TONE MAPPING OPTIMIZATION FOR REAL TIME APPLICATIONS

by

Siwei Zhao

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2019

Table of Contents

List of Tables

List of Figures

Abstract

Many tone mapping algorithms for natural and synthetic images have been proposed in recent years. However, manually tuning their parameters is difficult, especially if the scene is dynamically changing over time or the view point is altered, as is the case in real time applications. In this thesis, we propose modifications to an automated parameter tuning algorithm which can replace the manual tuning work. This algorithm can optimize the parameters of tone mapping operators by minimizing the perceptual distortion using an evolution strategy. The perceptual distortion is measured by a feature-based objective image quality metric. This metric utilizes a contrast-enhanced virtual photograph technique by using a content-adaptive tone mapping operator specially designed for rendered HDR content characteristics. We show that, our tone mapping results with optimized parameters preserve more visible contrast than other optimization algorithms.

A further related issue in real time applications is the interpolation of tone mapping parameters. Standard practice uses a color grading pipeline to encode tone mapping results and color grading modifications into look-up tables (LUTs). Using look-up tables, artists can reproduce and apply certain luminance and color grading processes efficiently and easily. However, the interpolation of tone mapping and color grading parameters between locations where look up tables have been designed is problematic. We present a further algorithm for real-time interpolation of multiple look-up tables in video games accelerated by compute shaders on the GPU. Our method utilizes compute shaders to parallel counting the pixels belonging to each LUT and change the color grading effects using a post processing volume. We show that our results have a closer visual appearance to the reference images when the player changing the view point and can be executed with a fluent frame rate without overly taxing the CPU.

List of Abbreviations Used

HDR	High Dynamic Range
HVS	Human Visual System
LDR	Low Dynamic Range
TMO	Tone mapping operator
ACES	Academy Color Encoding System
LUT	Look-up Table
EA	Evolutionary Algorithm
DRIM	Dynamic Range Independent Metric
TMQI	Tone Mapped Image Quality Index
EOTF	Electro-Optical Transfer Function
JND	Just Noticeable Difference
PQ	Perceptual Quantization
SSIM	Structural Similarity Index
FSIM	Feature Similarity Index
BRISQUE	Blind /Referenceless Image Spatial Quality Evaluator
NSS	Natural Scene Statistics

Chapter 1

Introduction

High Dynamic Range (HDR) technology has been extensively studied in recent years. The goal of using HDR technology is capturing and displaying a wide range of luminance values which is close to the dynamic range perceived by the human visual system (HVS). The dynamic range of real-world luminance which can be perceived by the human eyes has a very high dynamic range of 10000 to 1 from direct sunlight to shallow starlight [1]. The human eye can perceive a dynamic range of about 5 orders of magnitude in real life scenes at a single adaptation time [2]. However, the dynamic range of traditional Low Dynamic Range (LDR) displays is only about 2 to 3 orders of magnitude which is smaller than the brightness and color range that humans can see.

In recent years, physical-based rendering techniques in HDR [3] have been widely used in the video gaming industry. They aim to render more realistic gaming contents with richer colors, brighter highlights and more details in the shadows to simulate light propagation in real life. The new generation of game engines has the ability to render HDR light values between 0 and 65,000. Therefore, HDR images allows more intensity levels to capture and store the visual data of real world and virtual scenes.

HDR images are created and stored in a format which has from 16 bits to 32 bits in each of the RGB channels. However, traditional LDR displays cannot show the darker and brighter HDR luminance with only 8 bits. The process of compressing the HDR luminance values for LDR displays is known as tone mapping. The goal of tone mapping is using tone mapping operator (TMO) to achieve a visual match between the observed HDR scenes and the tone mapped images on LDR display. A good tone mapping algorithm improves the visual quality of images. As Figure 1 shows, compared with rendered images with linear tone mapping, well-designed tone mapped rendered images preserve more details in the bright and dark areas. Over the past decade, a variety of TMOs have been introduced such as Academy Color Encoding System (ACES) TMO [4], Hable TMO [5] and Reinhard TMO [6].

Figure 1. Difference in visual quality between rendered image with linear tone mapping (The left) and well-designed tone mapped rendered image (The right).

HDR color grading in video games has also attracted considerable attention in recent years. Color grading is the process of modifying the colors of input images in various ways such as changing brightness, contrast and hue to achieve a specific artistic tone. In particular, a look-up table (LUT) based color grading pipeline has been widely used in video games such as Uncharted 4 [7]. Using look-up tables, artists can reproduce and apply certain color grading processes easily. Previously, artists enhanced the image color by adjusting parameters like vibrancy and saturation using intuitive sliders and controls. However, color grading effect is usually complex to reuse and it is hard to copy specific color grading processes from professional tools to a game engine. To solve this problem, Selan [8] shows how to store the color grading process into a three-dimensional LUT. LUT is a three-dimensional lattice of output RGB color values that can be indexed by sets of input RGB color values. The LUT explicitly stores the input to output conversion efficiently. It can be used for any mapping from input to output colors under the resolution limit. One of the advantages of using a LUT is to reproduce a complex grading process easily by baking a grading process into 3D lookup tables which can be authored by professional grading software such as DaVinci Resolve [9] and Adobe Photoshop [10]. Another advantage is that potential speedups are enormous. The color grading process using a 3D LUT is approximately 100 times faster than standard color correction without a LUT, reducing the process from 2.5 million operations to 32 thousand operations per frame [8]. Recent video games introduce a color grading workflow which applies various color grading effects easily using LUTs [7][11]. The artist can interactively choose a final effect from predefined LUTs and tweaks them as needed in real-time.

1.1 Research Problems

In recent years, many studies have focused on HDR images which have been used widely in many aspects such as digital photography, physical-based rendering, and virtual reality. One of the most important research problems for HDR images is tone mapping. Tone mapping operators usually have several tunable parameters to control the contrast of the tone mapped image. However, manually tuning many parameters is a challenging task for inexperienced developers. This challenging problem is also faced by photographic task which mapping high dynamic range of real-world luminance to the low dynamic range of the photographic print [6]. The default parameters of photographic global and local TMOs need to be manually tweaked to achieve satisfying performance. **Therefore, it is worthwhile to study automatic parameter tuning algorithms which can replace the work of manually tweaking parameters.**

LUT can encode tone mapping and color grading results into efficient color mapping functions. The LUT based HDR color grading pipeline has been extensively used in video games. To apply multiple LUTs in a single scene, the Unity and Unreal game engines introduce post processing volumes which is a box collider with boundaries to blend different LUTs. However, while the colors inside the volume are predefined by the LUT, the colors between multiple volumes are linearly blended together based on the distance towards the volume which may ruin the atmosphere of the game. Using simple linear interpolation of LUT values between the defined post processing volumes is problematic and not sufficient. **It is necessary to research the problem of preserving more LUT colors of the direction in front of the players when they are moving among multiple LUTs and adjusting their viewpoints.**

1.1.1 Objective Quality Evaluation for Tone Mapped Images

To solve the parameter tuning problem of tone mapping for single photographic images, Gao et al. [12] propose an evolutionary algorithm (EA) to automatically tune the tone mapping parameters by using objective quality of tone mapped images without human input. The tone mapping parameters with the best objective scores are selected during the iterations. In recent years, many objective image quality assessment methods have been introduced.

Aydin et al. [13] propose a dynamic range independent metric (DRIM) which defines the visible distortion based on the detection and classification of visible changes in the image structure. In this method, the contrast change is evaluated by three kinds of contrast variations including contrast loss, contrast amplification and contrast reverse when image pairs (tone mapped images and reference HDR images) are compared. The image pairs could have arbitrary dynamic ranges.

Yeganeh and Wang [14] have proposed Tone Mapped Image Quality Index (TMQI) which combines measures of structural fidelity and statistical naturalness in to a score. Nevertheless, TMQI exhibits some drawbacks which lead to inaccurate prediction for rendered HDR images. First, the TMQI quality is sensitive to the visibility threshold in the structural fidelity. In some cases, tiny local detail change of tone mapped image may lead to a significant difference in quality measure [15]. Secondly, the statistical naturalness measurement in TMQI tends to assign high scores for the tone mapped images with "average" brightness and contrast which is problematic for accurate quality prediction for dark and bright scenes with extreme luminance regions [16]. Ma et al. [15] have proposed TMQI-II which improve the structural fidelity and statistical naturalness of TMQI. However, both TMQI and TMQI-II involves naturalness quantification which was conducted with a subjective-ranked natural image database. The naturalness performance is not a crucial factor to evaluate computer rendered tone mapped images.

Gao et al. [18] propose a feature-based objective quality metric by taking the virtual photograph sequence from a HDR image and measuring the distortion of important image features. They apply a simple transfer curve often used in modern photography to take virtual photographs. This transfer curve is designed for tone mapping natural images. In this thesis we propose a method of automatic parameter tuning specifically for synthetic images.

1.1.2 Lookup Tables (LUTs) Interpolation In Video Games

Using a LUT is an efficient way to encode tone mapping results and color grading modifications. Waylon et al. proposed the color grading workflow of manual tweaking LUTs in Uncharted 4 [7]. In this workflow, the object colors in each view direction of player are carefully tweaked by the

artists to present a desired overall atmosphere. But this process needs the professional experience of artists. Therefore, it is necessary to research how to automatically tweak the LUT in real-time when the players move among multiple scenes and look around to replace the work of manual tweaking.

A method to blend LUTs uses a post processing volume and the world position of each pixel. Post processing volumes define the affect range of LUTs. The world position of each pixel on the screen can be reconstructed from the camera's depth. For the pixel inside the boundary of volume, the game engine can modify it by using the LUT color of that volume, so that every object can show their original color when the player moves between LUTs. Nevertheless, the color of each pixel needs to be recalculated in each frame which is very time-consuming. To solve this problem, we propose a pixel-based real-time interpolation method for blending multiple LUTs using compute shaders to accelerate the algorithm.

1.2 Objectives

The purpose of this thesis is to present an automatic tone mapping parameter tuning algorithm for video games which can replace the work of manually tweaking parameters and develop a pixel-based look-up table interpolation method which preserves more LUT colors of direct sight when moving among multiple LUTs and adjusting the viewpoint.

1.3 Contributions

The main contributions of the thesis are summarized as follows:

We propose modifications to the feature-based quality metric introduced Gao et al. [18] that make it suitable for use with synthetic images. Our algorithm utilizes a content-adaptive tone mapping operator to take virtual photograph for rendered HDR content. We show that, our results with optimized parameters preserve more visible contrast than other optimization algorithms (Chapter 3).

We also present an algorithm for real-time interpolation between multiple look-up tables in video games. We record the pixel count belonging to each LUT on the current screen which is optimized by GPU parallel processing of compute shaders. Compared with linear interpolation results, our results are closer to the colors of predefined LUT which should be presented exactly. After optimized by GPU, our algorithm can be executed fluently in real-time (Chapter 4).

1.4 Structure of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 describes the research topics and the existing methods that serve as the background of the thesis which covers HDR tone mapping, rendered HDR gaming content, perceptual quantization, full-reference and no-reference image quality metrics and the HDR color grading workflow. Chapter 3 presents the tone mapping optimization for HDR gaming content. Chapter 4 focuses on the real-time interpolation between lookup tables. Chapter 5 concludes the thesis and discusses the directions for future research.

Chapter 2

Background

This chapter provides an overview of the various fields related to research topics found throughout the thesis. First of all, we describe the concept of HDR tone mapping which include recent research about photographic and video game tone mapping operators (Section 2.1). Then, we discuss the techniques behind creating rendered HDR gaming content and unique characters of the rendered HDR content (Section 2.2). After that, we outline the concept of perceptual quantization (Section 2.3). Then, we discuss the objective image quality assessment of tone mapped images (Section 2.4). We also discuss the no-reference quality assessment for synthetic images (Section 2.5) and tone mapping for HDR video (Section 2.6). Finally, we discuss the HDR color grading workflow including 3D lookup tables and post-processing volumes (Section 2.7) and evolutionary optimization of objective tone mapped image quality metric (Section 2.8).

2.1 HDR Tone Mapping

In the real world, our visual system is presented with a wide range of colors and luminance. The standard unit of luminance is the candela per square meter (cd/m^2) or nits. Typical real-world luminance has a very high dynamic range of over 14 orders of magnitude ranging from direct sunlight (10^5 up to 10^8 cd/m^2) to shallow starlight (10^{-3} down to 10^{-6} cd/m^2). However, the human eyes can only perceive about 5 orders of magnitude in a single adaptation time [2] and LDR displays can display only 2 up to 3 orders of magnitude [19], as demonstrated in Figure 2.

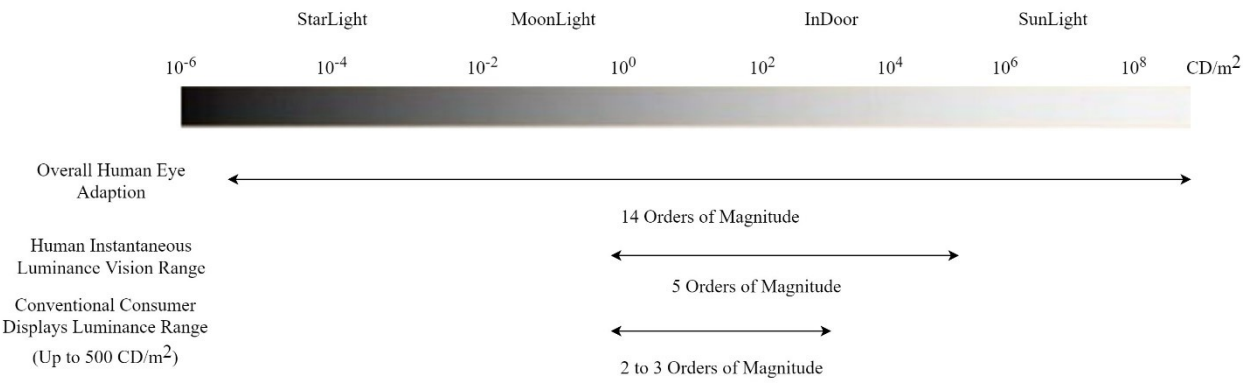


Figure 2. The dynamic range of real-world luminance and the capabilities of the human visual system and display devices.

The process of compressing the HDR luminance values for human eyes and LDR displays is known as tone mapping. However, the loss of visual quality such as contrast and detail information in dark and bright areas is inevitable during the process of tone mapping. To reproduce more visual accuracy of natural and synthetic HDR images, many tone mapping operators have been proposed in recent years as described in the following section.

2.1.1 Tone Mapping Operators

Tone mapping operators (TMOs) are designed to reproduce visibility as well as the brightness, contrast and color of the real world onto LDR displays. A good TMO should ideally compress the dynamic range of reference HDR image while maintaining details. In recent research, many TMOs are widely used in both photographic techniques and video games.

Reinhard et al. [6] propose the Photographic Reproduction TMO which focuses on creating pleasant photographic look to the results. First, this method employs a linear scaling to simulate the exposure adjustment in a camera. Then, contrast of image is locally adjusted using a computational model which is similar to photographic dodging-and-burning. By using this technique, dark regions of images are given more exposure and light regions are given less exposure.

More and more TMOs also have been proposed for video games. ACES [4] propose a carefully crafted filmic sigmoid curve. The curve is designed to match film characteristics in order to make games look like film. Considering real-time requirements of algorithm, developers usually use a simple approximate curve to fit the data of the full curve. Using this predefined curve might be an effective tone mapping solution, but it only provides a fixed contrast between light and shade. Thus, the tone mapped result does not always meet the requirements of developers very well.

In order to render video games into multiple film styles, Hable [5] propose a customizable filmic tone mapping curve by introducing some artistic controllable parameters. This filmic curve can be divided into three separate segments, a linear section, a shoulder and a toe. The strength and slope of each segment can be controlled by parameters.

GT tone mapping curve [20] proposed more intuitive controls with a smoother connection between toe and shoulder. However, the quality of tone mapping depends on personal experience of artists.

2.1.2 Content Adaptive TMO

Khaldieh [21] proposes content adaptive TMO which introduces a global piecewise-linear tone mapping curve. Content adaptive TMO can adjust the slopes of line segments based on the image histogram. For high populated luminance areas of the histogram, it will allocate more ranges during tone mapping to preserve contrasts of the image.

The procedure of this tone mapping operator is comprised of the following steps. First, the linear luminance values are transformed into the PQ domain which contains 1,024 JND thresholds. The PQ domain simulates the ability of human eyes to receive light in different levels. Compared with physical domain information directly extracted from the image, the PQ domain is a more perceptually meaningful domain with respect to the human visual system's properties. JND is the minimum difference between two consecutive light values that makes them distinguishable to our eyes. Then, because a Reinhard tone mapping curve [6] provides a good set of initial slopes, it is divided into 1024 straight line segments.

Secondly, divided bins are categorized into heavily populated and low populated by using maximum entropy thresholding [22]. Entropy is the measure of uncertainty of the output of an experiment, the higher the entropy, the more uncertain of the output. The maximum entropy thresholding function helps us to find the bin with the highest uncertainty of whether it belongs to the low populated or the heavily populated category. The purpose to find the range of concentrated luminance values for making specifically tone mapping. As Figure 3 shows the heavily populated bins and low populated bins are colored into green and purple.

Thirdly, in order to achieve the best visual quality in the tone-mapped content, the slopes of tone mapping curve for under-populated bins are adjusted based on the predefined lower bound function. The lower bound slope is calculated from the HVS response curve. The sensitivity of human eyes is changed under different luminance levels. However, there is a minimum sensitivity level that our eyes will not go below at any luminance level which defined by the HVS response curve. We use the HVS response curve proposed by Khaldieh [21] which calculates the response of human eyes at all light levels. This HVS response curve has been mapped into PQ domain and can be used as low boundary of our tone mapping curve. As Figure 3 left shows, after adjusting the slope of under-populated bins, a tone mapping range is extruded comparing with the upper boundary of Reinhard tone mapping curve. Then, that extruded range is equally allocated to the highly populated bins by increasing the slopes of line segments, as shown in Figure 3 right. Finally, the tone mapped image is compressed based on the piecewise tone mapping curve.

