BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

HDR VIDEO PLUG-IN FOR ADOBE PREMIERE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. LUKÁŠ SVATÝ

BRNO - 2015

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HDR VIDEO PLUG-IN FOR ADOBE PREMIERE

HDR VIDEO PLUG-IN PRE ADOBE PREMIÉR

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. LUKÁŠ SVATÝ

SUPERVISOR
VEDOUCÍ PRÁCE

PAVEL ZEMČÍK, prof. Dr. Ing.

BRNO - 2015

Abstract

The goal of this thesis is to enable support for editing HDR video. For this purposes, Adobe Premiere Pro was chosen and to achieve specified results, plug-in to Adobe Premiere Pro was developed. This thesis describes fundamental points of creating, displaying and storing high dynamic range content. Principles of plug-in development for Adobe Premiere Pro CS6 SDK are mentioned. Details of implementation, problems solved during the development process and description of the final plug-in providing support for HDR content inside Adobe Premiere Pro in the part of this thesis dealing with implementation. The design was done based on future development on this plug-in, adding additional functionality and for usage of this thesis on future progress in high dynamic range field.

Abstrakt

Cílem práce je vytvoření podpory pro editovaní videa v HDR formátu. Pro editaci videa je zvolen program Adobe Premiere Pro a na dosažení požadovaného výsledku je vytvořen plugin do zmiňovaného softwaru, který poskytuje požadovanou funkcionalitu. V práci jsou vysvětleny principy vytváření, zobrazení a ukládáni obsahu z rozšíreným dynamickým rozsahem. Zároveň jsou vysvětleny principy vytváření pluginů pro Adobe Premiere Pro za pomoci SDK verze CS6. V praktické části této práce jsou vysvětleny detaily implementace, problémy, které byly řešené, a popis samotného pluginu. Návrh pluginů je vytvořen tak, aby byla možná další práce na tomto softwaru, přidaní další funkcionality a pro umožnění využití tohoto díla na rozvoji obsahu s rozšíreným dynamickým rozsahem.

Keywords

Adobe Premiere, video processing, High dynamic range, HDR video, Adobe Premire SDK, plug-in development

Klíčová slova

Adobe Premiér, spracovanie videy, rozšírený dynamický rozsah, HDR video, Adobe Premiér SDK, vývoj plug-inov

Citation

Lukáš Svatý: HDR Video Plug-in for Adobe Premiere, master's thesis, Brno, FIT VUT v Brně, 2015

HDR Video Plug-in for Adobe Premiere

Statement

Cílem práce je vytvoření podpory pro editovaní videa v HDR formátu. To

....
Lukáš Svatý
July 31, 2015

Acknowledgement

II take this opportunity to express gratitude to my supervisor Pavel Zemčík, Prof. Dr. Ing. for his help and support. I also thank Professor Alan Chalmers for the guidance and support as a consultant for the practical part of this thesis.

Contents

Chapter 1

Introduction

Since 19th-century people are trying to store visual memories by photographies and video files. As the technological change progresses, researchers are attempting to achieve the same visual effect of images and video files captured as the human eye can see on the real life scenery. To solve this problem, high dynamic range content was created. HDR content is trying to expand the visual effect of photographies and movies to simulate the visual effect of real life scenery. This thesis introduces method of video editing of this kind of files so they can be used in film industry, scientific capturing of medical operations with better visual properties of the video or virtual reality displaying content so similar to real life world that human brain would not notice the difference.

Adobe Premiere Pro is a video editing software developed by Adobe Systems. This closed-source application is a successor to older version Adobe Premiere and was introduced to the market in 2003. Adobe Premiere was one of the first software that enabled video editing in a non-destructive way, where a video could be cut, and effects could be added without editing the source video. In this thesis, a plug-in for Adobe Premiere Pro is introduced which enables the functionality of high dynamic range videos.

Dynamic range is a velocity which describes the difference between darkest and lightest parts of an image. High dynamic range or HDR is expanding this dynamic range above the dynamic range of regular cameras and displays such as LCD or CRT. By expanding this range, high dynamic range images can simulate scenery which can be viewed by a human vision system. This way more realistic images can be acquired.

High dynamic range images can be acquired by many different ways. One of them is a technique developed in nineteenth century which is using merging of various images captured of the same scenery and composing them in a way that only parts of image which appears the same as human eye would see them are displayed. Pictures were created by using two different capture methods for the sky and for the sea.

In the twentieth century, more techniques to high dynamic images were introduced such as tone mapping. This technique reduces the dynamic range of image while trying to persist the local contrast and details of images. As high dynamic range images were becoming more popular, demand for high dynamic range video has started. This introduced new problems in capturing of high dynamic range content as well as storing it. These are being solved by the continuous research in the high dynamic range imaging field, and some of them are explained in this work.

This thesis introduces plug-in which enables support of high dynamic range videos in Adobe Premiere Pro. On current market only editing tools which are available for public use are working with images. In case of video only single frames could be edited. Because of this and progress of high dynamic range techniques for images and video it is necessary to have a solution to video editing which is provided by introduced HDR plug-in for Adobe Premiere Pro.

It is necessary to understand that there are many limitations mainly connected with the progress of high dynamic range imaging. Big companies and standards are trying to introduce high dynamic range imaging for public use however the process is time demanding and the market changes slowly. In future, there is a considerable amount of development which is waiting for new formats which can efficiently store HDR content or release of new software development kits with support current or future formats. The first step is to make high dynamic range imaging available for everyone by creating more and more high dynamic range content. This way companies designing hardware for the high dynamic range such as displays and cameras will have to lower the prices of these products which will make them available for public usage. At this moment, the high dynamic range will be available for everyone and development process on software and editing tools for HDR content will speed up rapidly.

However, some software can be developed prior to these future changes. As some high dynamic range video files are currently available, it is necessary to provide software for video editing. Creating this type of plug-in will push on support for high dynamic range images and videos in more editing tools, and this will again lead to higher demand for high dynamic content. In the future version of Adobe Premiere Pro CC, it will be available to create full support for high dynamic range video files with their export in HDR format. Once we are able to efficiently edit video files in high dynamic range format it is a matter of time when high dynamic range will be fully supported by movies, virtual reality, computer games or simulation of human visual system.

This thesis consist of three chapters necessary for understanding the state of the art used, design and implementation if this plug-in. In the first chapter, necessary terms and concepts are introduced for understanding design and development process. This chapter describes the state of current high dynamic range imaging, methods of generating HDR content and solutions for displaying high dynamic range content. Next it is necessary to explain the current state of Adobe Premiere Pro SDK which supports different kind of plug-ins. The second chapter deals with the design for this plug-in as well as usage of current state of technologies and techniques mentioned in the first chapter. Last chapter explains implementation details and works done on the HDR plug-in. The overall structure of the plugin is described, with problems and solutions to them reached during the development process. Finally, the plug-in itself and its functionality is mentioned here. Last part of this document is a conclusion of the work creating a summary of all the work done in this thesis as well as all the chapters mentioned before. A lot of future work could be done on this topic, and more development awaits in the future.

Chapter 2

State of art

This chapter introduces the current state of the art having an impact on this thesis. Necessary terms and concepts are described for understanding design and development process. This chapter defines the state of current high dynamic range imaging, methods of generating and displaying HDR content. After this state of Adobe Premiere Pro SDK by different plug-in types is mentioned. The appropriate type of plug-in must be chosen for the development process which will enable the desired behavior.

First basics about high dynamic range are explained. It is necessary to understand the concept of dynamic range and the difference between low and high dynamic range imaging. The notion of capturing and displaying of HDR content is needed. Since there are many different solutions for high dynamic image capturing, only few most used concepts are explained. The same applies for high dynamic range image capturing. In both capturing and displaying high dynamic range content, there are two different approaches. The first choice is to work with more expensive hardware that can capture and display high dynamic range content natively. On the second hand it is possible to simulate techniques of high dynamic range content with software such as tone mapping high dynamic range images to be displayed on LCD or capturing high dynamic content with low dynamic range camera with multiple captured images with different exposures. Finally, high dynamic range content must be stored efficiently. The end of this section describes concepts about high dynamic range encoding of images and video files.

The second part of this chapter is dealing with Adobe Premiere Pro development process for plug-ins. Adobe Premiere Pro in version CS6 supports nine different types of plug-ins, which are appropriate in various parts of a video editing pipeline. The necessity for these plugins is their registration to Adobe Premiere Pro and definition of resources used. This section also describes two of most critical resources used by plug-ins that are also used and later mentioned in this thesis. Pixel formats provide Adobe Premiere options for storing content in a most efficient way, based on different factors like compression or rendering times. The notion of time in Adobe Premiere is another necessary component of the development process as different plug-in types use a different notion of times specified by Adobe Premiere Pro.

This chapter is not meant to be a walk through all the information for high dynamic range imaging and Adobe Premiere Pro plug-in development process. For this, all the literature with more information is cited and listed at the end of this thesis. This chapter explains only necessary terms and concepts that are essential to understanding the design and development process for software included in this thesis.

Video editing pipeline

This thesis mentions the video editing pipeline in various chapters. This pipeline consists of few necessary steps that needs to be considered in every video editing and processing and can be seen in figure 2.1.

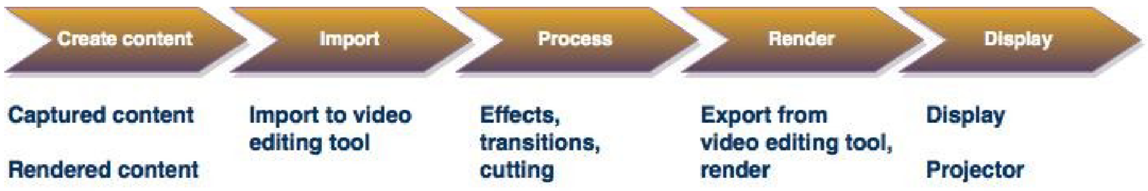

Create content
Import
Process
Render
Display
Captured content
Rendered content
Import to video editing tool
Effects, transitions, cutting
Export from video editing tool, render
Display
Projector

Figure 2.1: Video editing pipeline

The first step consists of generating video content. This can be achieved by two different ways. First one is capturing by a video recorder, which in case of high dynamic range videos can be HDR camera or low dynamic range camera and software merging multiple images. The second used technique is software rendering by software which can create video files.

Next, it is necessary to import this video file to editing tool, in our case Adobe Premiere Pro. Editing tool must support video format and pixel format of captured or rendered video.

The third step is the actual video processing and editing inside the video editing tool. This is many times the most complicated process as it requires human interaction.

After the video is processed by editing tool. It is exported to the output format. This format can be same as the captured format or any supported format available for export by editing tool.

Last step of this pipeline is the actual displaying of the content. As in this thesis, only high dynamic range content is considered it is necessary to solve this problem of displaying HDR video. This can be achieved either by using specialised software for playback of HDR video which is tone-mapping this video for low dynamic range displays or by using HDR screen for direct displaying.

2.1 High dynamic range

This section introduces basic principles of High dynamic range imaging. First it is necessary to understand what dynamic range is. Easily said the dynamic range is a range of light on the image. Currently, a human eye can see a much larger range of light in real life then current devices can capture and display and this problem can be solved by high dynamic range imaging.

Few points of interests rise with the topic of high dynamic range content. First one is capturing of HDR content. This can be acquired either by a specific camera for HDR capturing or software rendering of low dynamic range images by their composition.

But if we acquire and efficiently store high dynamic range content is it useful without representing the actual content? This problem can be solved by two different options. First one is the hardware kind of view where new HDR displays are being created which can represent high dynamic range. Second is at the moment more user-friendly as no need for additional purchase is required. High dynamic range content can be efficiently mapped to low dynamic range to be presented on current LDR devices.

The mentioned problems are described in this section. After the detailed explanation of dynamic range, techniques of high dynamic range capturing and storing, in the last part high dynamic range encoding is described.

Dynamic range

Dynamic range is a dimensionless quantity that may relate to different physical measurements.

In digital photography, dynamic range is measured as a ratio between darkest and lightest pixel of the image. Value based on this definition can be non-informative, because of the pixels choice. Many times darkest and lightest pixels are extreme values compared to average pixel lightning value [31]. More accurate measurement of dynamic range in digital image or photography can be achieved by excluding few percent of brightest and darkest pixels.

For displays, dynamic range is defined as a ratio between the most and the least luminance which the display can emit [32].

The dynamic range of a digital camera is based on a ratio of luminance that saturates the sensor and the luminance that lifts the camera response to one standard deviation above the noise floor [31].

Dynamic range can be represented in two different ways. In most literature, it is represented as ratio mentioned above. However, different representation using exposure value is sometimes used when we are talking about a dynamic range of cameras and high dynamic range capturing [16].

Ratio representation

Ratio 1:100 describes that the lightest pixel of a digital image is hundred times lighter than the darkest pixel. This representation is mostly used for pictures or devices displaying low or high dynamic range images or video. For easier representation, many times ratio is not used and instead an order value is mentioned. For example ratio 1:100 is a 2nd order dynamic range.

Exposure value representation

Exposure value is a photographic quantity describing the amount of light that is received by camera sensors [29]. This value can be defined by a combination of camera's shutter speed and f-number. Shutter speed is the length of time a camera's shutter is open while taking a photograph. By this camera can limit an amount of light camera sensor will receive [21].

The f-number is the ratio of lens's focal length to the diameter of the entrance pupil [33].

Zero exposure value is defined by International Organization for Standardization (ISO) as exposure value with f-number f/1 and a shutter speed of 1 second. Increasing or decreasing the exposure value by one causes doubling the amount of received light or dividing by two respectively. This can be achieved by two separate ways. Either by multiplying or dividing the shutter speed by two or modifying the f-stop number by dividing or multiplying by $\sqrt{2}$. This causes the scale of exposure value to represent a dynamic range in the logarithmic way.

Both ratio representation and exposure value representation are easily transformed into another one as $2^{EV} = k$, where k is a ratio and EV is exposure value of a dynamic range.

