INTERNATIONAL STANDARD

ISO/IEC 15938-16

First edition
2021-06

Information technology — Multimedia content description interface —

Part 16: Conformance and reference software for compact descriptors for video analysis

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

Partie 16: Conformité et logiciels de référence pour les descripteurs compacts pour l'analyse vidéo

Reference number
ISO/IEC 15938-16:2021(E)

Contents

Page

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

A list of all parts in the ISO/IEC 15938 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

ISO/IEC 15938 (all parts) provides a standardized set of technologies for describing multimedia content. It 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 ISO/IEC 15938 (all parts):

Description schemes (DS) describe entities or relationships pertaining to multimedia content. Description schemes specify the structure and semantics of their components, which can 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, etc.

ISO/IEC 15938 (all parts) is subdivided into 16 published parts with further parts in development:

— **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 series 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 series.

— **Part 7: Conformance testing**: specifies the guidelines and procedures for testing conformance of implementations of the series.

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

— **Part 9: Profiles and levels**: provides guidelines and standard profiles.

— **Part 10: Schema definition**: specifies the schema using description definition language.

— **Part 11: MPEG-7 profile schemas**: listing of profile schemas using description definition language.

— **Part 12: Query format**: contains the tools of the MPEG query format (MPQF).

— **Part 13: Compact descriptors for visual search**: specifies an image description tool for visual search applications.

— **Part 14: Reference software, conformance and usage guidelines for compact descriptors for visual search**: provides the reference software and guidelines, specifies the conformance testing.

— **Part 15: Compact descriptors for video analysis**: specifies a video description tool designed to enable efficient and interoperable video analysis applications, allowing visual content matching in videos.

— **Part 16** (this document): **Conformance and reference software for compact descriptors for video analysis**: describes conformance testing of descriptors specified in ISO/IEC 15938-15, and provides the reference software for extracting and matching such descriptors.

The structure of this document is as follows:

— Clause 5 specifies conformance testing of descriptors.

— Clause 6 describes the reference software for extracting and matching descriptors. The deployment of the reference software using Docker containers is described in Annex A.

Information technology — Multimedia content description interface —

Part 16: Conformance and reference software for compact descriptors for video analysis

1 Scope

This document specifies the assessment of conformance to ISO/IEC 15938-15 as well as the reference software.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15938-14, *Information technology — Multimedia content description interface — Part 14: Reference software, conformance and usage guidelines for compact descriptors for visual search*

ISO/IEC 15938-15, *Information technology — Multimedia content description interface — Part 15: Compact descriptors for video analysis*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15938-15 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4 Abbreviated terms, relational operators, bitwise operators and functions

For the purposes of this document, the abbreviated terms, relational operators, bitwise operators and functions in ISO/IEC 15938-15 apply.

5 Conformance testing

5.1 Conformance video dataset and reference bitstreams

The CDVA conformance data set, containing 3 318 items, is used for conformance testing.

The data set is available at https://standards.iso.org/iso-iec/15938/-16/ed-1/en.

For deep feature descriptors, two levels of conformance are considered:

— Conformance using default neural network: This is a stricter level of conformance, which ensures that the bitstreams match those generated with the reference software as-is.

— Conformance using custom neural network: This is a less strict level of conformance, that allows using a custom neural network. The deep feature descriptors generated with implementations using different neural networks are not interoperable.

5.2 Conformance test conditions for strict conformance

5.2.1 Test setup

— Bitstreams extracted with the reference software on Ubuntu 16.04 are provided.

— Implementation under test is used to extract bitstreams from the conformance test set, including all three descriptor components, and varying the parameters listed in subclause 6.5.4.

— Use the reference implementation to perform pairwise matching between the bitstreams of corresponding videos.

— Check that the resulting similarity is above the thresholds listed in the following subclauses.

5.2.2 Test conditions

Vary *minLocalDiff* for the local feature descriptor in the conformance tests. At least the following values for *minLocalDiff* should be tested: 1; 31.

5.2.3 Global descriptor test

Perform pairwise matching of global descriptors.

An implementation is considered conforming to the specification, if at least 99% of the descriptors of the conformance dataset yield a score greater or equal to 0,9*s*, when matching bitstreams extracted with the implementation under test with the reference bitstreams for the same video. *s* is the matching score for the same pair of videos when matching the reference bitstreams to itself using the reference implementation.