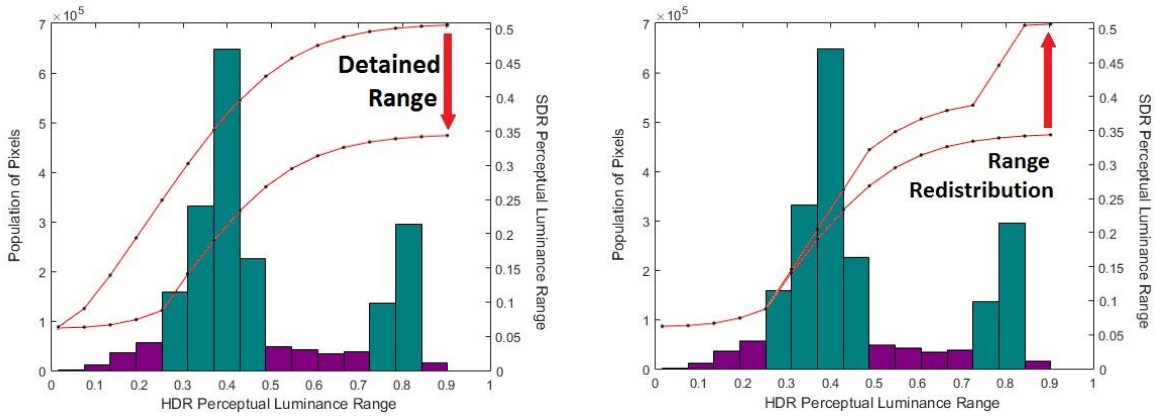


Figure 3. The demonstration of slopes readjustment of range detaining(left) and range redistribution(right). The images are taken from [21].

2.2 Rendered HDR Gaming Content

Rendered HDR contents are generated by physical-based rendering techniques [3] in game engines which can mimic light propagation in real life to render realistic results. However, simulating light propagation of every indirect light is too costly to be performed for computers. To save computation time, game engines use some algorithms to precompute light.

For example, in the Unity game engine, the precomputing of global illumination techniques can calculate the bounce of indirect light in the scene. For large objects, Unity creates lightmaps for each of them. Using lightmaps is accurate but it is computationally expensive to bake light and requires significant space to store the lightmap information. Considering these tradeoffs, for small and moving objects, light probes [23] have been proposed which preserve illumination information in advance and reconstruct it quickly in the rendering stage. Light probes are samples distributed throughout the 3D geometry of the rendered scene to store reflected light from surfaces as Figure 4 shows. In this figure, the yellow balls represent the light probes capturing the reflection light from the barrels and rocks. The faint purple lines among light probes show the paths of light calculated by the game engine. Due to computational expense, game engines only calculate the paths of light that belong to active light probes near the barrels.

The number of light probes affect the game performance, because the indirect lights among the object are calculated in real-time by interpolating the light probes around the objects. More light probes are used, more indirect light will be captured which will give more accurate HDR image.

Considering the requirements of real-time performance, the number of light probes is limited by game engines which leads to some indirect light not being captured by the light probes. Therefore, the number of illuminations rendered in the game scene is much less than the number of illuminations captured in real life [24]. Unlike natural images, luminance values in rendered HDR images are concentrated in smaller areas of the histogram and not spread all over the HDR range which lead to the spiky characteristic of rendered image histogram.

Figure 4. Multiple light probes generated by Global Illumination System in Unity.

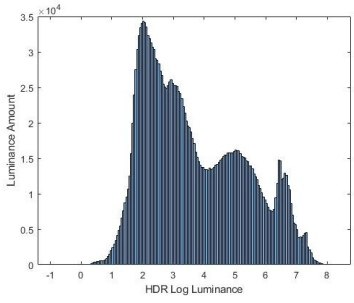
2.2.1 Unique Characteristics of the Rendered HDR Content

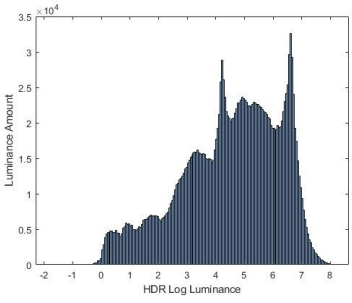
In this section, we compare the differences between the histograms of natural images and synthetic images. Physical-based rendering techniques in HDR [3] have been widely used in the video gaming industry. Unlike natural images, luminance values in rendered HDR images are

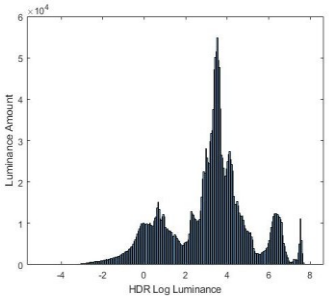
concentrated in smaller areas of the histogram and not spread all over the HDR range which lead to the spiky characteristic of rendered image histograms [21].

Figure 5 and Figure 6 show natural HDR images and synthetic HDR images with their luminance histograms proposed by the database of "TMIQD: Database of Tone-Mapped Natural and Computer Generated HDR Images" [17]. These images are tone mapped using the Reinhard TMO which is a traditional TMO widely used in photographic technology. During tone mapping, we get rid of the outlier luminance which is less than -15. To compare the difference of HDR content, these histograms are generated based on the original HDR images before tone mapping.

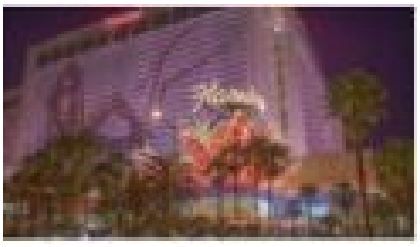
For each luminance distribution histogram, x axis is the log luminance of each pixel and y axis is the number of pixels. Comparing the natural histogram in Figure 5 and synthetic image histograms in Figure 6, we notice that the synthetic images are more spiky than natural image such as S3, S4, S7, S8. For spiky synthetic images such as S5, S7 and S8, the Reinhard TMO has difficulty to tone map these images. Because spiky characteristics are not usually found in natural images so traditional photographic TMOs designed for real-life captured HDR images are not good at processing these images. Many details in bright and dark areas are lost and S5, S7 and S8, and the Reinhard TMO does not well process the bright areas in the image. Therefore, it is necessary to discuss another TMO designed for synthetic images and consider the spiky characteristics.

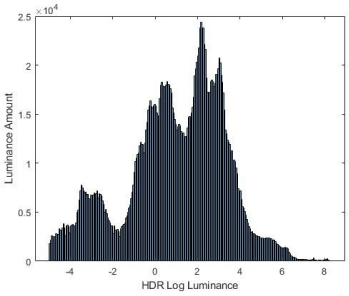


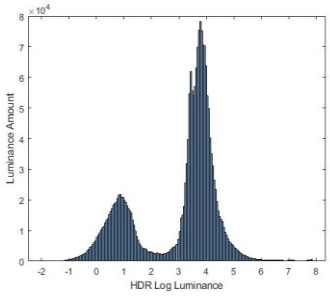


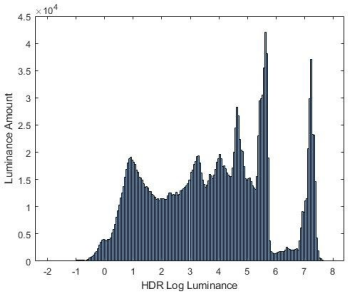


N1 N2 N3

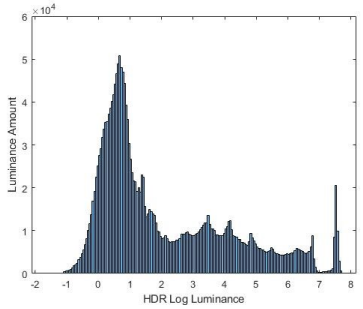


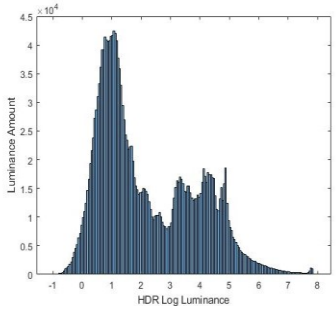


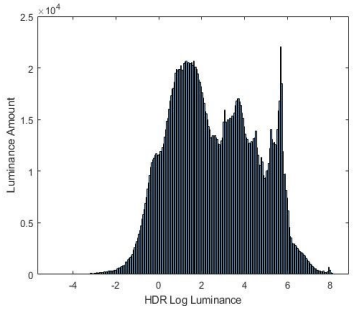




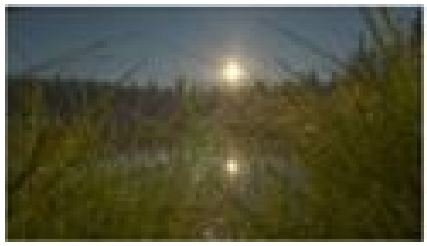
N4 N5 N6

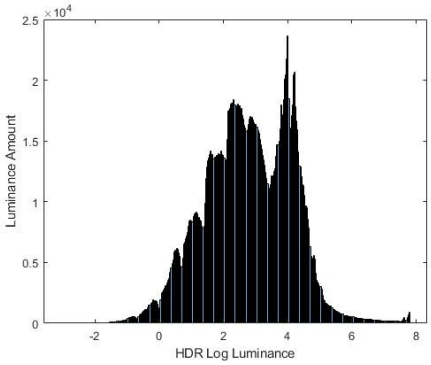






N7 N8 N9

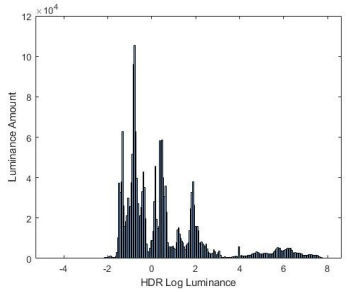


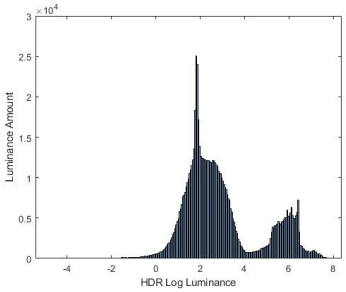


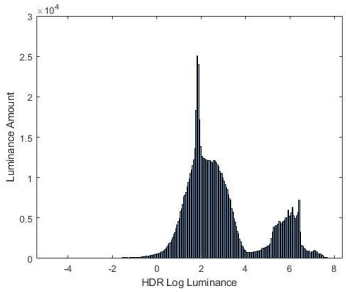
N10

Figure 5. Tone mapped natural HDR images with their corresponding distribution of luminance in the histograms.

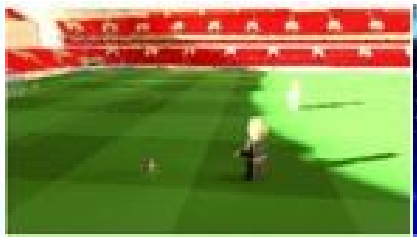


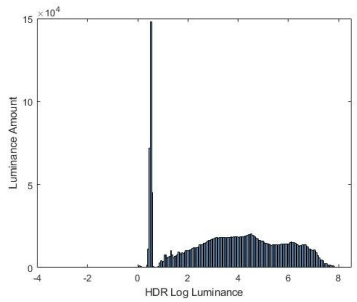


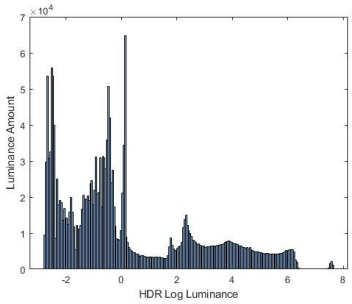


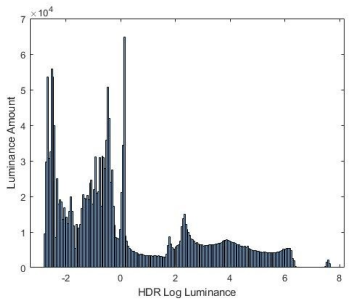


S1 S2 S3

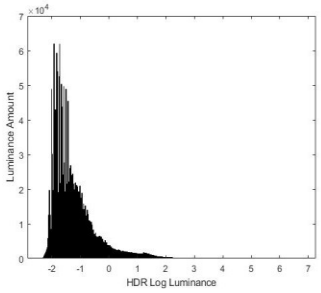



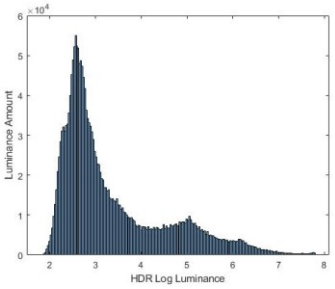



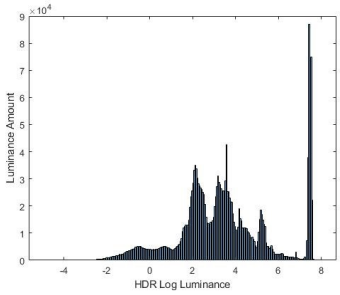




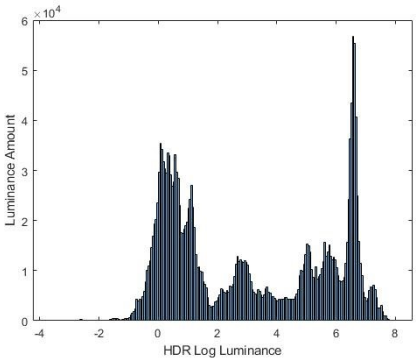
S4 S5 S6







S7 S8 S9



S10

Figure 6. Tone mapped synthetic HDR images with their corresponding distribution of luminance in the histograms.

2.3 Perceptual Quantization

Electro-Optical Transfer Function (EOTF) is a mathematical equation translating physical digital values into brightness values which can be present in the display devices. The traditional EOTF curve is called Gamma 2.4 which simulate the behavior of the cathode ray tube approximated by a 0-1 exponential curve with a power value of 2.4. Gamma 2.4 curve can efficiently compress luminance values that fall into the LDR luminance range (0.1 to 100 cd/m2) [25]. However, for high dynamic range monitors which can display a large range of light between 0.1 and 10000 cd/m^2, Gamma 2.4 curve is not well adaptive to wide dynamic range of displays.

Perceptual Quantization (PQ) is a new EOTF curve proposed by Dolby and standardized in SMPTE ST.2084 [26]. It is designed to allocate luminance range as efficiently as possible with respect to how the human vision perceives changes in light levels. Our human visual system does not perceive differences between consecutive light values equally along the full high dynamic range. The Just Noticeable Difference (JND) threshold [27] is the minimum difference between two consecutive light values that makes them distinguishable to our eyes. This minimum difference threshold increases in a nonlinear way as light values increase, and any two light values whose difference falls below the corresponding JND threshold will be perceived by our eyes as one light value. PQ is designed to convert light values from the physical domain to a perceptually linear domain which respect to the HVS properties. Comparing with traditional log curve, the PQ curve saves luminance range on the low side which gives it the ability to preserve more details in the bright areas. Compared with the PQ curve, the traditional gamma curve wastes luminance range on the high side which leads to steps in the dark areas.

Figure 7 shows the comparison between the PQ curve and the Gamma 2.4 curve. The x axis represents the display luminance. The y axis shows the physical luminance value. Compared with Gamma 2.4, the PQ curve is more sensitive to brightness less than 0.01 nits. It also can also present the brightness more than 100 nits which beyond the displayable range of Gamma 2.4.

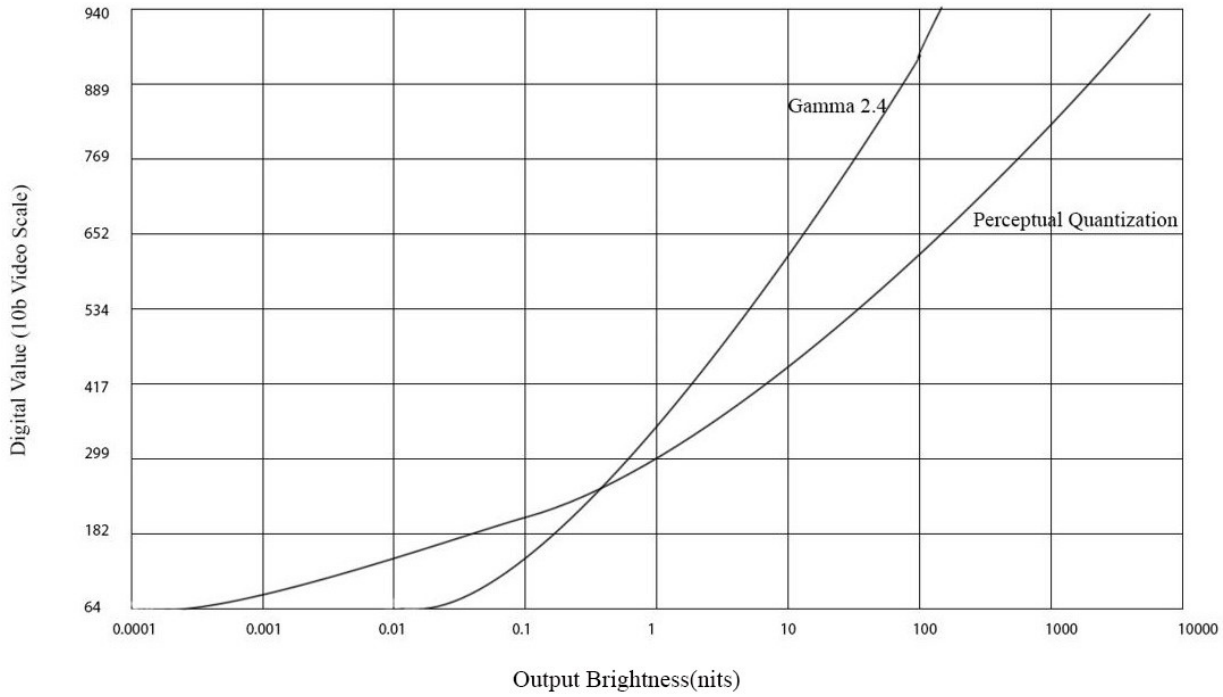


Figure 7. The comparison between Gamma 2.4 and Perceptual Quantization.

