Executive Summary

The aim of this project is to propose a novel algorithm for the creation of the HDR image, and develop a HDR image painting application based on this algorithm to provide an unprecedented approach to paint an HDR image with user-defined luminance.

The main contributions and achievements in this project are as follow:

- A research on the possible development of the HDR image applications has been carried out based on the HDR imaging background investigation and comparison and evaluation of the current HDR imaging applications, see pages 24-25
- Focusing on the research on creation of HDR image, this project proposed a novel algorithm to construct an HDR image and developed a brand new approach for the HDR image painting process. 25-26
- A HDR image painting application was developed as the implementation of the proposed algorithm and the approach of painting process. 26-38
- A set of new painting tools has been developed for this specific HDR painting application, with the purpose of more intuitively operation and the ease of use for the users. 30-35
- The conclusion of the HDR image painting application is given based on the evaluation of the testing and the user trials. 47-49
- The main aim and objectives of this project have been achieved. However, there is still improvement space. The outline of a paint tool to prevent details losing based on the Bilateral filtering algorithm has been designed, which considered as a valuable future development. 50-52

Acknowledgement

I would like to express my deep gratitude to Dr Erik Reinhard for his guidance, enlightening advice and constant encouragement through all the stage of this project.

I would like to thank Ms Foteini Pouli for helping me to understand concepts and giving me advices and direction to improve my project..

I would like to thank my markers for their feedbacks and their constructive criticisms.

I would like to express my deep appreciation to my family and my friends. Their encouragement and support always with me even I have not been seeing them for more than one year.

Table of Contents

1. Introduction and Context

1.1 Aims and Objectives

This project is about researching the High Dynamic Range (HDR) imaging, and developing a HDR image painting application with the ability to paint with the luminance that out of dynamic range of the LDR devices. This painting panel tends to solve the problem of how to draw HDR image with the contrasts that are much larger range than the typical display devices can display in an intuitive way.

This project is an attractive combination of the research carried on the HDR imaging creation theory and the software design that allowing the interaction with users. And as the project is considered as a mix of practical development and research, the background theory and techniques study are essential phase in order to achieve this project.

As aim of this project is to develop HDR painting application based on the proposed algorithm for the creation of an HDR image, the objectives for this project could be divided into several states as follow:

1. Based on the research about the HDR image processing and creation, propose a novel algorithm to create HDR images.
2. Propose a solution for the HDR image painting with the luminance out of dynamic range of LDR displayer.
3. Construct a basic painting application as a start point for the implementation of the HDR painting solution by Java programming.
4. Develop an HDR image painting application to implement the algorithm and solution for the HDR painting.
5. Develop some novel luminance modulation tools for this HDR painting application.
6. Improve the user interface based on the user trials and makes the HDR image painting process a more intuitive way.
7. Carry on the research about the possible improvement both on the HDR creation algorithm and the performance of the HDR painting application.

1.2 Organization of this dissertation

In this dissertation, chapter 1 will introduces the background of the project and the previous work done by the others in the focusing area.

Secondly, the algorithm proposed in this project as well as the design of the HDR image painting application will be introduced in chapter 3.

Thirdly, chapter 4 will be the tests of the solution and the HDR image painting application.

Then, a critical evaluation of the work been carried out and a conclusion for this project will be given in chapter 4.

Finally, the suggestions on possible improvement on the better performance of the HDR image painting application and some ideas for the future work will be discussed in chapter 6.

2. The context and background

2.1 Basic concept of HDR

The dynamic range is the radiance contrast between the brightest and the darkest regions in an image. In the real-world scenes, the dynamic range will be very wide if there is bright object like sun or artificial light sauce in the scenes. Normally, the dynamic range tends to be out of the range of the display device can represent.

The high dynamic range (HDR) imaging is an emerging field in image processing, computer graphics, and photography. Compare to the standard imaging techniques and photographic methods, HDR allow a greater dynamic range of luminance between the lightest and darkest areas of an image. With this feature, HDR gives a solution to the problem of how to simultaneously representing both the brightest highlights and the deepest shadows in an image. It has the ability to show the details at two ends of the tonal range, in this case, High dynamic range images provide a vastly higher visual quality than conventional low dynamic range images. Nowadays, with the help of HDR, photography, the film industry, and computer graphics have a significant improvement and HDR imaging is already widely accepted in these domains.[4]

Dynamic range is a ratio between any maximum and a minimum physical measure. For a natural scene, HDR is the ratio between the darkest and lightest parts of a scene. For a monitor, HDR is the ratio between the maximum and minimum intensities emitted from the screen. For a digital photograph, HDR is the ratio between the maximum and minimum luminance values expressed in candelas per square meter. [4]

The dynamic range of values that could be afforded by a traditional image is approximately two orders of magnitude. Data in each of the red, green and blue channels in a single pixel are stored as a byte. Thus it is impossible to represent images with a much higher dynamic range directly. It is known that the real world produces a much greater dynamic range than just two orders of magnitude.

The table 2.1 indicates the typical ambient luminance levels for commonly encountered scenes.

Condition	Illumination (in cd / m^2)
Starlight	10^{-3}
Moonlight	10^{-1}
Indoor lighting	10^{2}
Sunlight	10^{5}
Maximum intensity of common CRT monitors	10^{2}

Table 1.1 Source: Ref [3]

The sunlight could be 100 million times brighter than starlight [1, 2]. As can be seen from table 1.1, the maximum intensity of common CRT monitors is only capable of representing of 2 orders of magnitude. While the human visual system could adapt to nearly 10 orders of magnitude of lighting conditions, and within one scene, the human visual system could accept a range of about 5 orders of magnitude simultaneously. [1] The common modern liquid crystal display (LCD), whose operation range is limited by the strength of backlight, would be brighter than CRT displays, yet not greater enough to be orders of magnitude.

Human visual system adjusts the amount of light entered the eyes by controlling the pupil, photoreceptors and neurons in the chemical and neural processes. Photographic devices using the lens aperture and exposure time to simulate this behavior. However, images captured in this way will lose the data that cannot be afforded by the lens and exposure state. In other words, we may need to capture a scene in the entire dynamic range of all of the information, and only had to wait until after the decision to choose which parts of discarded. As we will demonstrate in this article, the use of HDR image support in the source image information stored in more light, so to support the end-user access to the optical information and run them when the amendment.

In this case, capturing the scene with a range of intensities and level of quantization representative of the scene would be a better method, compared with matching to the display devices when capture the scene. In another word, we may need to capture the scene with the entire dynamic range of all the information, all this information will be retained to be selected when need.

2.2 The capture of HDR images

In order to obtain high dynamic range images, some particular methods will be used based on the existing equipment with limited display dynamic range. One way to achieve this goal is to capture HDR images as the raw file directly from real world. This process may be achieved by using the professional camera with the 12-16 bits AD converter. With the development of technology, the HDR camera with the ability to generate high-quality HDR images directly is beginning to enter the market.

Another more widely used technique to obtain a HDR image is using multiple images under different exposure situation. Firstly set different exposure time to obtain multiple images, and then analyze the dynamic range of these images, and combine each of the dynamic range to finally get a HDR image. In this way, low dynamic range images are used to composite a HDR image. On the other hand, the output HDR image is down-sampled as a low dynamic range image.

The following picture demonstrates the method of capturing a HDR image by the combination of multiple images with different exposure time.

Fig. 2.1 Source: Ref. [4].

2.3 The encodings of HDR images

An encoding is a mechanism about the raw bit representation of a pixel value in an image; a format refers to the data wrapper around the pixels to compose a complete image. [6]

The encoding of HDR image pixel is the organization of the image information. HDR image file should record the entire data of gradation and dynamic range in a real scene. Most computer graphics software works in a 24-bit RGB space, with 8-bits per primary color channel in a power-law encoding, while the way that HDR images store pixel values record a much wider gamut than the standard ones. In another word, the encoding of HDR images asks for a more efficient method. Several HDR encodings and relative formats have been developed already; more exciting fact is that some of these have already been used for a long time.

The colors of the traditional 24-bit RGB images are encoded based on the ability of their relative target output device, thus the traditional images are usually seen as output-referred. The dynamic range of the real-world scenes is very high, while most of the output devices cannot represent the full range of HDR images, as a logical result, HDR encoding is not necessarily concern about the output devices, so their pixels directly related to radiance in either real or virtual scene. Scene-referred represents the light values of an unbounded scene. [6] Output-referred images represent the light values of a light-receiving or light-transmitting device. [4] Although, as we known, human eye is dichromatic, the information recorded by HDR is more than our naked eyes can see. But in many situations we wish to record more than we could see of a scene. For instance: the different wavelengths in the satellite imagery, the abstract data in scientific and medical visualization, etc. In these cases, HDR formats are independent from display process, providing a method to achieve these.

HDR encoding holds the scene-referred data, in order to be represented in the output devices; it can be transform into output-referred data. It does not make any sense to reverse the transformation because up till now there's no output device can reproduce the scene in the real world. [6]

This technique that transforms a scene-referred into output-referred is called tone mapping. (See the section 3'tone mapping' for details)

The following lists some key encoding format:
Radiance RGBE format: also known as the HDR format (.hdr, .pic), it was firstly introduced in 1985 by Ward when he was developing the Radiance painting system. This format uses 32 bits to represent a pixel. The mantissa of each color channel occupies 16 big, while another 16 bits was used to store exponent. The RGBE refers

to red, green, blue and exponent. The RGBE components are computed with the equation [6] below:

$$E=\lceil \log_2(\max(R_W, G_W, B_W)) + 128 \rceil \quad (2.1)$$

$$R_M=\left\lfloor \frac{256R_W}{2^{E-128}} \right\rfloor$$

$$G_M=\left\lfloor \frac{256G_W}{2^{E-128}} \right\rfloor$$

$$B_M=\left\lfloor \frac{256B_W}{2^{E-128}} \right\rfloor$$

The dynamic range of these encoding is over 76 orders of magnitude and its high accuracy is enough for most of the HDR applications.

The TIFF format: the Tagged Image File Format (.tif, .tiff) stores three color channels, which are Red, Green, and Blue. The values after the log transformation are indicated by 11bits. The dynamic range provide by TIFF format is 3.6 orders of magnitude (3600:1).

The OpenEXR format: the Extended Range format (.exr) was provided in an open-source C++ library in 2002. [8] This format uses 48 bits to store a pixel. Each color channel occupies 16 bits IEEE-754 float format, which is one bit for sign, five bits for exponent, and ten bits for mantissa. The dynamic range of the format could be 10.7 orders of magnitude. [13]

In the three formats above, the dynamic range of RGBE format is 76 orders of magnitudes, which is much wider than the dynamic range that human eyes can sense or the normal scenes in real world. In this case, it would be more effective if reduce its dynamic range while increase its accuracy. Thus OpenEXR format will be more suitable for the realistic use, and until now, some advanced Graphics cards directly support this format. But as radiance RGBE format has been used earlier, there are still many HDR applications with this format.

All the existing HDR encoding formats can be seen as a lossless version for the original scene. Although the information will be hold in these formats, the cost of the addition dynamic range and the high accuracy should be kept under some level considering the burden these will bring. In fact, the “lossy” HDR formats which offering much better compression with current JPEG [14] format will wider the application of HDR in commercial markets.

2.4 Human Visual System and HDR Tone Mapping

2.4.1 Why using HDR tone mapping.

In the recent years, high dynamic range image plays a more common and more important role in computer graphic. As introduced in section 2, HDR is a new challenge of display device technologies and the image processing technologies. Displaying HDR images is an advancing ability required of hardware equipment. Thanks to the development of physical simulation techniques, combination of multiple exposure photography [18] and a variety of new sensor technologies [19], the HDR images with the ability of reflecting the absolute luminance values and real brightness of a real scene is also available. Moreover, on the perspective of the recent process in digital photographic technology, it is likely that in a near future the digital camera and digital video will be capable to capture the HDR images directly. As the availability of HDR images grows due to both these advanced new technologies in lighting simulation and hardware support, there is a growing demand to display the HDR images on low dynamic range devices. [10]

Although various techniques are available to generate HDR images, yet it is not possible to linearly display such wide luminance range. To translate HDR to a display media requires the reduction the dynamic range. Thus tone mapping is generally employed for this requirement.

2.4.2 Human visual system.

Human visual system is the world's oldest, best image processing system, but its functionality is far from perfect. In human visual system, the perception of the image information is non-uniform, non-linear process rather than a linear process. Not all the changes in the images can be perceived.

There are some Human Vision System Characteristics:

1. The resolving power of human eyes is relatively weaker in the dark areas compare to the bright area.
2. The resolving power of human eyes will decrease if the contrast is too high or too low; in this case, the brightness details of an image are more important than color details.
3. Mach effect can result in local threshold effect.

The dynamic range of the human visual system (HVS) is 100 times larger than a

printed page. And it is capacity of function over the huge range of illumination during a day. The intense of sunlight can be million times more than moonlight, which means the effective range of illumination is more than a billion to one [3]. The mechanism, provided by dark and light tonal adaptation, allows the HVS to distinguish content at any area of the dynamic range. In another word, HVS can simultaneously perceive the detailed contrast in both the brightest and darkest parts of an high dynamic range scene.

Modeling of human visual adaptation is crucial step to realistic HDR tone mapping.

In order to accurately analyze an image, or compare with the real scene, we hope that the output image has the same visual effect with the original scene. Ideally, when we observed under the same conditions of a real scene and an image representing the scene, whether the image is computer-generated or generated by the camera to produce images and real scene should have the same tone, which is the brightness levels are matched.