Real life dynamic range

Real life has much bigger dynamic range than what we are able to capture with general photography. The human vision system can receive and adapt to the light condition changing by ten orders of luminance [12] and in one scenery human eye can capture up to 5 orders of dynamic range at the same time. For example sun during maximum brightness at noon is 10^{12} times brighter than starlight. Values for different lighting conditions and surroundings are mentioned in the table 2.1.

	Ratio	Order	Exposure value
Range of noon sunlight to starlight	1: 1 000 000 000 000	12	40
Human eye vision after adaptation	1: 1 000 000 000	9	30
Sunlight on scenery	1: 1 000 000	6	20
Human eye vision in a scenery	1 : 100 000	5	17
LCD display	1 : 350	2	8-9
CRT display	1 : 200	2	7-8
Digital camera	1 : 100	2	7
Printed photography	1 : 50	1	5-6

Table 2.1: Dynamic range of devices and sceneries

As seen in table 2.1 many standard devices are representing a dynamic range in mostly 2nd order of dynamic range. As human eye system can receive and recognize 5 to 9 orders of dynamic range, it is clear that devices such as LCD or CRT display are either losing quality or displaying lower quality images. Due to this limitation it is not effective to store high dynamic range images just for representation on these devices and because of this most cameras are storing images in lower quality mostly using 8 bits per pixel per channel. This is sufficient for storing and also displaying on these devices.

To display high dynamic range images that would be able to represent real life sceneries with a higher order of dynamic range we need displays that can display high dynamic range.

At the moment, we have the technology that can capture high dynamic range images that are more realistic, that are called HDR cameras mentioned in section 2.1. This would be however not efficient as high dynamic images are saved with higher quality than mentioned 8 bits per pixel per channel. For displaying images stored with high dynamic range we need to lower the range of images to low dynamic images such as LCD or CRT displays by tone mapping HDR image to LDR which is described in section 2.1 or use HDR display mentioned in section 2.1.

HDR image capture

Generation of high dynamic range images is based on few different methods such as acquiring HDR image by combining images of multiple exposure values. Easiest but most expensive is acquiring an HDR camera that can capture HDR images or video natively. The second method of producing HDR images is by software rendering artificial scenery in higher dynamic range than regular images. However, as these two methods can create new images with a bigger dynamic range, we also want to port legacy images of low dynamic range (LDR) to more suitable format for representation of HDR displays.

Multiple exposure

Current cameras for digital image capturing can capture a dynamic range of 9-10 EV, more professional devices up to 13 EV. This fact is limiting factor for the creation of HDR images. However moving their dynamic range with different exposers enables photographers to manipulate images to a wider dynamic range.

The idea of using multiple exposers for high dynamic range images is based on the fact that composing multiple images with smaller dynamic range can widen the range of composed image [26]. To achieve this, multiple (2, 3 or 9) images with different exposures are taken from under-exposed up to over-exposed images. Each of this images will have specific parts correctly exposed. Composing parts of images with the correct exposure and eliminating pixels that are either over-exposed or under-exposed will result in creating an image with correct exposure value on every part of the image. Due to the fact that captured images were taken with a different exposure, resulting image had a wider range than a regular image taken from the camera (high dynamic range).

This technique of composing HDR image from multiple low dynamic range images is a cheap and useful version for static images. However for shots of moving objects or capture of the video some problems may occur. Because of the composition of the final image from multiple images, these images should be preferable of static objects, and all the pixels should align. In case of movement between capture of images with different exposure noise and ghosts may appear [7]. At the moment multiple de-ghosting algorithms exists, however, the effectiveness of them is not absolute. De-ghosting is a complicated process that takes a high amount of time. Because of this it is not used very much in video capturing. More suitable devices that can capture HDR directly are better for capturing moving objects without creating ghosts or noise.

Direct HDR capturing

At the moment, there are many ideas and prototypes of HDR cameras that can capture a high dynamic range of images directly. This field of study is pushed forward by constant demand on HDR videos which being captured by multiple exposure techniques are causing too much noise.

Devices capturing HDR images directly are based on many different techniques. Two of the proposed solutions are mentioned in HDR camera section. Cameras supporting this type of capture are more expensive than regular LDR camera because of additional hardware they need for support of high dynamic range condition.

Expanding dynamic range of legacy images

A standard analog black-white negative can capture images with 13-14 EV. Because of this we can capture standard dynamic range images with an analog camera, but print them only on camera tape that can store up to 7 EV. The idea behind this technique is scanning these images from an analog camera with a scanner [5]. However, scanners can scan only low dynamic range images and for creating high dynamic range images they are inadequate. The same way as multiple exposures capturing works, scanners might be able to scan these images multiple times with varying exposures and compose high dynamic range image.

High dynamic range devices

Market with capturing and displaying devices has every year higher demand on devices able to capture and display high dynamic range content. Big companies like Dolby digital are rather pushing forward HDR displays than expanding the resolution of the screen to Ultra HD and above. This causes big filmmaking companies to produce more and more content in high dynamic range quality, which is also able to be displayed on LDR displays. With movies being created in the future in HDR quality, HDR displays will be in the future more available to public usage.

HDR displays

BrightSide Inc. release prototype of HDR display that is now also accessible to the public. This device is designed to combine features of LCD and LED and by that expanding dynamic range to HDR [30]. LCD handles colors on output while a layer of LEDs under LCD handles luminance of pixels. While regular LCD has on average 8-9 EV, BrightSide's technology can produce images with approximately 18EV.

A new development in HDR displays brought us also a technology called *Surface-conduction electron-emitter display* (SED) [23]. This technology was developed by Canon and Toshiba and is based on the idea of emitting electrons by sensors with few nanometers length. These displays can display images with a range of 17 EV.

HDR cameras

High dynamic range capturing devices can directly capture images of high dynamic range. Multiple ideas were proposed for solution of this problem, and this section will mention two of them.

Fujifilm Holdings Corporation in 2003 introduced *Super CCD SR* with extended dynamic range [3]. The small secondary sensor was added to every sensor in a camera. This secondary sensor has smaller sensitivity and is used for acquiring light information while a larger primary sensor is capturing a regular image. Information from both of these sensors is merged to single image with high dynamic range. The dynamic range of this device is approximately 13 EV, which is much greater than a regular digital camera with approximately 7 EV.

Next to *Super CCD SR* were another camera designed based on the idea of capturing multiple exposures at once with multiple sensors [28]. This is done by placing a prism that divide light into multiple sensors, which are set to capture different EV. Images from all sensors are afterward composed into single HDR image. This process is very effective. However, the technology behind it must be detailed, and these devices are too expensive for public usage.

Tone mapping

It is clear that dynamic range of real life scenery is much higher then what are regular displays able to display. Using techniques discussed in previous sections, we are able to capture high dynamic range images. Unfortunately, regular displays are not able to display high dynamic range images. This is a problem which needs to be solved. Mapping high dynamic images to be presented on a low dynamic range display is possible by tone mapping [4]. By tone mapping content acquired by high dynamic range, we are trying to find the closest visual appearance on presented image on display to real life scenery. Because of the fact that lightning intensity of real life scenery is much greater than what display are able to achieve, tone mapping is used to lower the dynamic range of high dynamic range images. The problem of the visual appearance of scenery on LDR display is defined by the level of illumination and range of contrast [10]. Some examples are *„Scene appears more colorful and contrasty on a sunny day,“ „Colorful scenes of the day appear gray during the night,“ „Moonlight has a bluish appearance.“* Because of this is a simple linear transformation of high dynamic range to a dynamic range of regular displays not usable. Using linear transformation images would appear dark, and details of images would be lost. The overall linear transformation would change the visual appearance of the scenery. Solution to this problem is a usage of visual models for better representation, which can be seen in figure 2.2 [37].

The goal of this is to simulate the same visual effect of real life scenery on LDR display.

This problem is being solved by tone mapping operators. These operators are trying to simulate the human visual system. The human visual system is able to differ between local contrast and global contrast. When perceiving details of image global contrast of scenery is suppressed. This way details are not lost because of the higher dynamic range of the image.

Global operators

These operators are mapping same function to every pixel the same way. Pixels with same value stays the same after the tone mapping. Global operators are based on transformation by the tone curve. This curve can be based on many different techniques such as computed from the histogram or calculated for each color channel separately. These operators are used for middle dynamic range images [22].

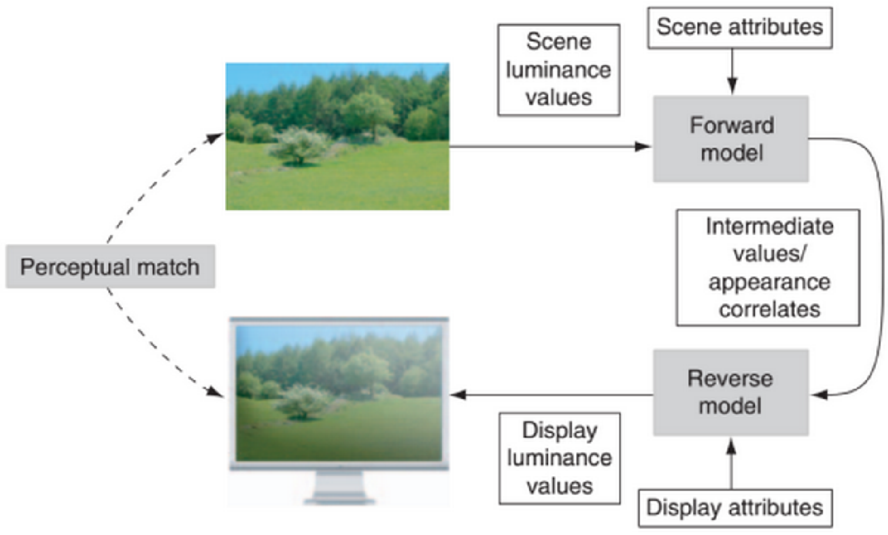


Figure 2.2: The goal of the tone-mapping problem are visually matching images of the real life scenery and HDR image displayed in LDR display. (Image taken from [31])

Local operators

Local operators are processing groups of pixel but not all the pixels at once. They were created on the concept of human visual system, which is identifying local contrast, but at the same time trying to suppress global contrast. Local operators are taking into account surrounding of selected pixel. Moving exposure value based on this selective process is creating better results when tone mapping images with significant contrast range [35].

Frequency domain operators

These operators are working on reduced domain of dynamic range. This domain is reduced by the spatial frequency of the image region. Converting data in the image to frequencies which are simulating contrast in details. Operators are manipulating content with low and high frequency [25].

Gradient domain operators

Gradient domain operators are based on same principle as frequency domain operators. Change in gradient is used for discovering a change in global and local contrast. Bigger changes in global contrast are defined as long and steep slope while a big difference of local contrast in details can be seen as a gradual slope of shorter range [22]. Mentioned operators can be seen in a figure 2.3 which was taken from [1]

High dynamic range encoding

In this section few fundementals for storage and encoding of high dynamic range images and video content is presented. Storing of images is a necessary factor to be considered

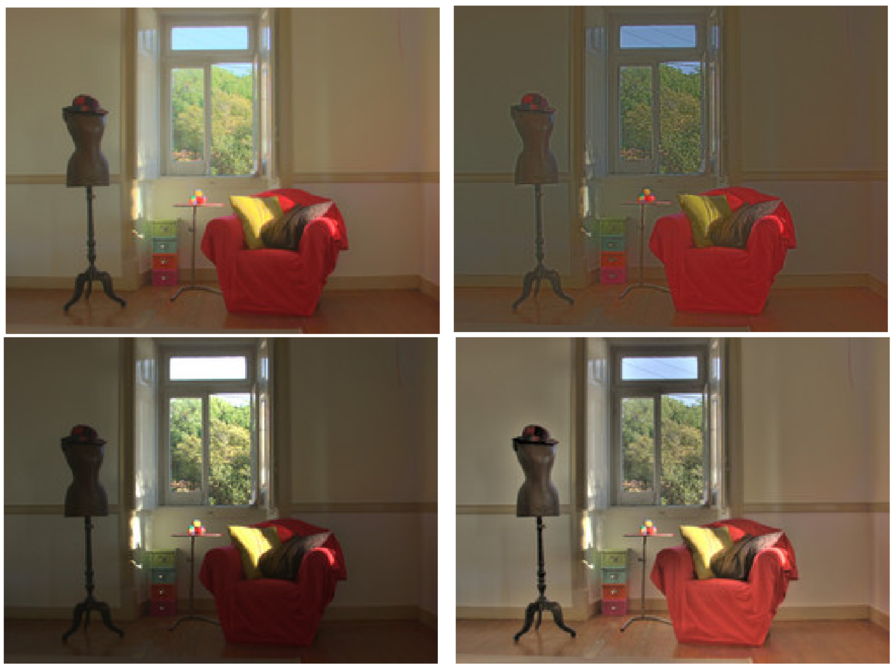


Figure 2.3: Four tone mapping operators applied to same high dynamic range image. On top left image, global operator is used (Drago), top right is using local operator (Pattanik), bottom left is using frequency domain operator (Durand and Dorsey), and finally, bottom right is using gradient domain operator (Fattal)

in photography industry. This is even bigger problem for high dynamic range images as they are. In the high dynamic range point of view it is necessary to take into account that HDR image or frame needs to have stored more bits (24-bits per channel) than regular low dynamic range image.

Also it is to necessary to consider lossy and lossless compression factor. High dynamic range images should be stored in lossless format to keep all the details. However this is not always the fact as sometimes efficient storage size is a necesasry component to be taken into account. Video files are generally stored in lossy format. This is because storing few thousands of raw uncompressed frames will increase the video size massivly.

High dynamic range encoding

In this section, few fundamentals for storage and encoding of high dynamic range images and video content is presented. Storing of images is a necessary factor to be considered in the photography industry. In the high dynamic range point of view, it is necessary to take into account that HDR image or frame needs to have stored more bits (24-bits per channel) than regular low dynamic range image.