2.4 Objective Quality Assessment of Tone Mapped Images

Using objective metrics for evaluating tone mapped images is a challenging task. Many full reference image quality metrics such as Structural Similarity Index (SSIM) [28] and Feature Similarity Index (FSIM) [29] assume that the dynamic range of the reference and target image are the same. However, in the case of tone mapped HDR images, the dynamic range between reference and target are different. The following objective quality metrics for tone mapped images discussed in this section are designed to overcome this issue.

2.4.1 Feature-based Quality Metric

Gao et al. [18] propose Feature-based Quality Metric which analyzes image features by using the virtual photograph technique to evaluate the tone mapped image. The framework of the metric is shown in Figure 8 [18].

As Figure 8 shows, the virtual photos are used to extract features from the HDR reference image. Therefore, the final distortion output is influenced by the quality of the virtual photography sequence. During the process of taking virtual photo, the luminance of the virtual photo needs to be calibrated to adapt to different lighting conditions. Reinhard [30] introduces a luminance calibration algorithm with the key of the scene. The key indicates whether a scene is subjectively light, normal, or dark. For the tone mapping algorithm used in taking the virtual photos.

The procedure for evaluating the quality of tone mapped image is comprised of three steps: i) Calculate image features of tone mapped image including brightness, visual saliency and detail in bright and dark areas; ii) Take virtual photos to extract features from the HDR reference image using Reinhard TMO; iii) Calculate the distortions between tone mapped features and HDR image features and combine the normalized distortions together as a single quality score.

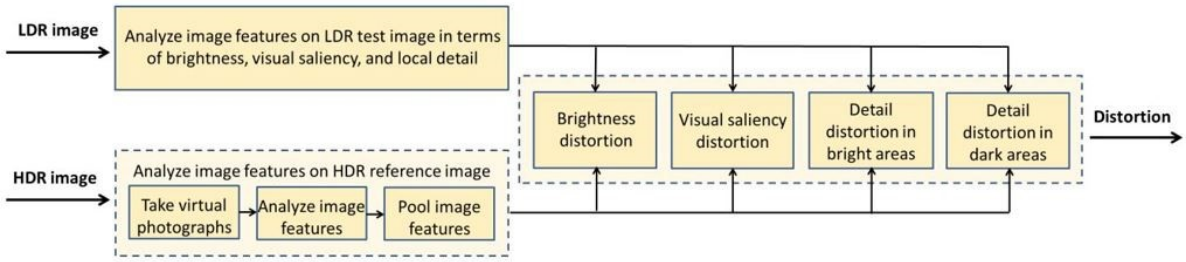


Figure 8. The framework of the feature-based quality metric for tone mapped images. The image is taken from [18].

2.4.2 TMQI, TMQI-II

Yeganeh and Wang have proposed TMQI [14] which combines measures of multi-scale structural fidelity and statistical naturalness into a score.

The structural fidelity measurement is an improved SSIM index [31]. The original SSIM algorithm consists of three comparison components including luminance measurement, contrast measurement and structure measurement. Due to the huge luminance change during tone mapping, it is inappropriate to compare these components between HDR and LDR images

directly. To solve this problem, the improved SSIM index create a new contrast component which does not penalize the difference in signal strength if the HDR and LDR images are both below or above the visibility threshold. The structural fidelity is measured at multiple scales of HDR and LDR images. Afterwards, the overall structural fidelity is computed by combining the scores of different scales.

The statistical naturalness measurement is based on the assumption that the high quality real-life tone mapped image should look "natural". And the naturalness of the tone mapped image can be calculated by probability distributions of brightness and contrast in natural images. The statistical naturalness is analyzed based on a database of 3000 images including many natural scenes in different light conditions. The overall statistical naturalness can be calculated by the probability model above.

The final TMQI is a combination of the two measures defined as:

$$Q = a * S^{\alpha} + (1 - a) * N^{\beta} \tag{2.1}$$

where S and N represent the structural fidelity and statistical naturalness, respectively, a controls the proportion of two components, and α and β are obtained from subjective data to control the components sensitivities ($\alpha = 0.3046$, $\beta = 0.7088$)

However, TMQI is too reliant on the HDR reference database during the measurement of structural fidelity and statistical naturalness. Ma et al. argue that the score of a tone mapped image in TMQI depends on the mean and the standard deviation of the reference image database. They propose a new version of the quality measure called TMQI-II [15]. To make the means and standard deviations more accurate, they designed a subjective experiment and let people adjust the means and standard deviations of 60 natural images, in order to find the lower and upper bounds for naturalness. However, both TMQI and TMQI-II involves naturalness quantification conducted with a subjective-ranked natural image database. The naturalness performance is not a crucial factor to evaluate computer rendered tone mapped images. Human observers tend to focus on some low level properties of the images such as contrast when evaluating synthetic images [17].

2.4.3 DRIM

Aydin et al. [13] propose DRIM which can generate a distortion map by comparing two images having different dynamic ranges. It includes the distortion detection model similar to that used in the HDR-VDP [32] which enables precise detection of visible contrast changes between HDR images and its tone mapped images.

The contrast detection predictor in DRIM generate three distortion maps including contrast loss, contrast amplification and contrast reverse distortion maps.

The loss of visible contrast evaluates the contrast which is perceivable in HDR but imperceptible in LDR image. The amplification of invisible contrast shows the contrast which is invisible in HDR reference but visible in tone mapped result. This distortion can be caused by some contrast enhanced tone mapping algorithms. Reversal of visible contrast is the contrast that can be seen in both reference and test images but with different polarity which is mostly caused by halo artifacts during the tone-mapping process.

To combine three distortion values into a single score, Krasula et al. [17] propose a simple algorithm which calculate the mean value of distortion values. In this thesis, I sum three types of distortion together to evaluate my results. Because my test images have the same number of pixels, the sum distortion and mean distortion have same results in different scales.

2.5 No-reference Quality Assessment for Synthetic Image

TMQI and TMQI-II discussed in section 2.4.2 are full-reference quality metrics designed for natural images based on natural scene statistics (NSS). NSS is an important tool for no-reference visual quality assessment, because the reference image is not needed for comparison. To apply this tool for synthetic images, Kundu and Evans [33] propose the synthetic image database which contains 500 distorted images (20 distorted images for each of the 25 original images) with 5 different distortion types such as blur, fade, and gaussian noise. Each distorted image has a corresponding subjective score ranging from 0 to 100 obtained by subjective tests for human observers.

Using this synthetic image database, Kundu and Evans [34] evaluate the performance of 17 no-reference image quality assessment algorithms using synthetic scene statics. The Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) proposed by Mittal et al. [35] is one of the best metrics among them. They propose that the distribution of normalized pixel intensities of natural images follows a Gaussian-like distribution while pixel intensities of distorted images do not. The NSS based features can be extracted from database. The differences between natural image distribution and distorted image distribution can be measured as distortions which can be used to evaluate images.

2.6 Tone Mapping for HDR Video

Tone mapping algorithms for HDR video is another related area in the HDR field. Many TMOs have been proposed and designed for static images. However, if these operators are applied directly on HDR video sequences, they may cause visual artifacts such as visual noise, flickering, ghosting and brightness and color inconsistencies. In this section, we will discuss two main type of artifacts which are flickering artifacts, temporal brightness incoherency and temporal noise. We will also discuss two video TMOs to reduce visual artifacts.

2.6.1 Flickering Artifacts

The main type of temporal incoherency that has been investigated is flickering artifacts. During tone mapping, the parameters of TMO control the shape of curve which affects the final tone mapped results. Flickering artifacts occur when the parameters of TMO change rapidly between consecutive frames which leads to similar HDR luminance values being tone mapped into different LDR luminance values in a short time. These artifacts appear because the TMO change their parameters using image luminance statistics such as logarithmic mean, minimum and maximum which may not stable over time. For example, the sudden appearance of a new object in the scene may lead to huge changes of maximum and minimum luminance. Although these changes of TMO parameters might be insignificant when tone mapping a static image, the

brightness of each tone mapped frame can change quite noticeably from one video frame to the next, leading to flicker in the final video stream.

These artifacts can either be global or local depending on the type of TMO used. Global flickering artifacts mostly occur with TMOs that rely on the maximum and minimum values of image which are unstable over time. Figure 9 illustrates such an artifact occurring in two successive frames of a tone mapped video sequence. The overall brightness has changed because the relative area of the sky in the second frame is larger which influences the image luminance statistics. These statistics change the normalization factor of TMO leading to different tone mapping result.

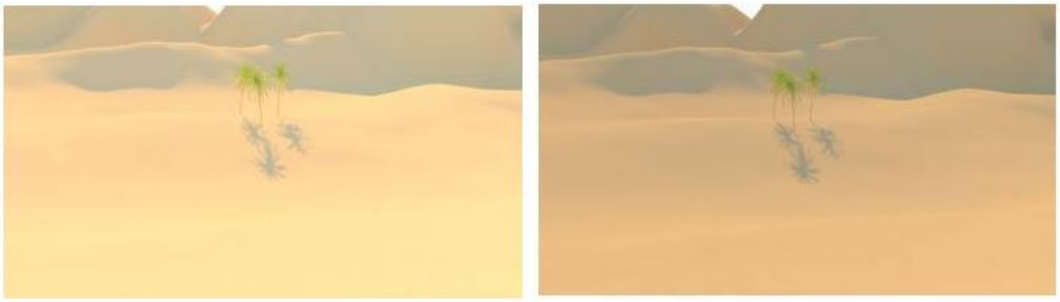


Figure 9. Global flickering artifacts due to small change of sky area within two successive frames. The images are taken from [36].

In local TMOs the tone mapped result of each pixel is influenced by its neighbor pixels. Therefore, small changes of this neighborhood in consecutive frames may lead to a different mapping result. Some local TMOs such as virtual exposures TMO [37] and domain transform TMO [38] decompose HDR images into multiple layers which is a base layer and multiple detail layers for detail enhancement and noise visibility control. Because each layer is tone mapped independently, any changes in a detail layer can lead to local flickering artifacts. Figure 10 shows an example of local flickering artifacts when applying local TMO to three consecutive frames. Comparing the left and the right frames, the edges of the object in the middle frame is briefly much clearer due to the enhancement of one of the detail layers.

Figure 10. Example of local flickering artifacts when applying local TMO to three consecutive frames. The images are taken from [39].

2.6.2 Tone Mapping Operators for HDR Video

Many recent HDR video TMOs focus on reduce these temporal artifacts. Kiser et al. proposes a global TMO [40] widely used in many video games aiming to reduce global flickering artifacts. This global TMO extends the photographic operator [6] with automated parameter estimation [30] for video applications. It uses a key value to calibrate tone mapped image luminance. The key indicates whether a scene is subjectively light, normal, or dark.

This TMO has two advantages. First, in order to better utilize the available LDR range and make the TMO less influenced by extreme values, the Kiser TMO clamps the input HDR frame based on the black and white levels of HDR light histogram before tone mapping. Second, to reduce temporal brightness incoherency, the Kiser TMO proposes a temporally stable tone mapping system that relies on a leaky integrator to smooth out the parameters estimated at every frame.

The parameters computed at the i^{th} frame are defined in the formula below:

$$s'_i = \begin{cases} s_i & if\ i = 0, \\ (1-\alpha) * s'_{i-1} + \alpha * s_i & otherwise, \end{cases} \qquad (2.6)$$

where α is a time constant which is in the range [0, 1], and the authors suggest setting α = 0.98.

However, Kiser's method uses Reinhard TMO [6] which is a photographic algorithm designed for the natural image. The synthetic image has unique characteristics comparing with the natural

image. The details of the comparison between natural and synthetic images are provided in Section 2.2.1. Our approach uses the content-adaptive TMO proposed by Khaldieh [21] considering the spiky character of synthetic image and preserving more contrasts for tone mapped image.

Eilertsen et al. [41] propose a noise-aware global TMO which is a fast display adaptive video TMO aiming to preserve the contrast of the original HDR content without increasing the noise present in the original content. This TMO decomposes the input HDR frame into details layers and base layer. First, a noise model is introduced to avoid the amplification of noise. The second step is applying a spatial edge-aware filter which is designed to extract base and detail layers avoiding error around soft edges. Then, a piece-wise linear tone mapping function is calculated based on the histogram of luminance of base layer. The details layer is recombined with the tone-mapped base layer as the final SDR result. To ensure smooth brightness changes between consecutive frames over time, the nodes in piece-wise-linear function are filtered over time. However, Eilertsen's method needs user inputs to control the local contrast and details which are hard to be used in the optimization tasks.

In this section, we introduce two video TMOs to reduce brightness flickering between consecutive frames. These TMO demonstrate some algorithm to smooth the abrupt brightness changes. But meanwhile, they limit the change of tone mapping curve without considering the scene change. Our approach bakes tone mapping curve into LUT and changes it dynamically based on the view point of the player. Any change in the scene is detected in real-time by using compute shaders to parallel counting the change of scene. We show that our results are closer to the colors of predefined LUT which should be presented exactly. In Unity, the luminance and color of current screenshot are regenerated in each frame by using LUT. The comparison between TMOs for video and the modification of LUT in each frame need to be considered in the future.

2.7 HDR Color Grading Workflow

One of the simple methods to encode color modifications is using parameters. These parameters control the toe, body and shoulder of tone mapping curve as well as the hue and saturation of color. However, parameter-based workflow has a high computation cost because some tone mapping functions are complex including many power and exponential functions which are time consuming to be processed on the CPU [11]. Another drawback is that some parameters in the workflow are too complex to be understood by an inexperienced developer and it is hard to be reused cross platforms.

Considering these problems, recent video games introduce a color grading workflow which applies various color grading effects easily using LUTs [7][11]. The artist can interactively choose a final effect from predefined LUTs and tweak them as needed.

A LUT changes color from one value to another based on a list of values. It is commonly used in video games to accelerate color transformations. Tone mapping curve and color modifications can be encoded into a LUT to be easily reused and applied cross the platforms. To utilize LUTs in video games, Waylon et al. proposed the color grading workflow of manual tweaking LUTs in Uncharted 4 [7].

As Figure 11 shows, the LUT is placed in the left top color of the screen. In this workflow, the colors in a player's sight are modified by the LUT in the direction that the player faces and it is tweaked by the artist. After every edit, the artist saves the LUT to disk and sends a refresh message to the game. This refresh happens at about 5 fps to make sure that the LUT is real-time updated in the game. In the same way, the colors in each view direction are carefully tweaked by the artists to present a desired overall atmosphere.

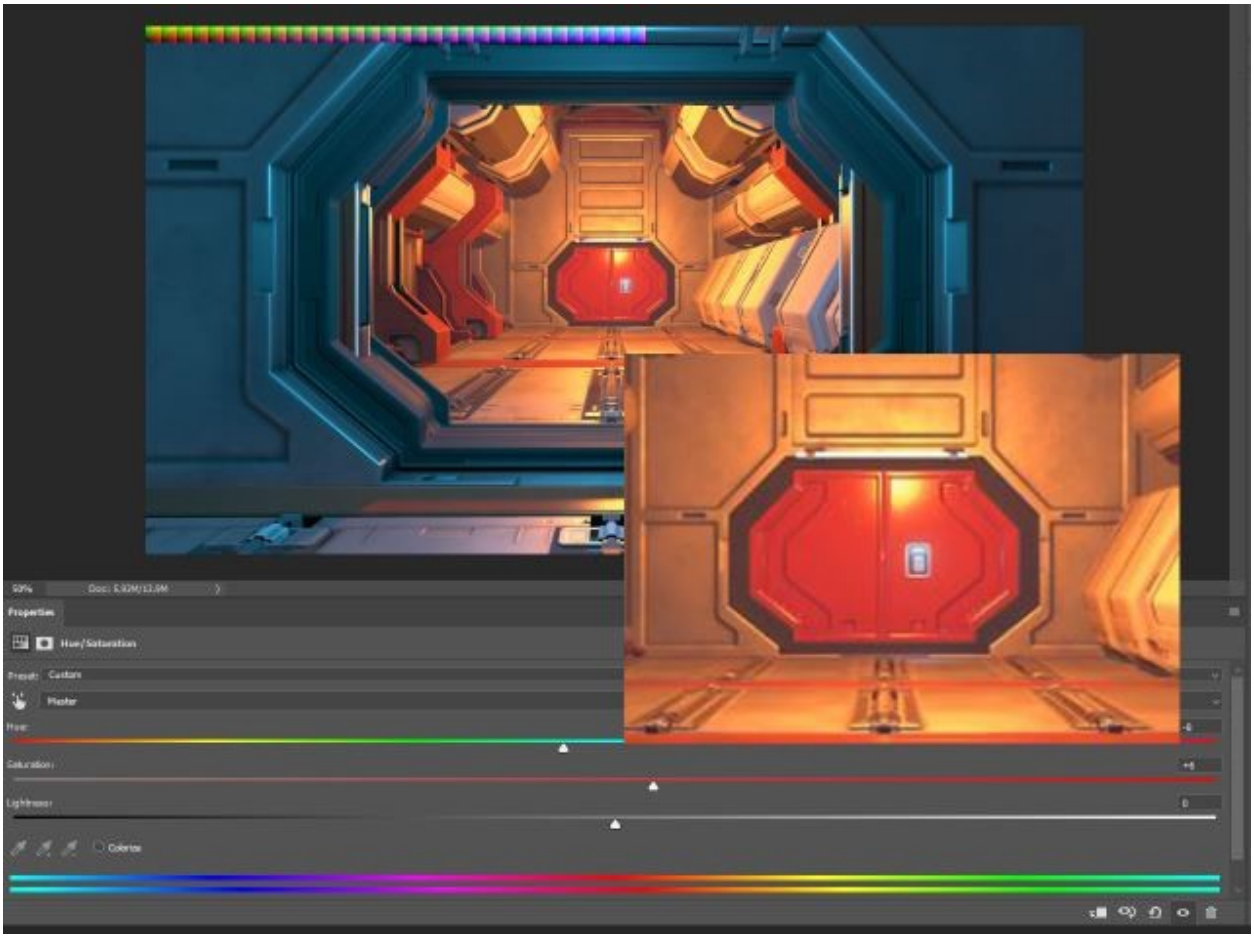


Figure 11. The workflow of tweaking LUTs.