5.2.4 Local descriptor test

Perform pairwise matching of local descriptors.

An implementation is considered conforming to the specification if at least 99% of the descriptors of the conformance video dataset yield a score greater or equal to 0,9*s*, when matching bitstreams extracted with the implementation under test with the reference bitstreams for the same video. *s* is the matching score for the same pair of videos when matching the reference bitstreams to itself using the reference implementation.

5.2.5 Deep feature descriptor test with default NN

Perform pairwise matching of deep feature descriptors, using the default NN.

An implementation is considered conforming to the specification if at least 99% of the descriptors of the conformance video dataset yield a score greater or equal to 0,9*s*, when matching bitstreams extracted with the implementation under test with the reference bitstreams for the same video. *s* is the matching score for the same pair of videos when matching the reference bitstreams to itself using the reference implementation.

5.2.6 Combined global and local descriptor test

Perform pairwise matching combining local and global descriptors.

An implementation is considered conforming to the specification if at least 99% of the descriptors of the conformance video dataset yield a score greater or equal to 0,9*s*, when matching bitstreams extracted with the implementation under test with the reference bitstreams for the same video. *s* is the matching score for the same pair of videos when matching the reference bitstreams to itself using the reference implementation.

5.2.7 Combined global and deep feature descriptor test with default NN

Perform pairwise matching combining local and global descriptors, using the default NN.

An implementation is considered conforming to the specification if at least 99% of the descriptors of the conformance video dataset yield a score greater or equal to 0,9*s*, when matching bitstreams extracted with the implementation under test with the reference bitstreams for the same video. *s* is the matching score for the same pair of videos when matching the reference bitstreams to itself using the reference implementation.

5.3 Conformance test conditions using custom NN

As a prerequisite for this test, at least the conformance tests for global descriptors shall be run.

No reference data is provided for this test.

Only syntactic checking can be performed by using the reference software to parse and match bitstreams generated with the implementation under test. This shall complete without errors.

6 Reference software

6.1 Reference software location

The reference software is available at https://standards.iso.org/iso-iec/15938/-16/ed-1/en.

6.2 Reference software licence

The licence of the software is described in the file `CDVA_evaluation_framework/COPYING` included with the reference software.

6.3 Reference software documentation

The documentation of the software is contained in the folder `CDVA_evaluation_framework/docs` included with the reference software. An overview is provided in the file `CDVA_evaluation_framework/README`.

6.4 Reference software installation and compilation

The installation of the software is described in file `CDVA_evaluation_framework/INSTALLING` included with the reference software.

The CDVA reference software is entirely written in C and C++. On Windows, it has been compiled and tested on Windows 10 Enterprise 64-bit using Visual C++ 2015 and 2017 (64 bit). On Linux, it has been compiled and tested on Ubuntu 16.04 LTS (64-bit).