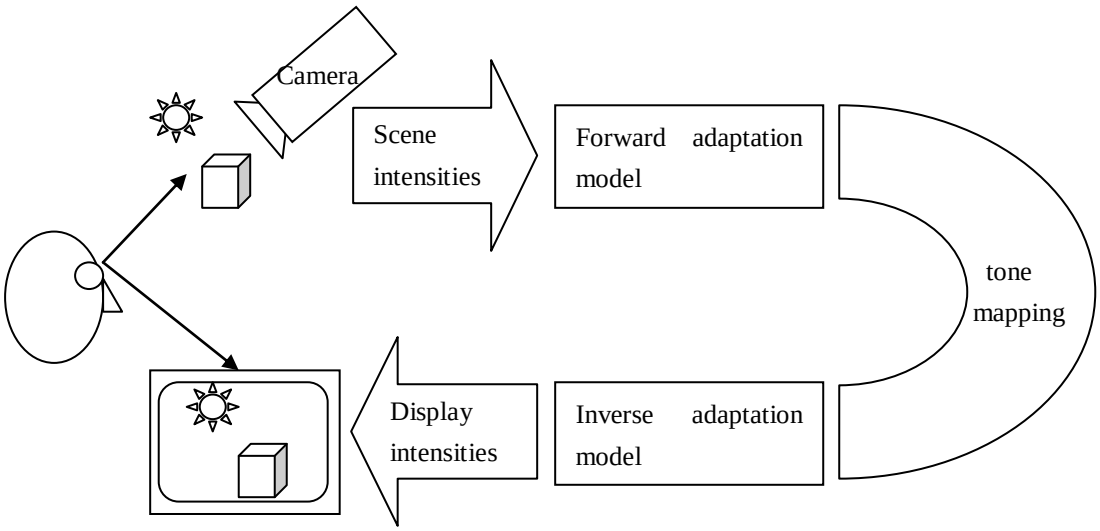


Fig 2.2 Source: Ref [6]

Fig 3.1 presents a basic framework for HDR tone mapping using the models of visual adaption. This graphic outlines two models – forward and inverse adaptation models. [6]

As indicated in the figure 3.1, the forward adaptation models will extract visual appearance parameters by dealing with the luminance values of the scene for appropriate tone mapping process. While the inverse adaptation model will take the visual appearance parameters and the adaptation parameters appropriate to the display viewing condition and will display the output luminance values. [6]

There are several visual adaptation models like photoreceptor adaptation model and

threshold adaptation model could be used for forward and inverse adaptation.

The photoreceptor adaptation is the most important factor responsible for visual adaptation. There strong tie between various tone mapping algorithms and this visual adaptation model. The following equation introduces the mathematical model of the photoreceptor adaptation.

$$\frac{R}{R_{max}} = \frac{I^n}{I^n + \sigma^n} \tag{2.2}$$

R refers to the photoreceptor response with the range $0 < R < R_{max}$, and R_{max} is the maximum response, I is light intensity, and σ is the semi saturation constant (the intensity which leads to the half-maximum response), n is a sensitivity control exponent with a value between 0.7 and 1.0 [5].

2.4.3 HDR tone mapping

Tone mapping combined with bit-depth scaling is a approach to map the pixels representative of the scene to the gamut of the display devices.

The implement of tone mapping from HDR to low dynamic range is to compress the dynamic range with the tone mapping operators.

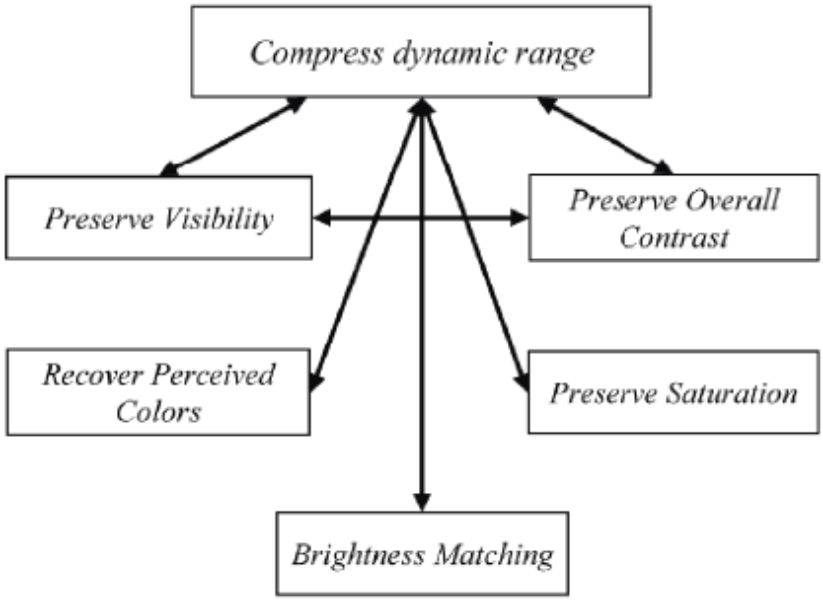


Fig 2.3 Source: Ref [9]

The figure 3.2 presents the sub-goals [9] of tone mapping:

(1) Compress dynamic rang
This is the main goals of tone mapping.

(2) Preserve Visibility
Keep visibility of local details of the original scene into the output low dynamic range images only if the details are viewable for human visual system. Prevent the image from being unrealistic.

(3) Preserve Overall Contrast
Preserve the overall contrast of the output image in a level that a human observer can obtain in the original scene.

(4) Preserve Saturation
Preserve the saturation acceptable for human visual system, and keep the saturation in a reasonable impression similar to the original scene, neither exaggerate nor reduce saturation.

(5) Recover Perceived Colors
Prevent the loss of color condition that perceived by human visual system in the source scene. Recovering process can be implemented via simulating the color constancy ability of the HVS.

(6) Brightness matching
This procedure is focusing on preserve the overall brightness. It related to the use of the output dynamic range.

Besides these are general requirements of the HDR tone mapping, the tone mapping methods require some level of computationally efficiency as well.

Fig 2.4 images with different tone mapping Source: Ref [17]

The tone mapping operators can be classified into two categories [10]: global operators and local operators. Global operators take the whole image as the neighborhood for each pixel, which means each pixel in an image is compressed via a same compression curve. In contrast, the local operators will compute the adaptation level for each pixel which is determined by pixel itself and a region surrounding the pixel. The compression process will be based on this local adaptation level.
The algorithm of global operators is to map every pixel of the image directly into a lower dynamic range under the same compress condition for the limited display devices. Global operators have a remarkable advantage which is computationally efficiency thanks to their working model. Thus many global operators can be executed in real time. Generally speaking, global operators executes much faster than local operators. However, global operators might not preserve the visibility of images if the source image is extremely high in dynamic range, in some situations, the visually unpleasant artifacts will occur. In other words, the capacity of compressing HDR images of global operators is limited because global operators could lead to loss of visibility or contrast.

The list of some important global operators is as followed:

(1) Miller Brightness-ratio-preserving Operator
(2) Tumblin-Rushmeier Brightness-preserving Operator
(3) Ward Contrast-based Scale Factor

(4) Ferwerda Model of Visual Adaptation
(5) Logarithmic and Exponential Mappings
(6) Drago Logarithmic Mapping
(7) Reinhard and Devlin Photoreceptor Model
(8) Ward Histogram Adjustment
(9) Schlick Uniform Rational Quantization

Local operators, instead of using a single tone-mapping curve for the entire image, compress each pixel according to its luminance value and the luminance values of a certain set of surrounding pixels. In image processing term, global operators executed by pixel, while local operators then executed by neighboring region. The implicitly is that the mapping function is not only decided by the current pixel's luminance, but its position as well. According to the research on the human visual system (HVS), the adaptation for the changing of luminance of each pixel is much more sensitive to luminance value with the absolute change. That indicates in the area with similar luminance values, observer will not notice even if the absolute luminance value changes a lot. The outline of local operators' implementation is as followed:

$$L_d(x,y) = m(x,y)L(x,y) \tag{2.3}$$

In equation 3.2, $L(x,y)$ is the luminance value for each pixel of a HDR image; $L_d(x,y)$ is the output luminance value mapped into the display devices. $m(x,y)$ is the local mapping factor for each pixel. The main problems to be solved for local operators are to determine what is the proper scale of the surrounded region for each pixel to be computed, how to decide the weight of surrounding region to the local adaptation level, and how to deal with the adaptation level in the implementation of dynamic range compression.

The list of some important local operators is as followed:
(1) Chiu Spatially Variant Operator
(2) Rahman Retinex
(3) Fairchild iCAM
(4) Pattanaik Multiscale Observer Model
(5) Ashikhmin Spatially Variant Operator
(6) Reinhard et al. Photographic Tone Reproduction
(7) Pattanaik Adaptive Gain Control
(8) Yee Segmentation-based Approach

2.4.4 Reinhard et al.'s tone reproduction.

Here I choose Reihard et al' tone reproduction as an example to indicate the photographic tone reproduction operator in details.

Reinhard et al.'s tone reproduction, following other tone-reproduction operators, views the log average luminance $\overline{L_w}$ as a useful approximation of a scene's key.

$$\overline{L_w} = \exp\left(\frac{1}{N} \sum_{x,y} \log\left(L_w(x,y)\right)\right) \tag{2.4}$$

For average-key scenes, the log average luminance $\overline{L_w}$ should be mapped to 18% of the display range, which is in line with common photographic practice. Higher-key scenes should be mapped to a higher value, and lower-key scenes should be mapped to a lower value. The dynamic range of each pixel in the image should be reduced by a method of local contrast, which is similar to the dodging and burning ("a technique for printing process to manipulate the exposure of a selected area on a photographic print, deviating from the rest of the image's exposure." from Wikipedia.)

The first step of this tone mapping is to scales the image using the following equation, in which α is a parameter defined by a user, and α sets the exposure of the HDR image effectively.

$$L_m(x,y) = \frac{\alpha}{\overline{L_w}} L_w(x,y) \tag{2.5}$$

Here the outcome of the first step $L_m(x,y)$ is the obtained value after the initial linear mapping.

Generally, many images have a good quality of average dynamic range with only a few luminance regions out of limited range near the highlights area or in the sky, etc. In order to solve this problem, the second step is to compress the images in dynamic range with the transfer functions to deal with the high luminance regions. This process could be modeled with the equation 3.4.

$$L_d(x,y) = \frac{L_m(x,y)}{1+L_m(x,y)} \tag{2.6}$$

This method linearly scales the small values, in contrast, the higher luminance are compressed by larger amounts. All the values for each pixel will be mapped into a display range between 0 and 1. As a fact, the original source image does not hold infinitely large luminance values. That means the largest display luminance does not closely reach the value 1. To turn the bright areas into a controlled fashion, the

function 3.4 is blended with a linear mapping, as the function 3.5.

$$L_d(x,y) = \frac{L_m(x,y)\left(1+\frac{L_m(x,y)}{L_{white}^2}\right)}{1+L_m(x,y)} \quad (2.7)$$

L_{white} is a new user parameter as the smallest luminance value to be mapped to white. It's effective in minimizing the loss of contrast when tone mapping a low dynamic range image.

To make this basic method locally adaptive, change the division into a value based on the largest surrounding area that does not contain any sharp contrasts for each pixel. To find this region, a reasonable method of scale-space is afforded by center-surround computations. Pixel by pixel, the Gaussian-weighted average is computed. Then compute the Gaussian-weighted average over a larger region surrounded the center of pixel. By computing the difference of Gaussians (DoG) at a given scale, we can detect if the surrounded region under the filter kernel has a sharp contrast or not. If there are no significant contrasts in this region, the difference of these two Gaussians will be close to zero. Through this method, repeat this procedure for the largest scales of region.

$$L_s^{blur}(x,y) = L_m(x,y) \otimes R(x,y,\sigma_s) \quad (2.8)$$

$R(x, y, \sigma_s)$ is a two-dimensional Gaussian defined with a standard deviation of σ_s for the scale s. The center-surround mechanism at the scale s is processed with the function 3.7 as followed.

$$V_s(x,y) = \frac{L_s^{blur} - L_{s+1}^{blur}}{2^{\emptyset}\alpha/_{s^2} + L_s^{blur}} \quad (2.9)$$

The DoG is defined by function 3.7. The normalization by $2^{\emptyset}\,\alpha/_{s^2} + L_s^{blur}$ allows the result to quantify relative differences, rather than absolute luminance differences. Now V_s is independent of absolute luminance values.

After this process, we try to find the largest region with the relatively low contrast for each pixel. To achieve this, we search the maximum scale S_{max} for which the DoG is less than a threshold value$\in$.

$$S_{max}: |V_{smax}(x,y)| < \in \quad (2.10)$$

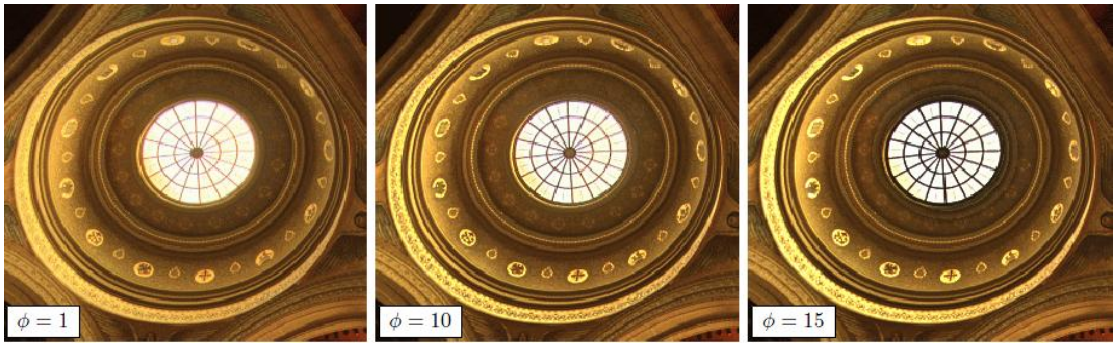


Fig 2.5 The free parameter φ in Equation 3.8 controls sharpening Source: Ref [16]

The corresponding center Gaussian blurred pixel, produced in this method, is taken as a local adaptation. The local operator implements a model of dodging and burning in photographic terms. Finally, the compression function is given by following:

$$L_d(x,y) = \frac{L_m(x,y)}{1 + L^{blur}_{smax(x,y)}(x,y)} \tag{2.11}$$

2.5 Filtering technology for the HDR imaging

2.5.1 Basic concept on filtering

Filtering is considered as one of the most fundamental operations of image processing and computer vision [21]. The concept on the broadest sense of filtering is that the value of the filtered image is determined by a function of the values of the original image in a small region of its neighborhood. Three basic filters are introduced as follow:

(1) Mean filter: the solution of the mean filter is to replace value of the pixel by the average (mean) value of the neighbor near its location.[22]

	97	103	83	82	81	
	98	103	105	108	97	
	99	255	102	101	95	
	101	103	107	255	93	
	93	101	112	108	107	

$*$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$=$

	97	103	83	82	81	
	98	103	105	108	97	
	99	255	137	101	95	
	101	103	107	255	93	
	93	101	112	108	107	

Fig 2.6 Mean filter Source: Ref [22]

(2) Median filter: the idea of the median filter is to replace the pixel value with the median value of its neighbors near its location. And Variance in neighboring values does not influence mean. Thus the solution of median filter is firstly sort the neighboring value, then find the middle one to replace the current pixel value.[22]