2.7.1 Three-Dimensional Lookup Tables

A LUT is a three-dimensional lattice of output RGB color values that can be indexed by sets of input RGB color values. It can be used for any mapping from input to output colors under the resolution limit. The resolution limit of LUTs start from 8 bits (values 0-255), 10 bits (values 0-1023), 12 bits (values 0-4095) to 32-bit floating point (values from 0.0-1.0).

For example, an RGB color of (0, 0, 125) can be directly transformed to (28, 0, 32). It is not necessary to use the resolution of LUTs more than 32 bits which will reduce the speed advantage of using LUT. Therefore, 3D LUTs usually have a set of 32 coordinates on each axis (red, green, and blue) from which other values are interpolated to various levels of accuracy.

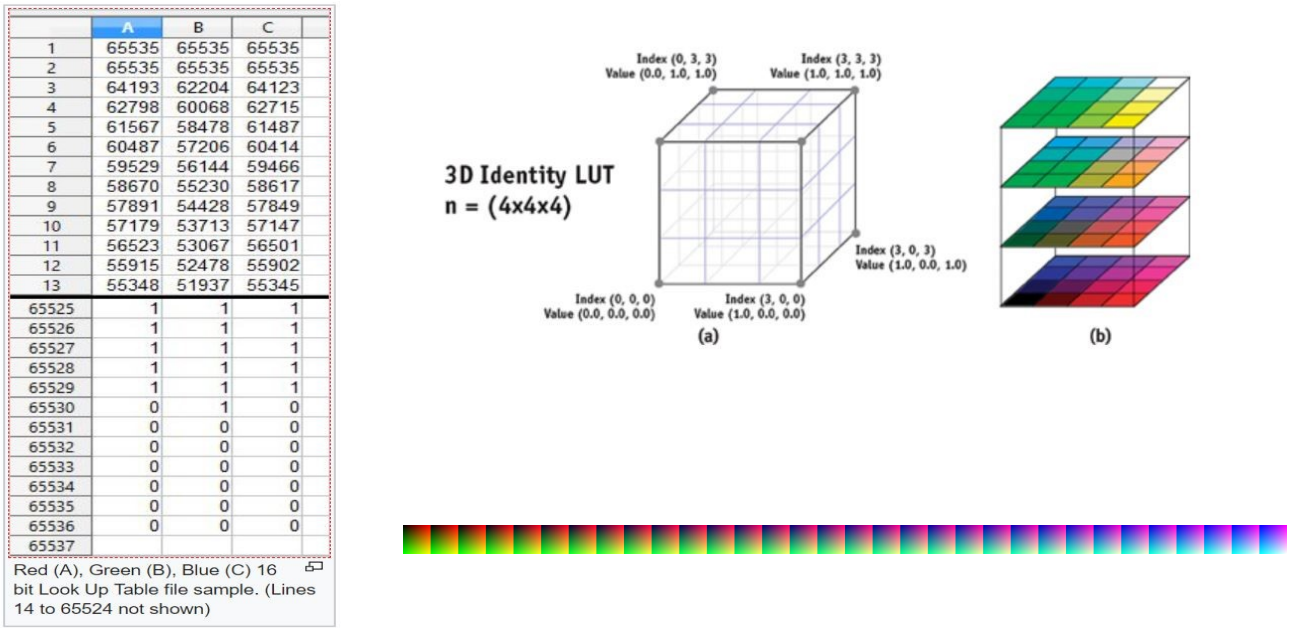


Figure 12. Ways of storing 3D LUTs. The left figure is CUBE format. The right figure is 2D texture format. The images are taken from [8].

As Figure 12 shows, there are two ways to store this 3D lattice (LUTs) in computer. Some formats (such as .3DL and .cube) store the RGB color values of each lattice point as the left figure shows. But in some game engines such as Unreal, they usually utilize a colorful grid image like the right figure shows. That grid image represents the cross sections of 3D lattice. In the game engine such as Unity and Unreal, both of the above approaches are available because these visualization methods represent the same mapping function.

One of the advantages of using LUT is to reproduce a complex color grading process easily by baking a grading process into 3D lookup tables which can be authored by professional grading software such as DaVinci Resolve [9] and Adobe Photoshop [10].

Another advantage is that potential speedups are enormous. The color grading process using a 3D LUT is approximately 100 times faster than standard color correction without LUT reduces it from 2.5 million operations to 32 thousand operations per frame [8].

The last advantage of using LUT is that modern GPUs provide hardware accelerated interface called tex3D to interpolate LUT in real time. At runtime, the programmer can pass arguments including input image color and LUT texture to the interface. The interface will return a 3D texture which represents the mapping function from input to output colors.

2.7.2 Post-Processing Volume

A video game may exist as multiple scenes which contain multiple LUTs. To blend multiple LUTs, a trigger zone called a post processing volume is introduced in many game engines such as Unity and Unreal. Usually, a post processing volume is a box collider. For each LUT, artists can assign a corresponding post processing volume. There are two parameters to control the blending of volumes which are blend radius and blend weight. Blend radius represents the range of post effect which is the boundary of wireframe box in Figure 13. Blend weight represent the intensity of the effect. However, in default settings, blend weight has the linear relationship with blend radius. For example, if the radius is 1000 and weight is 1, the player will get weight of 0.5 at a distance 500 to the post processing volume. Figure 13 shows a demo scene with three post processing volumes. The player will see three different atmospheres when he moves from left to right. During the movement process, instead of calculating color for each pixel, Unity produces current post processing effects by generating new lookup table in each frame, because using a look up table can massively reduce computing costs which has been introduced in detail in section 2.7.1.

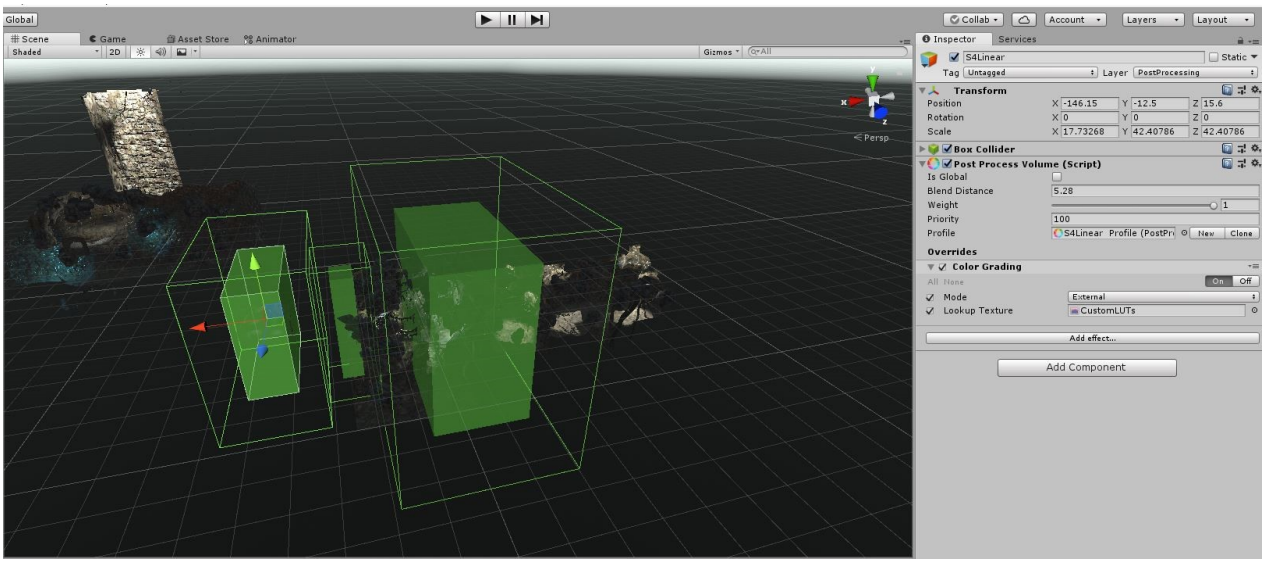


Figure 13. Post processing volume in the Unity game engine.

2.8 Evolutionary Optimization of Objective Tone Mapped Image Quality Metric

In this section, we discuss how to use an evolution strategy to optimize tone mapped image quality. Recently, TMOs used in video games such as GT TMO [20] and Hable TMO [5] are focused on improving the intuition of parameters to control the curve shape by humans. But Gao et al. decided to use the generic TMO proposed by Mantiuk and Seidel [42]. Generic TMO is a combination of sigmoidal function which contains seven parameters to flexibly control the brightness, contrast and lower and higher mid-tone of tone mapped results. The parameters of Generic TMO are shown in Table 1. Parameters b, dl, dh, and c are used to tune the curve shape and parameters m1, m2 and m3 are used to control the blurring and sharpening of image. Generic TMO aims to simulate different tone mapping operators by using fitted parameters which is suitable to be processed in the optimization task by a computer. For tone mapping n parameter optimization, we use the EA of (1 + λ)-ES proposed by Chisholm et al. [43]. The reason for using this algorithm is motivated by the lack of availability of analytical gradients and the potential for ruggedness resulting from the choice of quality criterion. In each iteration, we use the generic TMO compressed tone mapped image and measure the quality score based on the feature-based quality metric. Then, we select the best candidate solution with the highest quality

score from the parent. In each step, new candidates y_i are generated from the parental candidate x using the formula below:

$$y_i = x + \sigma z_i \qquad i = 1, \dots, \lambda \tag{2.8}$$

The offspring number λ is set to 10. The step size σ is predefined into 0.5 at the beginning. In each iteration, the step size is decreased by multiplication with 0.8 to guarantee the convergence of the function. z_i is the standard normally distributed mutation vector. The out-of-range new candidates y_i are clamped to the boundaries.

We stop the algorithm when the change in the best quality score has been less than 10^{-6} for six consecutive iterations. Then, we return the best candidate solution as the result. This candidate solution contains the optimized parameters of generic TMO which can be used for tone mapping.

Parameter	**Range**	**Description**
Parameters of tone mapping curve		
b	[-2.0,2.0]	Brightness factor
d_l	[0.0,2.5]	Lower midtone range factor
d_h	[0.0,2.5]	Higher midtone range factor
c	[0.2,1.5]	Contrast factor
Parameters of modulation transfer function		
m_1	[-2.0,2.0]	High frequency factor
m_2	[-2.0,2.0]	Medium frequency factor
m_3	[-2.0,2.0]	Low frequency factor

Table 1: Parameter of the generic TMO [42].

Chapter 3

Tone Mapping Optimization for HDR Gaming Content

3.1 Overview

Many tone mapping algorithms for natural images and synthetic images have been proposed in recent years. However, manually tuning these parameters is hard for developers without professional training. In this section, we propose modifications of feature-based quality metric to make it suitable for evaluating synthetic images. This algorithm can optimize the parameters of tone mapping operators by minimizing the perceptual distortion using an evolution strategy. The perceptual distortion is measured by a feature-based objective image quality metric. This metric utilizes a contrast-enhanced virtual photograph technique by using a content-adaptive tone mapping operator specially designed for rendered HDR content characteristics. We show that our results with optimized parameters preserve more visible contrast than other optimization algorithms.

Figure 14 shows the flow diagram of our algorithm. First of all, we take virtual photos from the HDR image using the Content Adaptive TMO [21]. We use the virtual photos approach proposed by Gao et al. [44] which decomposes HDR images into LDR images of multiple exposures. The reason for using virtual photo technology is the significant dynamic range differences between HDR images and LDR images. If traditional image processing algorithms are applied in HDR images directly, some HDR contents will be lost. The contrasts, local details and visual attentions of tone mapped images are influenced during this process. To preserve these image contents, virtual photos technology decomposes an HDR image into multi-exposed virtual photographs and then incorporates the virtual photograph sequence for visual saliency analysis. For each virtual photo, a feature-based quality metric calculates image features such as brightness, visual saliency and details for each virtual photo then incorporates each distortion category together as the HDR image features.

We use Content Adaptive TMO for taking the virtual photos. Content Adaptive TMO was proposed by Khaldieh [21] which considers the unique spiky properties of rendered HDR

gaming content. Considering that the video game requires high real-time performance, game engine precompute the indirect light information and store into light probes. When the game start, game engine can interpolate the light information of different light probes based on the position of player. Because more detail indirect light is captured near light probe, the luminance values of synthetic image are concentrated over small range leading to spiky. Content Adaptive TMO divides the image histogram into 1024 bins and allocates more tone mapping space for the high populated luminance. The reason for using this TMO is it preserves more global contrast and details than Reinhard TMO in tone mapped results for synthetic computer-generated images.

The initial HDR image is tone mapped using default parameters of the generic TMO [42] and an analysis of image features in the metric. The Generic TMO [42] is proposed by Mantiuk et al. which does not introduce a fixed mathematical formula to simulate tone mapping curves but contains a flexible four-segment sigmoidal function. This function can adjust the shape of tone mapping curve to satisfactorily approximate many existing global and local TMOs. The advantage of using generic TMO is it is computationally inexpensive and often provides visually indistinguishable tone mapped result when compared with the more expensive algorithms.

Second, a feature-based quality metric calculates the distortion between HDR image features and LDR image features and pool these distortions into a single score which can be used in the optimization task.

Third, in the evolutionary algorithm, we randomly generate new tone mapping parameter sets as children. In each iteration, we generate ten different parameter sets. Fourth, we tone map the HDR image using each new generated parameter set of the generic TMO. The results are evaluated by the feature-based quality metric again and we repeat the second step to get a score for the LDR image.

Fifth, we select the best child among ten children of the iteration as the parent. If the score of the parent is not changed more than 10^{-6} for six consecutive iterations, we stop the loop and consider this parent as the best parameter set. If not, we continue to randomly generate new parameters to find the best set. In the end, the optimized tone mapped image is calculated based on the best parameters of the generic TMO and the HDR image.

Comparing with feature-based quality metric proposed by Gao et al. [18], I replace the photographic TMO to content adaptive TMO designed for video game application as the red rectangle shows in Figure 14.

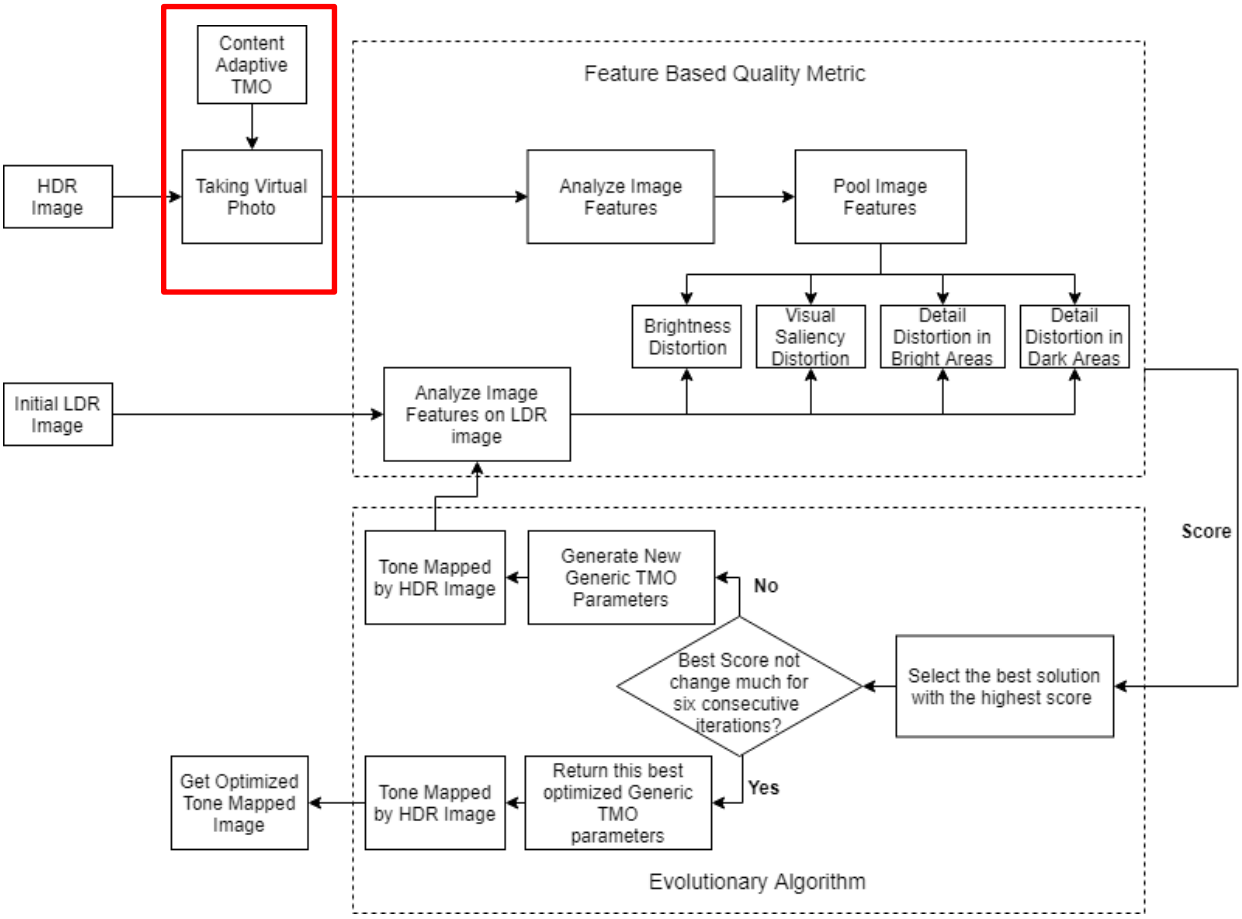


Figure 14. Flow chart of proposed automatic tone mapping parameter optimization algorithm.

