and Hue in HSV color space. The transformations for Diff and Sum have the following form:

$$\begin{aligned} \text{Diff} &= \text{Max} - \text{Min}; \\ \text{Sum} &= (\text{Max} + \text{Min}) / 2; \end{aligned}$$

The Max, Min and Sum components have values in the range [0,1] and the Diff component has values in the range [0,1]. The Hue component takes values in the range [0,360].

The HMMD color space has a double cone appearance as shown in Figure 10.

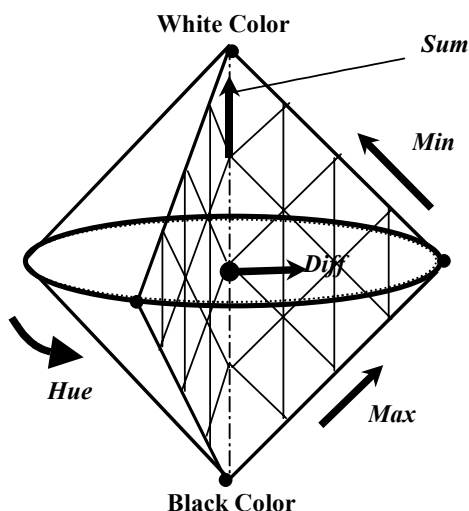


Figure 10 — Appearance of the HMMD color space

- type “LinearMatrix” is specified by a linear matrix transformation from the components of the RGB color space:

$$\begin{aligned} C1 &= \text{ColorTransMat}[0][0]*R + \text{ColorTransMat}[0][1]*G + \text{ColorTransMat}[0][2]*B \\ C2 &= \text{ColorTransMat}[1][0]*R + \text{ColorTransMat}[1][1]*G + \text{ColorTransMat}[1][2]*B \\ C3 &= \text{ColorTransMat}[2][0]*R + \text{ColorTransMat}[2][1]*G + \text{ColorTransMat}[2][2]*B. \end{aligned}$$

- type “Monochrome” is expressed by a linear transformation:

$$Y = 0.299*R + 0.587*G + 0.114*B$$

from the components of the RGB color space. The resulting values lie in the range [0,1].

Within a description, the components of a color space must be ordered in a specific sequence. For the color spaces described, the sequence shown in Table 9 shall be valid.

Table 9 — Ordering of color components

type	Component1	Component2	Component3	Component4	Component5
RGB	R	G	B	N/A	N/A
YCbCr	Y	Cb	Cr	N/A	N/A
HSV	H	S	V	N/A	N/A
HMMD	H	Max	Min	Diff	Sum
LinearMatrix	C1	C2	C3	N/A	N/A
Monochrome	Y	N/A	N/A	N/A	N/A

The binary representation of this attribute is specified in Table 10.

Table 10 — Meaning of type

type	Meaning
0000	RGB
0001	YCbCr
0010	HSV
0011	HMMD
0100	LinearMatrix
0101	Monochrome
0110-1111	Reserved

ColorTransMat

This element specifies a color transformation matrix. If the color space is expressed by the linear matrix transformation, the values of this matrix define the transformation with reference to the RGB color space. Matrix values are within the range of -1 to 1, and are expressed as 16 bit signed integers. The scaling factor between the integer value and the floating point value (from -1 to 1) is defined by the following equation:

$$V_{fp} = V_{int} * 2^{-14}.$$

6.3 Color quantization

6.3.1 Introduction

This descriptor defines the uniform quantization of a color space. It provides a mapping from the floating point values specified in subclause 6.2 to an integer representation in which each component is linearly scaled to the integer range [0, NumOfBins-1], where NumOfBins is the number of quantization levels of each color component (see below).

6.3.2 DDL representation syntax

```
<complexType name="ColorQuantizationType" final="#all">
  <sequence minOccurs="1" maxOccurs="3">
    <element name="Component">
      <simpleType>
        <restriction base="string">
          <enumeration value="R"/>
          <enumeration value="G"/>
          <enumeration value="B"/>
          <enumeration value="Y"/>
          <enumeration value="Cb"/>
          <enumeration value="Cr"/>
          <enumeration value="H"/>
          <enumeration value="S"/>
          <enumeration value="V"/>
          <enumeration value="Max"/>
          <enumeration value="Min"/>
          <enumeration value="Diff"/>
          <enumeration value="Sum"/>
        </restriction>
      </simpleType>
    </element>
  </sequence>
</complexType>
```

```

    <element name="NumOfBins" type="mpeg7:unsigned12"/>
  </sequence>
</complexType>