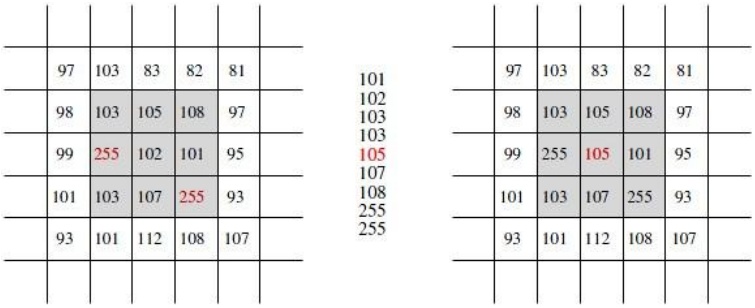


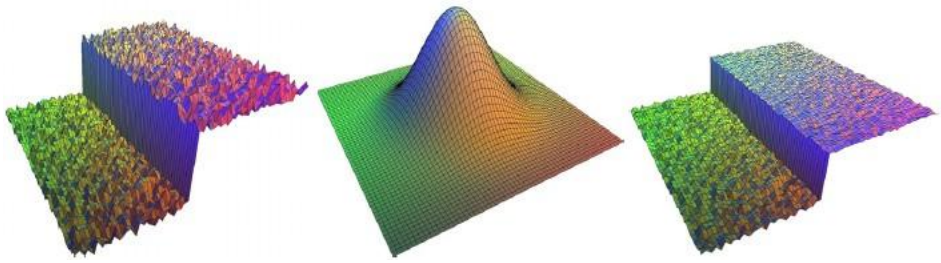
Fig 2.7 Median filter Source: Ref [22]

(3) Gaussian filter: The Gaussian blur is a type of image-blurring filter that uses a Gaussian function for calculating the transformation to apply to each pixel in the image. The pixel value is replaced by weighted average of its neighbors. Pixels near border of kernel are weighted lower

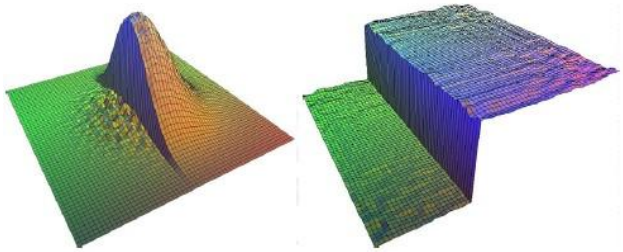
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.12}$$

(4) Bilateral filter: Bilateral filtering was developed by Tomasi and Manduchi as an alternative to anisotropic diffusion [21]. It is a non-linear filter where the output is a weighted average of the input. The weight of each pixel is computed using a Gaussian in the spatial domain multiplied by an influence function in the intensity domain that decreases the weight of pixels with large intensity differences. [26]

Therefore, the pixel value is influenced by pixels in neighboring area that are close spatially and that with a similar intensity.



(a) (b) (c)



(d) (e)

Fig 2.8 Bilateral filtering. (a)input; (b)spatial kernel *f*;(c) influence *g* in the intensity domain for the central pixel; (e) weight *f* g* for the central pixel. Source: Ref [20]

Original Mean: radius 6pixels Gaussian: σ= 4.0

Median: radius 6pixels Bilateral: σd =12.0, σ r = 3.0 Bilateral: σd =12.0, σ r = 6.0

Fig 2.9 output image with different filter. Source: Ref [22]

In figure ** illustrates the performance of these four filters. As can be seen, the mean filter removes simple noise but losses details of the image; Gaussian filter preserves details only for small σ; Median filter preserves some details, and is good at removing strong noise; bilateral filter blur the image and the edges of the image are preserved as well.

2.5.2 Bilateral Filtering for HDR Images

Bilateral filtering could be used for the display of high dynamic range images, and it could reduce the contrast of HDR image while preserving detail. Fredo Durand and Julie Dorsey proposed a fast bilateral filtering based on a two-scale decomposition of the image on a base layer, and encodes large-scale variations, and a detailed layer. The solution is to adjust the contrast on the base layer , in this case the detail of the original image will be preserved.[20]

Base Detail Color

Figure 2.10 color is treated separately using simple ratios. Source: Ref [20]

As illustrated in figure ** , The operator of this fast bilateral filtering takes a HDR image as a input, and is based on a two-scale decomposition of the image into a base layer and a detail layer. And only the base scale has its contrast reduced.

2.6 Previous work

With the growing demand of the commercial applications, the use of HDR images enters a new era. HDR images research is mainly focus on the display approaches on the current low dynamic range media. The compression of the HDR images is generally through the extension of the existing image compress approaches. Applications of HDR images have been used in photography, global illumination, realistic rendering and the digital cameras.

The tone mapping technique has been quite mature and sufficient, but the research on interfaces for HDR editing and manipulation is rare, especially for HDR painting.

Erik Reinhard, M. Colbert and C. E. Hughes have indicated an approach of painting in HDR imaging. They provide a brush-based interface and the ability to paint and choose values outside of the displayable luminance range. Their solutions for choosing and painting with HDR are from two aspects: one is quantitative and precise control and another is qualitative and visual-based manipulation. The tone mapping method for this approach is photographic tone mapping, which shows a strong overall performance and computational image quality metrics, and its support of real-time application. [6]

Fig 2.11 Source: Ref [6]

Adobe Photoshop now has provided painting HDR function. It is allowed user edit

and add effects to HDR/32-bpc images using a set of Photoshop tools. “When editing or painting on HDR images, you can preview your work at different exposure settings using either the 32-Bit Exposure slider in the document info area or the 32-Bit Preview Options dialog box (View > 32-Bit Preview Options). The HDR Color Picker also lets you preview your selected foreground color at different intensity settings, to match different exposure settings in an HDR image.”[22]

3. Algorithm and implementation

3.1 Overview

Bit depth determines the available number of colors in an image's color palette by using the numbers 0 and 1, or "bits," which are used to specify different colors. It is not necessary for an image to use all of these colors, but the colors can be differentiated with different levels of precision. In addition, the available number of unique shades in a grayscale image depends upon the bit depth. The relationship among them is that the higher bit depths the image has, the more combined sets of '0's and '1's are available and hence the more colors or shades can be encoded.

It is possible to create all the color pixels for a digital image by using different combinations of the three primary colors: red, green, and blue. Normally, each primary color is considered as a 'color channel', and its bit depth, which can differentiate the range of intensity values, is called the 'bits per channel'. The term 'bits per pixel (bpp)' is defined as the quantity of the bits in all of the three color channels. The 'bits per pixel' indicates the total available number of color for each pixel.

Most digital color images use eight '0's and '1's altogether, as they usually have 8-bit per channel, to form 256 different combinations to generate 256 different intensity values for each primary color. Since each pixel is formed by the combination of all the three primary colors, 16,777,216 different colors in total are allowed. This is called 'true color', which has 24 bits per pixel as each pixel consists of three 8-bit color channels.

Regular 8 and 16 bit per channel imaging is simply not cable of representing realistic or accurate processing real world levels of illumination, because they clamp brightness levels to discrete values between 0 and 255/65535.

The HDR image is able to make use of extra bits to save a lot more intensity range than that can be displayed all at once, which enables it to represent the real world levels of illumination. Besides, the HDR floating point image can also represent a great larger range of values. For example, a bright object may have brightness of 0.9 and the sun has the brightness of 10,000,000. It is possible to darken an image and the objects will become darkened, though the sun remains thousands of times brighter than the remaining of the scene.

The excess of 'full' black and white can be described by using floating point numbers. This enables an image to accurately demonstrate the intensity of the sun as well as the shadows in the same color regions with little distortion by intensive editing.

To develop a new approach to constructing an HDR image and to develop a painting

application allows for modulating the luminance of the HDR image intuitively in a LDR displayer. The main challenge lay in the fact that the Low Dynamic Range displayer cannot represent an HDR image, which brings difficulties in painting an HDR image, especially the modulation of luminance in the different area of the image.

As discussed in section 2.6, there are two main types of current HDR applications. One of them is the creation of the HDR image using the technology of combining different images with multiple exposures taken at varying shutter speeds into an HDR image, and then reducing it into a displayable image via tone mapping in order to get more details from the real scene or produce an artificial HDR effect. Another is the HDR image editing application, usually the normal function of these applications is to open up an HDR image and then use it for editing purpose. Normally, these HDR applications use tone mapping to solve this problem.

To paint HDR images with the existing difficulty in representing HDR on conventional display devices, this project has developed a novel method, instead of tone mapping the HDR color with a large range of luminance; a brand new method of painting the color with the high range of luminance was introduced.

3.2 The design of the algorithm

The idea of the algorithm design for this project is to break down the painting process into several stages, while keeping all stages of the process displayable in the LDR device. After the painting is complete, the result of every stage is merged and constructing them into an HDR image. Based on this thought, the painting process could be divided into 4 steps:

1. Firstly, constructing the shapes and the colors for the target image in an ordinary image. Each pixel channel is 8 bits as a normal 24bits image.

2. Secondly, based on the shapes of the result in the previous step, a grayscale image was drawn to indicate the luminance of the different areas of the image with the intensity of each pixel. Also, this grayscale image is 24-bits image with 8-bits possible intensities for each channel.

3. Merging the color image and the grayscale image. Each pixel of the two images has 8-bit on every channel. By multiply these two images pixel by pixel, the color channel of the resulting pixel can be expanded to 16-bit which makes the pixel 48 bits. Thus, 256*256 possible levels of intensity for each color channel could be achieved. This gives the image a much wider range of intensities.

In this approach, the process where the HDR image is created and the ease of use for this is unprecedented. As the resulting image is stored as a 48-bits file, it is the perfect material for any correction and further editing.

3.3 Design of the application.

3.3.1 Overview

The implementation of the algorithm is to paint the color and shape in one canvas, and manipulate the luminance of the image in the second canvas. Color and the shape of an image could be drawn on an ordinary canvas as the one in the normal painting application, and manipulated in the luminance detail of the image in another canvas. In these two stages, the images in the two canvas stay in the low dynamic range to suit the displayer. Via painting separately in two canvases, the whole painting process becomes intuitively visible for the users. After that, the image can be gained with a large brightness contrast via the merging two images.

Java is used as the programming language; because of its platform independency, it is easy to run on different platforms. As it is a painting application, Java GUI is used to construct the foundation framework. Figure ** shows the structure of the coding, and the key java files for the left canvas.

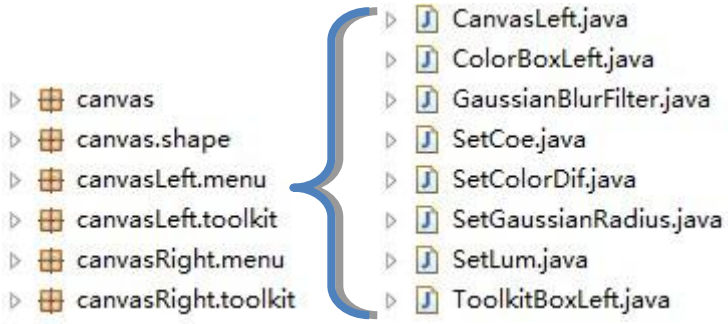


Fig 3.1 Implementation coding structure and some key Classes

3.3.2 Painting process

The painting process for this HDR painting application is as follows:

1. Draw or load an ordinary image on the right canvas.
2. Synchronize the area of the image where the luminance needs to be modified from the right to the left canvas.

3. Modulate the luminance in different areas of the image in the left canvas with user defined luminance and optional effects.
4. Set the linear scaling coefficient for the mean luminance for the result image.
5. Merge these two images into an HDR image.
6. Save the resulting image and write the image as a PFM file.

