08-6301-SIS-ZCH66
JUNE 26, 2008
1.2

Application Programmers Interface for Deinterlace

ABSTRACT:

Application Programmers Interface for Deinterlace

KEYWORDS:

Multimedia codecs, deinterlace

APPROVED:

Wang Zening

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	24-Dec-2007	Zhenyong Chen	Initial Draft
1.1	04-Mar-2008	Zhenyong Chen	Usage for frame group deinterlace algorithm added
1.2	26-Jun-2008	Zhenyong Chen	Change API for version string

Table of Contents

Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 Audience Description 4
1.4 References 4
1.4.1 Standards 4
1.4.2 Freescale Multimedia References 4
1.5 Definitions, Acronyms, and Abbreviations 1
1.6 Document Location 1
2 API Description 2
2.1 Data Structures 2
DEINTER 3
2.2 Macros 4
2.3 Application Programmer Interface 4
2.3.1 Deinterlacing methods Initialization and query 4
2.3.2 Deinterlace a frame 4
2.3.3 Frame reusable 5
2.3.4 Deinterlace two frames simultaneously 5
2.3.5 Get version description 5
3 Example Lib Usage 7

Introduction

1.1 Purpose

This document gives details of the application programmer's interface for deinterlace.

1.2 Scope

This document does not detail the implementation of the deinterlace algorithm. It only explains the APIs and data structures exposed to the application developer for using the deinterlace library.

1.3 Audience Description

The reader is expected to have basic understanding of deinterlacing.

1.4 References

1.4.1 Standards

- No relevant standards

1.4.2Freescale Multimedia References

- Deinterlace Requirements Book – deinterlace_reqb.doc
- Deinterlace Test Plan – deinterlace_test_plan.doc
- Deinterlace Release notes – deinterlace_release_notes.doc
- Deinterlace Test Results – deinterlace_test_results.doc
- Deinterlace Interface Header – deinterlace_api.h
- Deinterlace Application code – DeintApp.cpp

1.5 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
API	Application Programming Interface
ARM	Advanced RISC Machine
FSL	Freescale
Cortex	Official name for Arm12
Elvis	A chip with Cortex core
OS	Operating System
TBD	To Be Determined

1.6 Document Location

docs/deinterlace

2 API Description

This section describes the data structures followed by an example usage of deinterlace library. First, define basic data types used in this document:
BYTE: unsigned char, 8-bit
BOOL: int, 32-bit, 0 for false, non-zero for true

2.1 Data Structures

DEINTMETHOD

This structure specifies the information related to a method.

```
typedef struct tagDeintMethod
{
    unsigned int method;
    char name[64];
    char need_prev_frame;
    char need_next_frame;
    char safe;
}DEINTMETHOD;
```

Description of structure *DEINTMETHOD*

method
ID of method.
name
Name of method.
need_prev_frame
Whether previous frame is needed for reference.
need_next_frame
Whether next frame is needed for reference.
safe
Safety for use of this method. 1 for Freescale proprietary methods.

PICTURE

This struct specifies y, cb, and cr buffers for a picure.

```
typedef struct tagPICTURE
{
    BYTE *y;
    BYTE *cb;
    BYTE *cr;
}PICTURE;
```

Description of structure *PICTURE*

y
: Address of y buffer.

cb
: Address of cb buffer.

cr
: Address of cr buffer.

DEINTER

This struct specifies all parameters needed by deinterlace algorithm.

```
typedef struct tagDEINTER
{
    PICTURE frame[3];
    int y_stride;
    int uv_stride;
    int chrom_fmt;
    int width;
    int height;
    BOOL top_first;
    int method;
    void *dynamic_params;
}DEINTER;
```

Description of structure *DEINTER*

frame
: Array of frame addresses. frame[0] contains previous frame, frame[1] contains current frame, and frame[2] contains next frame. However, if some frame(s) are not required by the algorithm, then set them NULL.

y_stride
: Stride for y buffer of each frame.

uv_stride
: Stride for u/v buffer of each frame.

chrom_fmt
: Chroma format. Can be one of following values:

Value	Meaning
0	YUV 4:2:0
1	YUV 4:2:2
2	YUV 4:4:4

width
: Width of frame, in pixel.

height
: Height of frame, in pixel.

top_first
: Which field is first in temporal space (or sampled first).

method
: Which method to be used for deinterlacing.

dynamic_params
Reserved. Set NULL.