```

6.3.3 Binary representation syntax

ColorQuantization {	Number of bits	Mnemonic
for(k=0; k<NumOfComponents; k++) {		
Component[k]	5	bslbf
NumOfBins[k]	12	uimsbf
}		
}		

6.3.4 Descriptor components semantics

Component

This element specifies the color component to be quantized, allowing the components to be quantized in an arbitrary order. NumOfComponents is set to 1 when the color space quantized by this descriptor is monochrome and to 3 otherwise. The allowed color components combinations used for each color space are defined in Table 9. For the HMMD color space, the allowed combinations are {H,Max,Min} and {H,Diff,Sum}. In the binary representation, the meaning of the codes is specified in Table 11.

Table 11 — The meaning of Component

Component	Meaning
00000	R
00001	G
00010	B
00011	Y
00100	Cb
00101	Cr
00110	H
00111	S
01000	V
01001	Max
01010	Min
01011	Diff
01100	Sum
01101-11111	Reserved

NumOfBins

This element specifies the number of bins for the color component. In the case of a uniform quantizer, the normalized value range A of the color component is divided into NumOfBins equal intervals of width $A/\text{NumOfBins}$ each.

6.3.5 Decoding

The decoder decodes a color index that points to a specific color value.

Reconstruction of component k ($k=0, \dots, \text{NumOfComponents}-1$) with known **max_value** and **min_value** of each normalized component:

```

value_range = max_value - min_value;
comp_rec[k] = (comp_index[k]+0.5)*value_range/NumOfBins[k] - min_value;

```

In the above, comp_index[k] is a quantized index of the k -th color component, which is an unsigned integer between 0 and $2^{\text{NumOfBins}[k]}-1$. The color values residing on the boundaries between bins are assigned to the higher color index.

6.4 Dominant color

6.4.1 Introduction

This descriptor specifies a set of dominant colors in an arbitrarily shaped region. It targets content-based retrieval for color, either for the whole image or for an arbitrary region (rectangular or irregular).