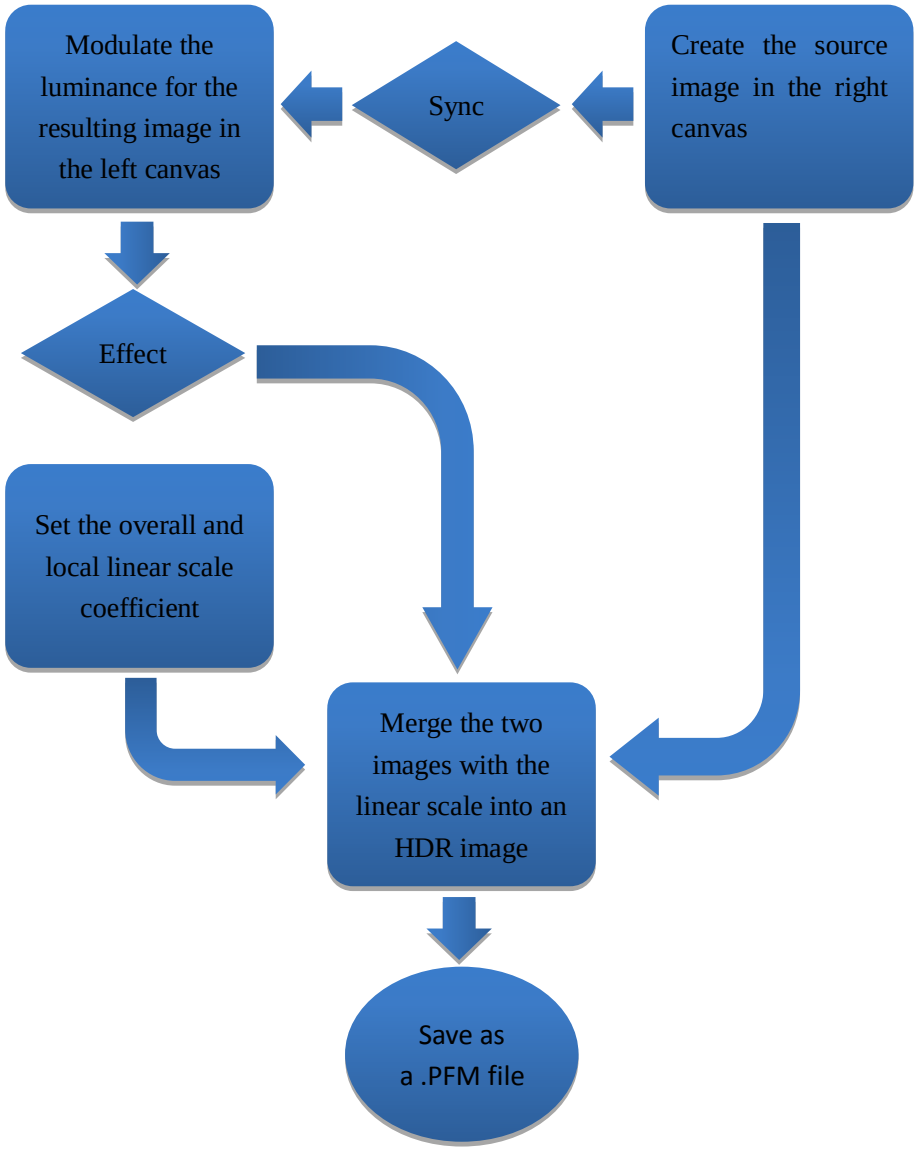


Fig 3.2 The painting process

3.3.3 User Interface design based on the user trials

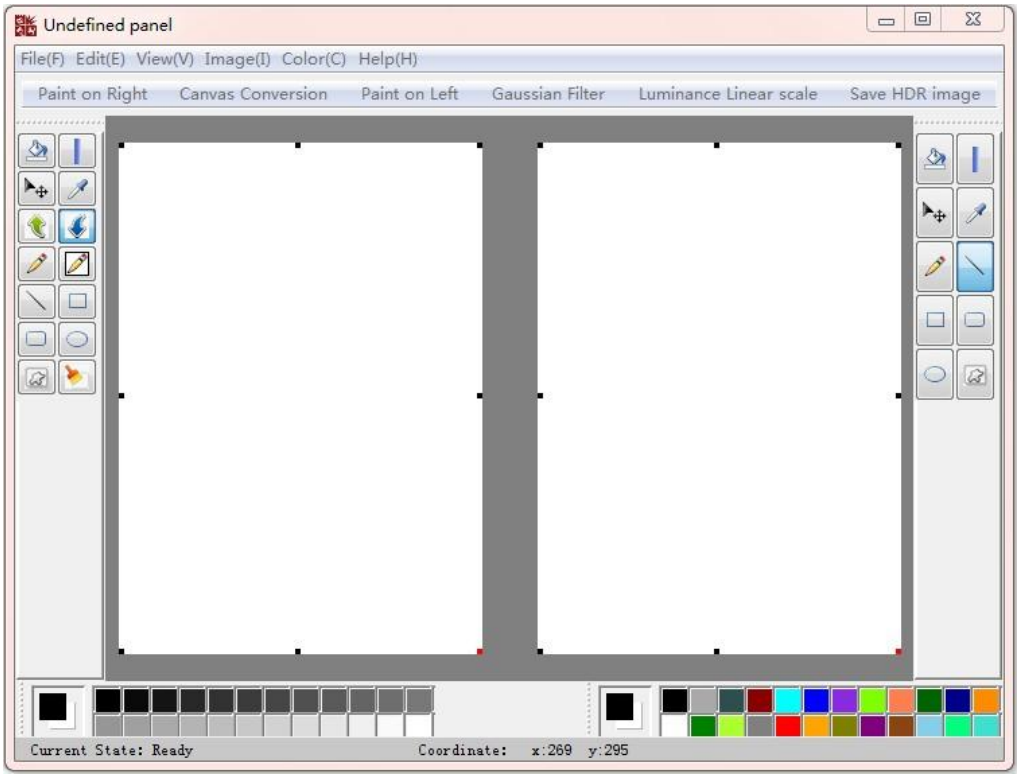


Fig 3.3 the interface of the application

The user interface of the HDR painting application is shown in Fig **.

In the center of the interface there are two canvases, the right one is used to paint the shapes and colors of an image, with the tool box on its right and a color box underneath it. The left one is used to manipulate the luminance corresponding to the different area in the image painted in the right canvas. The tool box and the gray color box next to it are provided for the toolkits for the painting on the left canvas.

The bottom of this user interface is a state box, which illustrates the current state or what kind of operation is being carried out, and also shows the coordinate of the cursor for the convenience of the users.

This user interface contains two menu bars at the top; the first is for the left canvas only. The second is the main operation menu for the HDR image painting.

By taking the user trials and the usability testing as an experimental investigation, the user interface improved significantly both in the understandability of the painting process and the simplification of the operation.

As it is a new HDR image painting panel without similar products, it is assumed that the user does

not have any clue about how to use this application. Based on this assumption, all the painting operations will be locked first, and an orientation window will pop up once the user click on either of the two canvases.

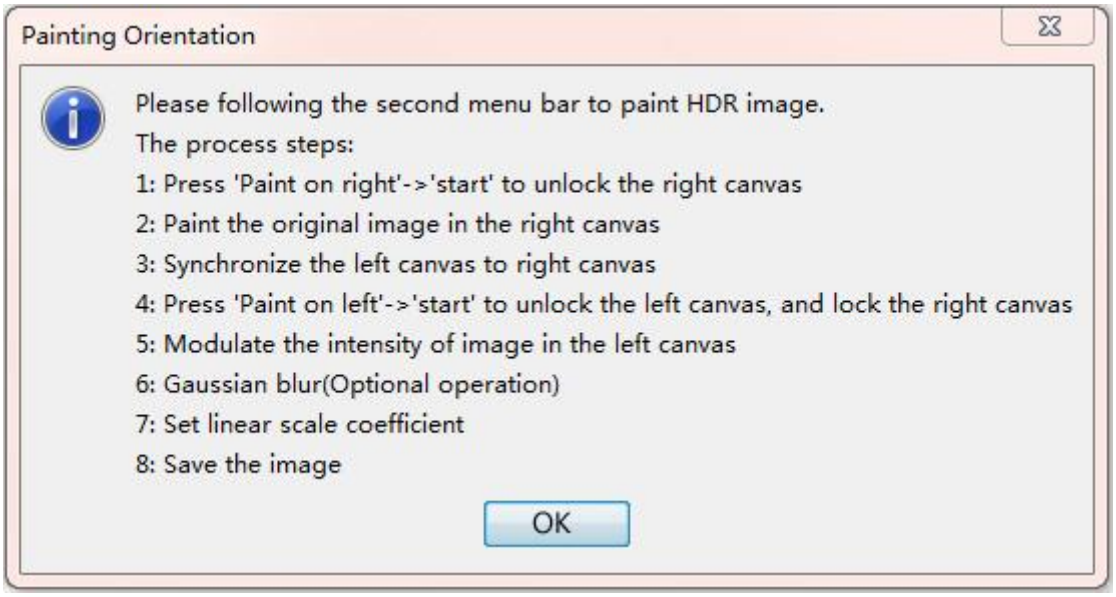


Fig 3.4 painting orientation window

In order to make this HDR painting application fully understandable and easy to use, the second menu bar illustrates the painting process.

Fig 3.5 the process menu

Following the painting orientation, the first operation required is to press the start button in “Paint on right” menu, and then the painting operations on the right canvas are unlocked; This allows the user to paint the shapes and colors, while the operations are still locked to prevent confusing operation in the proper sequence during the painting process.

Once finishing the canvas conversion, the left canvas needs to be unlocked. This would be achieved by pressing the “start” button in the “paint on left” menu; also, as with the previous painting, the right canvas is locked for preventing the wrong operations.

The luminance linear scale masks should be added only after the completion of the painting on both canvases. In this case, the two options here are disabled until two images are saved by pressing the button “ save images for merging”.

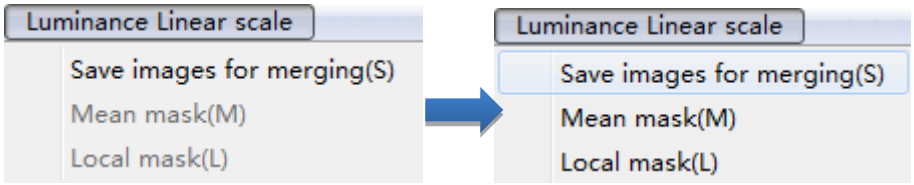


Fig 3.6 luminance linear scale options, press “Save images for merging” to unlock the Mean mask and Local mask operations

3.3.4 Paint on the right canvas

Paint on Right: The first step in constructing an HDR drawing is to do a normal drawing on the right canvas. This step is like the other paint panel, and some commonly used operations are provided.

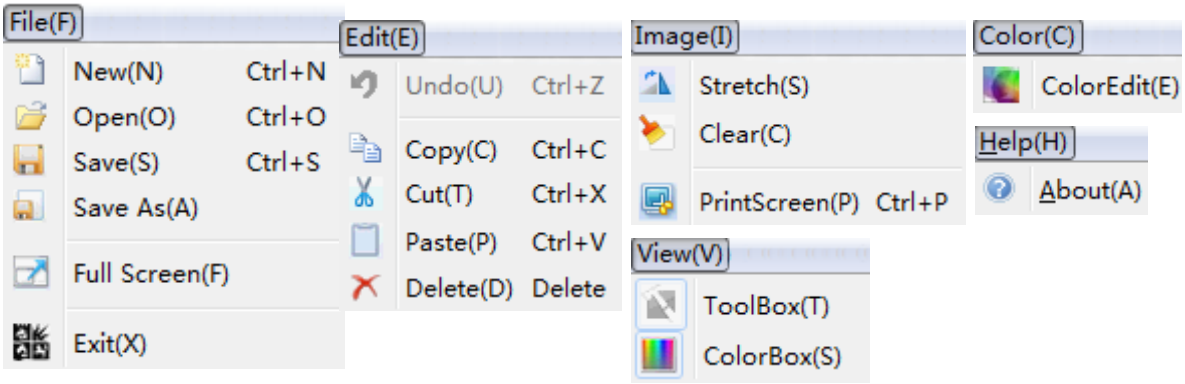


Fig 3.7 the function list of the left canvas

The user can either draw an original image or upload an existing image for the following operation.

3.3.5 Canvas Conversion

When finishing the drawing in the left canvas, the next step is to set the luminance value for the image. The Grayscale image is used for the editing of the luminance value, the larger the intensity of the grayscale color has, and the brighter the result image will be.

To precisely modulate the luminance of the image, the part of the image where the luminance needs to be edited should be transferred to the left canvas first, and then with the exact location the luminance can be changed with different operations.

Two kinds of method for convert the image from the right canvas to the left canvas are provided.

(1) The default conversion operation: Convert the whole image painted in the right canvas into a Grayscale image, and load it into the left canvas directly.

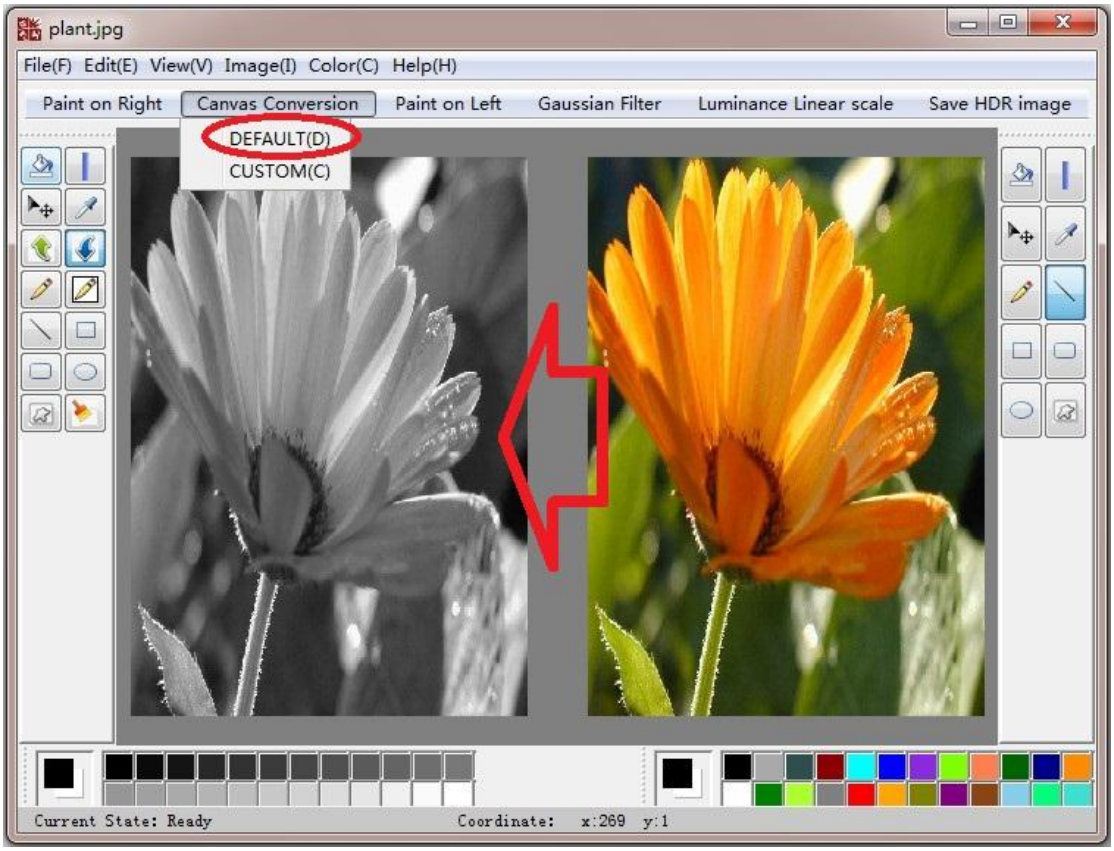


Fig 3.8 Default Conversion

The default canvas conversion is suitable for the luminance editing of the existing picture.

(2) The custom conversion operation: Convert the required parts of the image only.

Custom canvas conversion could be very useful for the luminance editing of the original painting. The image drawn in the right canvas consists of a sequence of different shapes. When using the custom canvas conversion, the selected shape will be synchronized into the left canvas as a shape too, which means this shapes could also be selected or changed its grayscale intensity.