Also, it is necessary to consider lossy and lossless compression factor. High dynamic range images should be stored in lossless format to keep all the details. However, this is not always the fact as sometimes efficient storage size is a necessary component to be taken into account. Video files are generally stored in a lossy format. This is because storing few thousands of raw uncompressed frames will increase the video size massively.

High dynamic range image encodings

Images are trying to store as much data as can be represented on displays. Becuase of this not all the light projected from scenes are stored. It is not required to store light in infrared or ultraviolet wavelength as this light is not visible for the human eye. As colors for a human eye can be represented by a combination of three different colors this representation is used. This may be defined by the RGB space or any other (e.g., CIE XYZ, YC_BC_R also later mentioned as YUV, CIELUV, etc.) [31].

Difference in storage of images depends on type of application, in databases storing images compression is the critical factor, in application sending images by network file size is the vital property, in photography industry accuracy is required.

Digital photographies are commonly using RAW formats. RAW format is specific file format for the camera and its software. Manufacturers are storing content in their design format, mostly without applying any video editing.

In video editing, multiple applications are able to edit high dynamic range images. For example Photoshop, since CS2 has included support for 32-bit pixel data [40]. This support is also included in applications like Cinepaint [6].

Other applications of HDR images are computer games where tone mapping is used to acquire most realistic effects or visual reality to simulate realistic sceneries.

Mentionable are few image formats that are able to store high dynamic range content without losing the quality. As this is necessary factors, sometimes efficient storage is required, and some lossy format might be needed.

The HDR picture format (or Radiance picture format) was introduced for efficient storage of HDR photographies and image based lighting [9]. Pixel data comes in 4-byte RGBE and CIE variant, XYZE.

Tagged Image file format support 32-bit per component floating point RGB encoding[2]. This representation is able to cover 79 orders of magnitude in miniscule steps. On this other hand, this image format takes three times more space than a regular image format. For efficient storage, it is necessary some compression which will lower the size of these files, which is a complicated process. Nevertheless, this lossless 96-bit per pixel is used as it is easy to use by many different video capturing devices.

The openEXR standard introduced Extended Range format in 2002 [24] which is based on 16-bit Half floating-point type similar to IEEE float with fewer bits. Each pixel is taking 48-bits in total. OpenEXR also supports 32-bit and 24-bit per channel pixel format, but they are not used as much.

There are also some lossy HDR formats which might be a unrequired factor in many applications for example when the image is required to go through the editing process with many steps of storing and retrieving.

Dolby's JPEG-HDR format is one of the lossy formats for high dynamic range content using 8-bit JPEG standard [39][38]. This format is storing tone mapped version of HDR image and storing important metadata within wrappers.

Another format for lossy HDR storing is XDepth format presented on www.XDepth.com, which uses a similiar logic of storage as JPEG-HDR.

High dynamic range video encodings

Video in the high dynamic range is required more and more in many different fields. Imaging sensors in medical applications produce a video with at least 10-bits. Digital cinema is using high dynamic range frames because of desired contrast in them and luminance range.

Special effects for movies are being rendered in high dynamic range to achieve a better visual result. In computer games, HDR video can be used for relighting of scenes or for realistic textures.

In all the applications of high dynamic range video, it is necessary to store this video with lossy compression. This is because of storage and bandwidth limitations.

MPEG-4 Part 2 Visual enables encoding of luminance and chromaticity up to 12 bits per color channel [34]. The most used international standard, H.264/AVC uses 10-bit video, and High 4:2:2 Profile can handle 12-bit coding. These formats allow frames to be stored with higher precision, but they were not meant for storage of high dynamic range content.

Many Custom HDR Video Codings are using custom transform to convert floating-point HDR pixel into multiple integers supported by the video format. By applying the inverse operation, the original HDR pixel can be acquired. This is possible because of MPEG-4 standard which 12-bit per channel encoding. It is clear that high dynamic range would not be able to be stored in 12-bits so a different color space must be created for this storage. For this YC$_B$C$_R$ compression can be used. Pixels have to be converted to this pixel format from RGB at the initial stage of MPEG encoding.

2.2 Adobe Premiere Pro SDK

Adobe Premiere Pro APIs support different types of plug-ins, which can be used on various points of the editing pipeline. This thesis is based on the documentation for Adobe Premiere Pro SDK CS6 [19][18][36].

The required development environment differs based on the chosen platform. For Windows Vista 64-bit, Windows 7 64-bit and Windows 8 64-bit Microsoft Visual Studio .NET 2010 is preferred. For Mac OS 10.6 the required environment is XCode 3.2.

Adobe Premiere Pro SDK provides multiple sample projects for mentioned development environments. The sample code is written in C++ and is essential for understanding different types of plug-ins. Another necessity for Adobe plug-in development is understanding the basic video editing pipeline and basic video topics such as resolution, frame rates, field interlacing, pixel aspect ratio, bit depth, timecode, compression and color spaces.

Suites

Adobe Premiere Pro SDK provides numerous suites for the easier development process. Suites are modules that provide additional functionality to Adobe Premiere Pro SDK for specific actions. Each suite consists of related functions and shares a common interface.

The *SweetPea Suites* are one of the newest. That are providing most of the additional functionality plug-ins. It is recommended to use SweetPea Suites whenever possible. However, some functionality is required from older suites such as *piSuite*. Both of these are common to all plug-in types as they are mostly providing a general function such as conversion between time or pixel formats.

Additionally some suites are unique for specific plug-in types. For example *Bottleneck Suite* is useful only for transitions and video filters.

Time and clock management

Adobe Premiere Pro uses two types of time and clock representations. Both representations are equivalent in their descriptive power but are used in different things because of their convenience in usability.

Scale over sampleSize representation is used to represent rational numbers in time-related fields like framerate.

PrTime is a measurement based on ticks of a processor clock. However, it is not synchronized with CPU's hardware clocks, but instead Adobe Premiere Pro has its internal clocks.

Scale over sampleSize

This time representation is defined by ratio of two numbers (two of *value, scale, sampleSize*), either separated or combined as *TBD_TimeRecord* structure. *Scale* over *sampleSize* represents the timebase. It is based on typical framerates. For example the NTSC standard of 29.97 frames per second can be represented as *scale* = 30000 and *sampleSize* = 1001. To represent other standards without decimal points, *sampleSize* is set to 1. For example PAL standard of 25 frames per second is represented as *scale* / *sampleSize* = 25 / 1.

Value is the timebase given by *scale* over *sampleSize*. For example 30 frames with *sampleSize* 1001 have a *value* of 30030. To convert *value* to seconds, divide by *scale*. To convert *value* to frames, divide by *sampleSize*.

This time representation is used not only for video but also for audio streams. In case scale over sampleSize is representing audio sampling (22050, 44100) *sampleSize* is set 1.

PrTime

PrTime is the newer of the two mentioned time representations. It is s a tick-based time value stored in signed 64-bit integer. Framerate represented in PrTime specifies the number of ticks in the duration of a frame.

Ticks are an internal Adobe representation and are not related to CPU's hardware clock. A number of ticks per second must be retrieved using the callback in the *Time Suite*[19]. This number is guaranteed to be constant during the run of Adobe Premiere application, however storing the value and restoring from the cache after a restart of the application can lead to timing problems.

Pixel Formats

Adobe Premiere CS6 SDK supports 66 different pixel formats. This does not include raw and custom formats. These pixel formats are necessary for various file formats and conversions. 8-bit formats are compact but lack quality. 32-bit formats are more accurate but may cause the video rendering pipeline to slow down significantly. Bit depth is not the only comparison in pixel formats, for example, compression is a necessary factor for pixel formats as well. Compressed formats are useful for storing raw frames, but not efficient for effect processing.

Selecting a specific pixel format is a crucial step the for the development of Adobe Premiere Pro plug-ins. Importers should use formats as close as possible to the source format, because of faster conversion. If necessary Adobe Premier Pro can convert between pixel formats. For effects editing, compression is a big factor to consider, as compressed pixel formats have significantly slower render times then uncompressed. Still, the general idea of Adobe Premiere Pro is to keep pixel formats compressed as much as possible to save memory.

Exporters and players require format closest to the output format. New since CS5 pixel format *PrPixelFormat_Any* can be used to allow Adobe Premiere Pro to use the best fit for the pixel format based on rendering and conversion times.

Plug-in support

Adobe Premiere Pro plug-ins are divided into multiple types based on their functionality. Most used plug-ins are importers and exporters as they are necessary at the start and at the end of editing pipeline, and they are mostly required for registration of new file types and video formats.

Recorders

Recorders are a type of plug-ins which create an interface between hardware, captured video/audio and a file format supported by Adobe Premiere. Recorders preview video in the Capture panel and play an audio preview to system sound or hardware output based on Adobe Premiere Pro settings. They are also responsible for management of all the meters in the Audio Master Meters panel. These plug-ins have the ability to provide a custom dialog for any necessary settings. Most important part of recorders consists of digitizing video and audio to a file or multiple files on disk.

Communication between Adobe Premiere Pro and recorders is rare and happens only when a recorder is providing source timecode information to Adobe Premiere Pro. In this case, the recorder notifies Adobe Premiere Pro about any video format changes so that the capture preview can be resized. There is no communication between a recorder and Premiere about the audio capture.

When a recording is complete, Adobe Premiere Pro imports the file using any available importers that support the file type. To enable the functionality of recording to multiple file types, the user must provide multiple recorders.

Importers

Importers provide video, audio and images from a media source. The media source is usually a single file, but in some cases it can also be a set of files or a communication link from another application. They are used whenever frames of video or audio from a clip are needed. Importers provide support to read media that use a new format or codec.

The standard importer is accessible from a drop-down menu in the main Adobe Premiere Pro screen in the *File* → *Import* dialog. Standard importer supports video, audio, images and sets of images.

Synthetic importer is available from the *File* → *New file* dialog. This type of importer synthesizes source material and is not reading anything from disk.

Custom importers are a particular type of synthetic importer. These were created for better titlers support [19]. The biggest difference, based on implementation, is that Custom importers can create files on disks while Synthetic importers are not. They can either generate new media or import existing media handled by an importer. Additionally the media can be modified by the importer.

Capabilities that are different for each type of importer can be seen in table 2.2.

Importer type	Reads from disk	Creates clips	Menu Location
Standard	Yes	No	File → Import
Synthetic	No	Yes	File → New
Custom	Yes	Yes	File → New File → Import

Table 2.2: Importers capabilities

Several importers for one file type can be created and used in case some importers fail to import a given file. The order of used importers is determined from their assigned priority. While standard importers provided by Adobe Premiere have priority ranging from 0 to 100. When developing an importer, it is recommended to set a higher priority to override all the existing importers.

Importers can add support for asynchronous calls to read frames for better performance. This is very helpful for high-resolution frames which used uncompressed pixel formats. All calls with asynchronous importers are reentrant, except for *aiClose*, which is only called once and is executed as last call of the importer.

Exporters

Exporters are in charge of exporting video, audio, and markers to any file format specified. Before exporter is run, Adobe Premiere Pro must provide individual frames of video in Pixel Format 2.2 specified by the exporter and uncompressed audio. They are also responsible for any compression of the video and audio data and creating the final output file and its format. To include existing exporter and use it for additional support of other exporters, a user can provide an export controller.

Exporters are available within Adobe Premiere Pro and from Adobe Media Encoder. Within Adobe Premiere Pro, they are accessible from the *File* → *Export* → *Media* dialog. In the Export Settings dialog, a user can choose the video format and an exporter provides a summary of settings with the parameter settings.

Finally, exporters have the optional hardware acceleration that communicates with a renderer plug-in and uses direct access to its frames.

Export Controllers

An export controller is a plug-in closely connected to exporters. It is accessible from a standard menu in *File* → *Export* submenu. The main functionality of export controllers is post-processing of video from an exporter plug-in. Export controllers use *Sequence Info Suite* to gather various properties of data.

Optionally export controllers can display any custom modal UI. This is the main way for the user to set additional parameters for export. This UI needs to be provided by the export controller plug-in.

Export controllers use the *Export Controller Suite* to communicate with exporter plug-in. Providing an exporter preset and path for the output, exporters use these values for standard export. When an exporter is done, the export controller can perform any post-processing on the exported file based on the return value from the exporter.

During the video export, the UI is blocked just as during standard export. At the moment, Adobe Premiere SDK does not have support for asynchronous export via export controllers.

Most used post-processing actions used by many export controllers consist mostly of transferring the rendered media file through the network or additionally filling meta-data of the output file.

Transmitters

Transmitters are a new feature of CS6. This preferred way of external hardware monitoring provides simplified support for pushing video, audio and closed captions to external hardware. Transmitter options are accessible from a standard menu in *Preferences* → *Playback*.

Transmitters are not bound to the sequence Editing Mode, which cannot be changed once a sequence had been edited, just as Players are. At the moment, transmitter plug-ins are supported for Adobe Premiere Pro, Encode, and Prelude and support only 32-bit plug-ins.

On creation, transmitters need to have formats defined in which the rendered video will be inputted. Adobe Premiere Pro will handle all the desired conversion to required format.

Adobe Premiere Pro has to handle scheduling for pre-fetching the media and asynchronous rendering.

Plug-in supports playing audio on the host, through the system's sound or sending it to external hardware by *Playmod Audio Suite* [19].

Synchronization is done by clock callbacks, which transmitter calls for an update of the host with the new time every frame. This also ensures video and audio synchronization.

Players

Player manages playback of video on different monitors such as Source Monitor, Program Monitor, Multicam Monitor or Reference Monitor[19]. Playing video on hardware devices such as SDI card or another third-party hardware is also supported. For each sequence, only one player is defined, once it is set it cannot be changed. Because of this players are designed to handle playback through the whole editing process, from the start till the final playout.

The player used for each sequence is chosen during the creation of sequence, by the Editing Mode. Editing Mode is defined by a combination of exporter and player.

Player used for sequences playback can be defined in *Preferences* → *Playback Settings* → *Default Player*. This selected player can be overridden by player chosen during sequence creation.