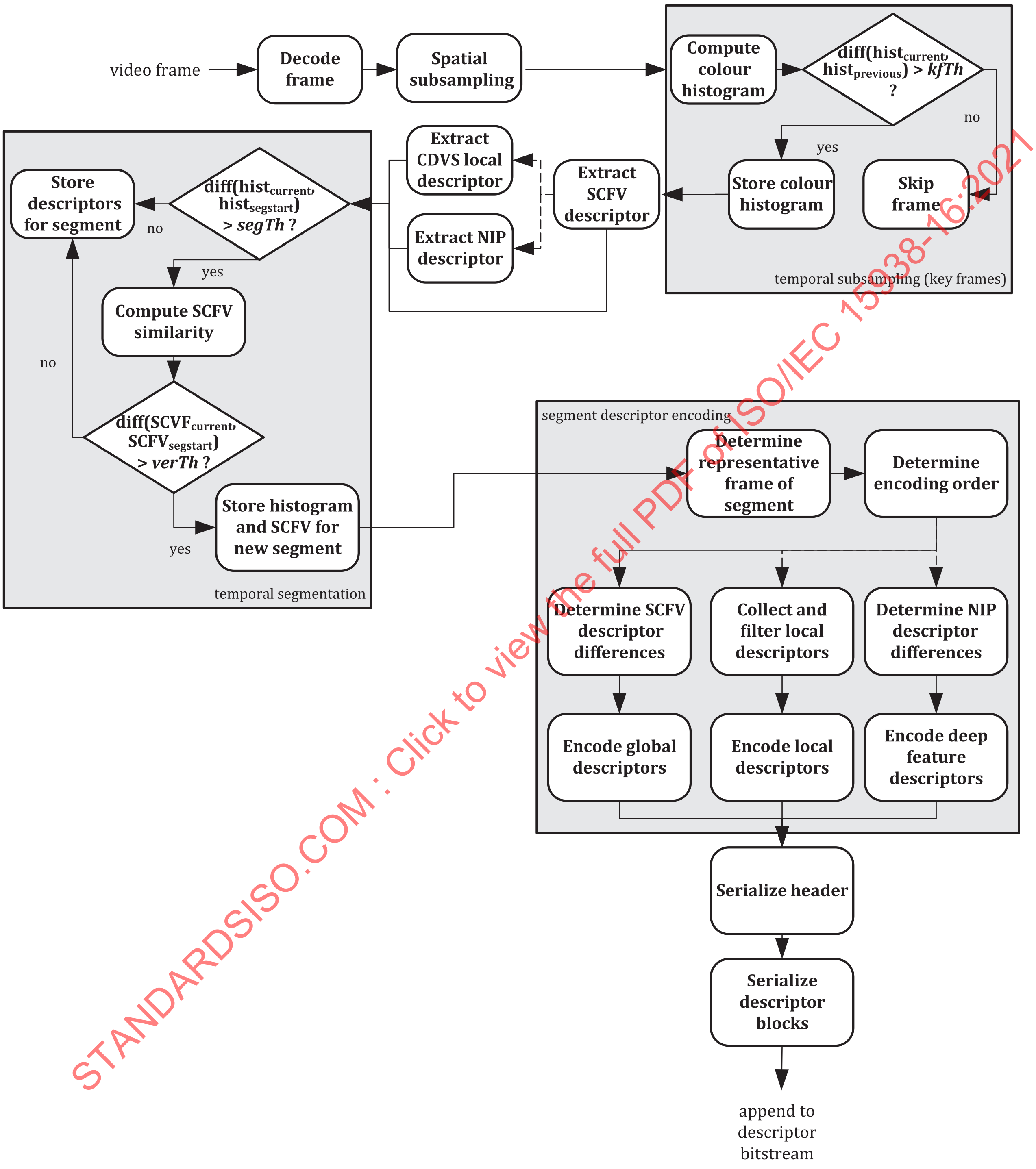
The CDVA reference software has a number of dependencies, in particular due to requiring feature extraction with CNNs. The build process including all dependencies is thus not trivial.

6.5 Reference software architecture

6.5.1 General

The software implementation uses the CDVS reference software (as published in ISO/IEC 15938-14) for extraction of the global and local descriptor components. It includes modules for temporal sampling and segmentation, extraction of deep features and temporal encoding of global and local CDVS features and deep features.

6.5.2 Extraction


video frame
Decode frame
Spatial subsampling
Compute colour histogram
diff(hist$_{current}$, hist$_{previous}$) > *kfTh* ?
no
yes
Store colour histogram
Skip frame
temporal subsampling (key frames)
Extract SCFV descriptor
Extract CDVS local descriptor
Extract NIP descriptor
Store descriptors for segment
diff(hist$_{current}$, hist$_{segstart}$) > *segTh* ?
no
yes
Compute SCFV similarity
no
diff(SCVF$_{current}$, SCFV$_{segstart}$) > *verTh* ?
yes
Store histogram and SCFV for new segment
temporal segmentation
segment descriptor encoding
Determine representative frame of segment
Determine encoding order
Determine SCFV descriptor differences
Collect and filter local descriptors
Determine NIP descriptor differences
Encode global descriptors
Encode local descriptors
Encode deep feature descriptors
Serialize header
Serialize descriptor blocks
append to descriptor bitstream

NOTE Dashed lines indicate optional steps in the process.

