특집논문-05-10-2-02

Implementation of MPEG-21 DIA Utility Software for Stereoscopic Video Adaptation

Hyukmin Kwon[a)‡] and Manbae Kim[a)]

Abstract

This paper presents recent works of the development of MPEG-21 DIA utility software as one of MPEG-21 activities. The main purpose of this work is to make integrated utility software for DIA C/C++ modules in a Java framework. To do this, JNI plays a role of the interface of a main module with C/C++ DIA adaptation modules. Furthermore, DIA and DID parsers are integrated to the utility software. Our software is designed such that all DIA modules are easily integrated. A variety of adaptation modules have been adopted in MPEG-21 standard. Among them, stereoscopic video conversion that is one of DIA adaptation modules is chosen and we show that our utility software is correctly implemented based on experiments.

Keywords : MPEG-21, DIA, Utility software, JNI

I. Introduction

Digital Item Adaptation (DIA) is one of main MPEG-21 parts[1][2]. The goal of the DIA is to achieve interoperable transparent access to multimedia contents by shielding users from network and terminal installation, management and implementation issues. The DIA is composed of resource adaptation and descriptor adaptation that produce a newly adapted (modified) Digital Item (DI). The descriptor adaptation transforms the input descriptor to the output one. As well, the input resource is adapted into the output resource according to the descriptor. Those outputs are then delivered to a user that has requested the adapted Digital Item. Fig. 1 illustrates the fundamental concept of the DIA processing. DIA adaptation engine generates an adapted CDI from two inputs CDI (Content Digital Item) and XDI (conteXt Digital Item). In other words, DIA modules consist of Digital Item A (input), Digital Item B (input), Digital Item A' (output), and methods such as adaptation engine, A' = f (A, B).

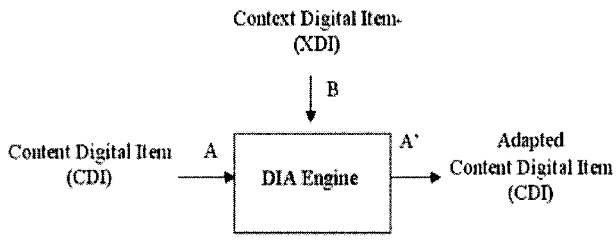


Fig. 1. MPEG-21 DIA processing

Currently, according to MPEG-21 DIA software implementation plan[3], DIA reference software packages and utility software modules are under development as a part of ISO/IEC Reference Software. Most of provided DIA software modules including stereoscopic video conversion (adaptation) are written in C/C++ language[4]. In

a) Department of Computer, Info. and Telecom., Kangwon National University, Chunchon, Republic of Korea

※ This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

order to integrate these C/C++ software modules with Java-based reference software packages (e.g. DIA Usage Environment Descriptor Tool Parser[5] and Digital Item Declaration (DID) Parser), the use of Java Native Interface (JNI) technology[6] is considered. The aim of this paper is to present the development of complete MPEG-21 DIA utility software modules, especially for stereoscopic video conversion, that could be applied to other DIA adaptation modules.

Following chapter describes the overall architecture of our DIA utility software. Chapter 3 presents JNI and C/C++ modules and their expansion to stereoscopic video adaptation. The integration of DID and DIA parsers is introduced in Chapter 4. Finally, experimental results that validate the correct implementation of our proposed software are presented in Chapter 5 followed by the conclusion.

Ⅱ. Overall Architecture

Fig. 2 illustrates the overall architecture that is used in JNI-based DIA utility software modules.