The player plug-in does not render sequences of media, however, the player can take over rendering for any segment it chooses. The player takes over rendering in case it can render selected part faster than the host's renderer. It is not supported to allow a player to take over rendering of every segment of a sequence since it may not support every file or pixel format.

Players are generally created in multiple instances during the editing process. One instance is created when project is created for titler and external monitor output. Another instance is created for each sequence that has been opened in sequence monitor. Multiple instances are created for all the clips opened by Source Monitor. Another instance may be for a Reference Monitor to the Sequence Monitor. However at each time only one player is active. This is managed by host application which triggers just one player at a time. Each player manages it's appropriate parts of sequences and are scheduled just for the editing process of their sequences.

In CS6, the transmit API is the suggested way to develop Players and Transmitters. However, the player API is still in CS6 for backward compatibility, but it is deprecated.

Transitions

Transition plug-ins are taking two sources of frames, which may be from different sequences and processes them into a single destination set of frames.

Transitions support the creation of own custom modal setup dialog that can be used for specification of additional functionality.

The transition is rendered on first playback of the frame. At this moment frame of transition is stored in frame cache. On the change of frame in transition, frame in the cache is not overwritten, but a new frame is created. This way revert of few changes is supported for complex transitions without losing computational power.

Real-time transitions must have a specific flag raised to notify players to obtain real-time preview (provided you can communicate directly with the player). Not every transition can be set as a real-time transition.

Video Filters

For video filters development, it is recommended to use Adobe After Effects SDK [20] as almost all the video filters and effects provided by Adobe Premiere Pro are Adobe After Effects plug-ins. And developers from Adobe suggests that all future development in this field will be done with Adobe After Effects SDK.

Device Controller

Device controllers manage hardware devices connected to Adobe Premiere Pro as third party device. With a use of cameras or tape decks they enable timecode-accurate, hardware assisted video capture. Device controllers also support output to tape. They are used while working with Capture panel, and during Export to Tape. Device controller plug-in enables support for one or more communication protocols and is able to handle various hardware. All hardware connected to device controller must be able to communicate with the same protocol.

Plug-in registration

Plug-ins are registered with resources which need to be specified during the initial phase development process. There are two types of registration for plug-ins. *Plug-in Property List* are mainly used for registering new names, formats while *IMPT* resource is mostly only defining a unique identifier for the plug-in.

Plug-in Property List

For many plug-in types, Adobe Premiere Pro loads the Plug-in Property List (PiPL) resource. The Plug-in Property List is defined in a file with „.r“ extension and follows the Adobe Premiere Pro Plug-in Property List syntax. An example of Plug-in Property List file is shown in figure 2.4

```
#define plugInName "SDK␣Custom␣Import"
#define plugInMatchName "SDK␣Custom␣Import"

resource 'PiPL' (16000) {
{
    //The plug-in type
    Kind {PrImporter},

    //The name as it will appear in a Premiere menu.
    Name {plgInName},

    //The internal name of this plug-in (not localized).
    AE_Effect_Match_Name {plugInMatchName}

//Transitions and video filters can define more PiPL attributes

}
};
```

Figure 2.4: Basic PiPL resource example

Exporters, players, and recorders do not need PiPLs.

From importer types, only synthetic and custom importers use a basic Plug-in Property List to specify their name and match the name that Adobe Premiere Pro uses to identify them. Standard importers do not use Plug-in Property List.

Device controller also uses basic Plug-in Property List for the same reason.

Video Filters use extended Plug-in Property List for name specification, the bin they go in, type of pixel aspect ratio handling, etc.

Transitions also use extended Plug-in Property List for the same reasons as Video Filters. Additionally information like reverse checkbox, borders, point controls are specified here.

IMPT Resource

Additionally, Adobe Premiere Pro can use IMPT to identify a plug-in as an importer. A unique draw-type *fourcc* is needed for the importer to function properly in Adobe After Effects or Adobe Premiere Pro on Mac OS. *Fourcc* is a 32-bit number represented by hexadecimal which is unique for each importer. In figure 2.5 you can see sample IMPT resource taken from Adobe Premiere SDK CS6 documentation [19].

```
1000 IMTP DISCARDABLE
BEGIN
    0x12345678 // Put your unique hexadecimal code here (fourcc)
END
```

Figure 2.5: Basic IMPT resource example

Plug-in deployment and installation

For development few necessary steps need to be done. At the moment, Microsoft Windows and Mac OS are fully supported for both usage and development of plug-ins. For development, appropriate path Adobe Premiere Pro libraries must be specified in Environment Variables for selected platform.

For the development process, plug-ins should be built into one of the many separate paths from where Adobe Premiere Pro loads plug-ins during its initial setup. On first launch, Premiere loads all the plug-ins, reads the PiPL resources and sends any start-up selectors to determine the plug-ins capabilities. For time-saving reasons, all the option are cached to what is called the Plug-in cache (the registry on Microsoft Windows, a Property List file on Mac OS). The next time Adobe Premiere Pro is launched information from a cache is loaded rather then reloading all the capabilities of plug-ins.

Caching capabilities is making the initial launching process faster, but may not be desired for plug-ins which require to be initialized every time. These are the type of plug-ins which require the run-time information about the system, such as installed codecs, clock synchronization or hardware specification.

Due to this fact by default importers, recorders, exporters and players are not cached. All of mentioned type of plug-ins supports caching by enabling specific flag for this purpose. This support was introduced in latter part of SDK development and, for this reason, there is no full support for all of them. The importer does not support caching on Microsoft Windows. Adobe Premiere Pro SDK CS6 also mentions recorders are causing a problem while being initialized from the cache

By default transitions, video filters, and device controllers are cached. To specify that these plug-ins should not be cached they need to implement a respond to *fsCacheOnLoad* and *esCacheOnLoad* calls respectively.

2.3 External libraries

As powerful as Adobe Premiere Pro SDK is, it is unable to manipulate natively video files and work with HDR content. Because of this few external libraries for support of additional functionality must be considered.

From the file manipulation point of view, it is necessary to have a library that can do various manipulations on a video file. First of all reading through all the video file headers (file header, stream header, codec header) is necessary as all this information will be provided to Adobe Premiere Pro plug-in. On the second-hand video editing is not a continues playback of video, so seeking to specific frame number is a crucial component for plug-in development, this seems to be a problem in many video editing libraries as they are mostly used for video players design that does not need this feature.

There are few libraries that support manipulation with HDR content. It is not necessary for a library to work with HDR videos as all the video editing is done on frames. Because of this, library with support for HDR images is sufficient. However, the most limiting factor will be a functionality of compressions. Uncompressed video in a video editing pipeline can result in excessive render times or necessity for acquiring hardware with bigger computational power than a computer of a regular user.

FFmpeg

FFmpeg is a cross-platform collection of free software providing functionality for recording, conversion and streaming of digital video and audio. As FFmpeg is being pushed by the open-source community, it supports most of the video formats and codecs that are available.

It contains three end-user applications that can be used for transcoding, streaming and playing. FFserver is a multimedia streaming servers used mostly for live broadcasts and streams of video and audio. FFplay is a simple media player with FFmpeg libraries with related content. FFprobe is an audio and video file format analyzer providing details about multimedia files, streams, codecs and all the information contained in files.

These three end-user application were created by libraries that are also publicly available. These seven libraries are essential for development with FFmpeg:

- **libavutil** is a library containing all the basics to video editing. It is meant to make development process with FFmpeg easier by providing functionality such as random number generators, mathematics routines, core multimedia utilities, etc.
- **libavcodec** is a library more used with video playback and broadcast and not so much in video editing. This library contains encoders and decoders for multimedia formats.
- **libavformat** is a library providing muxers and demuxers for multimedia content.
- **libavdevice** is containing input and output devices for communication with other multimedia frameworks (Video4Linux, Video4Linux2, Video for Windows, ALSA).
- **libavfilter** is providing functionality of filters for video files. These filters can be also applied to static images.
- **libswscale** is a library performing image scaling, color space conversion, and pixel format conversion. These actions need to be highly optimized as this is also used in real-time rendering.

- **libswresample** is an audio editing library that provides a functionality of audio resampling, rematrixing and sample format conversion operations.

OpenEXR

„*OpenEXR is a high dynamic-range (HDR) image file format developed by Industrial Light & Magic for use in computer imaging applicatios*“ [24]. This format is released as an open standard. With the format, OpenEXR Industrial Light and Magic (ILM) also release a set of software tools under a modified BSD Licenses [1].

OpenEXR standard was introduced with numerous features which are helpful for the editing process, displaying and capturing of HDR content. 16bit and 32bit pixel format are supported. These are stored as floating point numbers. This way it is possible to store a larger range of data in a pixel value. Pixel data can also contain arbitrary channels. With standard red, green, blue and alpha channel it is also possible to store channels such as luminance or depth. Image compression is solved by support for both lossless and lossy data formats. From other features of OpenEXR mentionable and interesting is support for multiple views of an image which enables the functionality of 3D stereo images. OpenEXR also provides libraries for editing, capturing and displaying of HDR content written in C with C and C++ interface.

OpenEXR also introduced EXRFormat plug-in for Adobe Photoshop which enabled import, edit and export for HDR content inside Adobe Photoshop [17]. This plug-in enables basic editing of high dynamic range content stored in EXR format such as exposure and gamma correction. It is also providing full functionality for import and export of HDR images stored in EXR format.

2.4 Current state of HDR video editing

In this section products that has similar functionality for video editing are mentioned. First of all, ProEXR plug-in which enabled HDR support for Adobe Premiere Pro seems like a solution to import and export of HDR content inside Adobe Premiere Pro. Another software enables editing HDR content only for images. However these programs are not meant to edit HDR software but rather create an image that is visually close to HDR photography, but instead of storing it in HDR format it is tone mapped into LDR image.

Image editting software

Among most used software for work with HDR images can be counted Photomatrix Pro [27], SNS-HDR [5], Oloneo HDR [15], EasyHDR [11] and HDR Darkroom 3 [8]. This photography software can create images that are visually similar to high dynamic range images. This is done by the technique of multiple exposure photographies which are on post-processing merged together and only correctly exposed pixels are inserted to the final image. This way this kind of software is creating an HDR image, but afterward it is tone mapping it to low dynamic range image and storing it the same way.

This is however very helpful for photographers to acquire images with the better visual effect but for editing of HDR content it is not providing any additional functionality. To

[1]Technical overview of the standard can be found on OpenEXR.org website

enable editing these photographies, HDR software would need to be able to store the image to high dynamic range format such as EXR.

ProEXR

Based on the OpenEXR Industrial Light & Magic in June 2014 released ProEXR plug-in that enables support for HDR images in Adobe Photoshop, Adobe Premiere Pro, and Adobe After Effects [13]. This allows import and export of HDR images into Adobe Premiere Pro, however, does not work with video files.

ProEXR consists of two plugins importer and exporter. While importer is focusing on importing of HDR images to Adobe Premiere Pro as single frames that can be used as still frames in a video, the exporter is able to work directly with video within Adobe Premiere Pro. At last phase of video editing pipeline when the video needs to be rendered to output file, it is able to be exported to multiple output images (frames). Afterwards it can be converted with appropriate information (framerate, timecode…) to video file without necessity for exporting directly to video file [14], but for this software additional software is required..

However, this enables functionality only for HDR images and disables the functionality of video editing in Adobe Premiere like effects or transitions on continuous video which is a necessary requirement for video editing. Even that export to single video frames might be useful in exporting to EXR format and merging these frames by additional software to video files. It is also necessary to provide functionality to export tone map video to the low dynamic range so it can be viewed by standard video players on low dynamic range displays.

Summary

Most of the software for photography editing is not meant actually to edit HDR content, but instead to create images that are visually similar to high dynamic range images. Afterward, they are storing these images tone mapped as low dynamic range images. Therefore, this software is not helpful for editing of HDR content. Another solution for HDR content is OpenEXR library which provides Adobe photoshop plug-in. This plug-in enables support for HDR content inside Adobe Photoshop. This might be helpful for editing for still images which can be used inside video for transitions etc., but inadequate for video editing. The same functionality but already integrated into video editing software (Adobe Premiere Pro) provides ProEXR. This plug-in enables support for HDR images in Adobe Premiere Pro and enables export of videos to EXR video format. However working only with still images will still lack some functionality of video editing as proper transitions or effects on more than one frame. For this purpose, it is necessary to create an importer which is able to import high dynamic range video, store it inside Adobe Premiere as video frames and not as images. This will enable all the features of editing process of Adobe Premiere Pro.

Chapter 3

Design

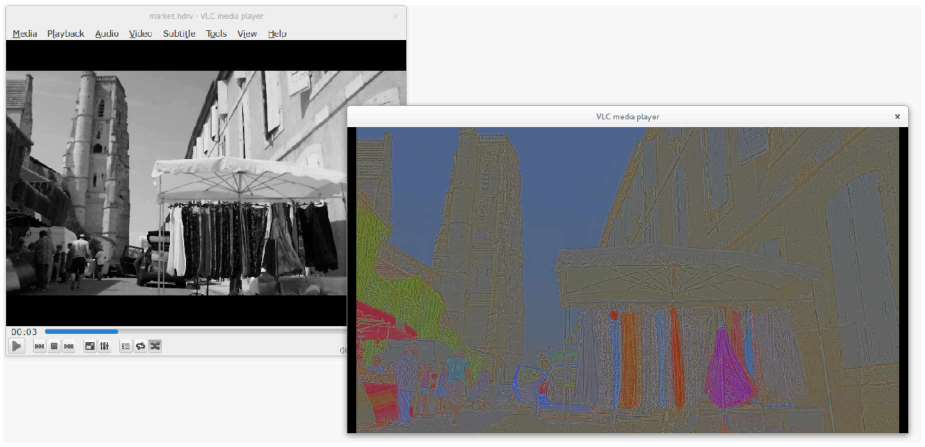
This chapter describes the design of plug-in that enables HDR support in Adobe Premiere Pro.