Figure 1 — CDVA descriptor extractor

Figure 1 illustrates how the CDVA reference software produces a compact descriptor of a video segment in a series of processing steps, when a video frame is given in input to the system. The process is repeated for all frames in the input video segment. The resulting descriptors are grouped by segments.

Segments are represented with their first key frame being identified as segment boundary. All following frames until the next segment boundary or the end of the file belong to the same segment. The output descriptor is updated by appending the output of the CDVA extraction process for a segment to a single CDVA descriptor.

1. **Frame subsampling:** Performs temporal frame subsampling (typically by a factor of 2–10).
2. **Decode frame:** Decode a frame present in the video.
3. **Compute colour histogram:** A histogram of the R,G,B planes is computed, using 32 bins for each plane.
4. **Check the difference between current and previous colour histograms:** If the difference is greater than a given threshold (*kfTh*), the frame is selected as *keyframe* and further processed. If not, the frame is dropped.
5. **Frame drop module:** If, according to step 4, the current frame is similar to the previously encoded one, the current frame is dropped.
6. **Store colour histogram:** The colour histogram is stored in memory, to be used as “previous histogram” in the next iteration.
7. **Extract SCFV descriptor:** Extracts the SCVF descriptors from individual frames, using mode 0 of ISO/IEC 15938-14.
8. **Extract CDVS local descriptor:** Extracts the CDVS local descriptors from individual frames, using mode 0 of ISO/IEC 15938-14.
9. **Extract deep feature descriptor:** Extract the deep feature descriptor using a pretrained VGG16 network, and optionally binarize the resulting descriptor.
10. **Check the difference of colour histograms between current frame and segment start:** If the difference is greater than a given threshold (*segTh*), the frame is selected as candidate of a segment boundary (first frame of a new segment).
11. **Compute global SCFV similarity between current descriptor and descriptor of first frame of segment:** If the similarity is below a given threshold (*verTh*), the frame is confirmed to be a segment boundary.
12. **Store colour histogram and SCFV descriptor:** For frames identified as segment boundaries, store the descriptors for comparisons with the subsequent frames.
13. **Store descriptors for segment:** Store all extracted descriptors for the current segment together with their time index.
14. **Determine representative frame:** Select a representative frame for the segment, selected as the frame with the medoid SCFV descriptor. This frame is used as the reference for encoding the global and local descriptor of the segment.
15. **Determine encoding order:** The encoding order is determined from the SCFV descriptors of the key frames in the segment. The first is the representative frame, followed by frames with decreasing distance to any of the frames encoded so far.
16. **Determine global descriptor differences:** For each key frame other than the representative frame, determine a difference descriptor as XOR between its SCFV descriptor and the SCFV descriptor of the reference frame.
17. **Determine deep feature descriptor differences:** For each key frame other than the representative frame, determine a difference descriptor as XOR between its NIP descriptor and the NIP descriptor of the reference frame.

18. **Encode global descriptor:** Encode the block formed from the SCFV descriptor of the representative frame and the difference descriptors using adaptive binary arithmetic coding.

19. **Collect and filter local descriptors:** Collect and order the local descriptors used in the segment, starting from the local descriptors of the representative frame, and continue with the temporally adjacent key frame frames (alternating in forward and backward direction). The descriptors are filtered as follows: if a local descriptor differs less than *minLocalDiff* elements from one already collected, it is discarded, and replaced by a reference to the already encoded descriptor.

20. **Encode local descriptors:** Generate a map of descriptor indices between the original per frame indices and the filtered list of descriptors. In case descriptors have been replaced by references, the map will have multiple pointers to the same index. The set of local descriptors remaining after filtering is encoded using adaptive binary arithmetic coding.

21. **Encode deep feature descriptor:** Encode the block formed from the deep feature descriptor of the representative frame and the difference descriptors using adaptive binary arithmetic coding.

22. **Serialize header:** Write a header structure, containing the start and end time of the segments, parameters needed for decoding, the number of frames, local descriptors and the sizes of the blocks containing the different types of descriptors.

23. **Serialize descriptor blocks:** Write the encoded global, local and deep feature descriptor blocks.

