Dirac Speci¯cation

Version 0.10.1

Abstract

This document is the speci¯cation of the Dirac video decoder and stream syntax.
Dirac is a video compression system utilising wavelet transforms and motion compensation.

1 Introduction

1.1 Purpose

Dirac was developed to address the growing complexity and cost of current video compression tech-

2 The conventions used in the speci¯cation

2.1 State machine decoder representation

This speci¯cation uses a state-machine model to express parsing and decoding operations. The state of the decoder/parser is stored in the global variable state, and individual variable values are accessed by means of namedof(of).e accessed


foo() :

for each The for each control loops over the elements in a list:

4 Arithmetic decoding

SIGN_ZERO
SIGN

REF2y_DATAREF2y

Part II

Stream parsing

5 Dirac stream specification

video_sequence() :	
parse_info()	5.3
while (*is_access_unit*()):	5.3
access_unit()	5.4

is_picture() :	

sequence_parameters() :	

frame_rate() :	
frame_rate_flag = *read_bool*()	
if (*frame_rate_flag* == True	

init_decode_params() :	
for each *var* in args(**default_state**):	
state[*var*] = **default_state**[*var*]	

State variables and default state variables are listed in the index.

5.9.2 Picture header

2. state[luma_xblen] ¸ state[luma_xbsep] and state[luma_yblen] ¸ state[luma_ybsep]
3. state[luma_xblen] ¡ state[luma_xbsep] and state[luma_yblen] ¡ state[luma_ybsep] shall be powers of 2 other than 1
4. state[chroma_xblen] ¡ state[chroma_xbsep] and state[chroma_yblen] ¡ state[chroma_ybsep] shall be powers of 2 other than 1

Informative:

ohnumewiFth1227x(pol3e84l-8.-9.TD[cb)]9Rn-orthogonalepr je289(38)].-9.rwithne343(xp)-/F80TD[()]TJ/F14ein!

- an integer pan/tilt vector

$$\mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$$

- an integer matrix element

wavelet_transform() :

The maximum number of codeblocks vertically shall be less than or equal *subband_height*(*level*)==2[illegible]

6 Motion data decoding

This section speci¯es the operation of the *block_data*() process for extracting block motion data from the Dirac stream.

Block data is aggregated into *superblocks*, consisting of a 4x4 array of blocks. The number of su-

6.2.1 Overall decoding loop

The decoding loop for block data iterates over all superblocks in raster order:

block_data() :	
initialise_motion_data()	6.2.2
length = *read_uint*()	
byte	

at position (*xpos; ypos*).

block_motion(*ypos; xpos*) :	

^{2} Sign = REF1x_SIGN

*ref*1

7 Wavelet coe±cient decoding

This section speci¯es how wavelet transform coe±cients are parsed.

7.1 Decoded subband data conventions7.1 ca aata ainitialisaionThisei¯es

is returned. If there is no parent (*parent* = *;*), then 0 is returned.

7.4.3 Zero neighbourhood

The *zero_nhood*() function returns a boolean indicating whether neighbouring values are all zero.

zero_nhood(*band; v; h*) :	
if (*v* > 0):	
if (*band*[*v* ¡ 1][*h*]! = 0):	
return False	
if (*h* > 0):	
if (*band*[*v* ¡ 1][*h* ¡ 1])! = 0*jjband*[*v*][*h* ¡ 1]! = 0):	
return False	
else:	
if (*h* > 0):	

parent	*zero_nhood*	*sign*	Context set	
0	True	0	Follow	[ZPZN

decode the coe ± cient sign.

7.4.7 Quantisation factors and o®sets

This section speci¯es the operation of the *quant_factor*() and *quant_*

ref_picture_remove() :	
for *i* = 0 to length(state[retired_picture_	

9.3 One-dimensional synthesis

This section speci¯es the one-dimensional synthesis process 1*d_synthesis*() applied to a 1-dimensional array of coe±cients of even length, consisting of either a row or a column of a 2-dimensional integral data array.