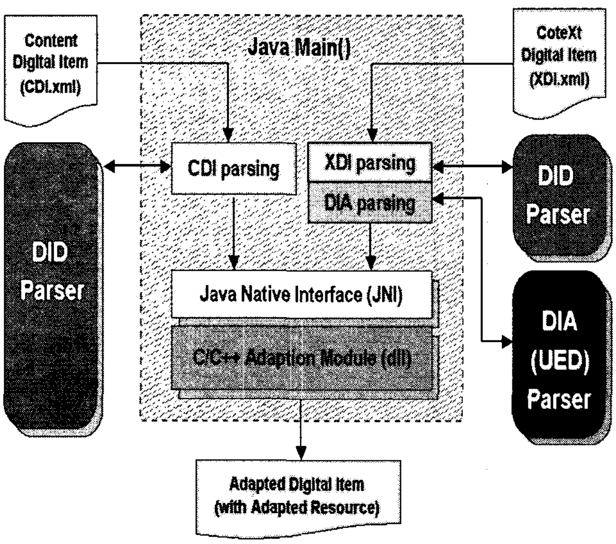


Fig. 2 The architecture of JNI-based DIA utility software

The utility software is composed of Java main, CDI/XDI xml, and DID/DIA parsers. In the Java main, JNI interconnects CDI/XDI/DIA parsers and C/C++ adaptation module that will be described in detail in the next two chapters. The parsing consists of CDI (Content Digital Item) parsing, XDI (ConteXt Digital Item) parsing and DIA parsing. CDI parser parses CDI.xml using DID parser and delivers the semantics of resource descriptors. XDI parser similarly parses XDI.xml using DID parser. Finally, DIA parser processes the outcome of XDI parser using DIA parser and delivers adaptation descriptors to JNI module. Finally, JNI calls C/C++ adaptation module (usually, in dll formats) and sends resource and descriptors. Then, C/C++ module implements actual resource adaptation according to its associated descriptors and returns the adapted resource to JNI.

Ⅲ. Java Native Interface

Java Native Interface (JNI) is the native programming interface for Java that is a part of the JDK. By writing programs using the JNI, it ensuresthat code is completely portable across all platforms. The JNI allows Java code that runs within a Java Virtual Machine (VM) to operate with applications and libraries written in other languages such as C, C++ and assembly. In other words, the JNI serves as glue between Java and native applications. The usage of JNI has been proposed and adopted in MPEG-21.

We present how the stereoscopic video conversion that is one of the MPEG-21 DIA utility software modules written in C/C++ language is integrated into JNI environments[4]. We show that several arguments are defined in Java code and then passed in to a method in native language (C/C++). Fig. 3 is a Java code which includes

```
class Converter {
public native void StereoscopicVideoConversion(
        String jCDIpath,String jXDIpath,int jConversionType,
        String jMotionType,int jStartFrame,int jEndFrame,
String jresourceFilename, String jresourceType,
String jparallxType,double jdepthRange,
int jmaxDelayedFrame,String jranderingFormat);
public static void main(String args[]) {
Converter p = new Converter();
  p.StereoscopicVideoConversion("cdi.xml","xdi.xml",0,"CR",35,37,
        "flower14sec.mpg","MPG","Negative",0.900000,4,"Anaglyph")
}
static {
  System.loadLibrary("Converter");
}
}
```

Fig. 3 Converter.java

```
#include "Converter.h"
#include <jni.h>

JNIEXPORT void JNICALL
Java_Converter_StereoscopicVideoConversion( JNIEnv *env,
jobject obj, jstring jCDIpath,jstring jXDIpath,
jint jConversionType,jstring jMotionType, jint jStartFrame,
jint jEndFrame,jstring jresourceFilename,jstring jresourceType,
jstring jparallxType,jdouble jdepthRange,
jint jmaxDelayedFrame,jstring jranderingFormat )
{
      CString EXEfile, CDIPath, XDIPath;
      CString TempMotionType;
      ...
}
```

Fig. 4 Converter.c

a main method to instantiate the class and calls the native method.

Note that the hard-coded variables are now moved and defined here in Java code. These arguments are passed in to the native method in C/C++ code through the JNI as follows:

p.StereoscopicVideoConversion ("cdi.xml", "xdi.xml", 0, "CR", 35, 37, "flower14sec.mpg", "MPG", Negative", 0.900000, 4, "Anaglyph");

Fig. 4 shows how the arguments from Java code are passed into a C/C++ code.

XDI and CDI are presented followed by the semantics

```
<Description xsi:type="UsageEnvironmentType">
<UsageEnvironmentProperty xsi:type="UsersType">
<User>
<UserCharacteristic xsi:type="DisplayPresentationPreferencesType">
   <StereoscopicVideoConversion>
        <From2DTo3DStereoscopic>
                <ParallaxType>Positive</ParallaxType>
                <DepthRange>4</DepthRange>
                <MaxDelayedFrame>3</MaxDelayedFrame>
        </From2DTo3DStereoscopic>
   </StereoscopicVideoConversion>
</UserCharacteristic>
</User>
</UsageEnvironmentProperty>
</Description>
```

Fig. 5 StereoscopicVideoConversionXDI.xml

```
<Item>
<Descriptor>
<Statement mimeType="text/plain">MPEG Video Item</Statement>
   </Descriptor>
        <Component>
           <Resource ref="flower14sec.mpg" mimeType="MPG"/>
        </Component>
</Item>
```

Fig. 6 StereoscopicVideoConversionCDI.xml

of the stereoscopic video conversion descriptors. For detailed, readers are referred to [17]. The StereoscopicVideoConversionXDI.xml is shown in Fig. 5. As well, Fig. 6 shows StereoscopicVideoConversionCDI.xml.