6.4.2 DDL representation syntax

```
<complexType name="DominantColorType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <sequence>
        <element name="ColorSpace" type="mpeg7:ColorSpaceType"
          minOccurs="0"/>
        <element name="ColorQuantization"
          type="mpeg7:ColorQuantizationType" minOccurs="0"/>
        <element name="SpatialCoherency" type="mpeg7:unsigned5"/>
        <element name="Value" minOccurs="1" maxOccurs="8">
          <complexType>
            <sequence>
              <element name="Percentage" type="mpeg7:unsigned5"/>
              <element name="Index">
                <simpleType>
                  <restriction>
                    <simpleType>
                      <list itemType="mpeg7:unsigned12"/>
                    </simpleType>
                    <length value="3"/>
                  </restriction>
                </simpleType>
              </element>
              <element name="ColorVariance" minOccurs="0">
                <simpleType>
                  <restriction>
                    <simpleType>
                      <list itemType="mpeg7:unsigned1"/>
                    </simpleType>
                    <length value="3"/>
                  </restriction>
                </simpleType>
              </element>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.4.3 Binary representation syntax

DominantColor {	Number of bits	Mnemonic
Size	3	uimsbf
ColorSpacePresent	1	bslbf
if(ColorSpacePresent) {		
ColorSpace	See subclause 6.2.3	ColorSpaceType
}		
ColorQuantizationPresent	1	bslbf
if(ColorQuantizationPresent) {		
ColorQuantization	See subclause 6.2.3	ColorQuantizationType
}		
VariancePresent	1	bslbf
SpatialCoherency	5	uimsbf
for(k=0; k<Size; k++) {		
Percentage	5	uimsbf
for(m=0; m<3; m++) {		
Index	1-12	uimsbf
if(VariancePresent) {		
ColorVariance	1	uimsbf
}		
}		
}		
}		
}		

6.4.4 Descriptor components semantics

Size

This field, which is only present in the binary representation, specifies the number of dominant colors in the region. The maximum allowed number of dominant colors is 8, the minimum number of dominant colors is 1. The following mapping of bit patterns is used: 000->1, ..., 111->8.

ColorSpacePresent

This field, which is only present in the binary representation, indicates the presence of the ColorSpace element. If set to 0, ColorSpace is not present and RGB color space is used.

ColorSpace

This element is defined in subclause 6.2.

ColorQuantizationPresent

This element, which is only present in the binary representation, signals the presence of the ColorQuantization element. If set to 0, ColorQuantization is not present and uniform color quantization of the components to 5 bits is used.

ColorQuantization

This element is specified in subclause 6.3.

VariancePresent

This field, which is only present in the binary representation, indicates the presence of the color variances in the descriptor.

SpatialCoherency

This element specifies the spatial coherency of the dominant colors described by the descriptor. It is computed as a single value by the weighted sum of per-dominant-color spatial coherencies. The weight is proportional to the number of pixels corresponding to each dominant color. Spatial coherency per dominant color captures how coherent the pixels corresponding to the dominant color are and whether they appear to be a solid color in the given image region (See Figure 11, where red pixels in the left image have

low spatial coherency and in the right image high spatial coherency). Spatial coherency per dominant color is computed by the normalized average connectivity (8-connectedness) for the corresponding dominant color pixels.

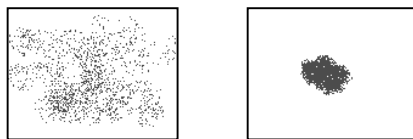


Figure 11 — Examples of low (a) and high (b) spatial coherency of color

The weighted sum of per-dominant-color spatial coherencies is normalized from 0 to 1, then non-uniformly quantized to the range from 1 to 31 as follows. Normalized values less than 0.7 are set to 1, while values between 0.7 to 1 are uniformly quantized to the range 2 to 31. 0 is used to signal that this element is not computed (note that if it is not computed it does not mean that the spatial coherency is low).

Value

This element specifies an array of elements that hold percentages and values of colors in a visual item. The array elements consist of Percentage, ColorValueIndex and ColorVariance.

Percentage

This element specifies the percentage of pixels that have the associated color value. The percentage value is uniformly quantized to 5 bits with 0 corresponding to 0 percentage and 31 corresponding to 100%. Note that the sum of the Percentage values for a given visual item does not have to be equal to 100%.

Index

This element specifies the index of the dominant color in the selected color space as defined in ColorQuantization. The number of bits for each component is derived from the ColorQuantization element.

ColorVariance

This element specifies an integer array containing the value of the variance of color values of pixels corresponding to the dominant color in the selected color space, i.e.

$$CV_j = \frac{1}{N} \sum_{k=0}^{N-1} (m_j - p_{kj})^2$$

where j indexes the color component, m_j is j -th component of the dominant color, p_{kj} is j -th component of the k -th pixel value, and the summation is over N pixels corresponding to the dominant color under consideration.

The dimension of this vector depends on the selected color space. Each component is quantized to 1 bit, with “0” corresponding to low variance and “1” corresponding to high variance. The quantization threshold is equal to 0.005 of the squared color component value range.

6.5 Scalable color

6.5.1 Introduction

This descriptor specifies a color histogram in the HSV color space, which is encoded by a Haar transform. Its binary representation is scalable in terms of bin numbers and bit representation accuracy over a broad range of data rates.

6.5.2 DDL representation syntax

```
<complexType name="ScalableColorType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <sequence>
        <element name="Coeff" type="mpeg7:integerVector"/>
      </sequence>
      <attribute name="numOfCoeff" use="required">
        <simpleType>
          <restriction base="integer">
            <enumeration value="16"/>
            <enumeration value="32"/>
            <enumeration value="64"/>
            <enumeration value="128"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

```

        <enumeration value="256"/>
    </restriction>
</simpleType>
</attribute>
<attribute name="numOfBitplanesDiscarded" use="required">
    <simpleType>
        <restriction base="integer">
            <enumeration value="0"/>
            <enumeration value="1"/>
            <enumeration value="2"/>
            <enumeration value="3"/>
            <enumeration value="4"/>
            <enumeration value="6"/>
            <enumeration value="8"/>
        </restriction>
    </simpleType>
</attribute>
</extension>
</complexContent>
</complexType>

```

6.5.3 Binary representation syntax

ScalableColor {	Number of bits	Mnemonic
numOfCoeff	3	bslbf
numOfBitplanesDiscarded	3	bslbf
for(k=0; k<numOfCoeff; k++) {		
CoefficientSign	1	bslbf
}		
for(k=0; k<8-numOfBitplanesDiscarded; k++) {		
Bitplane[k]	BitplaneSize	bslbf
}		
}		