6.5.3 Pairwise matching

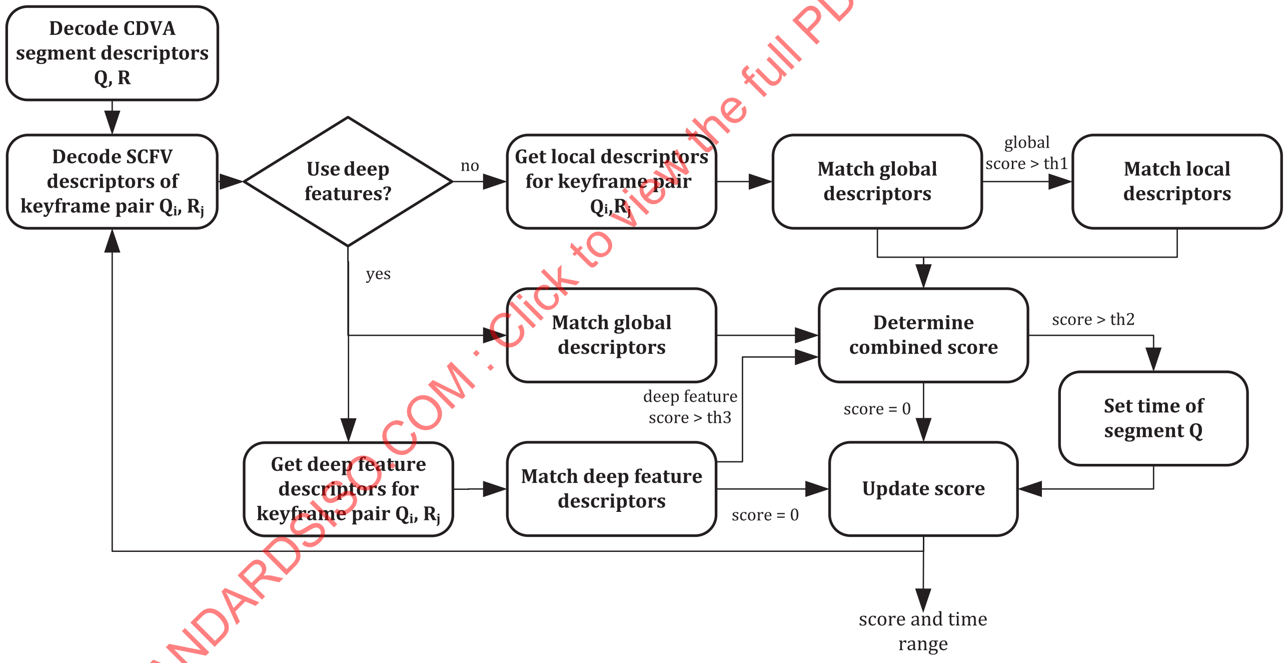


Figure 2 — CDVA pairwise matching

Figure 2 illustrates the CDVA pairwise matching operation.

1. **Decode:** Decode the query CDVA descriptor (*Q*) and the reference CDVA descriptor (*R*). Reconstruct SCFV and CDVS local feature or deep feature descriptors for all key frames in the segment.

2. **Get next descriptors for next key frame pair *Qi,Rj*:** For all pairs of query and reference key frames, execute the following steps.

3. **Branch between matching procedure with CDVS local and deep features.**

4. **Match global descriptor:** Perform a CDVS SCFV match operation using the global descriptors of key frames *Qi* and *Rj*.

5. **Match local descriptors:** If the match score is greater than *th1*, perform a CDVS match operation using the local descriptors.

6. **Match deep feature descriptors:** Perform a match operation (Hamming distance) of the deep feature descriptors of key frames *Qi* and *Rj* and convert the distance into a similarity score. If the score exceeds *th1*, continue with the combination of scores.