3.2 Results

We present a detailed comparison of the evolutionary optimized tone mapped results using our proposed quality metric with the evolutionary optimized tone mapped results using the feature-based quality metric proposed by Gao et al. et al. [18]. We use the full dataset which contains ten video game capture sequences and each sequence contains 9 sequential HDR game captures provided by the "DML-Video-Gaming-Content-HDR dataset" [45]. These HDR synthetic

images are captured from free demo game scenes and rendered using the Unreal Engine 4 game engine. The images in the dataset have been divided into four categories which are bright low contrast scenes, bright high contrast scenes, dark low contrast scenes and dark high contrast scenes. Our results also compared with the result of tone mapped images proposed in the DML dataset [45] using default parameters of the Content Adaptive TMO. We evaluated these results using DRIM proposed by Aydin et al. [13] and a no-reference quality metric called BRISQUE proposed by Mittal et al. [35]. BRISQUE extracts the pointwise statistics of normalized luminance value and evaluates image naturalness based on the measured deviations from a custom model. The custom model is trained from a set of synthetic images and corresponding subjective scores from a synthetic image database [33] proposed by Kundu and Evans [46]. Then we use the custom model to calculate BRISQUE scores for our results. Kundu and Evans [34] evaluated the performance of 17 no-reference image quality assessments and find that BRISQUE has good performance for evaluating synthetic image. A smaller BRISQUE score indicates better perceptual quality.

Figure 15 shows the DRIM visible contrast distortion maps with different tone mapping optimized method. The loss of visible contrast is marked in green; the amplification of invisible contrast is marked in blue, and the reversal of visible contrast is marked in red. For the pixel of distortion maps with multiple colors, we mark the color with the largest proportion. We also present the total contrast distortion value which indicate the percentages of marked pixels. The reason for measure amplification of invisible contrast (shown in blue pixels) is that when enhancing contrast and preserving detail in the tone mapped image, tone mapping algorithm may generate halo artifacts. Halo artifacts tend to appear near high-contrast edges if local details are significantly amplified. The reason for measure loss of visible contrast (shown in green pixels) is that when some details become invisible during tone mapping. The reason for measure reversal of visible contrast (shown in red pixels) is that when test images and reference images have different polarity of contrast visibilities. So that less loss of visible contrast, less reversal of visible contrast and less amplification of invisible contrast mean higher quality of tone mapped image. As shown in Figure 15, the DRIM results indicate that our method has fewer total contrast distortion values than other optimization results using Gao et al.'s method as well as Content Adaptive TMO in dark high contrast scene (Reflections and Effects Cave 2), dark low

contrast scenes (Effects Cave, Realistic Reflections, Temple and Realistic Reflections 2) and bright high contrast scene (Sun Temple and Research Lab). For bright low contrast scenes (Vehicle Game and Living Room) which are easy to be tone mapped, Content Adaptive TMO presents less contrast distortion than our results.

We also use a no-reference quality metric called BRISQUE to evaluate our results. The BRISQUE score is in the range from 0 to 100. Lower values of score reflect better perceptual quality of tone mapped image with respect to the input model. Our results have smaller BRISQUE score than other results which indicate our results have better perceptual quality predicted by our synthetic image model.

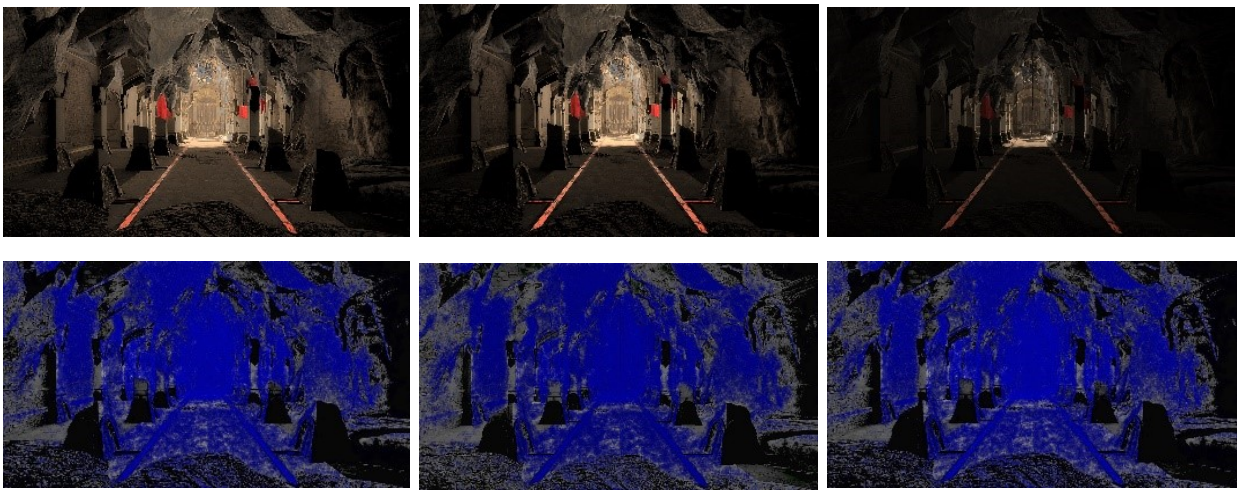
Sun Temple

Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 66.23	Distortion: 47.88	Distortion: 60.23
BRISQUE: 33.20	BRISQUE: 31.19	BRISQUE: 38.49

Vehicle Game

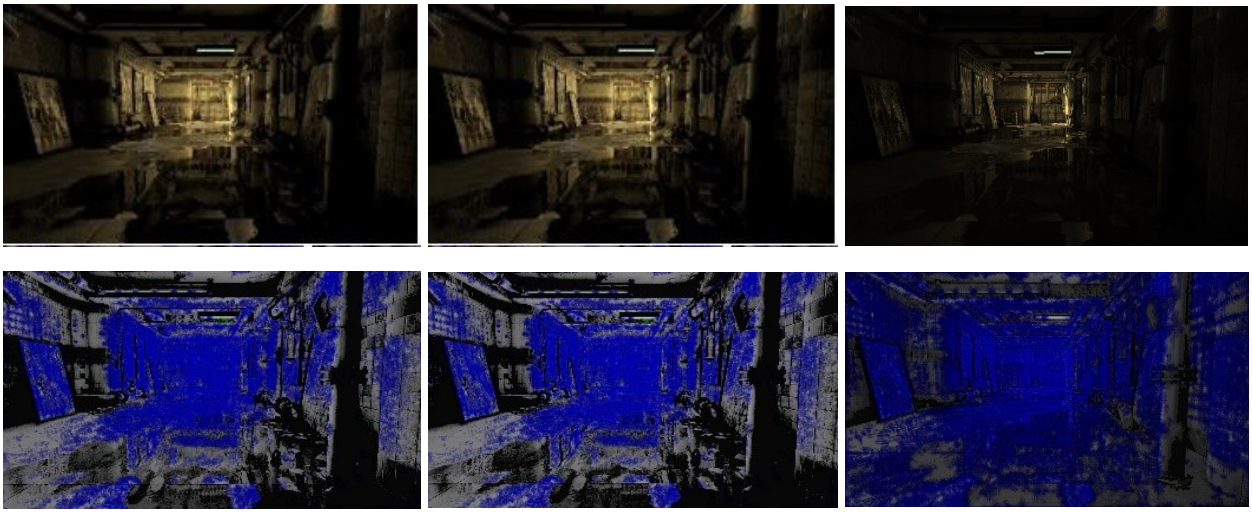
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 71.91	Distortion: 69.12	Distortion: 57.09
BRISQUE: 49.65	BRISQUE: 47.83	BRISQUE: 49.25

Effect Cave



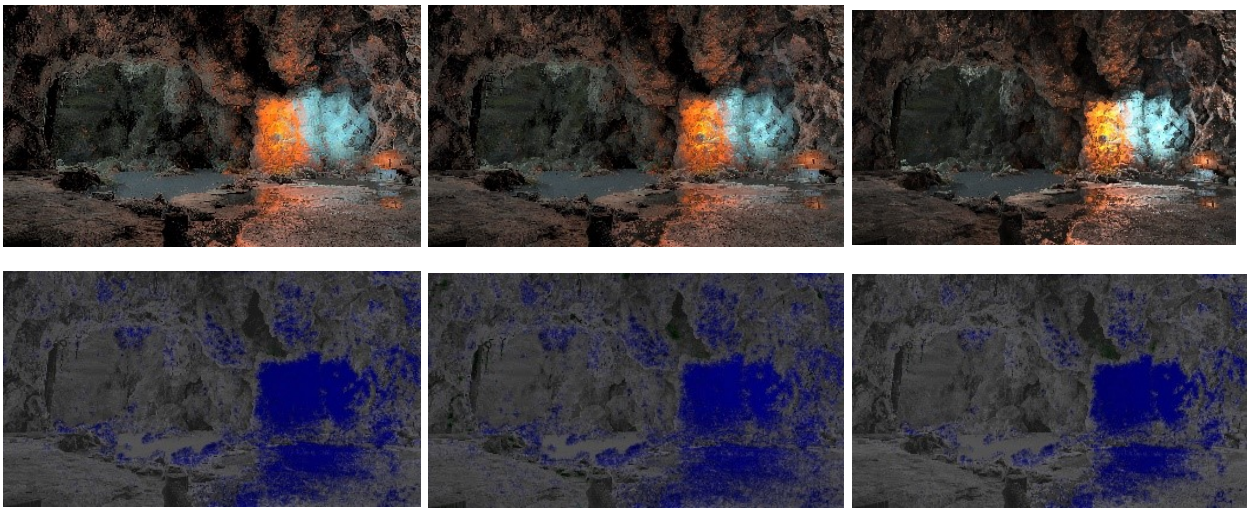
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 71.69	Distortion: 64.88	Distortion: 74.13
BRISQUE: 41.93	BRISQUE: 40.49	BRISQUE: 48.31

Realistic Reflections



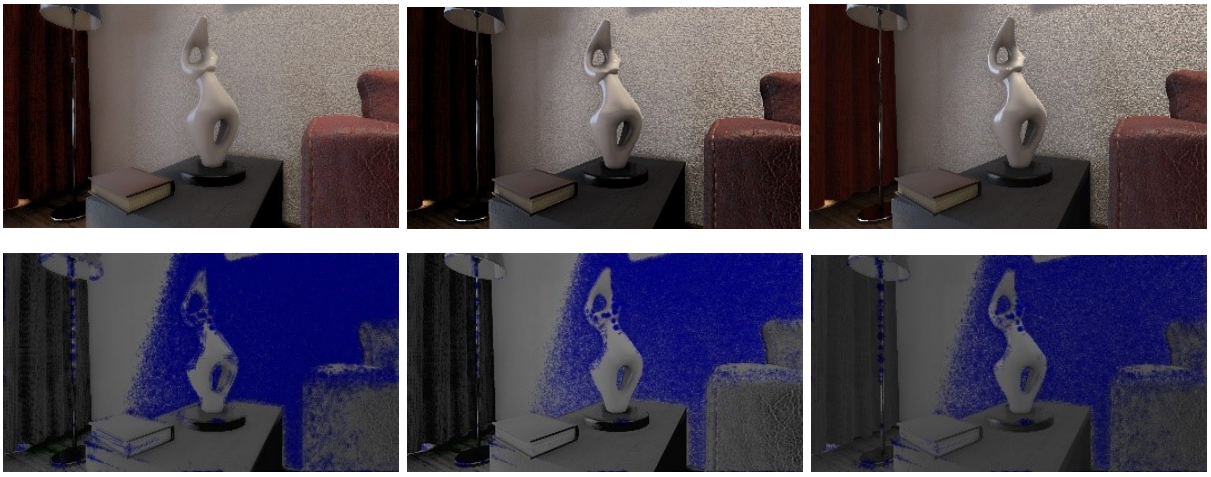
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 62.73	Distortion: 60.01	Distortion: 69.15
BRISQUE: 29.41	BRISQUE: 29.73	BRISQUE: 32.3

Reflections



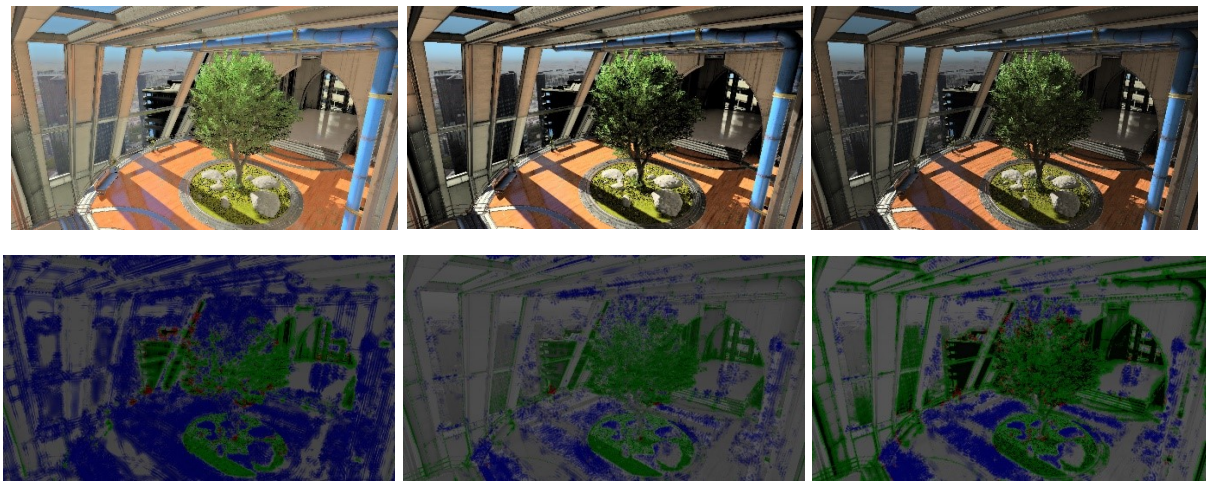
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 55.25	Distortion: 51.25	Distortion: 58.02
BRISQUE: 55.71	BRISQUE: 54.10	BRISQUE: 54.62

Living Room



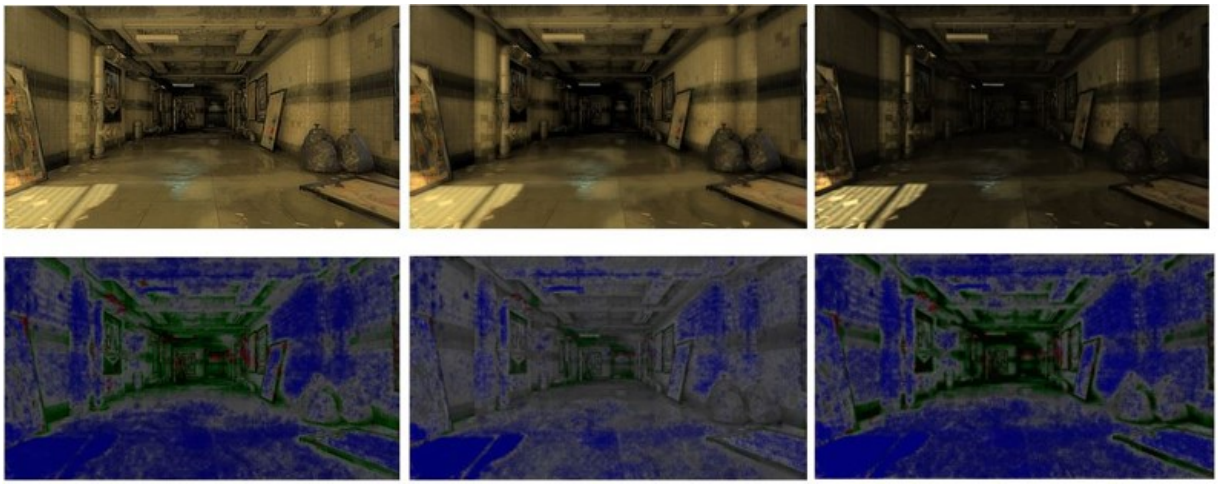
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 60.44	Distortion: 51.36	Distortion: 52.11
BRISQUE: 53.66	BRISQUE: 50.32	BRISQUE: 52.70

Research Lab



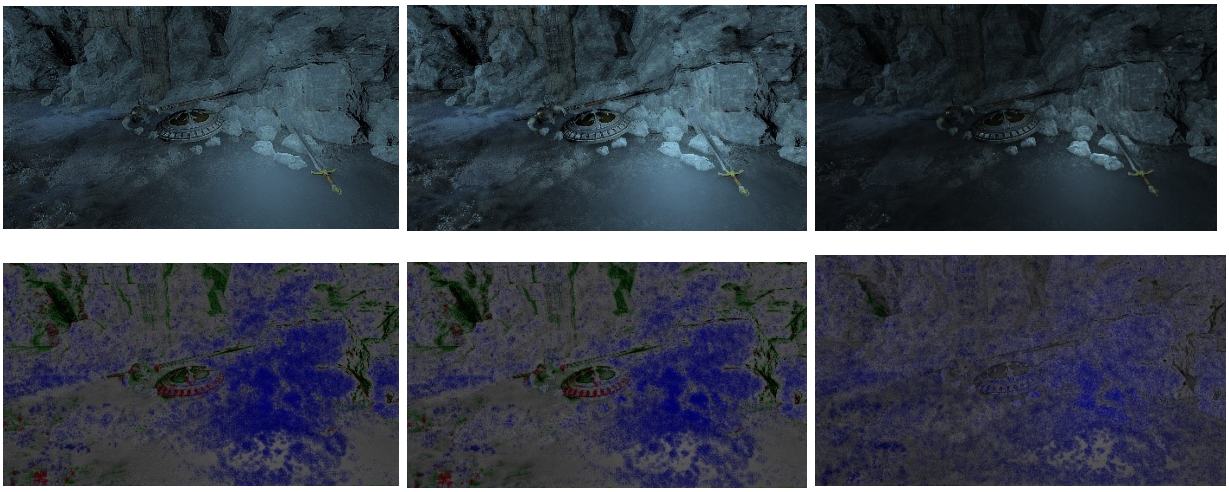
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 47.9	Distortion: 34.4	Distortion: 36.8
BRISQUE: 26.2	BRISQUE: 21.3	BRISQUE: 25.8