Table 1 shows the semantics of the stereoscopic video adaptation DS[7]. Note that Stereoscopic Video Conversion supports the video adaptation from 2-D video to stereoscopic video. A user can describe her/his own display presentation preferences by specifying the following descriptors: ParallaxType, DepthRange and

Table 1. The semantics of Stereoscopic Video Conversion DS

Name	Definition
ParallaxType	Identifies the type of parallax being composed of negative and positive parallaxes.
DepthRange	Describes the range of 3D depth perceived by the user
MaxDelayedFrame	Indicates the maximum interval of a previous frame.

MaxDelayed-Frame.

ParallaxType represents the type of the parallax being composed of positive parallax and negative parallax. This description can be used by the resource adaptation of stereoscopic video in order to deliver the perception of 3-D depth. In the negative parallax, the 3-D depth is perceived between the monitor screen and human eyes. On the contrary, the 3-D depth is perceived behind the monitor screen in the positive parallax. DepthRange indicates the range of 3-D depth perceived by the user and is defined as the distance between the monitor screen and the object in 3-D. It applies identically to the positive and negative parallaxes. The amount of DepthRange is varied at [Min, Max], which can be normalized to [0, 1]. For positive and negative parallaxes, shifting a right image to the right direction increases the range of depth. On the contrary, shifting it to the left direction decreases the range of depth.

Ⅳ. Parser Integrtaion

Parsers are integrated based upon the following features: (1) Usage of the DIA descriptors based on XML schema in [2], and (2) Integration of Java-based reference software packages (i.e., DID parser and DIA/UED parser [7]).

DID parser parses StereoscopicVideoConversion-CDI.xml and returns two parameters, resourceFilename and resourceType from <Resource> element in the xml file (Fig. 7). Because the element has all values that we need, we examine if it is 'Resource' or not, and then store it at DIDParam[]. After searching for the resource element, we parse it to obtain the two parameters using 'getRef()' and 'getMimeType()'.

On the other hand, DIA parser parses Stereoscopic-VideoConversionXDI.xml. Every CDI.xml has the same schema but XDI.xml does not. The DIA parser parses every significant element because it has its own special values which specify the type of child elements. In the following example, <Description> element specifies its child element type as "Usage-EnvironmentType". <Description> is one of significant elements. Therefore, the DIA parser should parse all significant elements to have necessary attributes. According to this procedure, the DIA parser returns internal parameters of xml file.

In case of StereoscopicVideoConversion, the parsing order is from <DIA> to <From2DTo3DStereoscopic>. As well, it is possible to use get() functions such as get-ParallaxType(), getDepthRange() and getMaxDelayed-Frame. DIAParam[] stores values of parameters as shown

```
class Converter {

private void getData(DIElement e){
        if (e instanceof ResourceElement) {
            myDIDParam.add(e);
         }
        // print children
        for (int i=0;i<e.getChildren().size();i++) {
        getData((DIElement)e.getChildren().get(i));
```

Fig. 7. The part of DID parser in Converter.java

```
class Converter {
String [] doDIAParse(String DIAParam[]) {  class
        //get the DIA Object
        DIA myDIA = (DIA) myDIDEngine.getDID().getItem()[0].
        getDescriptor()[0].getStatement()[0].getInlineData();
        // get the first Description
        Description theDescription = (myDIA.getDescription())[0];
        // check if the description contains a child of the UsageEnvironment
        if (!theDescription.getType().equals
             (DIA_UED_Definitions.XSITYPE_USAGEENVIRONMENT))
        {
          return null;
        }
        // get the first UsageEnvironment Object UsageEnvironmentProperty
        theUE = ( ( (UsageEnvironmentType) theDescription.getChild()).
                  getUsageEnvironmentProperty())[0];
        // check if the element is of type Users
        if (!theUE.getType().equals (DIA_UED_UC_Definitions.XSITYPE_USERS))
        {
                  return null;
        }
        if (theUE.getType().equals (DIA_UED_UC_Definitions.XSITYPE_USERS))
          // get the User Object
        {
          UsersType theUsers = (UsersType) theUE.getChild();
          UserType theUser = theUsers.getUser()[0];
          // get the UserCharacteristics
        UserCharacteristic theUserCharacteristic = theUser.getUserCharacteristics()[0];
        // check if the type the DISPLAYPRESENTATIONPREFERENCES
        if (!theUserCharacteristic.getType().equals
                  (DIA_UED_UC_Definitions.XSITYPE_UC_DISPLAYPRESENTATIONPREFERENCES))
        {
          return null
        }
        DisplayPresentationPreferencesType theDPPT = (DisplayPresentationPreferencesType)
                                                     theUserCharacteristic.getChild();
        //get the StereoscopicVideoConversion object StereoscopicVideoConversionType
        theSVCT = theDPPT.getStereoscopicVideoConversion()[0];
        //get the From2DTo3DStereoscopic object From2DTo3DStereoscopic
        theF2DT3D = theSVCT.getFrom2DTo3DStereoscopic()[0];
```