2.2 Macros

Chroma format

Macro name	Value	Meaning
CHROM_FMT_420	0	4:2:0
CHROM_FMT_422	1	4:2:2
CHROM_FMT_444	2	4:4:4

IDs of deinterlacing methods

ID	Method
0	Bob
1	Weave
2002	Block VT, field SAD, Bob
2003	Block VT, frame SAD, Bob
2004	Block VT, field SAD, 4-tap filter
2005	Block VT, frame SAD, 4-tap filter
2006	Block VT, frame group, frame SAD, 4-tap filter
2007	Block VT, frame group, frame SAD, Bob

2.3 Application Programmer Interface

2.3.1 Deinterlacing methods Initialization and query

InitDeinterlaceSafe

To query information of algorithms in the library.

C prototype:
```
void InitDeinterlaceSafe(DEINTMETHOD *pMethods, int *count);
```

Arguments:
- pMethods [out] Buffer to store information
- count [out] Count of methods implemented

Return value:
- None

2.3.2 Deinterlace a frame

DeinterlaceSafe

To deinterlace an interlaced frame, and output a progressive frame in same place as input frame.

C prototype:
```
void DeinterlaceSafe(DEINTER *pDeinterInfo);
```

Arguments:
- pDeinterInfo [inout] See definition of DEINTER.

Return value:
- None

2.3.3 Frame reusable

IsFrameReusableSafe

To determine current frame buffer can be reused when requires reloading.

C prototype:
```
int  IsFrameReusableSafe(int nMethod, BOOL bDrawTrack);
```

Arguments:
- nMethod [in] Deinterlace method.
- bDrawTrack [in] Whether tracking motion block is enabled. When this feature is enabled, frame buffer will be overwritten to show block edge.

Return value:

1 - reusable
0 - not reusable
-1 - undetermined

Remark:
For some deinterlace algorithms, current frame after filtering can be reused for another frame deinterlacing (next frame, current frame, or previous frame). To avoid reloading it, use this API to check whether current frame need (or not) reload.

2.3.4Deinterlace two frames simultaneously

The newly added methods (id 2006 and 2007) deinterlace two frames simultaneously, by function **DeinterlaceSafe**(see 2.3.2). No specific settings needed.

2.3.5Get version description

GetDeinterlaceVersionInfo

To get version description string.

C prototype:

```
const char * GetDeinterlaceVersionInfo();
```

Arguments:

- None

Return value:

- Version string.

Remarks:

Version string format is:
<BASELINE_SHORT_NAME> [OS_NAME] [CODEC_RELEASE_TYPE] [VERSION_STR_SUFFIX] “build on” <date and time information>