Currently, there is very low support for video processing of HDR videos available. As mentioned in the previous chapter it is mostly possible to edit HDR images either inside Adobe Photoshop by OpenEXR plug-in or import these still images to Adobe Premiere Pro by ProEXR plug-in. Direct editing of high dynamic range video files is a much more complicated process. The current solution for this problem is to split the video into single frames which can be edited. This is time-consuming and very inefficient. After all the editing on individual frames is done it is possible to store these frames again into a video which can by played by the player that supports HDR video. This player upon playback will tone map every single image to be displayed on low dynamic range device. This process of video editing is possible, however, many features of video editing such as transitions or video effects on continuous video can not be achieved.

Because of this new HDR plug-in must be implemented. A key feature of the HDR plug-in will be import of high dynamic range video without the necessity of splitting it into single frames. On import, all the frames will be tone mapped and stored as continuous video to enable video editing process. Because of storage in low dynamic range it is possible to export this video to different video formats which do not require any specialized software or hardware for playback. This leads to creating more content of high dynamic range videos which can be processed bu Adobe Premiere by using designed plug-in and after rendering displaying it on the standard low dynamic range screen available for public usage.

The goal of this plug-in is to allow video editing of HDR content in Adobe Premiere Pro. As a solution to this problem, a new importer must be implemented as plug-in to Adobe Premiere Pro. For this purpose, File type importer is used. Custom and synthetic type importer are creating new content inside Adobe Premiere Pro, which is not desired behavior. HDR plug-in supports HDR video type that is storing two kinds of frames. One is storing luminance information about every pixel of the frame and the second is storing color information.

There are two video file formats which are available for this importer. Both of them are storing two kinds of mentioned frames. These frames are stored based on a video format. First of them are storing both of these frames in single streams. By combining these frames and connecting them on top of each other, frame with double height is created. This frame has the frame with lighting information on top of the frame with color information and this way it is not necessary to read two streams at the same time or store them with same timing information.


market.hdrv - VLC media player
Media Playback Audio Video Subtitle Tools View Help
VLC media player
00:03
Figure 3.1: HDR video with two different streams for storing high dynamic range frames.

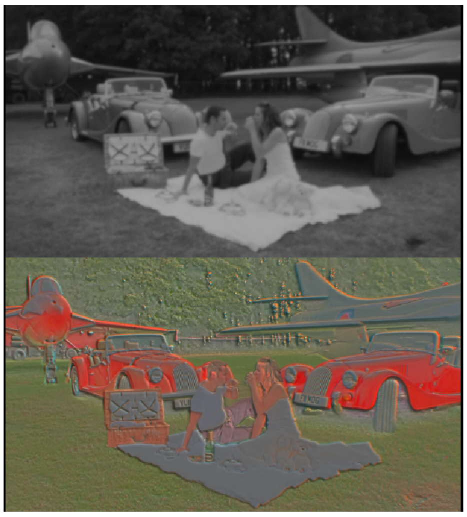


Figure 3.2: HDR video storing both frames with luminance and color information combined in single frame.

On the other hand, video format with two streams with the same timing information for both of them is available. This video format stores frame with lighting information in one of the streams and frames with color information in the other one. This way it is not necessary for editing process to manipulate the pointers and offsets to data to acquire pixels that are creating a pair for high dynamic range pixel, but it is possible to get both sub-pixels with the same offsets but in different streams. Both of these video formats can be seen in figures 3.1 and 3.2. These screenshots were created by opening a video in the standard player for low dynamic range content which is not tone mapping the frames, this way both frames with luminance and color information can be seen.

For the purpose of this plug-in first of the two were chosen for support. This way it is not necessary to work with streams of video files with FFmpeg, but tone mapping will need to address sub-pixels correctly.

Other problem consists of interpretation of HDR video inside Adobe Premiere Pro after the actual import. This problem can be solved by three different designs, however only one of them is supported in current version of Adobe Premiere Pro.

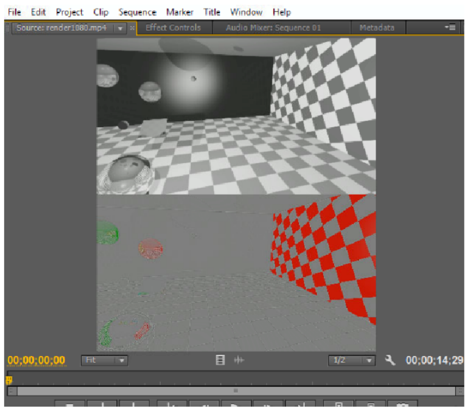


Figure 3.3: HDR video imported by a standard mp4 importer. Content is displayed as regular LDR frames without tone mapping. Therefore, they are displayed as two frames.

The first solution to this problem would be storing HDR frame and tone mapping them only when they are presented in a player of Adobe Premiere Pro. However, Adobe Premiere Pro importer plug-ins do not have separated storage and presenters. The format that frames are stored in is also presented to the player. In case that the frame would be stored as it is in an HDR video file, both frames with lighting and color information would appear. This way most of the functionality of Adobe Premiere Pro video processing would be lost. This problem can be seen in the figure 3.3 where HDR video is imported by a standard mp4 importer.

Another possible solution to this problem would be to store HDR frames in such a format that is supported by the Adobe Premiere SDK. This feature would enable presenting and editing HDR frames in Adobe Premiere Pro, also allowing export to HDR video either by the new custom exporter that would need to be developed or by ProEXR to single frames. After exporting video to individual frames, these frames can be converted to a video file with the additional software.

Unfortunately, Adobe Premiere Pro CS6 does not support such a format that can store HDR frames.

The current implementation of this problem consists of importing HDR video frames by plug-in, tone-mapping them and storing them as LDR frames. Storing these frames in one of supported formats of Adobe Premiere Pro SDK CS6 will enable all the editing features of Adobe Premiere Pro. Exporting to output video can be done after the editing process by any standard exporter of Adobe Premiere Pro. This way video can be rendered to any supported video format of Adobe Premiere Pro.

The implementation details of described importer are mentioned in chapter 4.

Chapter 4

Implementation

This chapter describes details behind the implementation of HDR plug-in for Adobe Premiere Pro. This plug-in was implemented with Adobe Premiere Pro CS6 SDK and can be used in the same version of the product. Because of the fact that plug-in is not using any features of mentioned SDk it is backward compatible with older versions of Adobe Premiere Pro.

In the first section of this chapter, the plug-in structure is described. Adobe Premiere Pro plug-in are being developed as handles to selector calls of Adobe Premiere. Each of these calls needs to specify different necessary part for import of the HDR video. This section describes this process and goes through the important selectors which are crucial for proper work of importer. After registering of the importer, it is necessary to create an option in Import menu for a user to select the high dynamic range video format. When this is provided, all the supported video formats can be stored for support. Next phase of the plug-in starts when a user selects video in Adobe Premiere Pro and selects HDR plug-in to import this video or selects video with „.hdrv“ extension as Adobe Premiere Pro does not support this extension. On selecting video, pixel format of the high dynamic range video file must be acquired and compared to supported pixel formats of HDR plug-in. Based on the video pixel format of the video file and its support inside HDR plug-in frame will be later converted to pixel format appropriate for video editing. When all the right pixel format is selected all the necessary information is parsed by FFmpeg library from header files of video file and filled to Adobe Premiere Pro structures. After this moment Adobe Premiere Pro can start asking for frames from HDR plug-in. On each call for a particular frame, this frame is tone mapped to low dynamic range frame and stored in Adobe Premiere Pro memory.

Next sections deals with the problem of seeking inside FFmpeg. As FFmpeg is mainly library for video editing and video players, seeking to specific frame number is implemented only to some extent. Not every frame is addressable in video file and because of this frame accurate seeking is not possible. As a solution to this problem new implementation of repetitive seeking and moving the number of sought frame was implemented. This solves the problem of seeking to the first frame of the video if this frame is not seekable and also of seeking backwards in the video stream which is not supported by FFmpeg for MPEG-4 video files.

In this final section of this chapter, the final product is discussed. There is a big difference in importing video by standard MPEG-4 imported which will consider HDR frame with both frames of luminance and color information to by one low dynamic range frame and display them as they are stored in video file and displaying tone mapped version of

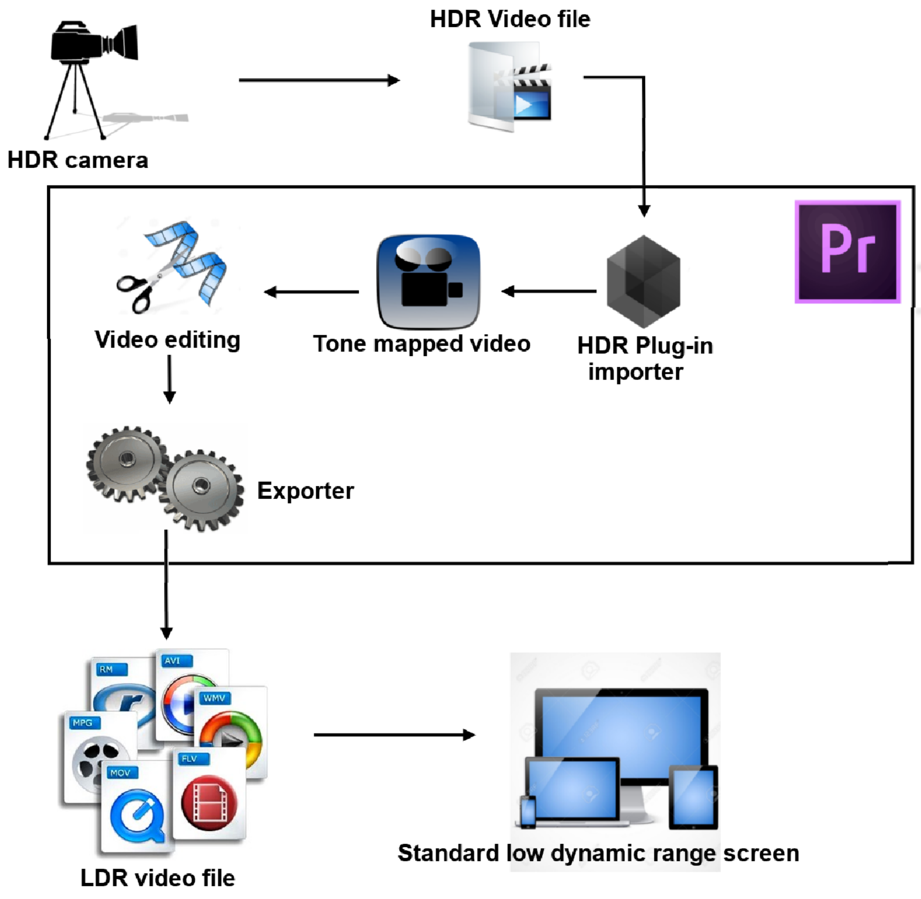


Figure 4.1: Workflow of video processing by HDR plug-in.

high dynamic range frame which had the desired visual effect. This plug-in is making HDR videos in the format of frames with both frames with luminance and color information on top of each other available to be seen in tone mapped version. This and other functionality of plug-in presented in this last section. Screenshot of rendered video can be seen here as well as HDR frames before tone mapping was applied to them.

Implemented plug-in will enable support for HDR content inside Adobe Premiere Pro. By importing HDR video with this plug-in video is stored inside Adobe Premiere Pro as low dynamic range frames. This video can be edited and exported by any standard importer of Adobe Premiere Pro or by any exporter provided by other exporter plug-in. Plug-in is automatically used when importing video with „.hdrv“ extension or by implicitly selecting chose „HDRV Video format“ on import of video with other extension. This way high dynamic range video files can be directly imported into video editing tool, tone mapped withing Adobe Premiere Pro and stored. After the import user can start the video processing and editing. When this is done it is possible to export the video by Adobe Premiere pro directly to low dynamic range video, as it was previously tone mapped, so it can be displayed on directly standard low dynamic range display (LCD, CRT). Recommended workflow for video editing of high dynamic range video with implemented HDR plug-in is depicted in figure 4.1

Acquired video from HDR camera is stored as high dynamic range video. By HDR importer, it is loaded into Adobe Premiere, and all the frames are tone mapped. At this moment editing process inside Adobe Premiere Pro can start, and afterward video can be rendered to one of supported output video formats displayable by the standard screen.

4.1 Plug-in structure

Adobe Premiere Pro has full support for plug-ins with selectors. These selectors are called by standard function as a one of the parameters and need to be handled.

It is necessary to implement individual selectors appropriate to the plug-in type and implement them. For example, all of the selectors that are called have prefix *im*. However not all selectors are necessary to be implemented as some of them are relevant only for synthetic or custom importers, some calls might be only appropriate for importers supporting asynchronous import. Sequence diagram for appropriate selector calls that are called from Adobe Premiere Pro can be seen in 4.2

The flow of data and code is going through three states. First one is on start-up when HDR plug-in is initialized. Calls *imInit* and multiple *imGeIndFormat* are executed to acquire plug-in information required for the user interface of Adobe Premiere Pro and to gather information about all the supported formats.

Second phase starts when user choose video file to be imported and it consists of *imGetInfo8*, multiple *imGetInfPixelFormat* selectors and call of *imPreferredFrameSize* to get resolution of frames.. At this moment information header is video file is being parsed by FFmpeg and all the information is stored to Adobe Premiere Pro structures.

After all the information is acquired, Adobe Premiere Pro asks for starting frames of video file to pre-cache them before the editing process on video file starts. Once the editing process began, frames are loaded from disk based on current position of editing the video clip. This is done only by *imGetSourceVideo* selector call.

In the close of application *imClose* selector is called, which does not have to be implemented if a plug-in is not using any child processes.

Plugin registration

For registering the HDR plug-in into Adobe Premiere Pro Plug-in Property list was used. For this unique fourcc number needed to be created. It is common practice in Adobe Premiere Pro plug-in development to use the hexadecimal representation of ASCII code of extension of video format supported. Because of this hexadecimal representation of string „HDRV“ was used and HDR plug-in is registered under fourcc 0x48445256. In case, another plug-in with the same fourcc would be included in Adobe Premiere Pro both plug-ins with the same fourcc would not be initialised.