6.5.4 Descriptor components semantics

numOfCoeff

This attribute specifies the number of coefficients used in the scalable representation. Possible values are: 16, 32, 64, 128, and 256. Its binary semantics are specified in Table 12

Table 12 — The meaning of numOfCoeff

numOfCoeff	Meaning
000	16
001	32
010	64
011	128
100	256
101-111	Reserved

numOfBitplanesDiscarded

This attribute specifies the number of bitplanes discarded in the scalable representation for each coefficient. Possible values are: 0, 1, 2, 3, 4, 6, 8. If the number of bits allocated to a specific coefficient is less than numOfBitplanesDiscarded, only the sign bit of this coefficient is retained. The binary semantics of this field are defined in Table 13

Table 13 — The meaning of numOfBitplanesDiscarded

numOfBitplanesDiscarded	Meaning
000	0
001	1
010	2
011	3
100	4
101	6
110	8
111	Reserved

Coeff

This element specifies the Haar transform coefficients. In the DDL representation, coefficients are expressed as signed integers, where the precision of the quantized representation follows from the numOfBitplanesDiscarded value. In the binary representation, coefficients are expressed through their sign and optionally their amplitude.

CoefficientSign

This element specifies the sign of a Haar coefficient representing a component of a transformed color histogram in the HSV color space. The positive sign is encoded as “1” and the negative sign as “0”.

Bitplane[k]

This represents one bitplane of all the coefficients that are encoded in this specific bitplane. Because each coefficient is encoded with a different number of bits, the size of each bitplane increases starting from the most significant bits (MSB=8) to the least significant bits (LSB=1). The first bitplane contains the MSB of all coefficients encoded on 8 bits, while the last bitplane contains the LSB of all coefficients encoded. Table 14 shows the size in bits (or equivalently in number of coefficients represented) of each bitplane, depending on the NumberOfCoefficients retained in the scalable color representation. Note that following the notation of 6.5.3, Bitplane[0] is the most significant bitplane, while Bitplane[7] contains the least significant bit of each coefficient.

Table 14 — Sizes of successive bitplanes, depending on the NumberOfCoefficients

Bitplanesize (significance)	Number Of Coefficients=16	Number Of Coefficients=32	Number Of Coefficients=64	Number Of Coefficients=128	Number Of Coefficients=256
Bitplane[0]	3	3	3	3	3
Bitplane[1]	5	5	5	5	5
Bitplane[2]	14	17	17	17	17
Bitplane[3]	16	24	25	25	25
Bitplane[4]	16	30	51	82	91
Bitplane[5]	16	31	63	102	138
Bitplane[6]	16	32	64	122	227
Bitplane[7]	16	32	64	128	256

Corresponding to Table 14, Bitplane[k] is assigned rank $r=8-k$. If for a coefficient $BN < r$ in Table 15, the coefficient is skipped in Bitplane[k].

Within the bitplanes the coefficients, if present, are ordered as follows with respect to the coefficient index CI in Table 15:

- Coefficients 0-15:
CI: 0, 4, 8, 12, 32, 36, 40, 44, 128, 132, 136, 140, 160, 164, 168, 172
- Coefficients 16-31:
CI: 2, 6, 10, 14, 34, 38, 42, 46, 130, 134, 138, 142, 162, 166, 170, 174
- Coefficients 32-63:
CI: 64, 66, 68, 70, 72, 74, 76, 78, 96, 98, 100, 102, 104, 106, 108, 110, 192, 194, 196, 198, 200, 202, 204, 206, 224, 226, 228, 230, 232, 234, 236, 238
- Coefficients 64-127:
CI: 16, 18, 20, 22, 24, 26, 28, 30, 48, 50, 52, 54, 56, 58, 60, 62, 80, 82, 84, 86, 88, 90, 92, 94, 112, 114, 116, 118, 120, 122, 124, 126, 144, 146, 148, 150, 152, 154, 156, 158, 176, 178, 180, 182, 184, 186, 188, 190, 208, 210, 212, 214, 216, 218, 220, 222, 240, 242, 244, 246, 248, 250, 252, 254
- Coefficients 128-255:
CI: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255.