Realistic Reflections 2



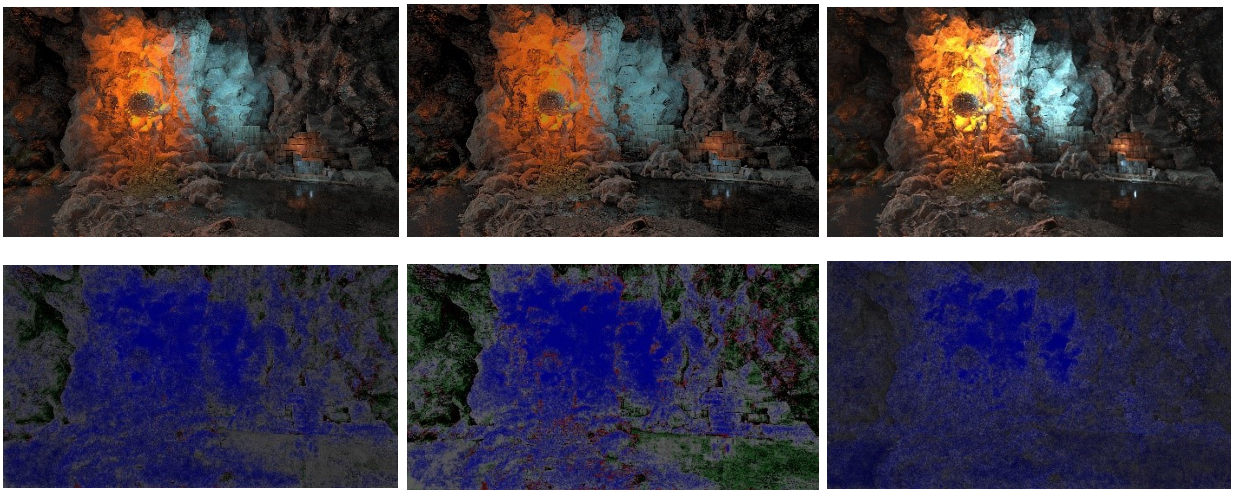
Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 61.33	Distortion: 59.7	Distortion: 68.2
BRISQUE: 53.1	BRISQUE: 51.8	BRISQUE: 59.7

Temple



Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 65.2	Distortion: 61.8	Distortion:63.9
BRISQUE: 47.6	BRISQUE: 45.7	BRISQUE: 52.9

Effects Cave 2



Gao et al. Method	Our Proposed Method	Content Adaptive TMO
Distortion: 42.1	Distortion: 40.3	Distortion: 41.9
BRISQUE: 53.6	BRISQUE: 49.3	BRISQUE: 51.7

Figure 15: Comparison of tone mapped LDR images (top rows) and distortion maps (bottom rows) using Gao et al. method [18], our method and content adaptive TMO [21]. The DRIM visible contrasts are also presented under each image.

	Reflections				Sun Temple			
	loss	amplification	reverse	total	loss	amplification	reverse	total
Content Adaptive TMO	39.3	22.3	2.3	63.9	36.1	12.6	1.8	50.5
Gao et al.	37.2	30.4	**1.9**	69.5	35.7	16.2	**1.5**	53.4
Our method	**28.1**	**18.0**	2.1	**48.2**	**30.7**	**10.4**	2.1	**43.2**
	Effects Cave				Vehicle Game			

Content Adaptive TMO	19.4	52.6	1.2	73.1	9.9	46.5	2.8	59.2
Gao et al.	21.3	**47.3**	0.6	69.2	11.7	60.9	**2.5**	75.1
Our method	**10.9**	49.3	**0.5**	**60.7**	**7.0**	**45.6**	4.4	**57.0**
	Realistic Reflections				Living Room			
Content Adaptive TMO	38.6	23.2	1.4	63.2	7.1	35.5	0.6	43.2
Gao et al.	38.5	27.1	1.1	66.7	9.6	42.7	1.1	53.4
Our method	**26.3**	**14.1**	**0.3**	**40.7**	**6.8**	**35.2**	**0.5**	**42.5**
	Research Lab				Effects Cave 2			
Content Adaptive TMO	13.9	**19.3**	1.9	35.1	25.5	**13.1**	1.8	40.1
Gao et al.	13.7	19.9	2.1	35.7	17.2	24.2	**1.5**	42.9
Our method	**11.4**	20.1	**1.7**	**33.2**	**11.3**	22.8	2.1	**36.2**
	Temple				Realistic Reflections 2			
Content Adaptive TMO	39.2	28.5	3.8	71.5	**8.1**	30.5	2.8	41.4
Gao et al.	33.3	28.5	2.1	63.9	11.7	28.1	**2.5**	42.3

Our method	**31.9**	**27.1**	**1.9**	**60.9**	8.3	**26.6**	4.4	**39.3**

Table 2: Analysis of the numerical distortion errors computed with Aydin et al. [13] using the Content Adaptive TMO method [21], Gao et al. method [18] and our method, averaged over six sequences of nine tone mapped LDR images each.

For an analysis of the numerical distortion errors, we evaluate the evolutionary optimized tone mapped results using our proposed method with feature-based quality metric [18] on ten video game sequences in the database and each sequence contains 9 sequential HDR game captures. One of sequences is shown in Figure 16. We use DRIM to calculate the distortion value for each pixel in the tone mapped image. For each sequence, we calculate the average DRIM distortion of tone mapped images to obtain a single score in each game scene. Every DRIM distortion has three categories which are loss of visible contrast, the amplification of invisible contrast and the reversal of visible contrast. Table 2 shows that tone mapping optimized results using our proposed method result has the least loss of visible contrast distortion and amplification of invisible contrast for most HDR images while causing a few more errors in reversal contrast comparing than the Gao et al. method. But our method shows less total contrast distortion compared with the other tone mapping optimized methods.

Figure 16: One of the example sequence of 9 sequential HDR game captures in Living Room demo scene.

Chapter 4

Real-time Interpolation Between Lookup Tables

4.1 Overview

The existing color grading pipeline uses parameters to control the shape of tone mapping curve and the modifications of color such as hue and saturation [5]. However, the implementation of this pipeline has a high computation cost because tone mapping curves, such as GT tone mapping [20], have many power and exponential functions which are time consuming to compute on the GPU in real time.

Khan et al. [47] propose a tone mapping algorithm that uses LUTs to map HDR luminance values to LDR values to make efficient on mobile devices. The LUT contains two columns which are the pairs of HDR and the corresponding LDR values. In recent years, many video games utilize the LUTs-based HDR Color grading pipeline [7][11] to accelerate the process of tone mapping and color grading. Waylon et al. [7] propose a color grading workflow which encodes both color grading and tone mapping curves into a single HDR LUT texture as shown in Figure 17. When artists tweak the LUT in external tools, colors of game maintain the live update by refreshing LUTs after every edit. Artists are not locked in the initial screenshot. They can freely look around the scene to make sure the LUT looks good everywhere.

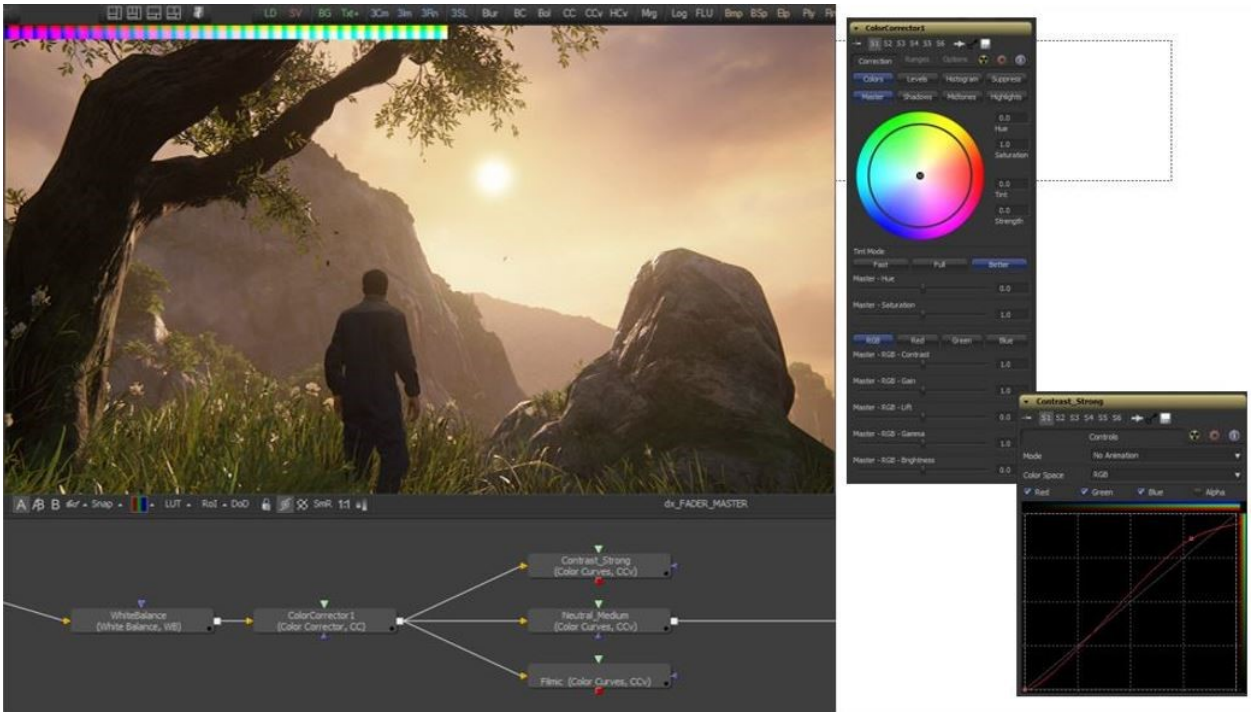


Figure 17. Tone mapping and color grading workflow in Uncharted 4. The image is taken from [7].

However, there may exist multiple scenes in a video game which contain multiple LUTs. To blend these LUTs, a trigger zone called a post processing volume is introduced. Each post processing volume has blend radius and blend weight to control the post processing effects. A common algorithm used by many game engines such as Unity and Unreal is linear interpolation representing the linear relationship between blend weight and blend radius. The details of this algorithm are shown in Figure 18. The game engine calculates the distance between the player and the boundaries of PPVs. The blend weight is proportional to the distance. For example, if the player stands in the middle position of two PPVs, the blend weight of each PPV will be 0.5. Based on the updated blend weights, the game engine merge multiple LUTs together into a new LUT in the background. This process will be recall in each frame.

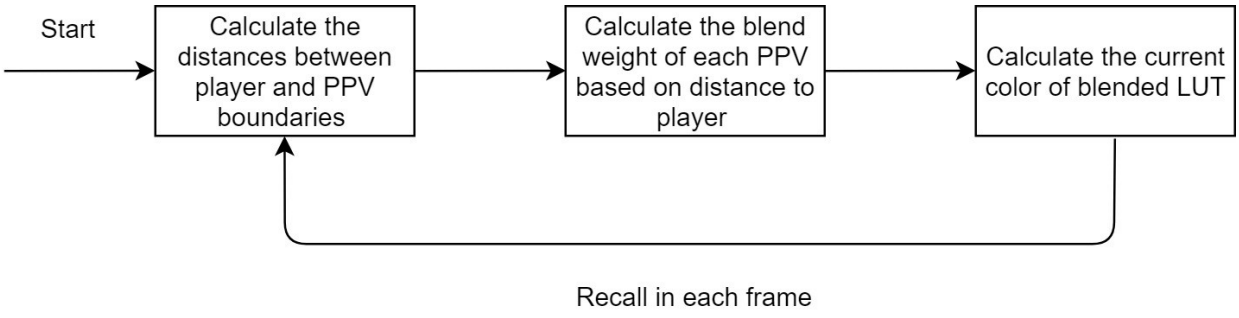


Figure 18. The diagram of linear interpolation pipeline.

However, the LUT color in a position-based linear interpolation algorithm is only influenced by the player position. In contrast, we propose a pixel-based linear interpolation algorithm which changes the LUT color dynamically based on the view point of player. During the game, the colors of objects are changed when the player move and look around. Comparing with the algorithm which change color based on player's position, our method preserves more original colors of objects.

By reconstructing screen pixel world position from the camera's depth, the color of each pixel inside post processing volume can be assigned based on the boundary of the volume. For the pixel inside the volume, we can modify it by using the LUT color of that volume, so that every object can show their original color when the player moves between LUTs. However, this method requires that the color of each pixel needs to be recalculated in each frame which is very time-consuming. To solve the previous problems, we propose a pixel-based real-time interpolation method for blending multiple LUTs. The process of this algorithm is demonstrated in Figure 19. We reconstruct the world position of each pixel from a depth texture. By comparing with the boundary of post processing volume, we record the pixel count belonging to each LUT on the current screen which is optimized by GPU parallel processing of compute shaders. Compute shaders are programs that execute on the graphics card for general tasks other than normal render tasks such as drawing triangles. Finally, we blend LUTs together based on the screen percentage of each LUT. Compared with linear interpolation, our results are closer to the colors of predefined LUT which should be presented exactly. We also show that our method satisfies the real-time requirement and can be executed fluently.

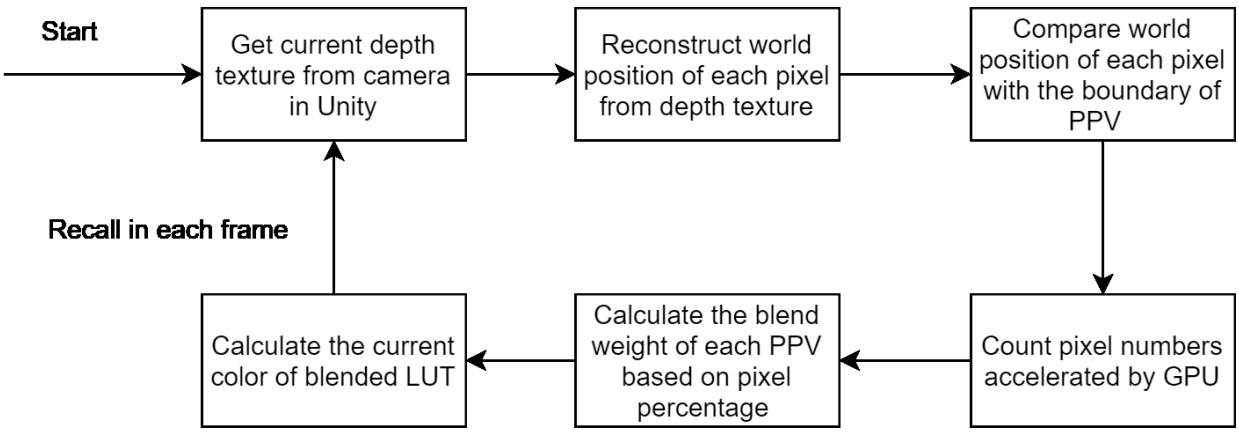


Figure 19. The diagram of pixel-based interpolation pipeline.

4.2 Reconstructing World Position from the Depth Buffer

In order to compare the world position of each pixel with the boundary of post-processing volumes, we reconstruct the 3D position of each pixel in world-space from the depth map. We use the reconstructing algorithm proposed by [48] which can be executed in parallel on the GPU with a compute shader. In the algorithm, the world position of pixel is calculated by two components which are world position of camera and the offset of each pixel related to the camera.

It is processed in the following steps shown in Figure 20. First, in each frame, each GPU thread operates in parallel on one of pixels of the depth map. The depth map can be accessed directly from the depth camera.

Then, we calculate the vectors from camera to the four corners of the near plane which are the top-right corner, top-left corner, bottom-left corner and bottom right corner. The near plane is the closest location that will be rendered by the camera. In this step, the corners of near plane are rotated to match the current orientation of the camera using the camera world matrix which is a matrix representing the camera's position and orientation in world space.

After that, the corners of near plane are interpolated to calculate rays from the camera to the near plane. The pixel position in the camera's coordinate space can be calculated by multiplying the depth value with the ray pointing from the camera to the near plane.

In the end, the relative position is added with the world space position of the camera to get the world space position of the pixel.