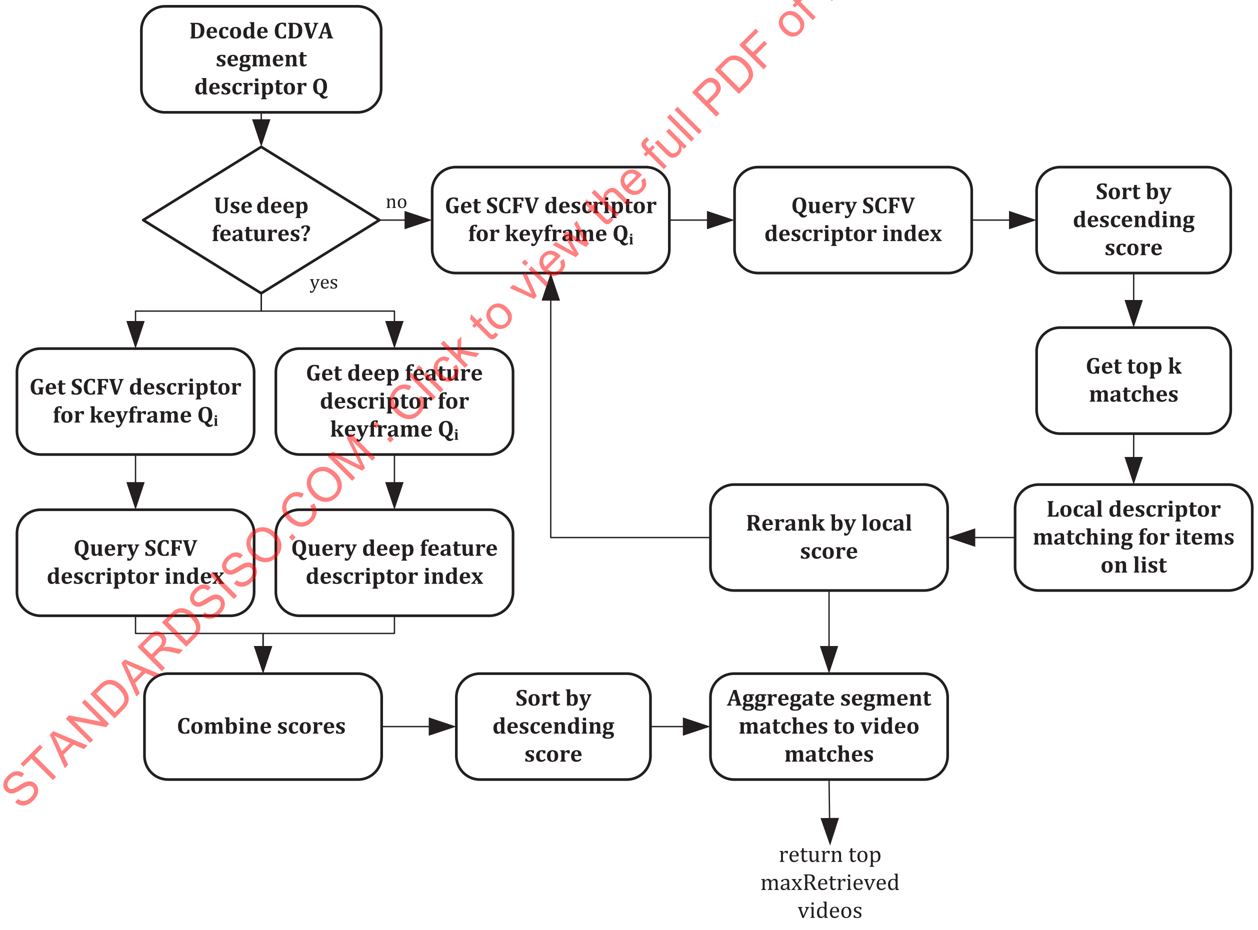
7. **Determine score:** Determine the current score as a combination of the local and global score or global and deep feature score respectively.

8. **Set time of *Q*:** If the current score is greater than *th2*, store the time of segment *Q* belongs.

9. **Update score:** (default strategy: max) Set the total score as the maximum of all current scores.

If CDVS local feature descriptors are used, the thresholds *th1* and *th2* are defined according to ISO/IEC 15938-14. If deep features are used, the parameters *th2* and *th3* defined in subclause 6.5.5 shall be used.

6.5.4 Retrieval



Decode CDVA segment descriptor Q
Use deep features?
no
yes
Get SCFV descriptor for keyframe Q_i
Query SCFV descriptor index
Sort by descending score
Get top k matches
Local descriptor matching for items on list
Rerank by local score
Get SCFV descriptor for keyframe Q_i
Get deep feature descriptor for keyframe Q_i
Query SCFV descriptor index
Query deep feature descriptor index
Combine scores
Sort by descending score
Aggregate segment matches to video matches
return top maxRetrieved videos

Figure 3 — CDVA retrieval pipeline

Figure 3 illustrates the CDVA retrieval operation. This figure assumes that a database index has been built and is in place.