For instance in Bitplane[1] with 16 coefficients, 5 bits expressing this bitplane for the coefficients (in sequence) CI=0, 4, 8, 12, 128 would be contained.

Bit representation of Coefficients

For each coefficient, what is encoded is the magnitude of a Haar coefficient representing a component of a transformed color histogram in the HSV color space. The Coefficient Index CI, the Number Of Bits allocated to the coefficients BN, as well as Offset values QO necessary for reconstruction of histograms, are given in Table 15. Reconstruction of one coefficient is defined as $CR=QO+SIGN*AMPLITUDE$. The AMPLITUDE value is constituted from all the bitplanes that are available, in their respective significance position. All bit values that are not available shall be supplemented by zero values.

Table 15 — Indexing, bit allocation and quantizer offsets of Haar coefficients

CI	BN	QO	CI	BN	QO	CI	BN	QO	CI	BN	QO
0	8	217	64	4	3	128	7	-29	192	4	-5
1	3	-1	65	3	1	129	4	1	193	3	0
2	6	4	66	5	3	130	6	0	194	4	-4
3	2	0	67	3	0	131	3	-1	195	3	-1
4	8	-54	68	4	0	132	6	-22	196	4	-2
5	2	-1	69	2	-1	133	2	0	197	2	0
6	5	-1	70	4	-1	134	4	0	198	4	-1
7	2	0	71	2	0	135	2	0	199	2	0
8	8	-71	72	4	2	136	6	-14	200	4	-3
9	2	0	73	3	0	137	3	0	201	3	0
10	5	-5	74	4	0	138	5	-5	202	4	-1
11	2	0	75	2	0	139	2	0	203	2	0
12	7	-27	76	4	0	140	6	-8	204	3	-1
13	2	0	77	2	0	141	2	0	205	2	0
14	4	3	78	4	0	142	4	1	206	4	-1
15	2	0	79	2	0	143	2	0	207	2	0
16	4	-1	80	4	0	144	4	0	208	4	-1
17	3	1	81	4	2	145	4	2	209	3	1
18	4	1	82	4	-1	146	4	-2	210	4	-2
19	3	0	83	4	-1	147	4	-1	211	3	-1
20	4	0	84	4	-3	148	4	-3	212	4	-2
21	2	-1	85	3	-1	149	2	0	213	2	0
22	4	-1	86	4	-3	150	4	-3	214	4	-2
23	2	0	87	2	0	151	2	0	215	1	0
24	3	0	88	4	-1	152	4	-1	216	4	-1
25	3	0	89	4	1	153	4	1	217	3	1
26	3	-1	90	4	-2	154	4	-2	218	4	-2
27	2	0	91	2	-1	155	2	0	219	2	0
28	2	-1	92	3	-2	156	3	-1	220	3	-1
29	1	0	93	1	0	157	1	0	221	1	0
30	3	-1	94	4	-2	158	4	-2	222	3	-1
31	2	0	95	2	0	159	2	0	223	2	0
32	6	-22	96	4	0	160	6	-22	224	4	-2
33	3	1	97	4	1	161	3	1	225	2	0
34	6	0	98	4	-3	162	5	-2	226	4	-1
35	3	0	99	4	-1	163	3	0	227	2	0
36	6	-19	100	4	-3	164	5	-11	228	2	0
37	2	0	101	2	0	165	2	0	229	1	0
38	4	-2	102	4	-2	166	2	0	230	2	0
39	2	0	103	2	0	167	1	0	231	1	0
40	6	-14	104	4	-2	168	6	-13	232	4	-1
41	3	0	105	3	1	169	3	1	233	3	0
42	5	-2	106	4	-2	170	4	-1	234	2	0
43	2	-1	107	2	0	171	2	0	235	2	0
44	6	-9	108	2	0	172	5	-6	236	2	0
45	1	0	109	1	0	173	1	0	237	1	0
46	4	2	110	4	-1	174	3	0	238	2	0
47	2	0	111	2	0	175	2	0	239	1	0
48	3	0	112	4	-2	176	4	-1	240	3	-1
49	3	1	113	3	0	177	2	0	241	2	0
50	3	-1	114	4	-2	178	3	-1	242	2	0