The one-dimensional synthesis process comprises the application of a number of reversible integer lifting ¯lter operations. An integral lifting ¯lter is characterised by four elements:

- a set of taps *t*

Informative:

Lifting steps	*filtershift*()
1. Even, Predict, $s = 1; t_1$	

10 Motion compensation

This section de¯nes the operation of the process *motion_compensate*(*ref*1; *ref*2; *pic*; *c*) for motion-compensating a picture component array *pic* of type *c* = *Y*; *U* or *V* from reference component arrays

$pixel\_pred(y; x; pic; ref1; ref2; c)$:	
$p = 0$	
for $(i; j)$ such that $(x; y)\ 2\ B(i; j)$:	

b[ ][ ] = 0	
for *j* = 0 to state[blocks_y] ¡ 1:	
for *i* = 0 to state[blocks_x] ¡ 1:	
m = state[block_	

If $c = Y$, set $A = \tilde{A}$, $b = \tilde{b}$ and $c = \tilde{c}$.

If $c = UAU^{-1}b =$

10.8.2 Half-pixel accurate motion vectors

If state[mv_precision] == 1 then the reference picture component *ref* is upconverted by a factor of 2 in each dimension to create an array *upref*. The value returned is *upref*[*cv*][*cu*].

upref

$$ru = u \ ¡ \ (hu \ ¿ \ (\text{state}[\text{mv\_precision}] \ ¡ \ 1))$$

A Parse diagrams

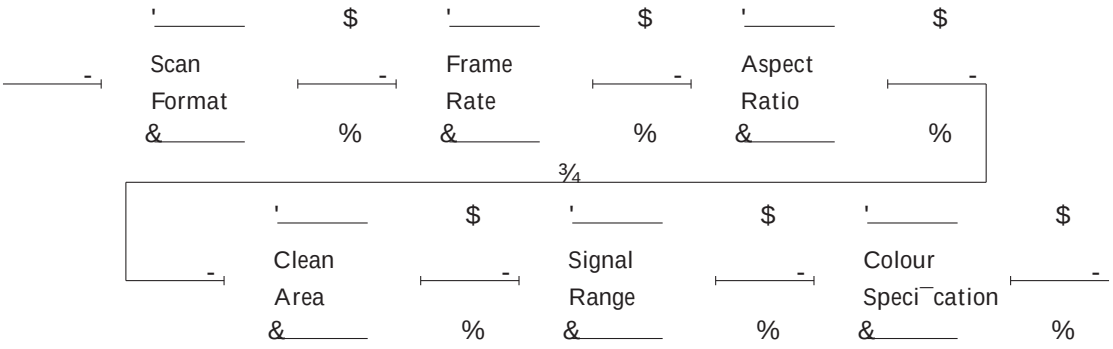


Figure 19: Access Unit Source Parameters

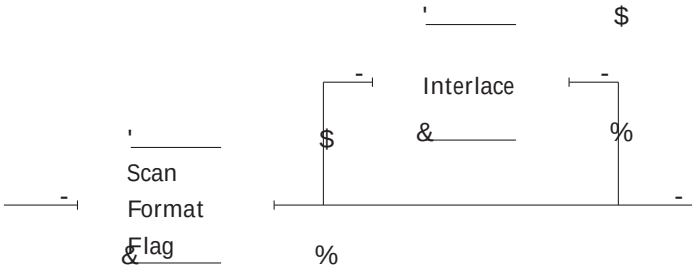
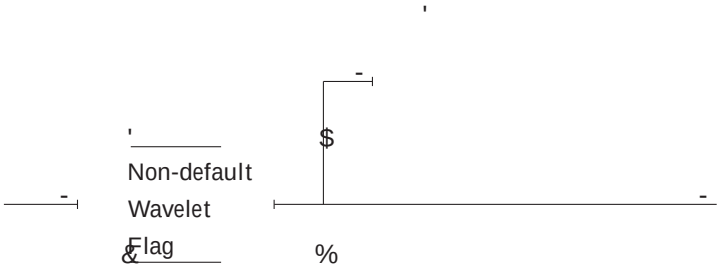


Figure 20: Scan Format


Non-default
Wavelet
Flag

[LUMA_OFFSET, LUMA_OFFSET+LUMA_EXCURSION]

and Cb, Cr to

[CHROMA_OFFSET-LUMA_

C Video format defaults

D Pro¯les and levels

Index