1. **Decode:** Decode the query CDVA descriptor *Q* and reconstruct the global and local or deep feature descriptors for each key frame in the segment.

2. **Branch between retrieval procedure with CDVS local and deep features.**

Retrieval procedure if CDVS local descriptors are present

3. **Get next:** For the CDVS global and local descriptors in the query CDVA descriptor, execute the following steps.
4. **Query in the global descriptor index:** Perform a CDVS retrieval operation using SCFV descriptor of key frame *Qi* on the global descriptor database index.
5. **Sort:** Sort the results by descending score.
6. **Local descriptor matching:** Perform CDVS local descriptor matching on the top 500 results.
7. **Rerank:** Rerank by local score.
8. **Merge results:** Merge results removing duplicates.
9. **Return the top *maxRetrieved* videos**.

Retrieval procedure if deep feature descriptors are present

3. **Query in the global descriptor index:** Perform a retrieval operation using the SCFV descriptor of key frame *Qi* on the global descriptor database index.
4. **Query in the deep feature descriptor index:** Perform a retrieval operation using the deep feature descriptor of key frame *Qi* on the deep feature descriptor database index.
5. **Combine scores:** Determine the score as a weighted combination of the global and deep feature descriptor scores, using *deepWeight* as the weight for the deep feature descriptor and (1-*deepWeight*) as the weight for the global feature descriptor.
6. **Sort:** Sort the results by descending score.
7. **Merge results:** Merge results removing duplicates.
8. Return the top *maxRetrieved* videos.

6.5.5 Parameters

Table 1 describes the parameters in the CDVA reference software. Note that additional parameters are in the CDVS reference software, which are used by the CDVA reference software.

Table 1 — Parameters used in CDVA reference software

Parameter	Process	Description	Default (16/64/256)
skipNum	Extraction	Number of frames skipped before a sampled frame.	8 / 6 / 4
kfTh	Extraction	Threshold (colour histogram) for selecting key frames.	0,7 / 0,6 / 0,5
segTh	Extraction	Threshold (colour histogram) for segment candidates.	1,98
verTh	Extraction	Threshold (CDVS global) for verifying segment candidates.	18
minLocalDiff	Extraction	Minimum local difference of local descriptors to be encoded (otherwise replaced by a reference).	31/16/1

Table 1 *(continued)*

Parameter	Process	Description	Default (16/64/256)
th2	Matching	[Deep feature descriptors only] Threshold for the matching the weighted sum of deep feature and SCFV global descriptor. If CDVS local features are used, this parameter is defined according to the CDVS specification.	0,46
th3	Matching	[Deep feature descriptors only] Threshold for deep feature descriptor matching. If CDVS local features are used, this parameter is defined according to the CDVS specification.	0,58
deepWeight	Matching and retrieval	[Deep feature descriptors only] Weight of the deep feature descriptor for combination. The SCFV descriptor will be weighted 1 – *deepWeight*.	0,75
maxRetrieved	Retrieval	Maximum number of segments to retrieve.	50

6.5.6 Implementation details for CDVA reference software

The software modules to extract, match, make the database index and retrieve matching images from the database index are implemented in C++.

The CDVA reference software code and the software documentation are available at https://standards.iso.org/iso-iec/15938/-16/ed-1/en. The code is documented by means of Doxygen (http://www.doxygen.org/).

Table 2 summarizes the modules that are included in the CDVA reference software.

Table 2 — Modules required by the CDVA reference software

CDVS library	The CDVS reference software (as published in ISO/IEC 15938-14).
OpenCV library	The Open Source Computer Vision library (2.4).
TensorFlow	Tensorflow 1.8 framework (for deep feature extraction).
CUDA toolkit	The CUDA toolkit (9.x) for GPU support in Tensorflow (for deep feature extraction, not required if Tensorflow is built without GPU support).
CuDNN	The CUDA DNN library (7.x) for GPU support in Tensorflow (for deep feature extraction, not required if Tensorflow is installed without GPU support).