51	2	0	115	3	0	179	2	0	243	2	0
52	3	0	116	2	0	180	2	0	244	1	0
53	2	0	117	2	0	181	1	0	245	1	0
54	2	0	118	2	0	182	1	0	246	1	0
55	2	0	119	1	0	183	1	0	247	1	0
56	3	-1	120	3	-1	184	3	-1	248	3	0
57	2	0	121	2	0	185	2	0	249	2	0
58	3	0	122	2	0	186	2	0	250	2	0
59	2	-1	123	2	0	187	2	0	251	2	0
60	2	0	124	2	0	188	1	0	252	1	0
61	1	0	125	1	0	189	1	0	253	1	0
62	3	1	126	2	0	190	2	0	254	1	0
63	2	1	127	2	0	191	2	0	255	1	0

The input to the transform is an HSV Color Histogram based on the definition of the uniform ColorQuantization descriptor with 16 bins in H, and 4 bins in each S and V (256 bins in total). The Index values of the input histogram bins equal the index values of the uniform ColorQuantization descriptor. Prior to the Haar transform, the histogram (probability) values are subject to a non-uniform mapping into a 4-bit index. Table 16 specifies the mapping function from a histogram with bin values of 11-bit integer precision into this 4-bit representation, which is used as the input to Figure 12(a). The flow diagram of the Haar transform is explained in Figure 12. The transform is performed in place, i.e. the indexing of the coefficients in Table 17 corresponds to the respective direct input path of the flow diagram. Figure 12(a) shows the first four levels of the transform, which has likewise to be applied with index offsets of 4, 8, 12, 32, 36, 40, 44, 128, 132, 136, 140, 160, 164, 168, 172. Figure 12(b) shows the remaining four levels, for which only the “intermediate lowpass” coefficients (to be found at the leading positions from the 16 index offsets) are used as inputs.

Table 16 — Equivalent quantization table of reconstructed HSV color histogram bin values

Histogram value	4-bit index	Histogram value	4-bit index	Histogram value	4-bit index	Histogram value	4-bit index
0	0	40	4	...	8	422	12
1	1	41	5	197	8	...	12
2	1	...	5	198	9	519	12
3	2	66	5	...	9	520	13
...	2	67	6	261	9	...	13
9	2	...	6	262	10	629	13
10	3	101	6	...	10	630	14
...	3	102	7	335	10	...	14
21	3	...	7	336	11	752	14
22	4	144	7	...	11	753	15
...	4	145	8	421	11	...	15
						2047	15