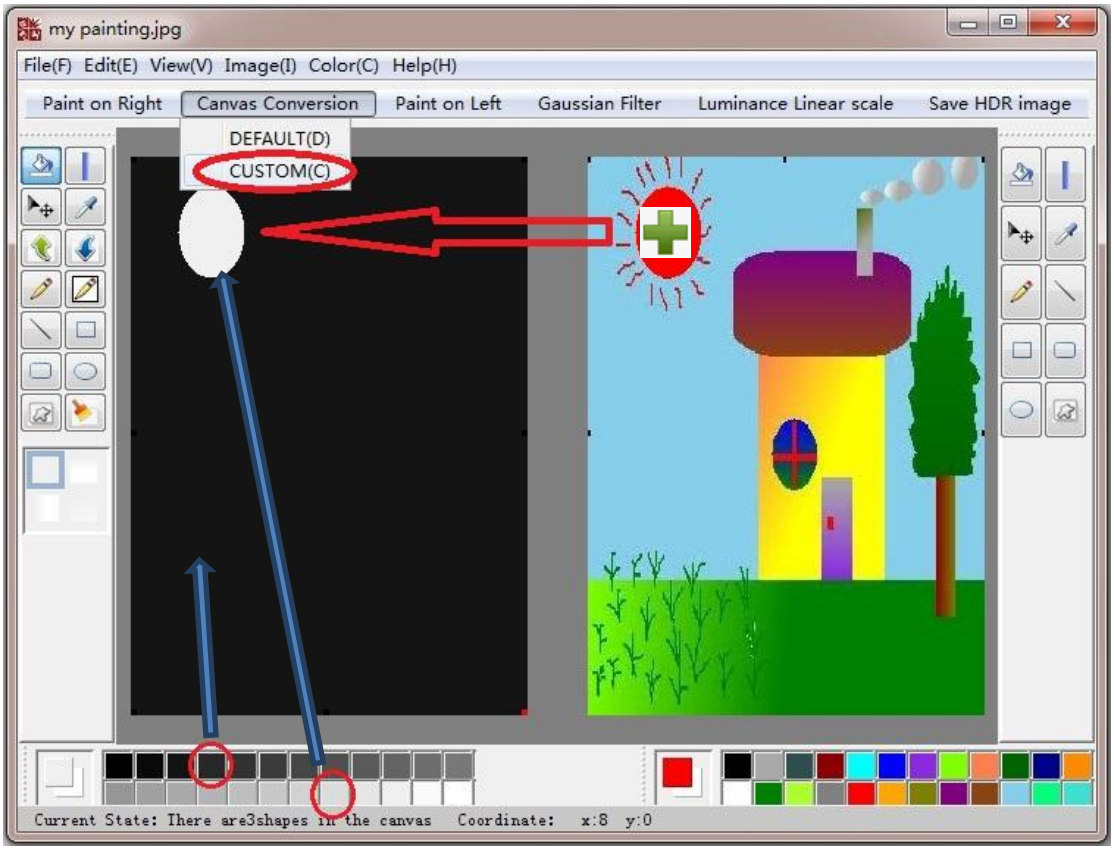


Fig 3.9 custom canvas conversion

The fig ** shows one example of the use of custom canvas conversion, the sun in the right canvas is chosen and synchronized to the left canvas. The following operation will be the luminance setting, for this specific image, the sun in the left canvas could be painted with a high intensity gray value, with the other parts of the image being assigned relatively dark value; Thus, in the resulting image which is constructed from two images in the left and right canvas, the luminance contrast between the sun and the rest of the image will be increased.

3.3.6 Luminance modulation

As introduced in section 3.2 and section 3.3.2, the algorithm breaks down the luminance modulation into a sequence of stages. After a conversion from the right canvas to the left canvas, the shapes whose luminance required to be manipulated are attained. Then the painting process moves to the third stage, modulating the luminance via a manipulation of an intensity of the converted image on the left canvas.

(1) Intensity chooser

In order to manipulate the luminance for the resulting image, the left canvas is allowed to paint directly with the custom intensity using a set of painting tools. The

intensity box on the bottom of the left canvas provides the some samples of gray scaled intensities, ranging from 0 to 255, and the chosen intensities of the foreground and the background are shown in the left of the intensity box.

Fig 3.10 Intensity box

Alternatively, by double clicking the intensity box, an intensity setting window will pop up, which allows the user to choose a specific intensity with a value in the range of 0~255 by dragging the slider..

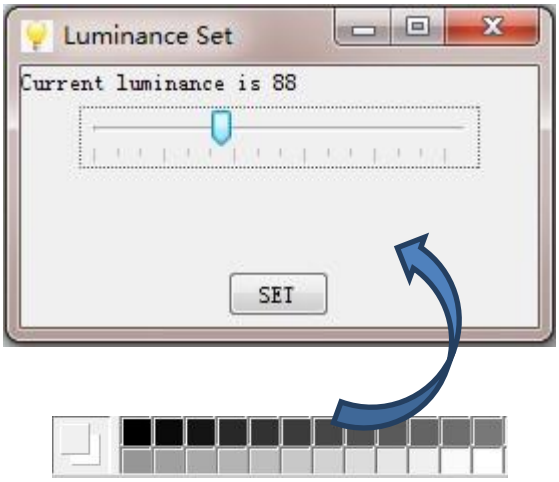


Fig 3.11 Intensity Setting window (pops up when double clicks the intensity box)

(2) Painting toolkit.

The toolkit box on the left side offers the painting tools for the manipulation on the left canvas. Besides the ordinary painting tools, this toolkit introduces a set of new tools exclusively for this stage of the luminance modulation.

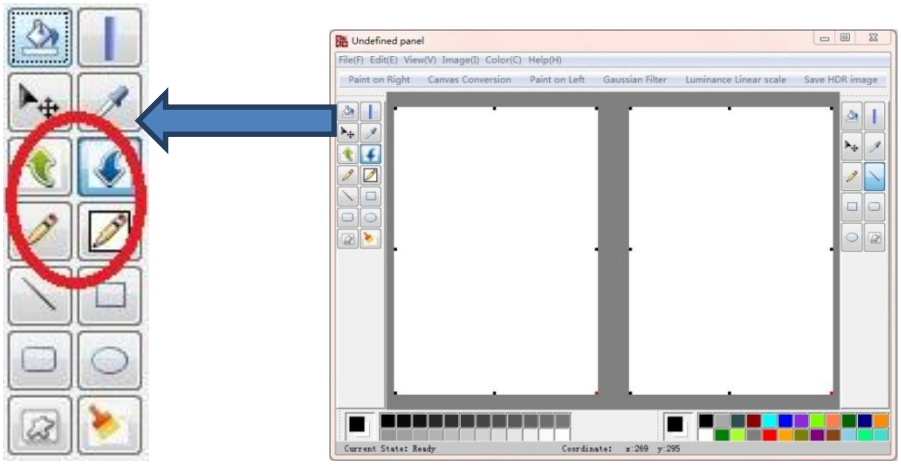


Fig 3.12 Toolkit for the left canvas

Figure 3.12 illustrates the new tools: intensity increase, intensity decrease, limited pencils.

(1) Intensity increase/decrease

The green rising arrow and the blue falling arrow are designed as a pair of tools to increase and decrease the intensity of the shapes where cursor clicks.

To implement these, once the mouse clicks on the left canvas, the coordinate of the pixel on which the mouse clicking happens will be stored and used to check whether this pixel is inside a shape. If positive, the intensity of the pixel will be increased or decreased for a step (step is set to 5 according to the user trials), and then repaint the shape containing the very pixel to update the whole image.

These two tools offer an intuitive way to manipulate the intensity value of shapes in the left canvas, which makes the modulation process much more convenient for the users.

(2) Limited pencil

The effect of the limited pencil is that it is only allowed to paint on certain limited area while dragging the mouse. The design is that once the mouse is being pressed, the pixel intensity under the cursor will be recorded as the source value, and while dragging the mouse, the pixel intensity under the cursor will be compared with the source value; The painting operation is permitted if the difference between the current pixel intensity value and the source value is within the permissible values, otherwise the operation will be denied. The permissible range is set when choosing the limited pencil as illustrated in figure 3.13.

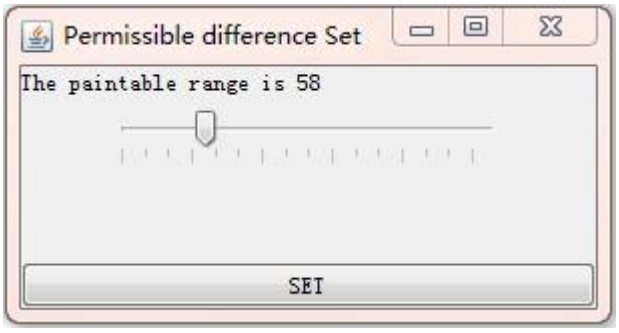


Fig 3.13 permissible difference set window

3.3.7 Gaussian Filter

Apart from the basic functions for the HDR painting application, a Gaussian Filter is added as an additional option.

Gaussian function in one dimension:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

Gaussian function in two dimensions:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.2}$$

Because of the linearly separable fact of Gaussian blur, the Gaussian blur can be applied to a two-dimensional image as two independent one-dimensional calculations besides being circularly symmetric. In this case, this application implements a series of dimensional Gaussian blur in the horizontal and the same process of Gaussian matrices in the vertical to achieve the effect of the two dimensional Gaussian blur. With the different times between horizontal and vertical Gaussian blur, or with different radius between horizontal and vertical process, the Gaussian blur effect will be more flexible in two directions.

The painting application provides three operations about the Gaussian filter:

(1) The horizontal Gaussian blur
(2) The vertical Gaussian blur
(3) Modify he radius of the Gaussian filter.

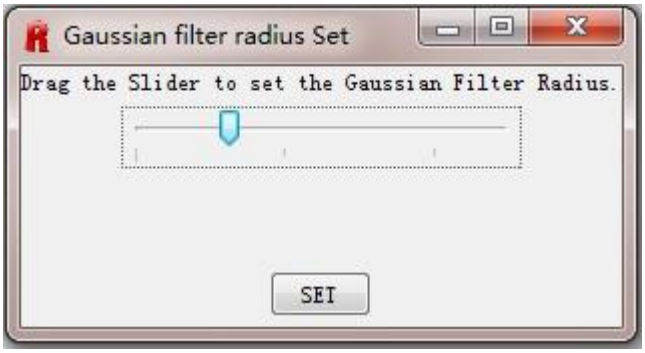


Fig 3.14 Gaussian filter radius set

In fact, with the Gaussian blur effects, some particular image will have a much better representation. When we paint something very bright, we normally want the object to have a shiny visual effect. For instance, the sun in sky, the shiny stars in the night sky or the light in the dark area, the Gaussian blur on these objects will give them the effects of a halo, which gives the whole image an interesting and

sensible representation.

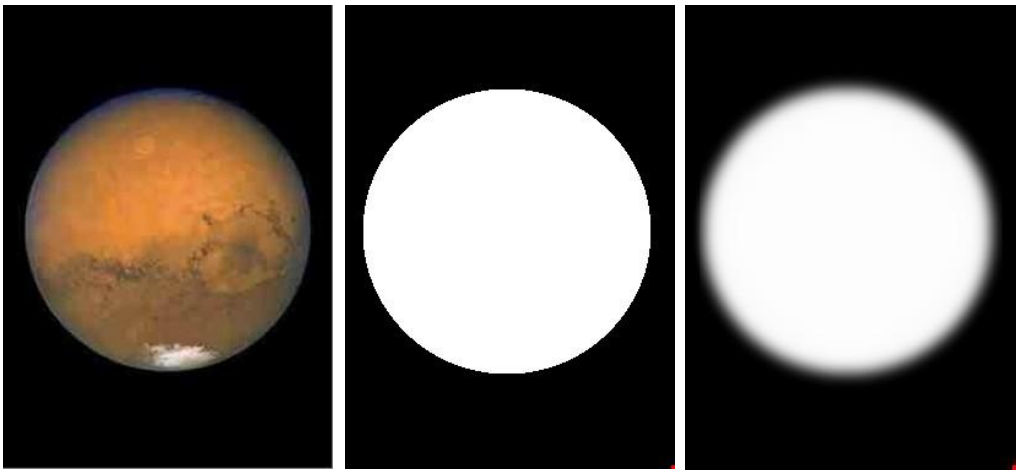


Fig 3.15 Gaussian filter for the effect of shininess

After the completion of the painting on the left canvas, the Gaussian Filter is also a considerable option to smooth the drawing on the left canvas. The cursor based drawing on the left canvas could be relatively rough especially when painting with the pencil tools. Normally, the drawing should be along the outline of or within the synchronized shapes from the right canvas, but the hand-painting may go slightly out of the boundary of the shapes, which leads to making the accuracy of the luminance modulation functioning at a low level.

To solve this problem, the Gaussian Filter could be used to smooth the roughly drawn part in the left canvas.

3.3.8 Luminance linear scale Mask

The luminance linear scale is considered as a fairly important step in the painting process. At this stage, the painting application provides two luminance linear scale masks, which are used in the final merge stage to scale the luminance.

The application provides two linear scale masks:

(1) The global mask

The global mask is used to linear-scale the overall luminance of the result image by setting a global coefficient, which is for the merging operation of the two images on the left and right canvases.

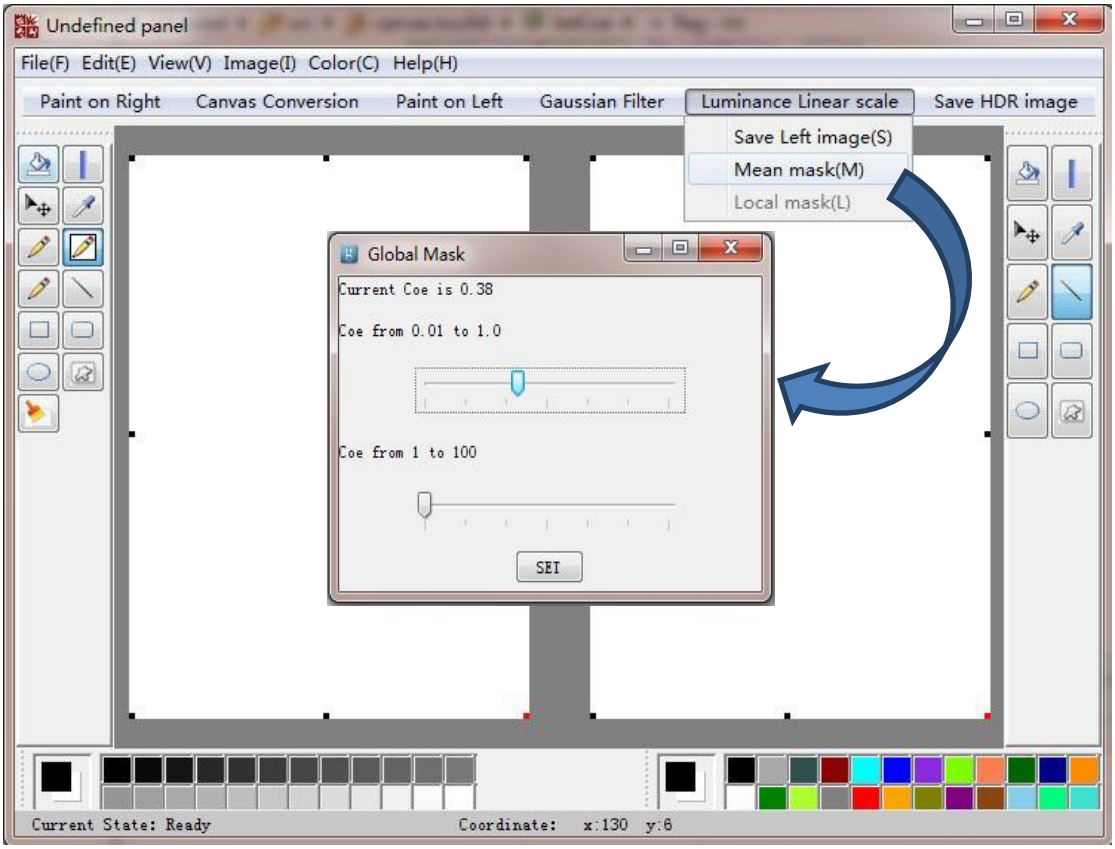


Fig 3.16 Global Mask

(2) The local mask

The local mask is used to linear-scale the luminance of the selected area of image by setting a local coefficient, which also works as part of the merging operation in the last step of the process. After setting the coefficient value in the setting windows, an area or a shape need to be selected to be assigned with the local linear scale value by click on the left canvas.