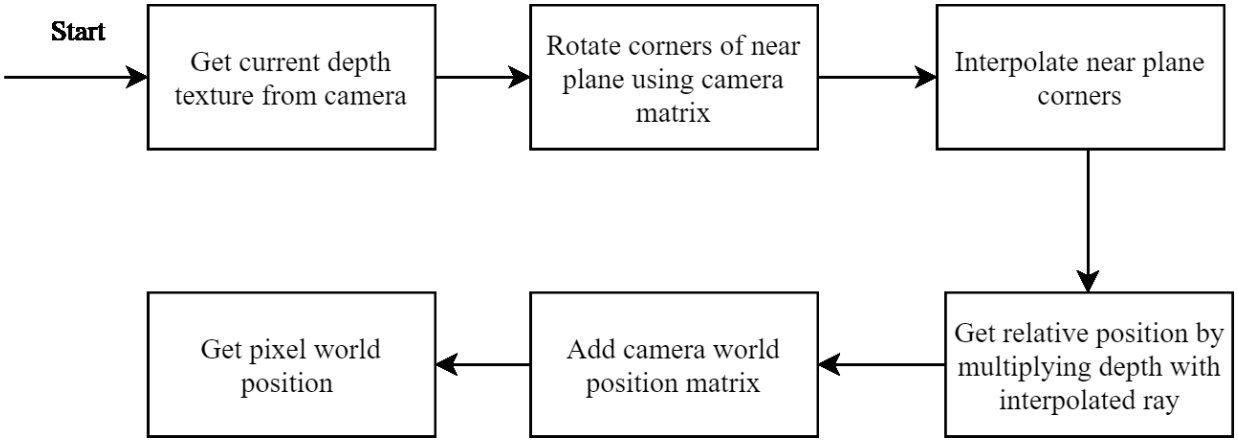


Figure 20. The diagram of reconstructing world positions of pixels.

4.3 GPU Optimization by Parallel Processing

It is time-consuming to count pixels inside LUTs one by one. In order to speed this process up, we use compute shaders which can run on the GPU outside of the normal rendering pipeline. The advantage of a compute shader is that it contains multiple threads for reading and writing data in parallel. In our application, the world position of each pixel is calculated in a compute shader program. We allocate a compute buffer for each post-processing volume. A compute buffer is a memory buffer which can be read and write in parallel by compute shader programs. For each pixel within a post-processing volume boundary we increase the count by one. But the same compute buffer might be accessed simultaneously by more than one thread. To avoid this problem, we utilize the atomic function for thread synchronization which makes other threads wait for the current thread before finishing. We count the number of pixels belonging to each post processing volume in GPU parallel computing. The count number divided by the sum of all pixels is the percentage used as the blend weight for the LUT associated with that volume. When the player looks around and moves around between LUTs the blend weight of each post processing volume changes dynamically in real-time to show different LUT colors.

Below we provide the pseudocode for the above LUTs interpolation algorithm as Algorithm 1.

Algorithm 1 LUTsInterpolation (*P*, *T*, *M*). Modifying blend weight of LUTs based on the pixel numbers in each post processing volume.

Input: *P* is a set of post processing volumes containing their positions and boundaries. *T* is the camera depth texture. *M* is camera matrix which contains the position and rotation information of camera.

Output: *S*, which is a set of blend weights for post processing volume to blend multiple LUTs for use in the current scene.

---

computeBuffer[] = initComputeBuffer(*P*): *Allocating compute buffer for each post processing volume to store the counting result of pixels. The type of compute buffer is RWStructuredBuffer which ensures the buffer is not accessed by multiple thread simultaneously.*

for each Pixel *pix* in *T*: **do**

worldPos = ReconstructingWorldPosition(pix, *M*): *Reconstructing world position of each pixel by multiplying the pixel depth value with the ray pointing from the camera to the far-clip plane.*

for each PostProcessingVolume *ppv* in *P*: **do**

isInside = boundaryDetection(pix, ppv): *Comparing the position of the current pixel with the boundary of the post processing volume. Return true if the pixel is inside current post processing volume.*

if *isInside* is **true**: **then**

computeBuffer[index] ++: *Add one into the compute buffer corresponding to current post processing volume.*

break;

---

end if

end for

end for

S = computeBuffer / ***Sum****(T)*: *For each post processing volume, calculate the percentage of pixels inside it.*

return *S*;

4.4 Creating LUTs from Examples

We create LUTs from examples to generate the LUTs of test scenes. A LUT can be tweaked by artists with professional experience. Figure 21 shows an example of realistic tweak LUTs from an industry example using professional color correction software called 3D LUT Creator [49]. We change the color of input image by bending the grid tied to the CIELAB color plane. This interface allows the artist to drag the desired color on the grid to get the desired hue and saturation without affecting other colors. With the help of this software, the artists can map colors from the input image to the reference image using the professional experience of color grading. One of the examples of manual color transformation is shown in Figure 22.

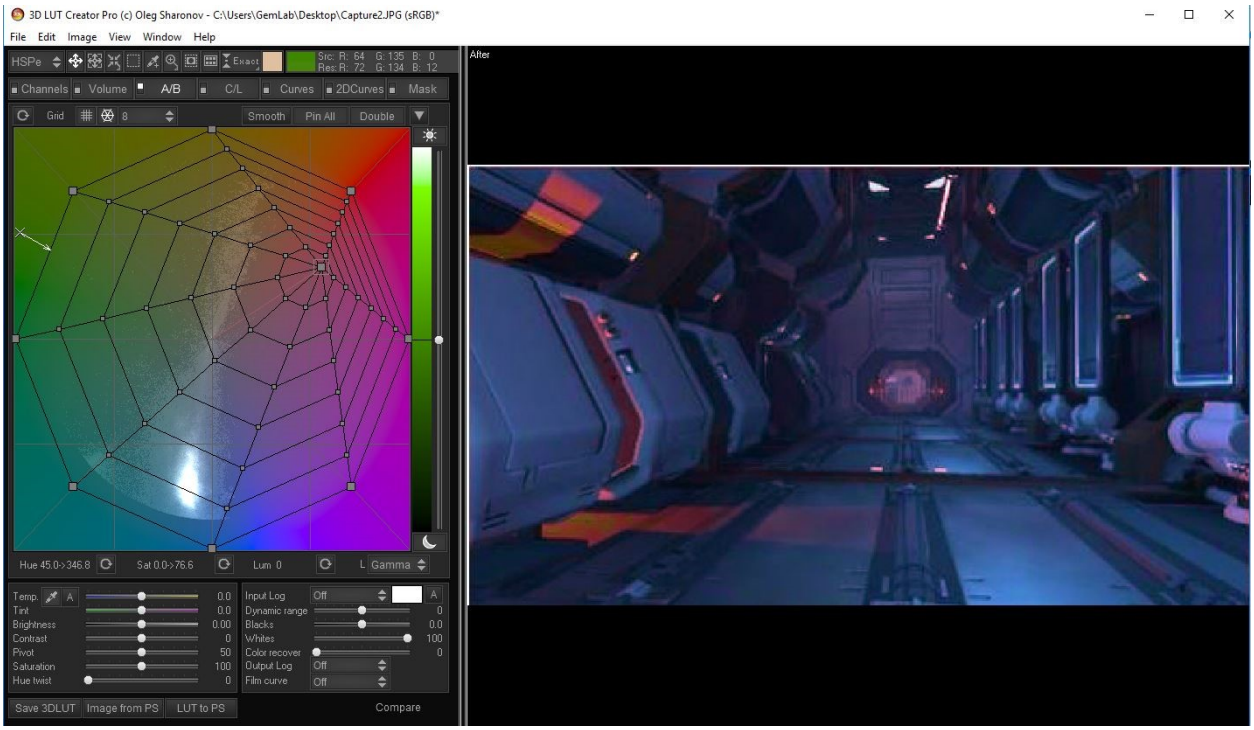


Figure 21. Manually tweak LUT in professional color correction software.

Input Image Reference Image Manual Tweaked Result

Figure 22. Manual color transfer results based on video game screenshots.

However, it also can be automatically generated from video game captures using a color transfer technique. Color transfer algorithms recolor a given image by deriving mapping function between that image and reference image. We decided to use an automatic color transfer method proposed by Pitie and Kokaram [50] which modifies the 3D color distribution of the input image by iterative matching random 1D projections of the color distribution with the reference image. Specifically, this algorithm treats the colors as a distribution in three-dimensional CIELAB color space and repeatedly projects this three-dimensional distribution into a series of random 1D

marginal distributions. The color distribution of the target image is converted to that of the source image by repeatedly mapping its 1D marginal distributions to those of the source image until convergence. The advantage of this method is that it is fast and flexible enough to capture a wide range of effects including hue and saturation variations represented by two-dimension transformations in the ab plane of CIRLAB color space. Figure 23 shows some results of this automatic color transfer method. The input images are captured from Sci-Fi Unity Scene [51]. The reference images are download from the Uncharted 4 website [52]. Based on the original input image and color transferred results, we generate the LUTs by Photoshop and apply these LUTs in Unity as shown in Figure 23.

Room1 Original Reference Room1 Color Transferred

Room2 Original Reference Room2 Color Transferred

Room3 Original Reference Room3 Color Transferred

Room4 Original Reference Room4 Color Transferred

Room5 Original Reference Room5 Color Transferred

Room6 Original Reference Room6 Color Transferred

Figure 23. Automatic color transfer results based on video game screenshots.

4.5 Results

We have evaluated our proposed method on a game project with some predefined post-processing volumes in Unity proposed by Sci-Fi [51]. The LUTs used in post-processing volumes are automatically generated based on the color transfer algorithm discussed in 4.4.

One of our demo scenes is shown in Figure 24. There are three post processing volumes in the scene. Each post processing volume has been assigned a LUT. If multiple LUTs are overlapping, we assign the most internal LUT to the volume. The largest volume contains background LUT which gives a default color mapping for the pixels which do not belong to any defined LUT area. The player represented by the camera icon stands between two neighboring post processing volumes. If the player is inside the volume, the object colors in every direction are predefined by

the LUT. However, it cannot avoid the player moving into another scene which is dominated by another LUT. The color mappings used when the player moves between LUTs are automatically calculated by the computer.

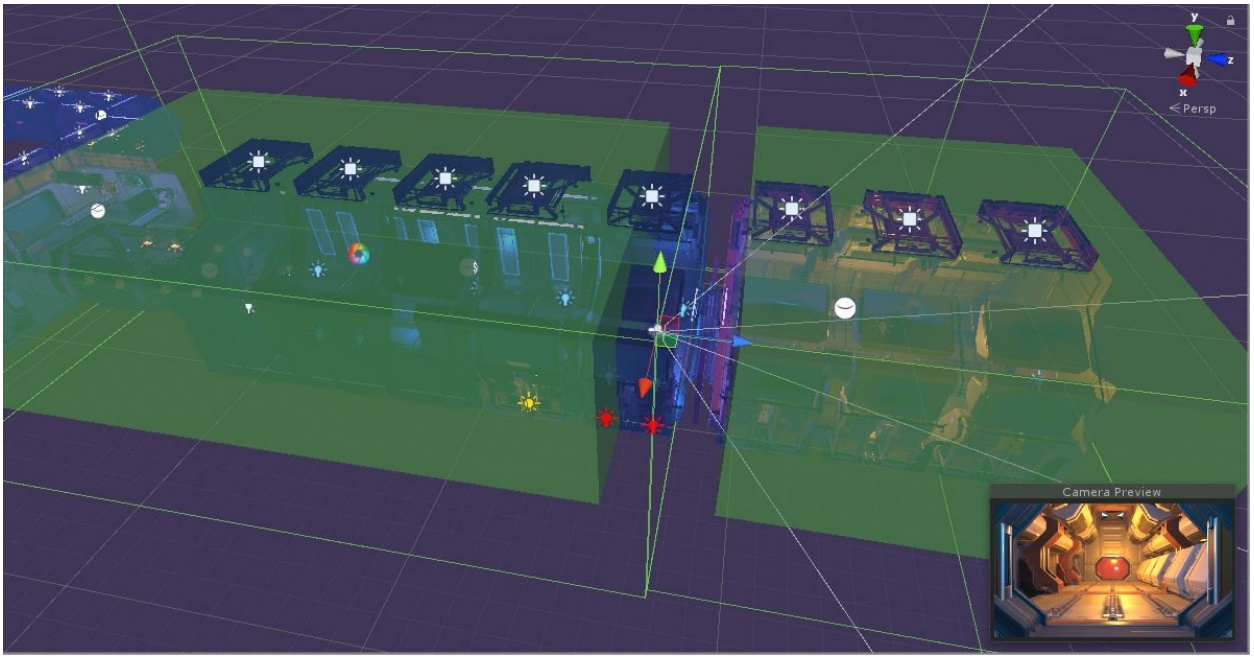
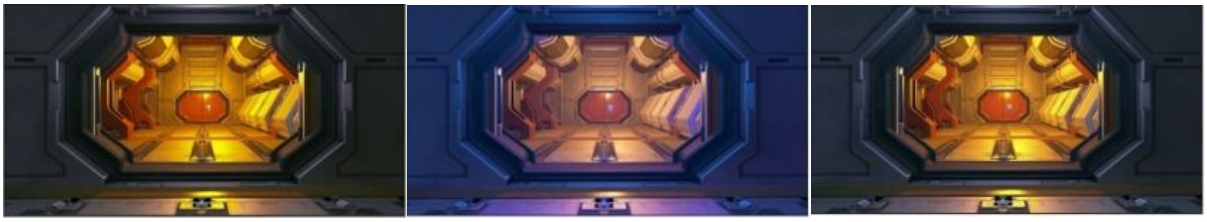


Figure 24. Demo scene of player looking round between LUTs in Unity.

We compare our proposed pixel-based interpolation algorithm with linear interpolation which is used in the game engine by default. Figure 25 illustrates the screenshots of two methods when the player looks around between two post-processing volumes. As shown in Figure 25, when more pixels of indoor objects come into sight, the pixel-based results have a closer visual appearance to the reference images. The reference image colors are modified by the LUT in front of the player. In contrast, the colors of the target area in the linear method do not change much in each frame capture. At the end of the rotation, when the player directly looks at the objects inside the door, pixel-based results preserve more LUT colors in front of the player.

Linear based result for each frame

Pixel based result for each frame

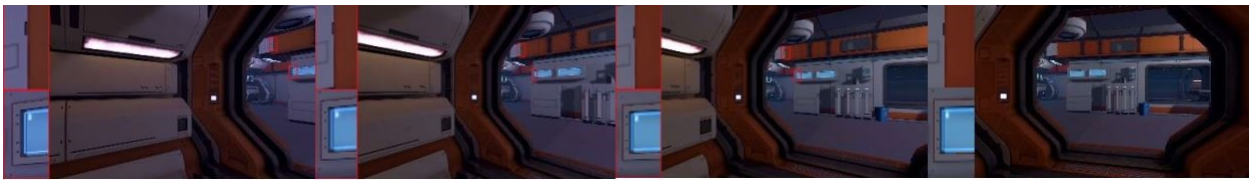


Final Reference Image Final Linear Result Final Pixel-based Result

Linear based result for each frame

Pixel based result for each frame

Final Reference Image Final Linear Result Final Pixel-based Result



Linear based result for each frame

Pixel based result for each frame

Final Reference Image Final Linear Result Final Pixel-based Result

Linear based result for each frame

Pixel based result for each frame

Final Reference Image Final Linear Result Final Pixel-based Result

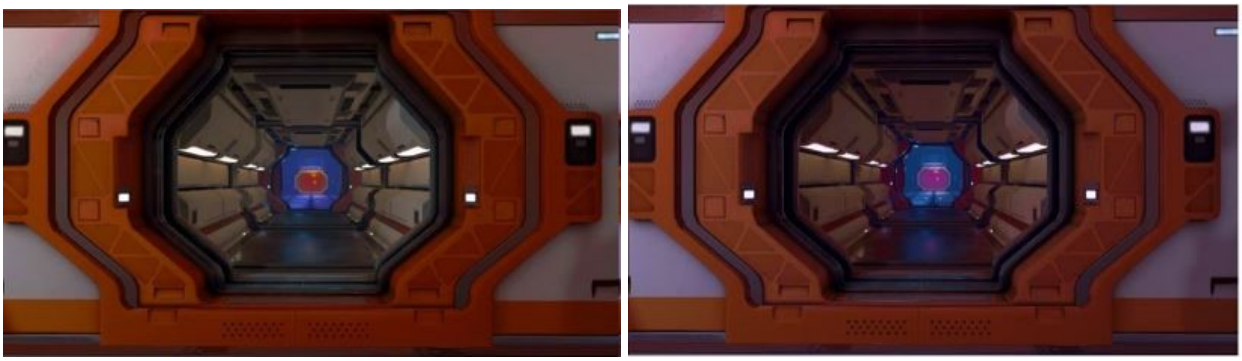
Figure 25. Result comparison between linear interpolation and pixel-based interpolation.

We also compare our method with a mixed image method. In this method, the colors of the image are stitched together from multiple LUTs based on the depth image. By reconstructing the world space position of each pixel, we compare the boundary of each object in the scene with the post-processing volume boundary. For the object inside the volume, we assign it colors based on the LUT of the volume. For the object which crosses two or three volumes, we find the volume which contains most of the object and use its LUT. In the end, we assign each object a specific LUT color. Figure 26 illustrates the comparisons between this mixed image method and the pixel-based method. The depth image shows the segmentation of the image based on the object boundary. As shown in Figure 26, mixed images present diverse colors in different areas with strong contrast. Conversely, the pixel-based results keep a more consistent atmosphere of post-processing effect.