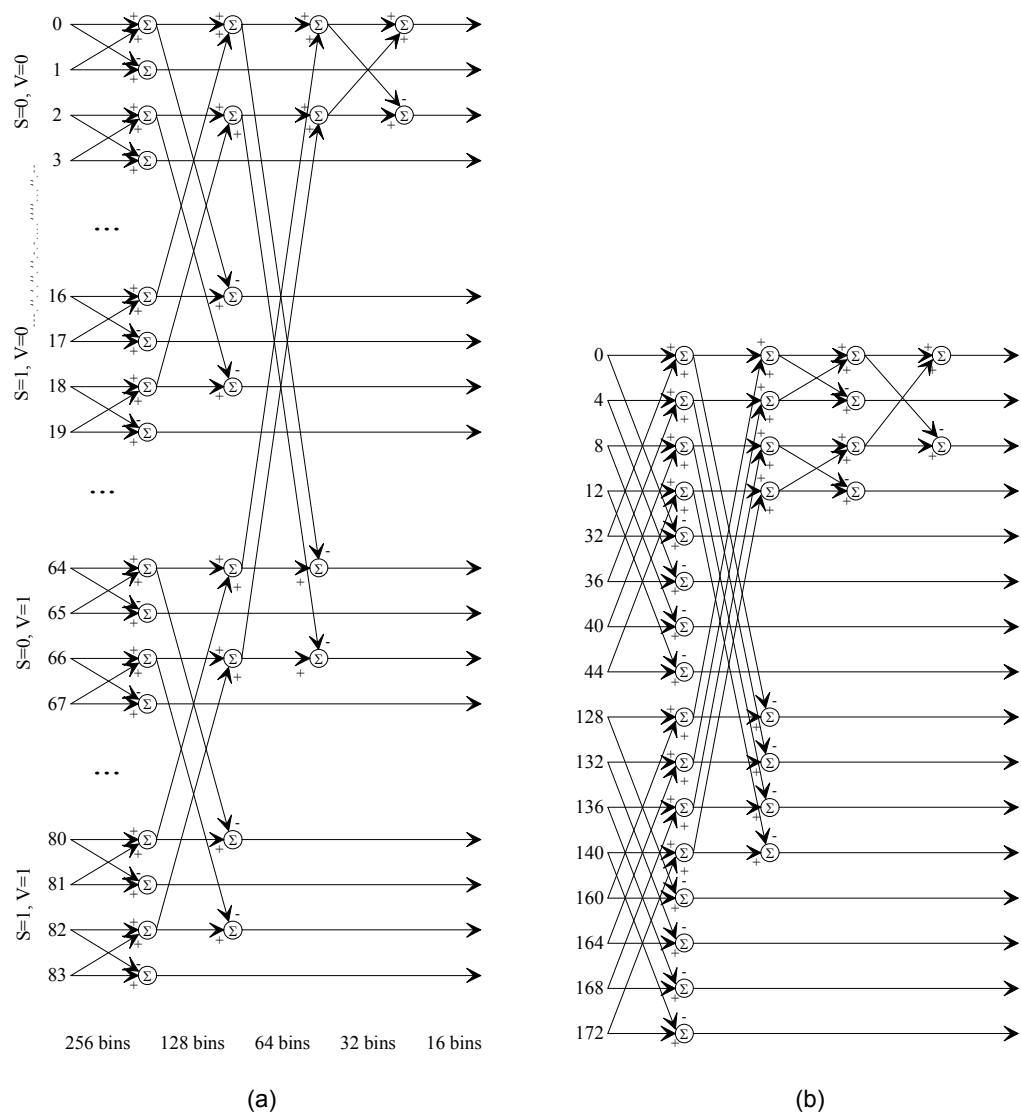


Figure 12 — Haar transform of 256-bin HSV color histogram (a) levels 1-4 (b) levels 5-8

Table 17 — Bin numbers of reconstructed HSV color histograms

Number of coefficients	Number of bins H	Number of bins S	Number of bins V
16	4	2	2
32	8	2	2
64	8	2	4
128	8	4	4
256	16	4	4

6.6 Color layout

6.6.1 Introduction

This descriptor specifies the spatial distribution of colors for high-speed retrieval and browsing. It targets image-to-image matching and sequence-to-sequence matching. It can also be used for color layout-based retrieval, such as sketch-to-image matching. This descriptor can be applied to images or arbitrarily shaped image regions. When applied to a video segment or a moving region, the descriptor specifies the spatial distribution of the color of a representative frame selected from the corresponding video segment or a representative region selected from the corresponding moving region.

The ColorLayout descriptor uses the YCbCr color space with quantization to 8 bits performed in the following way:

$$\begin{aligned} Y &= 219 \cdot Y_{\text{norm}} + 16 \\ Cb &= 224 \cdot Cb_{\text{norm}} + 128 \\ Cr &= 224 \cdot Cr_{\text{norm}} + 128 \end{aligned}$$

Here, the Y_{norm} , Cb_{norm} and Cr_{norm} are the normalized YCbCr color values as defined in subclause 6.2.

6.6.2 DDL representation syntax

```
<complexType name="ColorLayoutType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <sequence>
        <element name="YDCCoeff" type="mpeg7:unsigned6"/>
        <element name="CbDCCoeff" type="mpeg7:unsigned6"/>
        <element name="CrDCCoeff" type="mpeg7:unsigned6"/>
        <choice>
          <element name="YACCoeff2">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="2"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff5">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="5"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff9">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="9"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff14">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="14"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff20">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="20"/>
              </restriction>
            </simpleType>
          </element>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

        </simpleType>
        <length value="20"/>
      </restriction>
    </simpleType>
  </element>
  <element name="YACCoeff27">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="27"/>
      </restriction>
    </simpleType>
  </element>
  <element name="YACCoeff63">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="63"/>
      </restriction>
    </simpleType>
  </element>
</choice>
<choice>
  <sequence>
    <element name="CbACCoeff2">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="2"/>
        </restriction>
      </simpleType>
    </element>
    <element name="CrACCoeff2">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="2"/>
        </restriction>
      </simpleType>
    </element>
  </sequence>
  <sequence>
    <element name="CbACCoeff5">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="5"/>
        </restriction>
      </simpleType>
    </element>
    <element name="CrACCoeff5">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
        </restriction>
      </simpleType>
    </element>
  </sequence>

```