Fig. 8 The part of DIA parser in Converter.java

in Fig. 8.

In addition, parameter.txt stores parameter values that are defined by user for controlling the output data such as the frame numbers converted, a motion type and so forth. Note that those do not belong to a normative part of DIA standard. They are used only for obtaining necessary information on the user side.

V. Experiments

In this chapter, we present the output adapted images displayed from carrying out our presented utility software. One of stereoscopic video conversion schemes is to make use of a previous (delayed) image. Suppose the image sequence is {, I_{K-3}, I_{K-2}, I_{K-1}, I_K, } and I_K is the current frame. One of the previous frames, I_{K-i} ($i \geq 1$) is chosen. Then, a stereoscopic image consists of I_K and I_{K-i}. If the current and previous images are appropriately presented to both human eyes according to camera and object motions as shown in Table 2, the user then feels the stereoscopic perception. MaxDelayedFrame determines the amount of i value. Thus, the larger it is, the more depth the user feels. ParallaxType is controlled by switching the left and right images. For varying values of DepthRange, a left image is horizontally shifted to the left direction.

Fig. 9 shows the printed result. The appropriate stereoscopic images are generated according to the values of

Table 2. Motion types and left/right image selection

Camera motion	Object Motion	Left Image	Right Image
Right	None	Previous	Current
Left	None	Current	Previous
None	Right	Current	Previous
None	Left	Previous	Current

```
XDIFilename :: StereoscopicVideoConversionXDI.xml
CDIFilename :: StereoscopicVideoConversionCDI.xml
DIDParam[0]=StereoscopicVideoConversionCDI.xml
DIDParam[1]=flower14sec.mpg
DIDParam[2]=MPG
DIDParam[3]=CD2Soccer.avi
DIDParam[4]=AVI
DIAParam[0]=StereoScopicVideoConversionXDI.xml
DIAParam[1]=Positive
DIAParam[2]=4
DIAParam[3]=3
parameter[0]=MotionType :: CR
parameter[1]=StartFrame :: 0022
parameter[2]=EndFrame  :: 0027
parameter[3]=RanderingFormat :: Interlaced
Input command:
StereoscopicVideoConversionCDI.xml
StereoScopicVideoConversionXDI.xml
0 CR 22 27
```

Fig. 9 Output from the utility software

Fig. 10 Horizontal motion images. DepthRange is varied

(a) (b)

Fig. 11 Horizontal motion images. (a) Positive parallax and (b) Negative Parallax.