6.5.7 Usage

6.5.7.1 General

Once built and installed, the CDVA reference software can be started using the command

```
cdva <subcommand> <arguments>
```

where `<subcommand>` is one of the following:

— `extract`: extract descriptors from a list of files;

— `match`: match a list of descriptor pairs (using global or local feature descriptor components, or both);

— `retrieve`: run a list of queries against a set (database) of descriptors (using global or local feature descriptor components, or both);

— `match_by_deep_only`: match a list of descriptor pairs (using deep feature descriptor components);

— `match_by_deep_combination_SCFV`: match a list of descriptor pairs (using global and deep feature descriptor components);

- `retrieve_by_deep_only`: run a list of queries against a set (database) of descriptors (using deep feature descriptor components);
- `retrieve_by_deep_combination_SCFV`: run a list of queries against a set (database) of descriptors (using global or deep feature descriptor components, or both).

The parameters are specific to the types of subcommands.

6.5.7.2 Extraction arguments

```
cdva extract <vlist> <bitrate> [-k][-s][-c][-h][-t][-d][-n num][-w dir][-v][-f drop_frame_
th][-e encode_th][-x][-D][-b]
```

where:

- `vlist` – text file containing the relative pathname of the video files to process (one file name per line)
- `bitrate` (16, 64, 256)– bit rate of descriptors
- specific options:
- options:
  - -k: keep old descriptors (do not overwrite) if they exist
  - -s: determine descriptor component sizes
  - -x: read/write compressed
  - -D: extract deep feature (NIP) descriptors
  - -b: binarize deep feature (NIP) descriptors (only in combination with -D)
  - -c: generate CSV file
  - -h: generate HTML file
  - -t: produce text output
  - -d: dry run, try operation but make no changes
  - -n num: process only the first num elements
  - -w dir: set workspace directory (where descriptors are stored)
  - -v: verbose

EXAMPLE 1 Extract from the conformance data set, writing to directory ~/cxm, using the bitrate 16K, extracting deep feature descriptors and producing a CSV output:

```
cdva extract ConformanceExtract_CC.txt 16 -w ~/cxm -c -D
```

EXAMPLE 2 Extract from the conformance data set, writing to directory ~/cxm, using the bitrate 256K, not extracting deep feature descriptors, writing compressed output streams and producing a CSV output:

```
cdva extract ConformanceExtract_CC.txt 16 -w ~/cxm -c -D -x
```

6.5.7.3 Matching arguments

```
cdva match <vlist> <query_bitrate> <reference_bitrate> [-c][-h][-t][-d][-n num][-w dir]
[-v][-x]
```

where:

— `vlist` – text file containing the relative pathname of the video files to match (two file names per line)

— `query_bitrate` (16, 64, 256) – use query descriptors of this type

— `reference_bitrate` (16, 64, 256) – use reference descriptors of this type

— options:

  — -x: read/write compressed

  — -c: generate CSV file

  — -h: generate HTML file

  — -t: produce text output

  — -d: dry run, try operation but make no changes

  — -n num: process only the first num elements

  — -w dir: set workspace directory (where descriptors are stored)

  — -v: verbose

EXAMPLE Match the pairs in the conformance data set, using 16K bitrate, reading descriptors from ~/cxm in compressed format, and writing a CSV output:

```
cdva match ConformanceExtract_CC.txt 16 16 -c -x -w ~/cxm
```

NOTE Which descriptor components are matched is determining from what is contained in the descriptor bitstreams.

6.5.7.4 Retrieval arguments

```
cdva retrieve <queries> <bitrate> <index> <idxrate> [-c][-h][-t][-d][-n num][-w dir][-v]
[-x][-m][-D][-b]
```

where:

— queries - text file containing the relative pathnames of the query video files to process (one file name per line)

— bitrate (16, 64, 256) – bit rate of query descriptors

— index – text file containing the relative pathnames of the reference descriptors (one file per line)

— idxrate (0, 16, 64, 256) – bit rate of reference descriptors

— options:

  — -x: read/write compressed

  — -c: generate CSV file

  — -h: generate HTML file

  — -t: produce text output

  — -d: dry run, try operation but make no changes

  — -n num: process only the first num elements