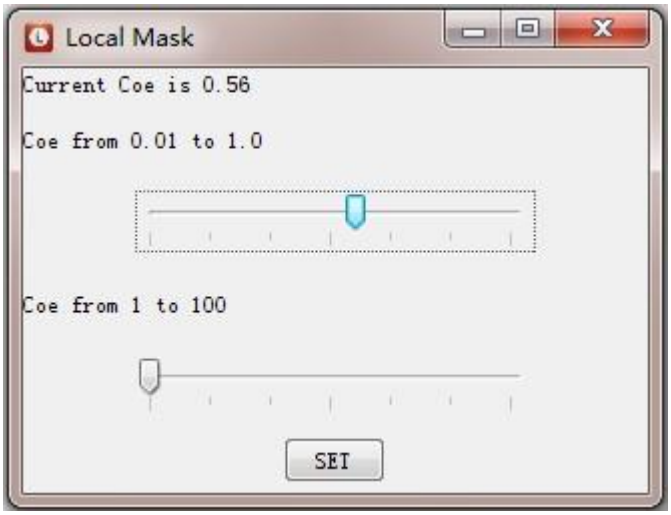


Fig 3.17 Local Mask

3.3.9 Merging of two images

When completing the painting on both canvases, the remaining work in the HDR image painting process is the merging of the two images.

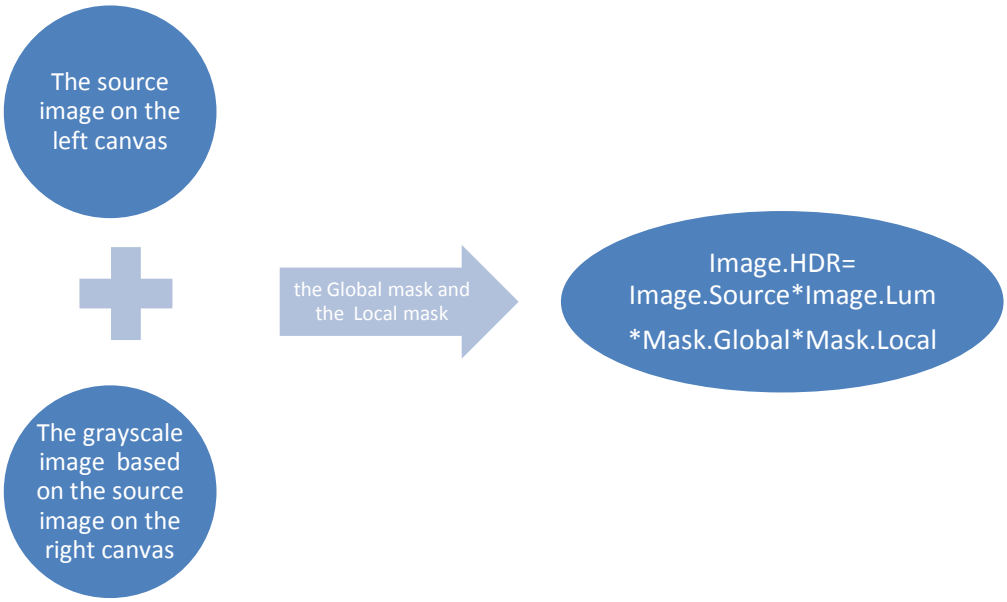


Fig 3.18 flow chart of merging of two images

For each pixel in the result image:
R.HDR = R.source*Lum.gray* LocalMask* GobalMask;
G.HDR = G.source*Lum.gray* LocalMask* GobalMask;
B.HDR = B.source*Lum.gray* LocalMask* GobalMask;

3.3.10 Save in the HDR format

The last step of the HDR image painting process is to save the resulting image in a HDR format. This painting application provides .pfm file format in which to store the image.

The PFM (Portable Float Map) [24] format evolved from the Net pbm format. The PFM format has many similarities with Net pbm format, though it is an unofficial extension of the Net pbm format that can support HDR. Besides, the goals of PFM format do not consist with those of the Net pbm format. [24][25][27]

A PFM image is a stream of bytes, consisting of a header followed by a raster. The structures of the two parts are described in detail as follows.

1. PFM header
   The PFM header consists of three consecutive lines of ASCII text, separated by a white space character between each two lines. Normally, the white space character is typically a newline character, which explains the term ‘line’.

2. Identifier Line
   The identifier line may contain the characters ‘PF’ or ‘Pf’. The ‘PF’ means that the image is a color PFM, whilst ‘Pf’ means the image is a grayscale PFM.[27]

3. Dimensions Line
   There are two positive decimal integers in the dimensions line, separated by a blank. The first integer represents the width of the image, and the second one represents its height (both are in pixels). The following three values represent the width, height, and aspect ratio in pixels respectively. These values may be placed on one, two or three lines, as there is not an agreed standard to define it except that a newline character is located at the end of the third value. The aspect ratio is floating point (usually 1) but also encodes the endian status of the following floating point binary numbers. A negative aspect ratio indicates little endian, or big endian.[27]

4. Scale Factor / Endianness
   The Scale Factor / Endianness line is a queer line that jams endianness information into an otherwise sane description of a scale. the Scale Factor / Endianness line contains a non-zero decimal number, which is not necessarily an integer. If the number is positive, the PFM raster has big endian, while on the other hand, it has little endian. The absolute value of the number represents the scale factor of the image.
   The scale factor also indicates the units of the samples contained in the raster. It is of great meaning to employ the scale factor together with some well understood unit information to endow a sample value with something meaningful, such as watts per square meter.[25]

5. PFM raster
   The raster consists of a sequence of pixels without any delimiters. They are arranged in the standard Western reading order. That is from left to right and from top to bottom. [25]
   There are one or three samples in each pixel, and as stated above, there are no delimiters in it. One sample is used for the grayscale PFM and three samples are used for the color PFM (see the section of Identifier Line).
   Each sample contains four bytes that represents a 32 bit string. The string is in an endian format, and the scale of the endian format depends upon the Scale Factor/ Endianness line of the PFM header. The string is a code of IEEE 32 bit floating point number. Considering that it has the same format that CPUs and compiler

mainly use, it is possible for program to be made to apply these bytes directly as a floating point number, on condition that the endianness variation is considered carefully.

6. Pixel data
   The pixel data is saved in the form of width*height floating point r, g, b, or as single grey scale values, which is then encoded as the IEEE floating point values in specified endian status.[27]

4. The performance measures

4.1 Overview

In this chapter, two kinds of HDR image creation tests have been carried out for the performance measures. The first test is to draw an original image from scratch and makes it an HDR image; the second one is to construct an HDR image with an existing LDR source image.

Because the visualization is not the included in the main task for this project, the visualization module is not provided in this HDR painting application. In the purpose of the performance measures, Adobe Photoshop CS5 is used to view the resulting image.

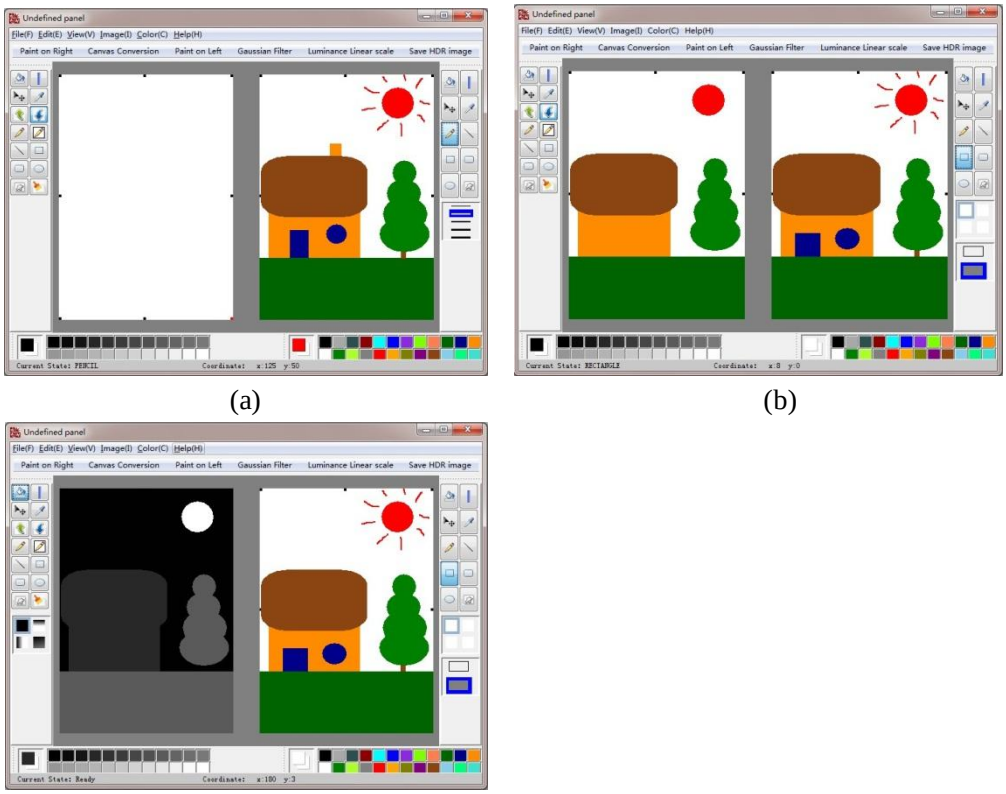
4.2 Paint an HDR image

In this test, firstly, an original drawing will be done on the right canvas, then the intensity value of each area of the drawing will be modulated on the left canvas. And an HDR image with the dynamic range out of the LDR displayable range would be created in the end.

The painting process is introduced in the following steps:

1. Select “Paint on right” → “Start(S)” to unlock the right canvas, thus the painting operation is permitted on the right canvas.
2. Draw an ordinary LDR image in the right canvas as the start point for the HDR image creation.
3. Select “Paint on right” → “Save the right image” to save the image in the right canvas as the source image.
4. Synchronize the right and left canvases. For this test, the custom synchronization is chosen. Select “Canvas Conversion” → “CUSTOM(C)”, and add the shapes whose luminance need to be modulated into the left canvas by double clicking on the right canvas,.
5. Select “Paint on left” → “Start(S)” to allow the manipulation operations for the left canvas, and lock the right canvas at same time to prevent incorrect manipulation.
6. Use painting tool from the toolkit on the left to modulate the intensity with the self-defined value selected from intensity box. To make the resulting image with a large luminance contrast, the intensity of sun is assigned a large value, while the rest part of the image is painted with a much lower value.
7. In this case, the Gaussian blur is not required, and the Local and Global masks are set as default.

8. Select “Save HDR image”→ “.pfm” to save the resulting image in pfm file format.

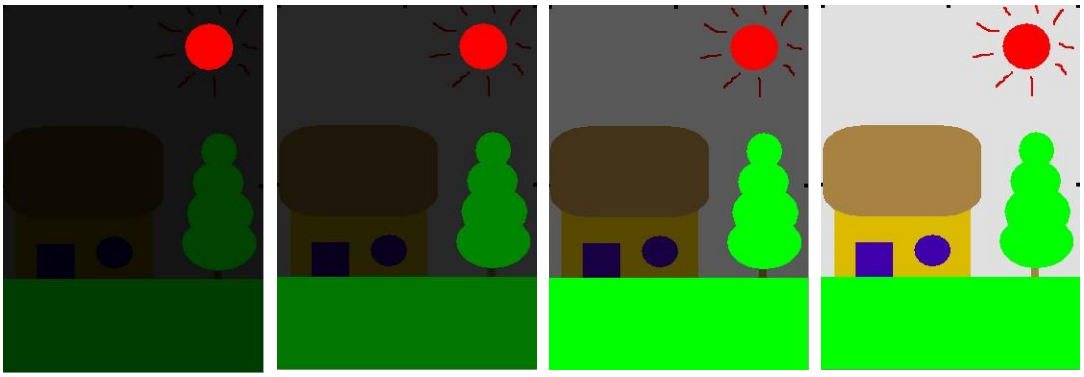


(a) (b)

(c)

Fig 4.1 the painting process to draw an HDR image

To view the resulting HDR image, Adobe Photoshop CS5 is used for the visualization. In order to represent the high luminance contrast of this HDR image, the exposure value is modulate to linear scale the mean luminance of this HDR image. As demonstrated in figure 4.1, the resulting image has a very large luminance range out of the LDR displayable range.



(a) (b) (c) (d)

Fig 4.2 the resulting HDR image with different exposure values to linear scale the mean luminance.

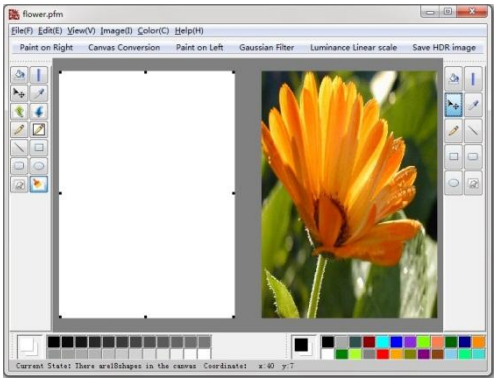
In the figure 4.2, the exposure value is increasing from picture (a) to (d). In the picture (a), the luminance of the image has been linear scaled down to make the flower displayable in the LDR displayer. The image apart from the sun is too dark to see the details because the dynamic range of the resulting image is nearly out of the range of the LDR device. With the increment of exposure value, the average luminance of the whole image becomes greater. However, there is no difference of the sun in four pictures because the luminance of the sun is out of the range of the displayer.