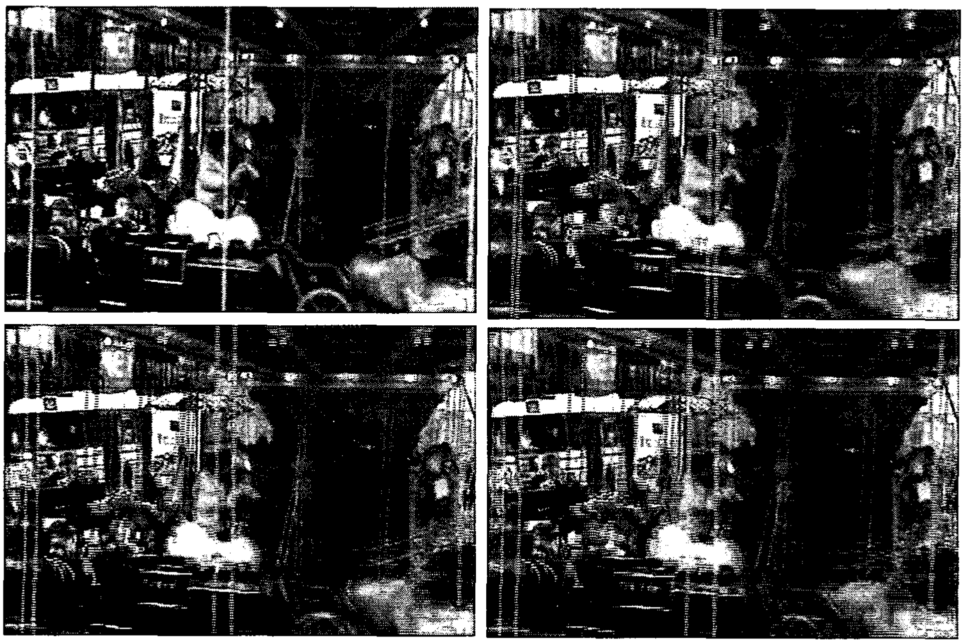


Fig. 12 Horizontal motion images. MaxDelayedFrame is varied in [1, 4].

DIAParam[]. Fig. 10 shows the interlaced stereoscopic images of 2D MPEG Fun sequence, where a merry-go-round is horizontally moving to the right. DepthRange is controlled by shifting the right image. A current image and a previous image are viewed to the left and right eyes, respectively. Since the image has negative parallax, 3D depthis perceived between a monitor screen and eyes. The parallax types being composed of positive and negative parallax are demonstrated in Fig. 11, where two images are switched for parallax exchange. As shown in Fig. 12, it is observed that as the MaxDelayed-Frame increases from one to four, the disparity between the left and right images accordingly increases, thus resulting in the greater 3D depth

Ⅴ. Conclusion

In this paper, we have presented the implementation of MPEG-21 DIA utility software and its application to stereoscopic video conversion. For this, JNI is adopted as an interface to DIA C/C++ adaptation module and DID and DIA parsers are integrated in our software. Furthermore, our developed software has been applied to other DIA modules such as AudioPresentationPreferences, Audio-SoundField, GraphicsPresentation-Preference, Audio-Impairment, VisualImpairment, ColorVisionDeficiency, and AudioEnvironment for its validation and it has been reported that all of them work correctly.

References

[1] "Information Technology Multimedia Framework- Part 1: Vision, Technology, and Strategy", ISO/IEC JTC1/SC29/WG11, March 2001.

[2] ISO/IEC 21000-7 FDIS Part 7: Digital Item Adaptation, ISO/IEC JTC1/ SC29/ WG11/N6168, Dec. 2003, Hawaii, Hawaii.

[3] MPEG-21 DIA Software Implementation Plan v.7, ISO/IEC JTC1/SC29/WG11/ N6414, March 2004, Munich, Germany

[4] "MPEG-21 DIA Description Tools for 3D Stereo- scopic Video Adaptation", ISO/IEC JTC1/SC29/ WG11, M8613, Klagenfurt, Austria, July 2002.

[5] Ghent University, "Digital Item Adaptation Usage Environment Descriptor Tool Parser," ISO/IEC JTC1/ SC29/WG11/m10387, December 2003, Waikoloa, Hawaii

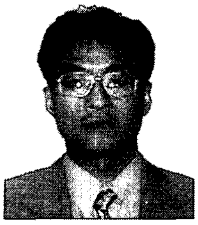
[6] Java Native Interface, http://java.sun.com/j2se/1.3/docs/guide/jni/, Sun Microsystems, Inc.

[7] Manbae Kim, et. al., "The adaptation of 3D stereoscopic video in MPEG-21 DIA", Signal Processing: Image Communication, Vol. 18, Issue 8, pp. 685-697, Sep. 2003.

저 자 소 개

Hyukmin Kwon

- Feb. 2001 : B. SC, Dept. of Info. and Telecom, Kangwon National Univ.
- Feb. 2004 : M. SC, Dept. of Info. and Telecom, Kangwon National Univ.
- Research : Image compression, MPEG-21 DIA



Manbae Kim

- 1983 : B. SC, Dept. of Electronic Eng., Hanyang Univ.
- 1986 : M. SC, Dept of Electrical Eng., University of Washington (Seattle)
- 1992 : Ph. D, Dept. of Electrical Eng., University of Washington (Seattle)
- 1993 : Georgetown University Visiting Scientist
- 1996 : University of Rochester, Visiting Scientist
- 1992~1998 : Senior Research, SAIT
- 1998~presence : Associate Prof., Kangwon National Univ.
- Research : Stereoscopic image processing, Realistic broadcasting system, MPEG-21, Video processing