Initialization

Initialization of HDR plug-in is done by handling the *imInit* call. This call is done on Adobe Premiere Pro startup.

Because of the fact that importer does not have any child files it is not necessary to provide any file handling. Therefore there is no need for implementing *inOpenFile8, imQuiteFile, imCloseFile, imSaveFile8, imDeleteFile8.*

Caching is disabled by setting *canDelete* flag to *kPrFalse*. This is caching of the importer on program start, and it is not suggested by Adobe Premiere Pro CS6 SDK to enable caching for importers.

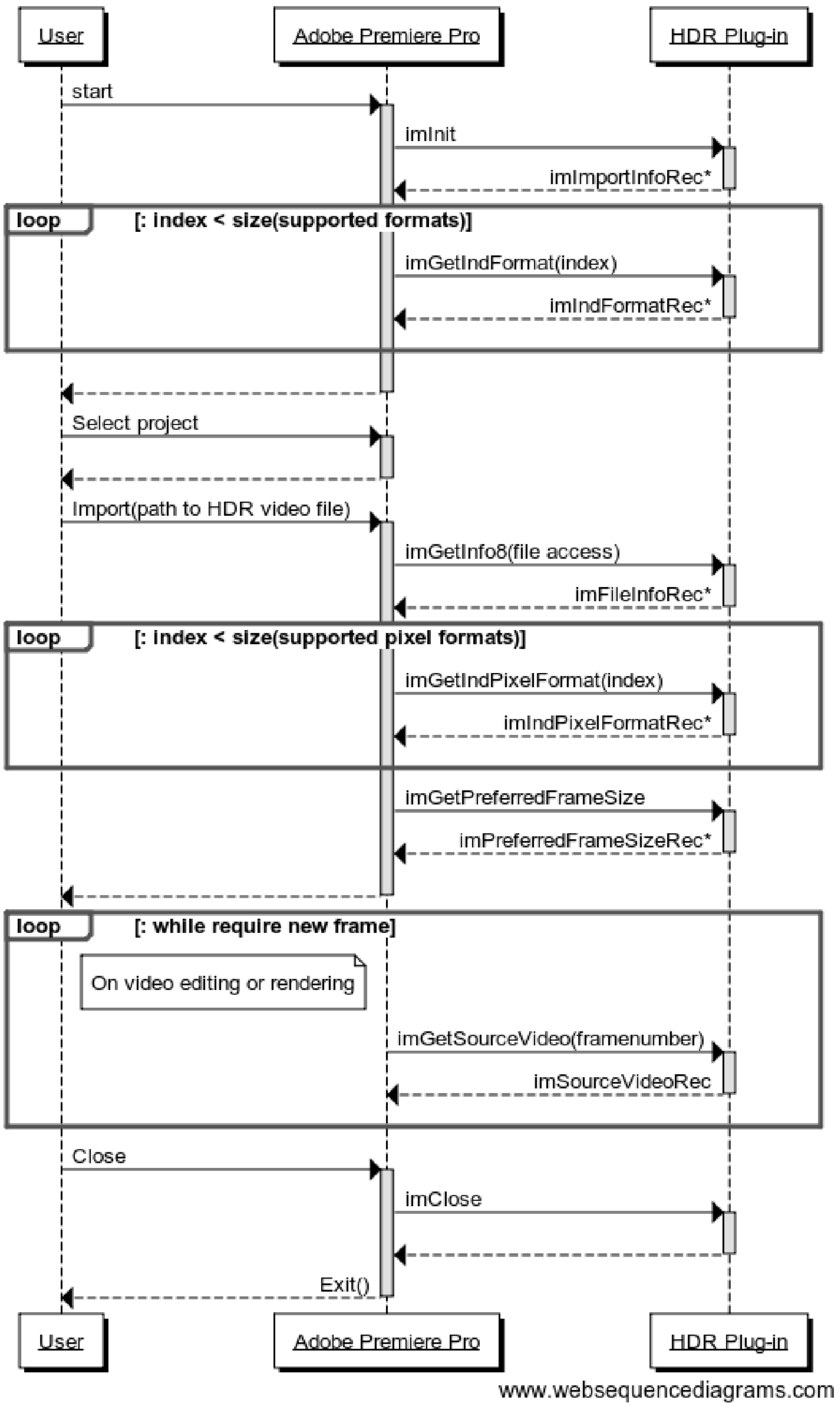


Figure 4.2: Sequence diagram of selector calls.

The importer does not support any further configuration on an import of the video as it is not necessary, because of this *hasSetup* is disabled.

Last fundamental option to set up is the priority of the plug-in. As this plug-in is not standard in Adobe Premiere Pro, the priority is set to 100, and this is to override all the standard plug-ins included in Adobe Premiere Pro.

Supported video formats

Next call executed during start-up of Adobe Premiere Pro is *imGetIndFormat*. This call is acquiring information about supported video format and is repeatedly called with the index rising after each supported video format. After the last video format on next call *imBadFormatIndex* is returned to notify Adobe Premiere Pro that there is no other video format supported. It is necessary to provide three string in this call handle. Full name of the supported file type, in this case, „HDRV Video Format“ that appears in import menu as seen in figure 4.3.

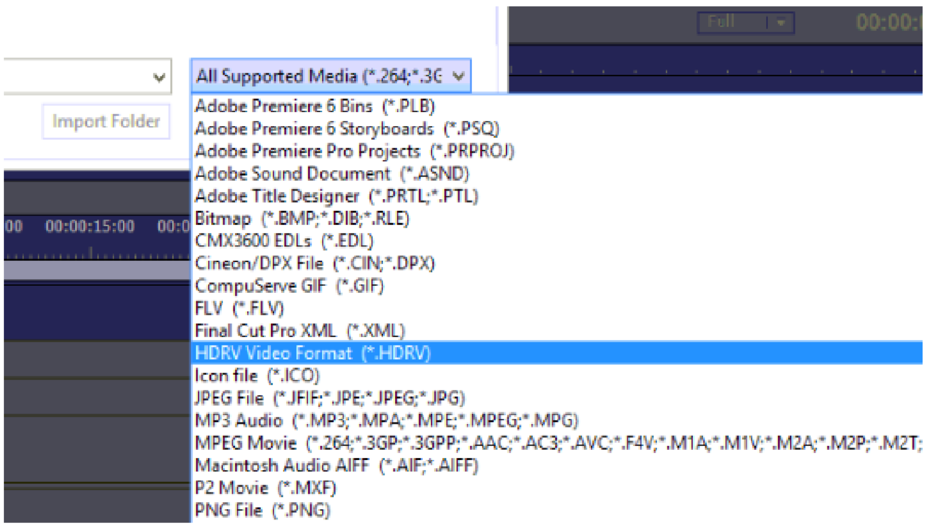

Import Folder
00:00:15:00
All Supported Media (*.264;*.3G
Adobe Premiere 6 Bins (*.PLB)
Adobe Premiere 6 Storyboards (*.PSQ)
Adobe Premiere Pro Projects (*.PRPROJ)
Adobe Sound Document (*.ASND)
Adobe Title Designer (*.PRTL;*.PTL)
Bitmap (*.BMP;*.DIB;*.RLE)
CMX3600 EDLs (*.EDL)
Cineon/DPX File (*.CIN;*.DPX)
CompuServe GIF (*.GIF)
FLV (*.FLV)
Final Cut Pro XML (*.XML)
HDRV Video Format (*.HDRV)
Icon file (*.ICO)
JPEG File (*.JFIF;*.JPE;*.JPEG;*.JPG)
MP3 Audio (*.MP3;*.MPA;*.MPE;*.MPEG;*.MPG)
MPEG Movie (*.264;*.3GP;*.3GPP;*.AAC;*.AC3;*.AVC;*.F4V;*.M1A;*.M1V;*.M2A;*.M2P;*.M2T;
Macintosh Audio AIFF (*.AIF;*.AIFF)
P2 Movie (*.MXF)
PNG File (*.PNG)

Figure 4.3: Import menu of the plug-in displaying supported HDRV video format.

Both Microsoft Windows and Mac OS are supported. As development on these two platforms requires different functions for example for string processing or file handling, development needs to be in branches based on PRWIN_ENV macro for Microsoft Windows operation system and PRMAC_ENV for Mac OS.

Header parsing

After the actual video is selected by importing menu shown in figure 4.3 *imGetInfo8* is called to acquire video and audio information. In this call, few necessary properties need to be set. Support for asynchronous importers is disabled.

There are two ways to acquire frames from video, either by *imGetSourceVideo* call or by *imImportImage* call. This plug-in enables *imGetSourceVideo* as it is the preferred way to work with videos with a particular resolution. This way HDRV plug-in can import video in native resolution or any specified resolution, rather than scale the frames to an arbitrary size. *imImportImage* call is useful in resolution-independent video, for example vector-based graphics. Because of this *imGetSourceVideo* is mostly used in standard importers which are importing video from video files and *imImportImage* call is mostly used in synthetic and custom importers where video is generated and does not have to have specific resolution.

After this suits that will be necessary for frame processing, caching of frames and memory management are loaded.

After all the initialization for plug-in processing is done it is necessary to acquire information about audio and video.

As HDRV files do not currently have any audio, HDRV plug-in does not support audio import. Video information is gathered through parsing the video header with FFmpeg libraries. Information like resolution, frame count, starting timecode, pixel format and sample aspect ratio is stored.

Pixel format is stored as ARGB_4444_8u as input pixel format will be converted to this one. Framerate needs to be also converted to Adobe Premiere Pro framerate that is in ticks per frame.

Supported pixel formats

Support for different pixel format is defined in the handle of call *imGetIndPixelFormat*. This selector is new in CS6 called to enumerate the pixel formats available for the particular file. Pixel formats should be returned in the preferred order

This call is executed iteratively until *imBadFormatIndex* is received. In case, no pixel format is supported *imUnsupported* should be returned on the first call. Implementation is the same as for video formats, but pixel formats are used.

It is recommended to use the closest pixel format to the data in the video file. However due to the further parsing, merging and tone mapping of frames it is not useful to store the frame in Adobe Premiere Pro in YUV420p pixel format. Because of this pixel format with easier representation and parsing is used. Standard ARGB with 4 bytes for channels is used which gives support for more straightforward frame tone mapping.

Frame processing

Most critical part in the HDR plug-in is the actual frame processing. This is done in *imGetSourceVideo* call. Adobe Premiere Pro provides the video frame number with the call and requires frame stored to allocated memory in a buffer.

As HDR frames and even theirs tone mapped version are mostly not efficient for storing purpose it is undesirable to load, parse, merge and tone map frame whenever Adobe Premiere requires a particular frame. Because of this caching is used. Every tone mapped frame is before returning it to Adobe Premiere Pro stored in the cache with PPixCacheSuite. This suit provides support for caching frames and their effective displaying in Adobe Premiere Pro.

In case the frame is not in a cache it will be loaded from the disk. Adobe Premiere Pro is loading frames by the bulks of five. When the specific frame is requested Adobe Premiere Pro iteratively asks for five frames following the particular frame. This is making implementation of frame loading more efficient for plug-ins that are doing post-processing on frames before saving it to the memory of Adobe Premiere Pro such as HDR plug-in.

After the correct frame is found (which is explained in section 4.2) parsing of the frame can start. As YUV420p format is not compatible for pixel by pixel merge and global tone mapping which is used, so the frame is converted to ARGB_4444_8u. This format is storing each channel as 8bit unsigned int value as also storing alpha channel which is ignored as it is not used in HDR frame

As the frame is stored in ARGB format frame containing information about lighting and frame containing information about color can be merged to single LDR frame. To acquire the effect of HDR image on LDR display, simple global tone mapping operator is used. This is because merging and tone mapping of HDR image must be done in real time. Computation of this simple operator is complicated enough to cause shattering on higher resolution videos (1920x1080). Parsing, merging, and tone mapping of two lines together from both color and lighting information frames can be seen in figure 4.4

4.2 Frame seeking problem

In the basic video playing software, there are only a few things that video player must support. Video playing, pausing the video and seeking in a video. First two mentioned are well supported by FFmpeg, as this library supports almost every current video format. The third necessary functionality seems to be a problem of implementation in FFmpeg.

FFmpeg library support frame number accurate seeking, however, this information is just partially true. There are three types of frames in a video.

- **I-frames** are uncompressed frames.
- **P-frames** can use data from previous frames for decompression, this way they support a higher level of compression than I-frames.
- **B-frames** use both past and future frames for decompression to get the highest possible amount of data compression.

Because of this compression in video frames FFmpeg library is only able to find frames that are not dependent on other surrounding frames (I-frames). This way frame accurate seeking is accurate just on the nearest I-frame.

Furthermore, FFmpeg are using two kinds of timestamps for frame timing.

- **Presentation time stamp** (PTS) is a timestamp used for specifying the displaying frame.
- **Decoding time stamp** (DTS) is a timestamp used for specifying the time of decoding and is necessary if video stream had B-frames that needs to be decoded in a different order than stream without B-frames.