4.3 The creation of HDR images with LDR images

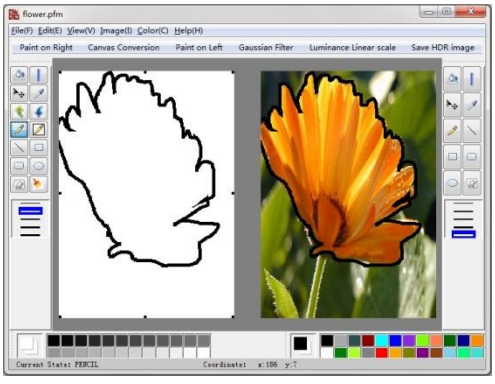
In this test, an existing ordinary LDR image is used as a source image to create a HDR image with the luminance contrast out of the range of the LDR displayer. Here, a simple image of a flower is taken as the source image.

The painting process could be divided into five main steps:

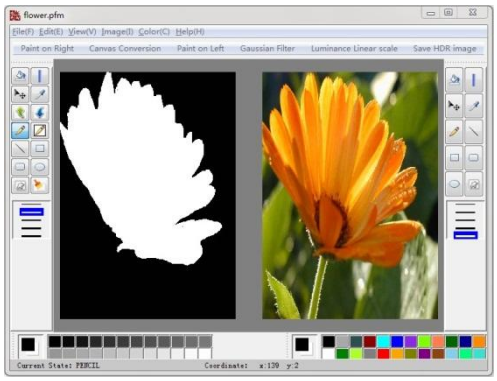
1. Load the existing ordinary LDR image into the right canvas as the start point for the HDR image creation.
2. Highlight the area where need to be manipulate the luminance. In this case, the flower is the outlined.
3. Synchronize the right and left canvases. For this test, the custom synchronization is chosen. Select “Canvas Conversion” → “CUSTOM(C)”, and add the line along the flower, which is drawn in step 4, into the left canvas.
4. Use painting tool from the toolkit on the left to modulate the intensity with the self-defined value selected from intensity box. To make the resulting image with a large luminance contrast, the intensity of flower shape is set as a large value, while the rest part of the image is paint as a much lower value.
5. Select “Save HDR image”→ “.pfm” to save the resulting image in pfm file format.



(a)



(b)



(c)

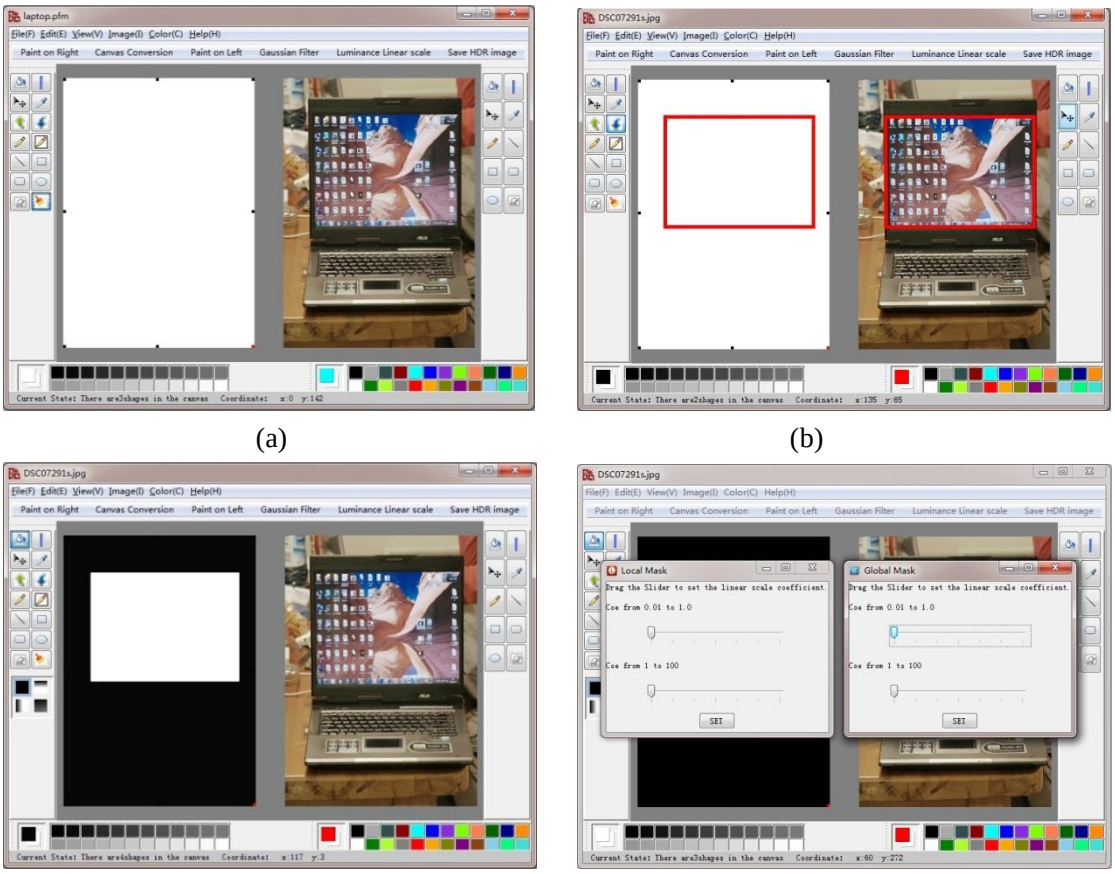
Fig 4.3 Painting process for creation of an HDR image with a LDR picture.

(a) (b) (c) (d)

Fig 4.4 the resulting HDR image with different exposure values to linear scale the luminance.

From picture (a) to (d) in figure4.4, the exposure value is from low to high. In the picture (a), the luminance of the image has been linear scaled down to make the flower displayable in the LDR displayer, however, because of the dynamic range of the resulting image is out of the range of the LDR device, the rest of the image are too dark to see the details. With the increase of the exposure value, the luminance value of the flower is out of the displayable range, thus the luminance difference cannot be distinguished between the pictures (c) and (d), while the luminance of the rest part of the resulting image is within the LDR displayable dynamic range.

Another similar test is carried out as the demonstration of the figure 4.4, besides the same content of the previous one, this test also tests the different resulting HDR images with different global mask and local mask. As demonstrated in picture (d) in figure 4.4, the local and global masks setting is added in the end of the painting process



(a) (b)

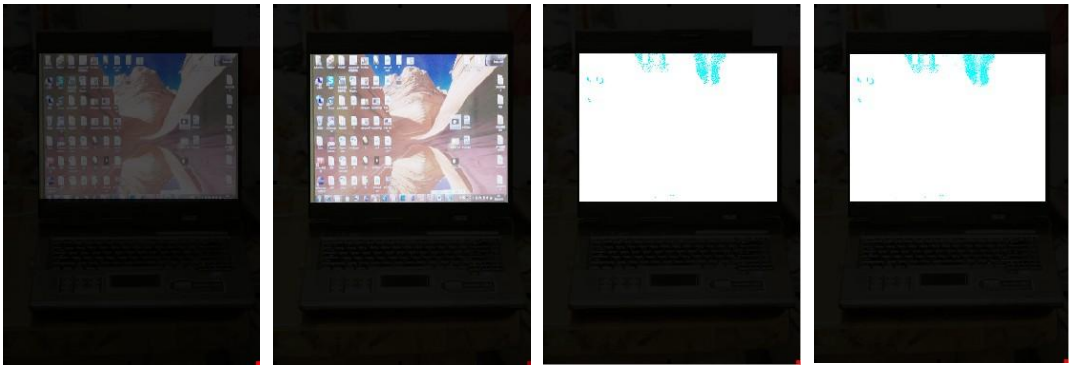
(c) (d)

Fig 4.5, Painting process to create HDR images with different mask setting.

According to the figure 4.6 and figure 4.7, the luminance contrast of the resulting image is out of the LDR displayable range. With the help of the local mask, the luminance difference between the lap-top screen and the rest part of the image could be scaled in a convieniant way.

(a) (b) (c) (d)

Fig 4.6, the resulting HDR image of lap-top with different exposure values to linear scale the luminance



(a) (b) (c) (d)

Fig 4.7 the resulting HDR images with different local mask with the same exposure

In the figure 4.7, different local mask coefficients are assigned to the tap-top screen, the value from picture (a) to (d) are 0.5, 1.0, 50, 100. It could be seen that, with the increasement of the local mask coeffiecient, the dynanmic range between the lap-top screen and the rest part of the image is increased. The difference between picture (c) and (d) cannot be distinguished because both of the tap-top screen luminance are out of the displayable range.

4.4 Summary

With lots of tests on this HDR painting application, it is found this application has reach the main requirements and the HDR image creation function gains an anticipated effect. The detailed evaluation will be given in the next section.

5. Conclusion

The first aim tackled in this project was to develop an intuitive method for painting images with radiance values that are outside the displayable range. Section 3.2 introduces the algorithm to create an HDR image with self-defined luminance. The core concept of this algorithm is that keep all the steps of the HDR image painting process displayable in the LDR devices by breaking down the painting process into several sub-processes. Based on this, the color and luminance of the Target image is manipulated separately in two synchronized images; both of these images are LDR displayable because the possible intensity of each pixel is within the range 0~255. After the color and luminance modulations have been completed, the resulting images of these two steps are merged, by multiplying the intensity of three channels in each and every corresponding pixel of these two images and constructing them as a 48 bits HDR image.

Based on the HDR image creating Algorithm, this project has developed an HDR image painting application, which provides an intuitive process to manipulate the luminance and contrast of HDR images. Aside from the implementation of the method to construct an HDR image, a focus of the project also concentrated on providing a variety of painting options.

Several special modulation operations and some brand new editing tools are introduced for this painting application as follows:

1. Two options are provided for the synchronization from the right and the left canvases. The default conversion synchronizes the entire image in the right canvas into the left canvas as a corresponding grayscale image, and the custom conversion allows the user to synchronize the selected shapes in the right canvas. These two synchronized options give a more flexible start for the following painting and editing.

2. Besides the ordinary image editing tools, this HDR painting application introduces a special pencil tool, and the difference from the normal pencil tool is that it only allows painting on a certain area on the image where the intensity difference with the source pixel is within user-defined intensity tolerance boundaries.

   This limited pencil tool has a much better performance than the general pencil tool, especially for this special editing case in the modulation of luminance on the left canvas.

3. In order to make the luminance modulation operates more conveniently and more precisely, a pair of intensity editing tools is provided. With the help of these tools,

the user can increase or decrease the intensity of the selected area of the image in the left canvas, not only has the option that repaint the selected area with the intensity value selected from the intensity box.

4. A Gaussian blur effect is added as an option to the painting process in order to offer a shininess visual effect for a better representation of bright objects such as the sun, or the light in the dark space, etc. The Gaussian blur on these objects gives them the effect of a halo, which provides the whole image with an interesting and sensible representation.

5. Two linear scale masks are provided as optional operations after the completion of the painting on the right canvas and the luminance modulation on the left canvas , global mask and local mask, are provided for the luminance linear scale, which are used to modulate the luminance of part or the whole image in the second modulation process.

As illustration and the evaluation in section4, it has been shown that the HDR image painting application provide a successful approach to create a 48 bits per pixel HDR image.

The last work in the HDR image painting process is to write up the resulting image, which created by multiplying the images in the left canvas and the right canvas, as a certain HDR file format. In this project, the HDR painting application choose .PFM file format to save the resulting HDR images.

To summarize this project, the background research mainly focused on the color space, basic image processing technology, HDR encoding, HDR capture, and tone mapping as well as the mains filter algorithm. In order to implement the algorithm to create an HDR image, the Java user interface programming and the GUI, which were new areas for the author, were also taken to be crucial for this project.

In this approach, the HDR image painting process is unprecedented and its ease of use for the HDR image painting application can be considered as a crucial achievement. Additionally, as the resulting image is stored as a 48bits file, it is the perfect material for any correction and further editing.

Another critical aspect of this project is the implementing of a good user interface for the painting and editing process. Thanks to the evaluation of the user trials, the design of the UI and the painting operations have made a remarkable improvement and it could be said that the painting process is now clearly guided and easy for operation.

Although most of essential work for this project was carried out, there is still much room for improvement. Most painting and editing operations are designed in a sensible way; and basically attain functioning requirements, although some of the

functions are still not very mature, and further work is required for some improvement of the performance.

The improvement of the painting tools and painting operations, as well as the expansion of the image formats could form the basis for future research. A detailed discussion about future work will be introduced in the section 6.

6. Future work

6.1 Overview

For this project, the essential aim and objects have been achieved, while as the development of this HDR painting application carries on, there are some ideas for the improvement appear. But because of time consuming, these ideas cannot be implemented during this period. The future work could focus on these area.

6.2Brush based on bilateral filtering

The HDR image painting application provides the ordinary pencil tool and the limited pencil for the manipulation of the intensity in the left canvas. Compared with the ordinary pencil, the limited pencil offers a more useful and reasonable way for the painting. But the performance is not quite satisfying when manipulating the region with the high contrast. The reason is that, it won't allow user to paint in the high intensity contrast area if the intensity tolerance is set to be a small value, the painting operation is permitted only when the tolerance value is big enough, but in this case, this kind of painting operation would makes the image lose the edge details, to solve this problem, one possible solution is using the bilateral filtering algorithm in the painting process.

Bilateral filtering is a technology usually used to smooth images via combination of the weighted average of nearby pixel values. In contrast with low-pass filtering like Gaussian filtering, it combines gray levels or colors not only according to their geometric closeness, but also the photometric similarity. [21]

Bilateral filter Operates both in the domain and the range of the image:

$$h(x) = k^{-1}(x)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\xi)c(\xi, x)s(f(\xi), f(x))d\xi \tag{6.1}$$

$$k(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c(\xi, x)s(f(\xi), f(x))d\xi \tag{6.2}$$

c (ξ , x) measures the geometric closeness between the neighborhood center x and a nearby point. s (f(ξ), f(x)) measures the photometric similarity between the pixel at the neighborhood center x and that of a nearby point.[]

The outline of the design of a brush tool with the bilateral filter algorithm is as

followed:

1. Get the source pixel information: Once the mouse pressed, the current pixel will be recorded as the source pixel, and its intensity value will be used as an original intensity value.
2. Allocate the weights for pixel: While dragging the mouse to paint with the user defined intensity value, the result intensity value will be determined by two factors, one is the intensity difference between the original intensity value of the source pixel and intensity value of the current pixel under the cursor; another one is the distance from the current pixel to the source pixel. The greater the similarity and the smaller the distance are, the greater the weight is given to the pixel.
3. Convolution with the user defined intensity and the corresponding pixels' weights.

With this kind of bilateral filter brush, the effect is that once the user picks an intensity value and paints on a certain region on the image, the first spot will be repaint with the user defined intensity value as the mouse pressed, and while dragging the mouse, the neighborhood of the first spot will be repaint with an intensity value calculated by the above algorithm. Thus, the intensity value will be adjusted automatically by the photometric similarity and geometric closeness between the current pixel and the source pixel.