Depth Image LUT Background LUTA LUTB LUTC

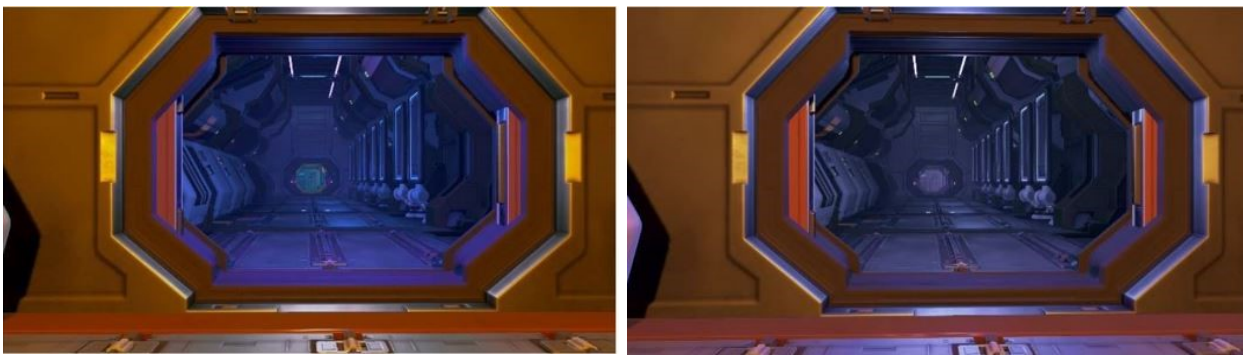
Stitched three images together Pixel based method result

Depth Image LUT Background LUTA LUTB LUTC



Stitched three images together Pixel based method result

Depth Image LUT Background LUTA LUTB LUTC



Stitched three images together Pixel based method result

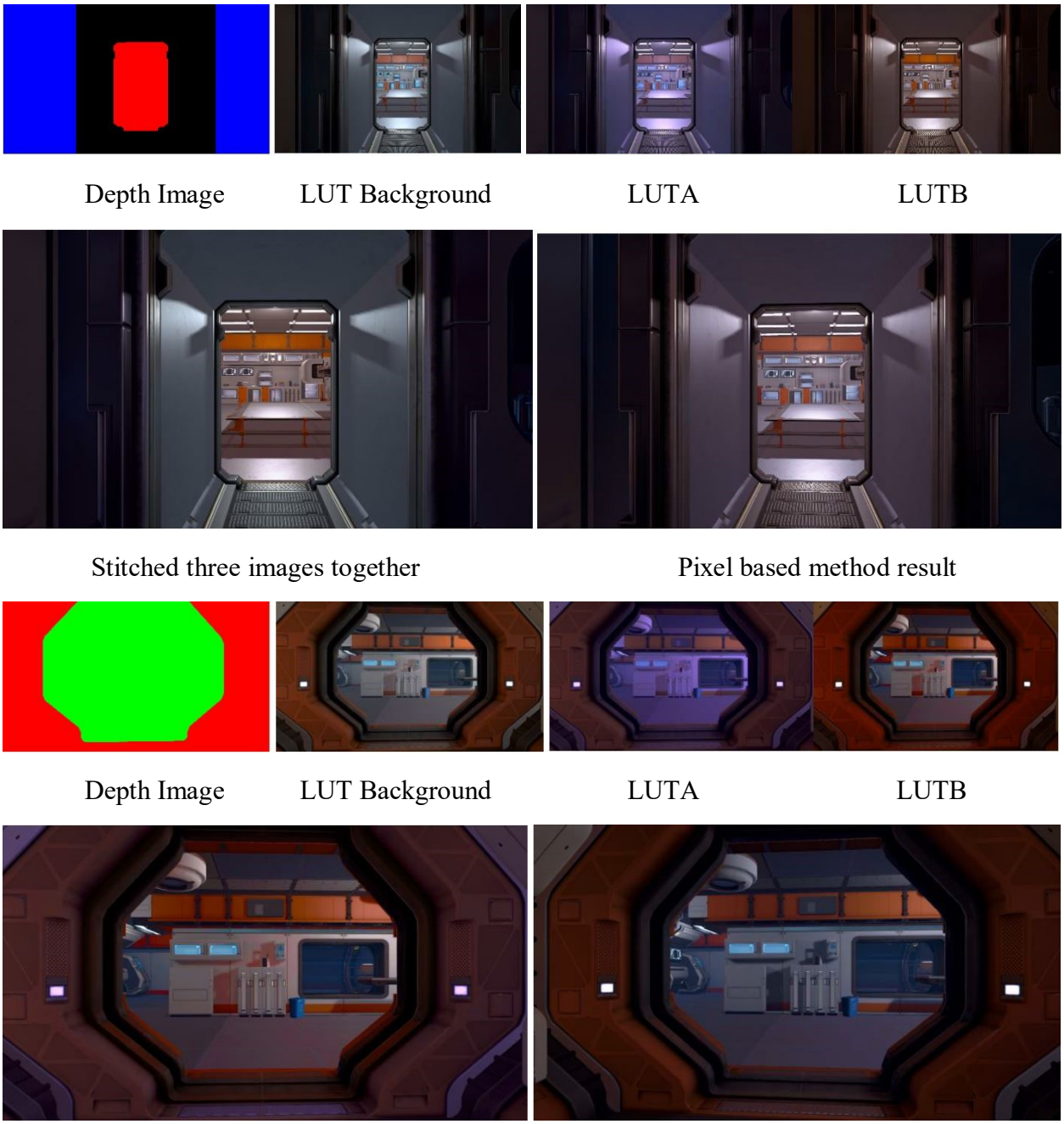


Figure 26. Result comparison between mixed image method and pixel-based interpolation method.

We implement our method in a Unity project of 1920 * 1080 resolution on the computer with an Intel Core i7-4770k 3.50 GHz CPU with 16 GB of RAM and GeForce GTX 1070 GPU with

8GB video memory. The mixed image method takes 0.2087s to 0.2264s to compute each frame and the frame rate is 5. It should be note that, one of the reasons for the slow computing speed is that it is hard to determine the irregular boundary shapes of the objects which cross multiple PPVs. For each pixel on the object, some information of neighborhood pixels are needed to be stored to determine the LUT color of current pixel.

In our approach, the boundary of PPV is regular box shape which can be obtained by GPU in real time. Therefore, the computation time of the pixel-based method enjoys a huge speedup which requires only 0.003s to 0.005s for each frame and the frame rate is between 80 and 95.

	Down Scale	Framerate (FPS)	Compute Time (s)
CPU Method	1	5	0.2187
CPU Method	1/2	20 - 25	0.035 – 0.045
CPU Method	1/4	42 - 60	0.008 – 0.012
CPU Method	1/8	57 - 93	0.0038 – 0.0067
GPU Method	1	65 - 95	0.005 – 0.0062
GPU Method	1/2	65 - 95	0.004 – 0.006
GPU Method	1/4	65 - 95	0.003 – 0.006
GPU Method	1/8	65 - 95	0.003 – 0.005

Table 3. Comparison of proposed algorithm processed on CPU and GPU.

We evaluate the pixel-based method implemented on CPU. As Table 3 shows, in the GPU method we count the pixel numbers by using compute buffer in parallel. This work can also be implemented by return a texture containing color marked pixels. For each post processing volume, we mark the same color to all pixels inside it. Then we calculate the percentage of colored pixels on the texture in CPU and modify the blend weights of LUTs. We also evaluate

the different situations by down scaling the resolution of camera from 1 (1920 * 1080) to 1/8 (240 * 135). The results show that our algorithm processed on a GPU significantly improves the computation time and framerate. For the CPU method, it is time consuming to make a for loop reading each pixel on the texture. The long processing time leads to low framerate which affect player's game experience under full resolution. However, in GPU method, pixels can be processed in parallel on multiple threads. The resolution has not been a performance bottleneck. Therefore, our GPU method has a consistently efficient performance under different resolution situations.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we discussed the problem of evaluating tone mapping quality for video gaming applications using an objective quality metric. We also discuss the problem of tweaking the LUTs automatically in real-time considering the sight of view.

In Chapter 3, we propose a modification to an algorithm which can optimize the parameters of tone mapping operators for synthetic image by automatically minimizing the perceptual distortion using an evolution strategy. The perceptual distortion is measured by a feature-based objective image quality metric which utilizes using a content-adaptive tone mapping operator for taking virtual photos. Our algorithm considers the unique characteristics of rendered HDR contents. We show that, our results with optimized TMO parameters preserve more visible contrast than other optimization algorithms evaluated by DRIM. The scores of a no-reference metric called BRISQUE also shows that our results outperform other results.

In Chapter 4, we proposed a pixel-based real-time interpolation method for LUTs which contain both tone mapping curves and color grading modifications. Our algorithm changes LUT colors based on the change of the player's sight. We use a camera depth texture to reconstruct the pixel world position and use a compute buffer to parallel count the pixel numbers in sight. We show that our results have a closer visual appearance to the fronting LUT colors when the player looks around. After accelerating the program using GPU, our algorithm can be executed fluently in real-time.

5.2 Future Work

In the thesis, we represent an objective quality metric based synthetic image tone mapping optimization algorithm and a real-time LUT interpolation algorithm based on player's sight of view. Here, we provide some suggestions for future directions which can be extended from our research.

The detail weight in feature-based quality metrics control the details in bright and dark areas. The detail weight can be modified by the script to change the visibility of game content such as non-player character. Some trigger conditions such as time remaining can be introduced to control the game difficulty.

The size of the post processing volume is also a interesting direction to be considered. The boundary of post processing volume can be determined by the three-dimensional geometry of objects. Also, it would be interesting to further analyze the interface for artists to input the number of volumes and automatically select the most suitable post processing volumes.

Bibliography

[1] Reinhard, E., Heidrich, W., Debevec, P., Pattanaik, S., Ward, G., & Myszkowski, K. (2010). High dynamic range imaging: acquisition, display, and image-based lighting. Morgan Kaufmann.

[2] Boitard, R., Pourazad, M. T., Nasiopoulos, P., & Slevinsky, J. (2015). Demystifying High-Dynamic-Range Technology: A new evolution in digital media. IEEE Consumer Electronics Magazine, 4(4), 72-86.

[3] Brooks, A. L., Brahnam, S., & Jain, L. C. (2014). Technologies of Inclusive Well-being. Springer.

[4] AMPAS. (2014). Academy Color Encoding System (ACES) Documentation Guide; Technical Bulletin TB-2014-001.

[5] Hable, J. (2010) “Uncharted 2: HDR Lighting”. In: Game Developers Conference.

[6] Reinhard, E., Stark, M., Shirley, P., & Ferwerda, J. (2002, July). Photographic tone reproduction for digital images. In ACM transactions on graphics (TOG) (Vol. 21, No. 3, pp. 267-276).

[7] Brinck, W., Andrew M., & Yibing J. (2016). "The technical art of uncharted 4." ACM SIGGRAPH 2016 Talks.

[8] Selan, J. (2005). “Using Lookup Tables to Accelerate Color Transformations” in Matt Pharr, ed., GPU Gems 2, Addison-Wesley, pp. 381–408.

[9] Blackmagic Design. DaVinci Resolve 16. [Online]. Available: https://www.blackmagicdesign.com/products.

[10] Andrew, F., & Conrad, C. (2019). Adobe Photoshop CC Classroom in a Book. Pearson Education.

[11] Hajime, U., & Kentaro, S. (December 2018) "Practical HDR and wide color techniques in gran turismo SPORT." ACM SIGGRAPH Asia 2018 Courses, Article No. 15.

[12] Gao, X., Porter, J., Brooks, S., & Arnold, D. V. (2017, October). Evolutionary Optimization of Tone Mapped Image Quality Index. In International Conference on Artificial Evolution (Evolution Artificielle) (pp. 176-188). Springer, Cham.

[13] Aydin, T. O., Mantiuk, R., Myszkowski, K., & Seidel, H. P. (2008). Dynamic range independent image quality assessment. ACM Transactions on Graphics (TOG), 27(3), 69.

[14] Yeganeh, H., & Wang, Z. (2012). Objective quality assessment of tone-mapped images. IEEE Transactions on Image processing, 22(2), 657-667.

[15] Ma, K., Yeganeh, H., Zeng, K., & Wang, Z. (2015). High dynamic range image compression by optimizing tone mapped image quality index. IEEE Transactions on Image Processing, 24(10), 3086-3097.

[16] Gao, X., Brooks, S., & Arnold, D. V. (2015). Automated parameter tuning for tone mapping using visual saliency. Computers & Graphics, 52, 171-180.

[17] Krasula, L., Narwaria, M., Fliegel, K., & Le Callet, P. (2016). Preference of experience in image tone-mapping: Dataset and framework for objective measures comparison. IEEE Journal of Selected Topics in Signal Processing, 11(1), 64-74.

[18] Gao, X., Brooks, S., & Arnold, D. V. (2017). A feature-based quality metric for tone mapped images. ACM Transactions on Applied Perception (TAP), 14(4), 26.

[19] Ledda, P., Chalmers, A., Troscianko, T., & Seetzen, H. (2005, July). Evaluation of tone mapping operators using a high dynamic range display. In ACM Transactions on Graphics (TOG) (Vol. 24, No. 3, pp. 640-648). ACM.

[20] Uchimura, H., & Suzuki, K. (2018, December). Practical HDR and wide color techniques in gran turismo SPORT. In SIGGRAPH Asia 2018 Courses (p. 15). ACM.

[21] Khaldieh, A. (2018). Tone mapping of high dynamic range video for video gaming applications (T). University of British Columbia. Retrieved from https://open.library.ubc.ca/collections/ubctheses/24/items/1.0366909.

[22] Burger, W., & Burge, M. J. (2016). Digital image processing: an algorithmic introduction using Java. Springer.

[23] Hooker, J. T. (2016). Volumetric global illumination at Treyarch. SIGGRAPH 2016 Course: Advances in Real-Time Rendering in 3D Graphics and Games.

[24] Shkarofsky, I. P., & Nickerson, S. B. (1982). Computer modeling of multipath propagation: Review of raytracing techniques. Radio science, 17(5), 1133-1158.

[25] ITU-R Recommendation BT.1886. (2011). Reference Electro-optical Transfer Function for Flat Panel Displays used in HDTV Studio Production.

[26] Miller, S., Nezamabadi, M., & Daly, S. (2013). Perceptual signal coding for more efficient usage of bit codes. SMPTE Motion Imaging Journal, 122(4), 52-59.

[27] Barten, P. G. (2003, December). Formula for the contrast sensitivity of the human eye. In Image Quality and System Performance (Vol. 5294, pp. 231-239). International Society for Optics and Photonics.

[28] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.

[29] Zhang, L., Zhang, L., Mou, X., & Zhang, D. (2011). FSIM: A feature similarity index for image quality assessment. IEEE transactions on Image Processing, 20(8), 2378-2386.

[30] Reinhard, E. (2002). Parameter estimation for photographic tone reproduction. Journal of graphics tools, 7(1), 45-51.

[31] Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003, November). Multiscale structural similarity for image quality assessment. In The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003 (Vol. 2, pp. 1398-1402). Ieee.

[32] Mantiuk, R., Kim, K. J., Rempel, A. G., & Heidrich, W. (2011, August). HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In ACM Transactions on graphics (TOG) (Vol. 30, No. 4, p. 40). ACM.

[33] Kundu, D., & Evans, B. L. (2015, September). Full-reference visual quality assessment for synthetic images: A subjective study. In 2015 IEEE International Conference on Image Processing (ICIP) (pp. 2374-2378). IEEE.

[34] Kundu, D., & Evans, B. L. (2015, November). No-reference synthetic image quality assessment using scene statistics. In 2015 49th Asilomar Conference on Signals, Systems and Computers (pp. 1579-1583). IEEE.

[35] Mittal, A., Moorthy, A. K., & Bovik, A. C. (2011, November). Blind/referenceless image spatial quality evaluator. In 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR) (pp. 723-727). IEEE.

[36] Boitard, R., Bouatouch, K., Cozot, R., Thoreau, D., & Gruson, A. 2012. Temporal coherency for video tone mapping. Proc. SPIE 8499.

[37] Bennett, E. P., & McMillan, L. (2005, July). Video enhancement using per-pixel virtual exposures. In ACM Transactions on Graphics (TOG) (Vol. 24, No. 3, pp. 845-852). ACM.

[38] Gastal, E. S., & Oliveira, M. M. (2011, August). Domain transform for edge-aware image and video processing. In ACM Transactions on Graphics (ToG) (Vol. 30, No. 4, p. 69). ACM.

[39] Boitard, R., Cozot, R., Thoreau, D., & Bouatouch, K. (2014). Survey of temporal brightness artifacts in video tone mapping. In HDRi2014-Second International Conference and SME Workshop on HDR imaging (Vol. 9).

[40] Kiser, C., Reinhard, E., Tocci, M., & Tocci, N. (2012, September). Real time automated tone mapping system for HDR video. In IEEE International Conference on Image Processing (Vol. 134).

[41] Eilertsen, G., Mantiuk, R. K., & Unger, J. (2015). Real-time noise-aware tone mapping. ACM Transactions on Graphics (TOG), 34(6), 198.

[42] Mantiuk, R., & Seidel, H. P. (2008, April). Modeling a Generic Tone-mapping Operator. In Computer Graphics Forum (Vol. 27, No. 2, pp. 699-708). Oxford, UK: Blackwell Publishing Ltd.

[43] Chisholm, S. B., Arnold, D. V., & Brooks, S. (2009, July). Tone mapping by interactive evolution. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 515-522). ACM.

[44] Gao, X., Brooks, S., & Arnold, D. V. (2013, July). Virtual photograph based saliency analysis of high dynamic range images. In Proceedings of the Symposium on Computational Aesthetics (pp. 87-92). ACM.

[45] Digital Multimedia Lab, "DML-Video-Gaming-Content-HDR dataset," [Online]. Available: http://dml.ece.ubc.ca/data/Video-Gaming-Content-HDR.

[46] Kundu, D., & Evans, B. L. (2015, November). No-reference synthetic image quality assessment using scene statistics. In 2015 49th Asilomar Conference on Signals, Systems and Computers (pp. 1579-1583). IEEE.

[47] Khan, I. R., Rahardja, S., Khan, M. M., Movania, M. M., & Abed, F. (2017). A tone-mapping technique based on histogram using a sensitivity model of the human visual system. IEEE Transactions on Industrial Electronics, 65(4), 3469-3479.

[48] Pharr, M., & Fernando, R. (2005). Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley Professional.

[49] 3D LUT Creator. [Online]. Available: https://3dlutcreator.com/.

[50] Pitié, F., Kokaram, A. C., & Dahyot, R. (2007). Automated colour grading using colour distribution transfer. Computer Vision and Image Understanding, 107(1-2), 123-137.

[51] "Sci-Fi Laboratory Pack 2" [Online]. Available: https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-laboratory-pack-2-35688.

[52] Uncharted 4 website [Online]. Available: https://www.unchartedthegame.com/en-ca/media/.