3 Example Lib Usage

This example shows how to use the deinterlace library.

```
static DEINTMETHOD methods_info[32];
static int method_count = 0;
int GetMethodCount(void)
{
   return method_count;
}
BOOL IsMethodNeedPrevFrame(unsigned int method)
{
   int methodIndex = GetMethodPosition(method);
   if(methodIndex == -1)
      return FALSE;
   return methods_info[methodIndex].need_prev_frame;
}
BOOL IsMethodNeedNextFrame(unsigned int method)
{
   int methodIndex = GetMethodPosition(method);
   if(methodIndex == -1)
      return FALSE;
   return methods_info[methodIndex].need_next_frame;
}
static BOOL IsMethodSafe(unsigned int method)
{
   int methodIndex = GetMethodPosition(method);
   if(methodIndex == -1)
      return FALSE;
   return methods_info[methodIndex].safe;
}
const char * GetMethodName(unsigned int method)
{
   int methodIndex = GetMethodPosition(method);
   if(methodIndex == -1)
      return NULL;
   return methods_info[methodIndex].name;
}
unsigned int MethodFromPosition(int position)
{
   if(position == -1)
      return (unsigned int)-1;
   return methods_info[position].method;
}
int GetMethodPosition(unsigned int method)
{
   int i;
```

```
    for(i=0; i<method_count; i++)
    {
        if(methods_info[i].method == method)
            return i;
    }
    return -1;
}
void InitDeinterlace(void)
{
    int i;
    for(i=0; i<32; i++)
    {
        methods_info[i].method = (unsigned int)-1;
    }
    method_count = 0;
    // Register implemented methods

    // Bob
    methods_info[method_count].method = DEINTMETHOD_BOB;
    strcpy(methods_info[method_count].name, "Bob");
    methods_info[method_count].need_prev_frame = 0;
    methods_info[method_count].need_next_frame = 0;
    methods_info[method_count].safe = 1;
    method_count++;

    // Weave
    methods_info[method_count].method = DEINTMETHOD_WEAVE;
    strcpy(methods_info[method_count].name, "Weave");
    methods_info[method_count].need_prev_frame = 0;
    methods_info[method_count].need_next_frame = 0;
    methods_info[method_count].safe = 1;
    method_count++;

    int count;
    InitDeinterlaceSafe(&methods_info[method_count], &count);
    method_count += count;
}

int Deinterlace(DEINTER *pDeinterInfo)
{
    int method = pDeinterInfo->method;
    int nFrameReUsable;

    if(method == DEINTMETHOD_WEAVE) // Do nothing
        return 0;


    // In-frame deinterlacing -
    // Following methods won't be affected
    // Bob, Weave, Repeat, VT_Median, Bob_Weave
```

```
    switch(method)
    {
    case DEINTMETHOD_BOB:
    case DEINTMETHOD_WEAVE:
        nFrameReUsable = 1;
        break;
    default:
        nFrameReUsable = IsFrameReusableSafe(method, distinguish_block);
        if(nFrameReUsable == -1)
            nFrameReUsable = 0;
        break;
    }

    // Parameter check
    if(IsMethodNeedPrevFrame(method) && pDeinterInfo->frame[0].y == NULL)
        method = DEINTMETHOD_BOB;
    if(IsMethodNeedNextFrame(method) && pDeinterInfo->frame[2].y == NULL)
        method = DEINTMETHOD_BOB;

    // deinterlace begin ...
    pDeinterInfo->method = method;
    if(IsMethodSafe(method))
        DeinterlaceSafe(pDeinterInfo);
    // deinterlace end.

    return nFrameReUsable;
}

int main(int artc, const char *argv[])
{
    printf(GetDeinterlaceVersionInfo());
    printf ("\r\n");

    InitDeinterlace();
    …
    DEINTER di;
    if(pic_prev)
    {
        di.frame[0].y = pic_prev->y;
        di.frame[0].cb = pic_prev->cb;
        di.frame[0].cr = pic_prev->cr;
    }
    else
    {
        di.frame[0].y = NULL;
        di.frame[0].cb = NULL;
        di.frame[0].cr = NULL;
    }
    if(picture)
    {
```

```
        di.frame[1].y = picture->y;
        di.frame[1].cb = picture->cb;
        di.frame[1].cr = picture->cr;
    }
    else
    {
        di.frame[1].y = NULL;
        di.frame[1].cb = NULL;
        di.frame[1].cr = NULL;
    }
    if(pic_next)
    {
        di.frame[2].y = pic_next->y;
        di.frame[2].cb = pic_next->cb;
        di.frame[2].cr = pic_next->cr;
    }
    else
    {
        di.frame[2].y = NULL;
        di.frame[2].cb = NULL;
        di.frame[2].cr = NULL;
    }
    di.chrom_fmt = m_StreamInfo.nChromaFormat;
    di.y_stride = m_y_stride;
    di.uv_stride = m_uv_stride;
    di.width = m_StreamInfo.nWidth;
    di.height = m_StreamInfo.nHeight;
    di.top_first = m_StreamInfo.bTopFirst;
    di.method = m_StreamInfo.nDeintMethodID;
    di.dynamic_params = NULL;
    pic_reusable = Deinterlace(&di);
    ...
}
```