The brush tool with the bilateral filter algorithm would allow the user to modulate the intensity value of an area on the image with the self-defined value without losing the edge details. The intensity modulation becomes more efficient and has a much better performance.

6.3 More file formats for saving HDR image

In this project, .pfm (Portable Float Map) is provided as the file format to save the HDR images. As PFM is an unofficial extension to the .pbm image format collection that supports HDR imaging, it could make a great improvement for this paint application to provide more commonly used HDR file formats. The extension about adding the main HDR file formats like .hdr and .pic is considered as a valuable future work.

6.4 HDR image visualization

Although the HDR image visualization is not included in the objects of this project,

based on the sufficient background research on tone mapping for this project, it could be an exciting extension to let the application has the ability to open up an HDR image and do some basic visualization as well as some manipulating.

The implementation of the novel approach introduced in this project for the HDR image creation was completely accomplished in the HDR painting application, with the future extension on HDR visualization, it would makes the application a fully capable HDR image editor, from manipulation, creation, visualization.

Reference

[1] J.A. Ferwerda. “Elements of Early Vision for Computer Graphics,” IEEE Computer Graphics and Applications. 21(5):22-33,2001.

[2] L.Spillmann and J.S. Werner (eds.). Visual Perception: The Neurological Foundations. San Diego: Academic Press, 1990.

[3] B.A.Wandell. Foundations of Vision. Sinauer Associates, 1995.

[4] High Dynamic Range Images.

[5] J. E. Dowling. The Retina: An Approachable Part of the Brain. Cambridge, MA: Belknap Press, 1987.

[6] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. High Dynamic Range Imaging. 2006.

[7] G.Ward-Larson and R. A. Shakespeare. Rendering with Raiance. San Francisco: Morgan Kaufmann, 1998.

[8] F. Kainz, R. Bogart, and D. Hess. “The OpenEXR Image File Format,” in SIGGRAPH Technical Sketches, 2003.

[9] Alessandro Rizzi, Carlo Gatta, Benedett Piacentini, Massimo Fierro, Daniele Marini “Human visual system inspired tone mapping algorithm for HDR images”, Via Comelico 39, 20100 Milano (Mi), Italy

[10] Fr´edo Durand, Julie Dorsey “Fast Bilateral Filtering for the Display of High-Dynamic-Range Images”, Laboratory for Computer Science, Massachusetts Institute of Technology

[11] Ward, Gregory, High Dynamic Range Image Encodings, Anyhere Software

[12] Larson, G.W., Overcoming Gamut and Dynamic Range Limitation in Digital Images, Proceedings of the Sixth Color Imaging Conference, November 1998

[13] Technical Introduction to OpenEXR, Industrial Light & Magic, 02/03/05. www.openEXR.org

[14] Ward, Gregory and Simmons Maryann, JPEG-HDR: A Backwards-Compatible, High Dynamic Range Extension to JPEG, Proceedings of the Thirteenth Color Imaging Conference, November 2000

[15] Patrick Ledda, Alan Chalmers, Helge Seetzen. “A Psychophysical Validation of Tone Mapping Operators using a High Dynamic Range Display”

[16] Erik Reinhard, Michael Stark, Peter Shirley, James Ferwerda, “Photographic Tone Reproduction for Digital Images”

[17] Patrick Ledda , Alan Chalmers , Tom Troscianko , Helge Seetzen, “Evaluation of Tone Mapping Operators using aHigh Dynamic Range Display”

[18] Ward, Gregory J., "The RADIANCE Lighting Simulation and Rendering System," Computer Graphics (Proceedings of '94 SIGGRAPH conference), July 1994, pp. 459-72.

[19] Mittsunaga,T. and Nayar, S.K.2000. High dynamic range imaging: Spatially varying pixel exposures. In IEEE CVPR, 472–479.

[20] Fr ́edo Durand and Julie Dorsey. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. ACM New York, NY, USA

[21] TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In Proc. IEEE Int. Conf. on Computer Vision, 836–846.

[22]http://help.adobe.com/en_US/Photoshop/11.0/WS801F7CBF-DF09-4ed9-9B6A-52B241608E01a.html.

[23] Mathias Eitz. http://user.cs.tu-berlin.de/~eitz/bilateral_filtering/index.html. November, 21st 2006

[24] Paul E. Debevec. http://ict.debevec.org/~debevec/Probes/ 10/9/04

[25] http://netpbm.sourceforge.net/doc/pfm.html. 10 April 2004

[26] R. Gonzalez and R. Woods Digital Image Processing, Addison-Wesley Publishing Company, 1992, p 191

[27] http://local.wasp.uwa.edu.au/~pbourke/dataformats/pbmhdr/

Appendices: Source Code

In this part, some main method in the following file will be list.

ProcessMenuBar.java:

```
public void actionPerformed(ActionEvent e) {
       if (e.getActionCommand().equals("startR")) {
          canvasRgb.setPaintable(true);
       }
       else if (e.getActionCommand().equals("saveR")) {
          saveRightImage();
          canvasRgb.setPaintable(true);
          startL.setEnabled(true);
       }
       else if (e.getActionCommand().equals("default")) {
          System.out.println("default");
          System.out.println("SYN");
          /*
           * default SYN
           */
          Robot r = null;
          Point p = canvasRgb.getLocationOnScreen();
          int canvasWidth = canvasRgb.getWidth();
          int canvasHeight = canvasRgb.getHeight();
          try {
             r = new Robot();
          } catch (AWTException event) {
             event.printStackTrace();
          }

          tempImage = r.createScreenCapture(new Rectangle(p.x,
          p.y, canvasWidth,canvasHeight));

          BufferedImage biGray = new
          BufferedImage(canvasWidth,canvasHeight,BufferedImage.
          TYPE_BYTE_GRAY);
          biGray.getGraphics().drawImage(tempImage,0,0,null) ;
          CanvasShape cs = new CanvasImage(0, 0, canvasWidth,
          canvasHeight,biGray);
          canvasGray.shapes.add(cs);
          canvasGray.repaint();
```

```
} else if (e.getActionCommand().equals("custom")) {
   System.out.println("custom");
   canvasRgb.setCustomConvert(true);
   canvasRgb.toolBox.resetSelect();
}
else if (e.getActionCommand().equals("startL")) {

   int canvasWidthSrc = canvasRgb.getWidth();
   int canvasHeightSrc = canvasRgb.getHeight();

   CanvasShape cs = new CanvasImage(0, 0, canvasWidthSrc,
   canvasHeightSrc,canvasRgbImage);
   canvasRgb.shapes.add(cs);

   canvasGray.setPaintable(true);
   canvasRgb.setPaintable(false);
}
else if(e.getActionCommand().equals("radius")){
   new SetGaussianRadius(this);
}
else if (e.getActionCommand().equals("horizontal")) {
   /*
    * get left canvas image
    */
   Robot r = null;
   Point p = canvasGray.getLocationOnScreen();
   int canvasWidth = canvasGray.getWidth();
   int canvasHeight = canvasGray.getHeight();
   try {
      r = new Robot();
   } catch (AWTException event) {
      event.printStackTrace();
   }
   tempImage = r.createScreenCapture(new Rectangle(p.x,
   p.y, canvasWidth,canvasHeight));
   BufferedImage filteredImage =
   GaussianBlurFilter.getGaussianBlurFilter(radius,
   true).filter(tempImage, null);
   CanvasShape cs = new CanvasImage(0, 0, canvasWidth,
   canvasHeight,filteredImage);
   canvasGray.shapes.add(cs);
   canvasGray .repaint();
}
else if (e.getActionCommand().equals("vertical")) {
```

```
    /*
     *get left canvas image
     */
    Robot r = null;
    Point p = canvasGray.getLocationOnScreen();
    int canvasWidth = canvasGray.getWidth();
    int canvasHeight = canvasGray.getHeight();
    try {
        r = new Robot();
    } catch (AWTException event) {
        event.printStackTrace();
    }
    tempImage = r.createScreenCapture(new Rectangle(p.x,
    p.y, canvasWidth,canvasHeight));
    BufferedImage filteredImage =
    GaussianBlurFilter.getGaussianBlurFilter(radius,
    false).filter(tempImage, null);
    CanvasShape cs = new CanvasImage(0, 0, canvasWidth,
    canvasHeight,filteredImage);
    canvasGray.shapes.add(cs);
    canvasGray .repaint();
}
else if (e.getActionCommand().equals("saveGray")) {
    System.out.println("进入saveGray");
    /*
     * get left canvas image
     */
    Robot rSrc = null;
    Point pSrc = canvasGray.getLocationOnScreen();
    int canvasWidthSrc = canvasGray.getWidth();
    int canvasHeightSrc = canvasGray.getHeight();
    try {
        rSrc = new Robot();
    } catch (AWTException event) {
        event.printStackTrace();
    }
    canvasGrayImage = rSrc.createScreenCapture(new
    Rectangle(pSrc.x,pSrc.y, canvasWidthSrc,canvasHeightSrc));
    local.setEnabled(true);
    mean.setEnabled(true);
}
else if (e.getActionCommand().equals("mean"))
{
    new SetCoe(this, 0, "Global Mask");
```

```
} else if (e.getActionCommand().equals("local"))
{
    new SetCoe(this,1,"Local Mask");
    canvasGray.setLocalCoe(co);
    canvasGray.setLocalMask(true);
}else if (e.getActionCommand().equals("pfm")) {
    System.out.println("saveHDR");
    MASK = canvasGray.getMask();

    if(canvasRgbImage==null){
        /*
         * get the right canvas image
         */
        Robot rRgb = null;
        Point pRgb = canvasRgb.getLocationOnScreen();
        int canvasWidthGray = canvasRgb.getWidth();
        int canvasHeightGray = canvasRgb.getHeight();
        try {
            rRgb = new Robot();
        } catch (AWTException event) {
            event.printStackTrace();
        }
        canvasRgbImage =rRgb.createScreenCapture(new Rectangle(pRgb.x,
        pRgb.y, canvasWidthGray,canvasHeightGray));
    }

    if(canvasGrayImage==null){
        /*
         * get the left canvas image
         */
        Robot rGray = null;
        Point pGray = canvasGray.getLocationOnScreen();
        int canvasWidthGray = canvasGray.getWidth();
        int canvasHeightGray = canvasGray.getHeight();
        try {
            rGray = new Robot();
        } catch (AWTException event) {
            event.printStackTrace();
        }
        canvasGrayImage =rGray.createScreenCapture(new
        Rectangle(pGray.x, pGray.y,
        canvasWidthGray,canvasHeightGray));
    }
    float[] rgbGray = new float[3];
```

```
float[] rgbSrc = new float[3];
int pixelGray;
int pixelSrc;

int width=canvasGrayImage.getWidth();
int height=canvasGrayImage.getHeight();
int minx=canvasGrayImage.getMinX();
int miny=canvasGrayImage.getMinY();

float[][][] rgbHDR =new float[width][height][3];


/*
 * WRIHT PFM
 */
ByteArrayOutputStream baos = new ByteArrayOutputStream();

try {
    baos.write(("PF" + "\n").getBytes());
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
try {
    baos.write("# this is a comment\n".getBytes());
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
try {
    baos.write((width + " " + height + "\n").getBytes());
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
try {
    baos.write(("-1.000000\n").getBytes());
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

int bits;
//System.out.println("co is: "+co);
```

```
for(int i=height-1;i>=miny;i--){
    for(int j=minx;j<width;j++){
        pixelGray=canvasGrayImage.getRGB(j, i);
        rgbGray[0] = ((float)((pixelGray & 0xff0000 ) >> 16))/255 ;
        rgbGray[1] = ((float)((pixelGray & 0xff00 ) >> 8 ))/255;
        rgbGray[2] = ((float)(pixelGray & 0xff ))/(255);

        pixelSrc=canvasRgbImage.getRGB(j, i);

        rgbSrc[0] = ((float)((pixelSrc & 0xff0000 ) >> 16))/(255) ;
        rgbSrc[1] = ((float)((pixelSrc & 0xff00 ) >> 8 ))/(255);
        rgbSrc[2] = ((float)(pixelSrc & 0xff ))/(255);

        rgbHDR[j][i][0]=rgbSrc[0]*rgbGray[0]*MASK[j][i]*co;
        rgbHDR[j][i][1]=rgbSrc[1]*rgbGray[1]*MASK[j][i]*co;
        rgbHDR[j][i][2]=rgbSrc[2]*rgbGray[2]*MASK[j][i]*co;
        if((i==30)&&(j==30))
            System.out.println("mask[][] is "+ MASK[j][i]+"; CO is
            "+MASK[j][i]);

        /*
         * write pfm data
         */
        bits = Float.floatToIntBits(rgbHDR[j][i][0]);
        baos.write( bits        & 0xff);
        baos.write((bits >>  8) & 0xff);
        baos.write((bits >> 16) & 0xff);
        baos.write((bits >> 24) & 0xff);
        bits = Float.floatToIntBits(rgbHDR[j][i][1]);
        baos.write( bits        & 0xff);
        baos.write((bits >>  8) & 0xff);
        baos.write((bits >> 16) & 0xff);
        baos.write((bits >> 24) & 0xff);
        bits = Float.floatToIntBits(rgbHDR[j][i][2]);
        baos.write( bits        & 0xff);
        baos.write((bits >>  8) & 0xff);
        baos.write((bits >> 16) & 0xff);
        baos.write((bits >> 24) & 0xff);
    }
}

try {
    baos.close();
```

```
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}//baos close
byte[] bytes = baos.toByteArray();

/*
 * saving dialog
 */
JFileChooser fileChooser = new JFileChooser(".");
FileNameExtensionFilter filter = new FileNameExtensionFilter(
        "PFM file", "pfm");
fileChooser.setFileFilter(filter);
int returnVal =fileChooser.showSaveDialog(canvasRgb);

if (returnVal == JFileChooser.APPROVE_OPTION) {

    CanvasWindow.FrameOwn.setTitle(fileChooser.getSelectedFile()
    .getName());
    File destinyFile = fileChooser.getSelectedFile();

    try {
        //FileOutputStream fileOutputStream = new
        FileOutputStream("try.pfm");
        FileOutputStream fileOutputStream = new
        FileOutputStream(destinyFile);
        try {
            fileOutputStream.write(bytes);
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        try {
            fileOutputStream.close();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    } catch (FileNotFoundException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }
}
```

```
            canvasGrayImage=null;//clear the buffered image for next operation
            canvasRgbImage=null;//clear the buffered image for next operation

            System.out.println("ok!");
        }
    }
```