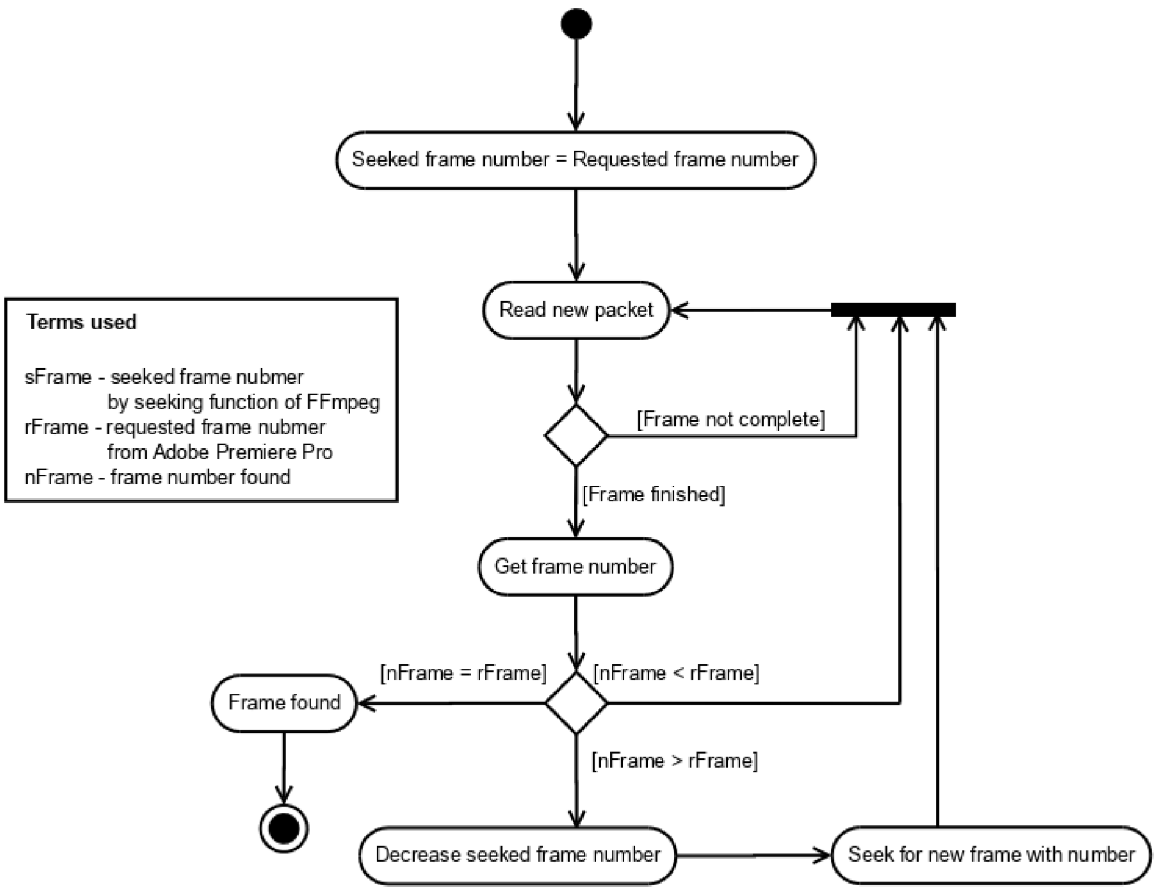
Most of the frame seeking in FFmpeg is done by Decoding time stamp, but this is not true for everything. Muxers and demuxers are using the presentation time stamp. However, just for frame seeking it is sufficient to get the frame number from the decoding time stamp.

Problem frame accurate seeking in FFmpeg is in tracking of frame numbers. Many flags through the development process were proposed for easier frame seeking. However, there are examples where these flags are not working, and they need to be checked if the frame found is frame sought.

Among most used flags in seeking in FFmpeg is AVSEEK_FLAG_BACKWARD. Using this flag on frame seeking, FFmpeg will found the closest I-frame before specified frame number. In some cases when trying to found I-frame, this flag provides I-frame

```
void mergeHDRLine(char* inTop, char *inBottom, int width, char* out) {
    for (int pixel = 0; pixel < width*4; pixel+=4) {
        // It is sufficient to get only one channel of lighting frame as all the channels are the same
        unsigned char topGreyScale = (*(inTop + pixel + 1));
        // Color channels are acquired from the frame with color information
        unsigned char botRed = (*(inBottom + pixel + 1));
        unsigned char botGreen = (*(inBottom + pixel + 2));
        unsigned char botBlue = (*(inBottom + pixel+3));

        // Values need to be normalized to interval <0.0, 1.0)
        float r = botRed / 256.0f;
        float g = botGreen / 256.0f;
        float b = botBlue / 256.0f;

        float lx = topGreyScale ? (topGreyScale / 256.0f) : 0.0f;
        // Tone mapping by global operator applied
        lx = lx / (1.0f - lx);

        lx = pow(lx , 0.4545f);
        b = pow(b, 0.4545f);
        r = pow(r, 0.4545f);
        g = pow(g, 0.4545f);

        float outR = r * lx * 8;
        float outG = g * lx * 8;
        float outB = b * lx * 8;

        // Normalization after tone mapping
        outR = outR > 1 ? 1 : outR;
        outG = outG > 1 ? 1 : outG;
        outB = outB > 1 ? 1 : outB;

        //Store as ARGB
        (*(out + pixel)) = (unsigned char) 255;
        (*(out + pixel + 1)) = (unsigned char)(outR * 255);
        (*(out + pixel + 2)) = (unsigned char)(outG * 255);
        (*(out + pixel + 3)) = (unsigned char)(outB * 255);
    }
}
```

Figure 4.4: Merge of two frames. *inTop* is a buffer storing the frame containing lighting information, *inBottom* is the bottom frame of double HDR frame containing information about color. For tone mapping simple global operator is used.



Seeked frame number = Requested frame number
Read new packet
Terms used
sFrame - seeked frame nubmer by seeking function of FFmpeg
rFrame - requested frame nubmer from Adobe Premiere Pro
nFrame - frame number found
[Frame not complete]
[Frame finished]
Get frame number
[nFrame = rFrame]
[nFrame < rFrame]
Frame found
[nFrame > rFrame]
Decrease seeked frame number
Seek for new frame with number

Figure 4.5: Activity diagram of seeking correct frame.

after specified frame, which is undesired. Other flags implemented for easier seeking are AVSEEK_FLAG_ANY, which makes seeking to B-frames and P-frames. However, this flag can be inaccurate and seek to nearest I-frame.

Most accurate solution to frame accurate seeking in FFmpeg (and proposed by FFmpeg developers) is reading the video file from start to the found frame. As this method is ineffective for video processing, we need to implement a better option for frame accurate seeking which can be seen in figure 4.5.

The problem of identifying the frame number is not always solved by video files. Some video files do not have frame numbers stored in each frame. As a solution to this problem decoding time stamp can be used. Dividing decoding time stamp by timebase (clock ticks per frame) and multiplying by fps (frames per second) correct frame number can be acquired.

This seeking algorithm is seeking for a particular frame in iterations. In each iteration, new seek of a frame is done. Afterward, three different conditions can happen based on the number of found frame.

- **Frame found is the desired frame.** In this case, the correct frame was found, and search for the frame is over and further frame processing can be done.
- **Frame found has lower number then sought frame.** No more seeking is re-

quired. In this case reading new frames and checking their frame number with a frame number of requested frame will lead to increasing frame number of found frame by one until the correct frame is found.

- **Frame found has higher frame number than required frame.** This is the main problem of frame-accurate seeking. In this case, there is no way to read back few frames. A new find for the frame must be done. However asking for the same frame number would give the same result. A number of sought frame is lowered to find I-frame with a lower frame number and the whole process is repeated.

Another problem of video seeking by FFmpeg in video files is finding the first frame of the video. In some video files, the first frame is not I-frame. Therefore, it is not seekable by FFmpeg. This is solved by the current implementation. On file opening decoding few frames without seeking will result in acquiring first frames even when they are not I-frames. Current implementation takes this into account and asks for a new frame only in case current frame number is bigger than desired frame.

4.3 Final product

The current implementation of HDR plug-in for Adobe Premiere Pro is implemented on top of Adobe Premiere Pro CS6 SDK. Because of this it is supported by the same version of Adobe Premiere Pro. As no features from Adobe Premiere Pro CS6 SDK were used, it is also able to be deployed on lower version of Adobe Premiere Pro up to version CS3. The plug-in enables support for import of video files with extension „.hdrv“ which are in specific video format containing both luminance and color information frames in single frames on top of each other. Video files with other extension but in the correct format can be also imported, but the user has to select specifically correct format provided by HDR plug-in in the import menu.

The difference between the import of the video by standard Adobe Premiere Pro MPEG-4 importer and HDR plug-in can be seen in figure 4.6. On the right player video imported with the standard MPEG-4 importer of Adobe Premiere Pro can be seen. While on the right player in Adobe Premiere Pro video was imported by HDR plug-in implemented, after import video was tone mapped and stored in a memory of Adobe Premiere Pro. Tone mapped frame is visible on the right side of the screenshot. Screenshots of exported video can be seen in figure 4.7[1]. For comparison, both frames with luminance and color information are shown as well.

As the video is imported into one of the supported pixel formats by Adobe Premier Pro, it is possible to do all the editing that Adobe Premiere Pro provides. Export of the video can be done by any standard exporter of Adobe Premiere Pro

[1]Borders added for better readability

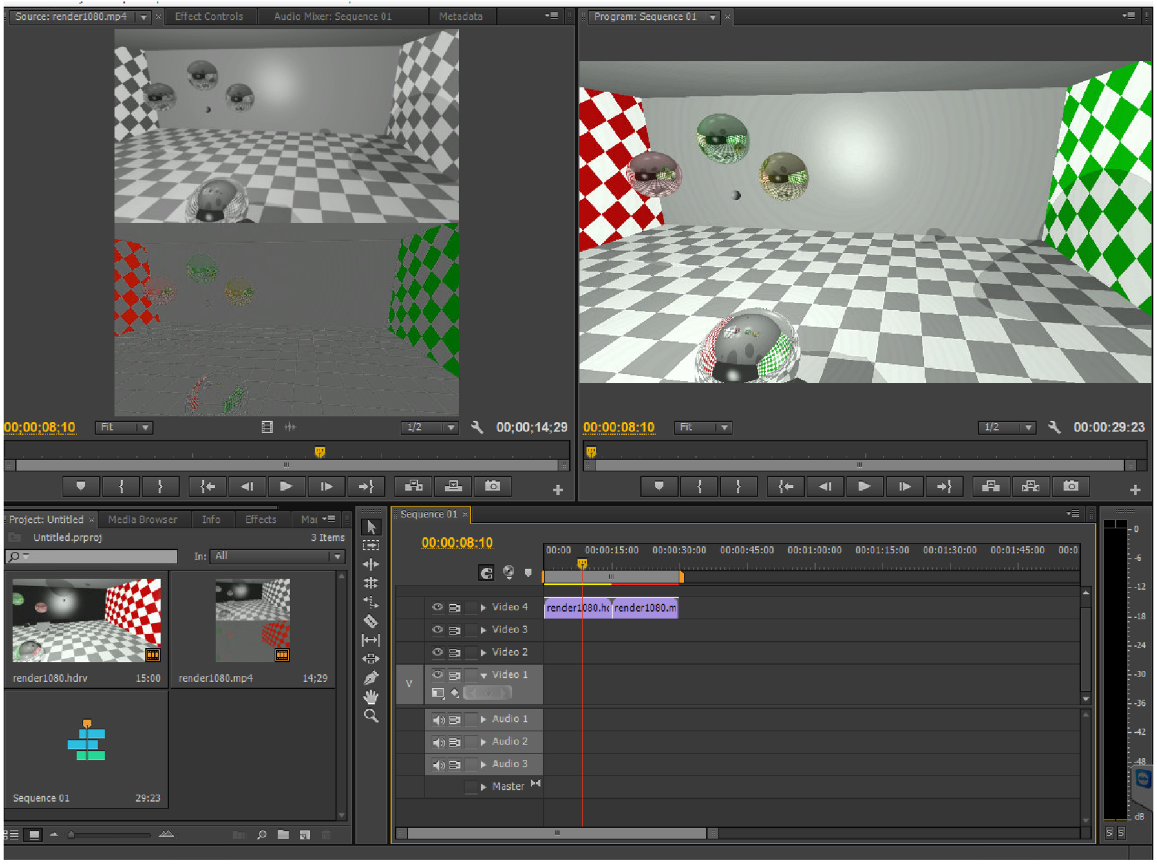


Figure 4.6: Import of HDR video by standard importer and by HDR plug-in.

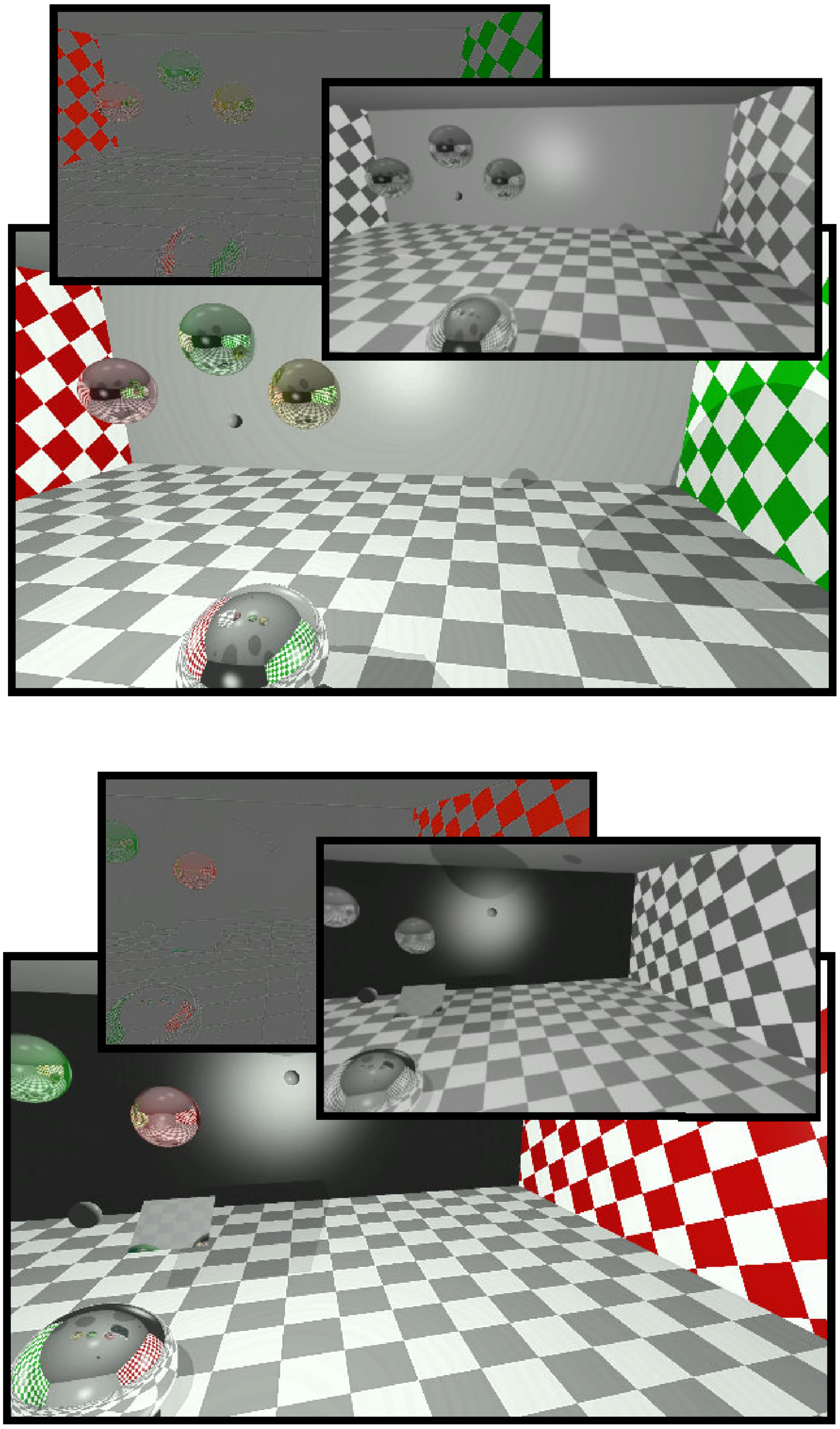


Figure 4.7: Frames of HDR video rendered into tone mapped frame.

Chapter 5

Conclusion

The goal of this thesis was to introduce new plug-in that will enable support of high dynamic range videos for Adobe Premiere Pro, which was successful. This is necessary because current plug-ins for this software only enables support for still images which makes import and export of these images to Adobe Premiere Pro available, but lacks all the features of editing process on video files. Importing video file to Adobe Premiere Pro with implemented plug-in will enable video processing such as transitions, effects, and more on video.

This practical part, HDR plug-in, was created under the supervision of Professor Alan Chalmers at the University of Warwick, who provided all the necessary hardware, software and information which was crucial for the success of this thesis. The theoretical part, this thesis, was written under the supervision of Prof. Dr. Ing. Pavel Zemčík, who provided vital help on the document structure, and supervision on all the chapters included.

Adobe Premiere development and all the specific plug-ins supported by current SDK version used are mentioned in chapter 2 as well as HDR algorithms with relevant literature cited. This chapter also describes external libraries needed for development and discuss their features. A suitable type of plugin for Adobe Premiere Pro to handle HDR video editing was outlined in the third chapter with the design used for the implementation and video format supported. Implemented plugin for HDR support is described in Implementation chapter, and its functionality is described in 4.3 as well as achieved results. Also screenshots of the final product and rendered video can be visited here. Future work will be described in the latter part of this conclusion.

Since this work has a strong connection to high dynamic range imaging, I gained knowledge that I could have gained only by working with the actual content. The problem of developing plug-in for Adobe Premiere seemed at start like a difficult task, mainly because long response time of community working on plug-in development for Adobe and Adobe SDK developers[1]. However after further investigation on all the sample code that are available and of some plug-in that are released as open source, the development process quickened. I would recommend reading through all he documentation of adobe SDK and for sure looking at example codes that is provided with Adobe Premiere SDK, as without reading through the code the development is a bit obscure and many times manual debugging is required.

For the future of plug-in for Adobe Premiere Pro, this plug-in should be moved to Adobe Premiere Pro CC version which provides a proper format for high dynamic range

[1]Response on unexplained structure parameters of documentation took 6 weeks, after which it was unnecessary.

frames storing. This way Adobe Premiere would be able to store HDR frames represent them in players but especially render video in a high dynamic range format. Once this is done development on other high dynamic range video format can start and support of this plug-in with ProEXR plugin will be available. Next step for HDR video would be to enable import and export of audio in high dynamic range video files, which is a necessary condition for video editing of movies. However for this more content of HDR videos with audio must be created. After this future development, the HDR plug-in will have full support for video files and this effort might lead to bigger public usage of HDR video files, which may result in making high dynamic range videos and images available for everyone.

Bibliography

[1] Jaloxa - Tone-mapping. http://www.jaloxa.eu/webhdr/tonemapping.shtml, accessed: 2015-07-08.

[2] TIFF Specification. 1995.
URL http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

[3] Bilissi, E.; Langford, M.: *Langford's Advanced Photography*. Taylor & Francis, 2007, ISBN 9781136093654.
URL https://books.google.cz/books?id=RridTrm9cWQC

[4] Birley, S.: *Real-time Tone Mapping: Rendering HDR Content Via the GPU*. Lap Lambert Academic Publishing GmbH KG, 2011, ISBN 9783846559734.
URL https://books.google.cz/books?id=AP-5pwAACAAJ

[5] Bloch, C.: *The HDRI Handbook 2.0: High Dynamic Range Imaging for Photographers and CG Artists*. Rocky Nook, 2013, ISBN 9781457179334.
URL https://books.google.cz/books?id=bju4BAAAQBAJ

[6] Books, H.: *Free Video Software, Including: Mplayer, Cinepaint, Xine, Cinelerra, Virtualdub, Virtualdubmod, VLC Media Player, Video4linux, Media Player Classic, Xbmc, Mythtv, Freej, Yatta, Cankiri, Cinefx, X264, Kino (Software), Lives, Avisynth, Mediaportal, Pitivi*. Hephaestus Books, 2011, ISBN 9781242801570.
URL https://books.google.cz/books?id=EB7pygAACAAJ

[7] Cooper, T.: *HDR Photography: From Snapshots to Great Shots*. From Snapshots to Great Shots, Pearson Education, 2015, ISBN 9780134181233.
URL https://books.google.cz/books?id=z7kDCgAAQBAJ

[8] Davis, H.; Davis, P.: *The Photoshop Darkroom: Creative Digital Post-Processing*. Taylor & Francis, 2014, ISBN 9781136097188.
URL https://books.google.cz/books?id=zJj_AwAAQBAJ

[9] Debevec, P. E.; Malik, J.: Recovering High Dynamic Range Radiance Maps from Photographs. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, New York, NY, USA: ACM, 2008, p. 31:1–31:10, doi:10.1145/1401132.1401174.
URL http://doi.acm.org/10.1145/1401132.1401174

[10] Fairchild, M. D.: Color Appearance Models. John Wiley & Sons Ltd, 2005.

[11] Farace, J.; Staver, B.: *Better Available Light Digital Photography: How to Make the Most of Your Night and Low-Light Shots*. Taylor & Francis, 2008, ISBN 9781136090387.
URL https://books.google.cz/books?id=75PAAwAAQBAJ

[12] Ferwerda, J. A.: Elements of Early Vision for Computer Graphics (Tutorial). *IEEE Computer Graphics and Applications*, volume 21, nr. 5: p. 22–33.
URL http://dblp.uni-trier.de/db/journals/cga/cga21.html#Ferwerda01

[13] fnord: ProEXR - Ultimate OpenEXR Plug-Ins.
http://www.fnordware.com/ProEXR, accessed: 2015-28-07.

[14] fnord: *ProEXR, Advance OpenEXR plug-ins*. June 2015.
URL http://www.fnordware.com/ProEXR/ProEXR_Manual.pdf

[15] Gerlach, J.: *Digital Nature Photography: The Art and the Science*. Taylor & Francis, 2015, ISBN 9781317805144.
URL https://books.google.cz/books?id=PGxKCAAAQBAJ

[16] Grossberg, M.; Nayar, S.: High Dynamic Range from Multiple Images: Which Exposures to Combine? In *ICCV Workshop on Color and Photometric Methods in Computer Vision (CPMCV)*, Oct 2003.

[17] Harrington, R.: *Photoshop for Video*. Taylor & Francis, 2012, ISBN 9781136065989.
URL https://books.google.cz/books?id=rXUqBgAAQBAJ

[18] Harrington, R.; Carman, R.; Greenberg, J.: *An Editor's Guide to Adobe Premiere Pro*. Pearson Education, 2012, ISBN 9780133065534.
URL https://books.google.cz/books?id=cuEtgLkLIUgC

[19] Incorporated, A. S.: *Adobe Premiere Pro CS6, Software Development Kit Guide*. 2012.

[20] Incorporated, A. S.: *After Effects CS6, SDK Guide*. 2012.

[21] Jacobson, R.: *The Manual of Photography: Photographic and Digital Imaging*. Media Manual Series, Focal Press, 2000, ISBN 9780240515748.
URL https://books.google.cz/books?id=Z95TAAAAMAAJ

[22] Jaeseok, K.; Shin, H.: *Algorithm & SoC Design for Automotive Vision Systems: For Smart Safe Driving System*. SpringerLink : Bücher, Springer Netherlands, 2014, ISBN 9789401790758.
URL https://books.google.cz/books?id=tfMkBAAAQBAJ

[23] Kriss, M.: *Handbook of Digital Imaging*. Wiley, 2015, ISBN 9780470510599.
URL https://books.google.cz/books?id=bh1wCAAAQBAJ

[24] Lucas Digital Ltd. LLC.: *Technical Introduction to OpenEXR*. November 2006.
URL http://www.openexr.com/TechnicalIntroduction.pdf

[25] Lukac, R.: *Computational Photography: Methods and Applications*. Digital Imaging and Computer Vision, CRC Press, 2010, ISBN 9781439817506.
URL https://books.google.cz/books?id=rGWxB7rkJl8C

[26] McCann, J.; Rizzi, A.: *The Art and Science of HDR Imaging*. The Wiley-IS&T Series in Imaging Science and Technology, Wiley, 2011, ISBN 9781119952121.
URL https://books.google.cz/books?id=l-YGi1KgBowC

[27] McCollough, F.: *Complete Guide to High Dynamic Range Digital Photography*. A Lark photography book, Lark Books, 2008, ISBN 9781600591969.
URL https://books.google.cz/books?id=6jK_jXsWZXsC

[28] Microfilms, U.; International, U. M.: *Dissertation Abstracts International: The sciences and engineering*. University Microfilms, 2008.
URL https://books.google.cz/books?id=VKAvAQAAIAAJ

[29] Mikhail, E.; Bethel, J.; McGlone, J.: *Introduction to modern photogrammetry*. v. 1, Wiley, 2001, ISBN 9780471309246.
URL https://books.google.cz/books?id=JdNTAAAAMAAJ

[30] Myszkowski, K.; Mantiuk, R.; Krawczyk, G.: *High Dynamic Range Video*. Synthesis Lectures on Computer Graphics and Animation, Morgan & Claypool Publishers, 2008, ISBN 9781598292152.
URL https://books.google.cz/books?id=2YteAQAAQBAJ

[31] Reinhard, E.; Heidrich, W.; Debevec, P.; et al.: *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann series in computer graphics, Elsevier Science, 2010, ISBN 9780080957111.
URL https://books.google.cz/books?id=w1i_1kejoYcC

[32] Seetzen, H.; Heidrich, W.; Stuerzlinger, W.; et al.: High Dynamic Range Display Systems. *ACM Trans. Graph.*, volume 23, nr. 3, August 2004: p. 760–768, ISSN 0730-0301, doi:10.1145/1015706.1015797.
URL http://doi.acm.org/10.1145/1015706.1015797

[33] Smith, W.: *Modern Optical Engineering, 4th Ed.* McGraw Hill professional, McGraw-Hill Education, 2007, ISBN 9780071593755.
URL https://books.google.cz/books?id=DrtM_bAnf_YC

[34] for Standardization, I. O.; Commission, I. E.: *Information Technology–coding of Audio-visual Objects: Part 10: Advanced Video Coding*. International standard, ISO/IEC, 2005.
URL https://books.google.cz/books?id=BBq2MQAACAAJ

[35] Tan, T.; Ruan, Q.; Wang, S.; et al.: *Advances in Image and Graphics Technologies: Chinese Conference, IGTA 2014, Beijing, China, June 19-20, 2014. Proceedings*. Communications in Computer and Information Science, Springer Berlin Heidelberg, 2014, ISBN 9783662454985.
URL https://books.google.cz/books?id=QszlBAAAQBAJ

[36] Team, A. C.; Systems, A.: *Adobe Premiere Pro CS6 Classroom in a Book*. Classroom in a book, Adobe Press, 2012, ISBN 9780321822475.
URL https://books.google.cz/books?id=a7VvNofJXZUC

[37] Tumblin, J.; Turk, G.: LCIS: A Boundary Hierarchy for Detail-Preserving Contrast Reduction. In *SIGGRAPH*, 1999, p. 83–90.
URL
http://dblp.uni-trier.de/db/conf/siggraph/siggraph1999.html#TumblinT99

[38] Ward, G.; Simmons, M.: Subband Encoding of High Dynamic Range Imagery. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, APGV '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-914-4, p. 83–90, doi:10.1145/1012551.1012566.
URL http://doi.acm.org/10.1145/1012551.1012566

[39] Ward, G. J.: The RADIANCE Lighting Simulation and Rendering System. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, New York, NY, USA: ACM, 1994, ISBN 0-89791-667-0, p. 459–472, doi:10.1145/192161.192286.
URL http://doi.acm.org/10.1145/192161.192286

[40] Willmore, B.: *Adobe Photoshop CS2 Studio Techniques*. Number v. 1 in Adobe Photoshop CS2 Studio Techniques, Adobe Press book is published by Peachpit Press, 2006, ISBN 9780321321893.
URL https://books.google.cz/books?id=3HK6rrxHQUYC

Appendix A

CD

A.1 Programs

A.2 Electronic version of this document