VFX High Dynamic Range Imaging Project Report

廖婕吟(R09922136) r09922136@csie.ntu.edu.tw

鄒汶橙(R09922131) r09922131@csie.ntu.edu.tw

Project Description

High dynamic range (HDR) images have much larger dynamic ranges than traditional images' 256 levels. It accurately represents the wide range of intensity levels found in real scenes ranging from direct sunlight to shadows. To produce an HDR image, we took a series of photographs for a scene under different exposures, then solved the response curve using Paul Debevec's method. This HDR image is finally synthesized using tone mapping methods to create an image with better visual effects.

The rest of the report is organized as follows. Section II introduces the algorithms we implemented. Section III presents the details of our implementations. Section IV shows the result of all the implementations.

Algorithms

I. Image Alignment

The purpose is to align multiple input photos to avoid the image's inconsistency at the same pixel due to vibration using **median threshold bitmap algorithm**.

1. Determine the median 8-bit value from a low-resolution histogram over the grayscale image pixels.
2. Create a bitmap image with 0's where the input pixels are less than or equal to the median value and 1's where the pixels are greater.
3. Create an exclusion bitmap that exclude the pixels that the values are close to our threshold by zeroed all bits where pixels are within ± 4 of the median
4. Computing an image pyramid for each grayscale image exposure, with log2(max offset) levels past the base resolution.
5. Start with the lowest resolution MTB pair and compute the minimum difference offset between them within a range of ± 1 pixel in each dimension. At the next resolution level, we multiply this offset by 2 (corresponding to the change in resolution) and compute the minimum difference offset within a ± 1 pixel range of this previous offset. This continues to the highest (original) resolution MTB, where we get our final offset result.
6. We need to compare exactly 9 candidate MTB offset at each level, taking the difference of these two bitmaps with an exclusive-or (XOR) operator shows where the two images

are misaligned. Then AND'ing it with both offset exclusion bitmaps to compute our final difference.

II. HDR

Using multiple photographs of the scene taken with different amounts of exposure to recover the response function of the imaging process. With the known response function, we can fuse the multiple photographs into a single, high dynamic range radiance map whose pixel values are proportional to the true radiance values in the scene.

1. Read 16 images with different exposure times and use an array to store each photo's exposure time.
2. Set the weight of constraint λ=5, define function w to give intensity 0-255 different weight
3. Define gslove(slove for imaging system response function) as the reconstruction response curve g, and the algorithm used here is the **Debevec's method**
4. Calculate each pixel's real exposure value in the image through the weighted average and the g obtained in Step 3. Because the calculation is done in the log domain in the mathematical derivation, the value calculated in Step 3 must be restored through math.exp(). All three channels of BGR do the above steps and finally get output
5. Calculate each pixel's real exposure value in the image through the weighted average and the g obtained in Step 3. Because the calculation is done in the log domain in the mathematical derivation, the value calculated in Step 3 must be restored through math.exp(). Do the above steps for all three channels of BGR, and finally get output.

III. Tone Mapping

Tone Mapping is the mapping of the potentially high dynamic range of real world luminances to the low dynamic range of the photographic print. By using this method we convert HDR images to LDR images. The algorithm here uses **Photographic Tone Reproduction** which is more robust in most scenarios.

1. First define the global operator, and linearly scale the brightness of the pixel value of the HDR image. Use parameter a to pull the pixel value back to 0-1, which indirectly indicates 0-255. However, pixels with too much brightness above the average are prone to overexposure, so it is necessary to use a non-linear factor to correct it. L_{white} is the minimum brightness non-linear factor to be mapped to 1, but such an operator will cause a lot of details to disappear.
2. Define the local operator. The main purpose of defining local operators is to solve the situation that the global operator will lose the images details, using the dodying-and-burning method in the reference paper. The method is roughly the same as the global approach, the difference is adding an Ls variable, and using a Gaussian filter to do convolution on Lm, and finding the largest flat area for processing, make the originally bright

place brighter, and the dark place darker. Define the local operator using the dodying-and-burning method in the reference paper. The main purpose of defining local operators is to solve the situation that the global operator will lose the images' details. The method is roughly the same as the global approach, the difference is adding an Ls variable, using a Gaussian filter to do convolution on Lm and finding the largest flat area for processing makes the originally bright pixels brighter and so as dark pixels.

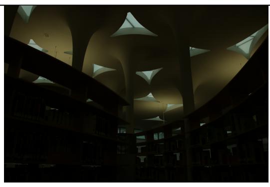
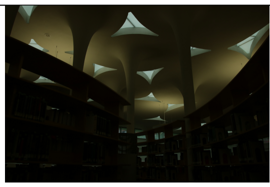
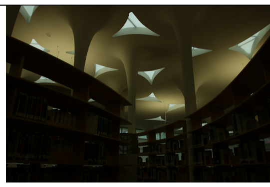
3. Use the output HDR obtained in HDR step as the input of this step, and finally get the LDR image.

Implementation Details

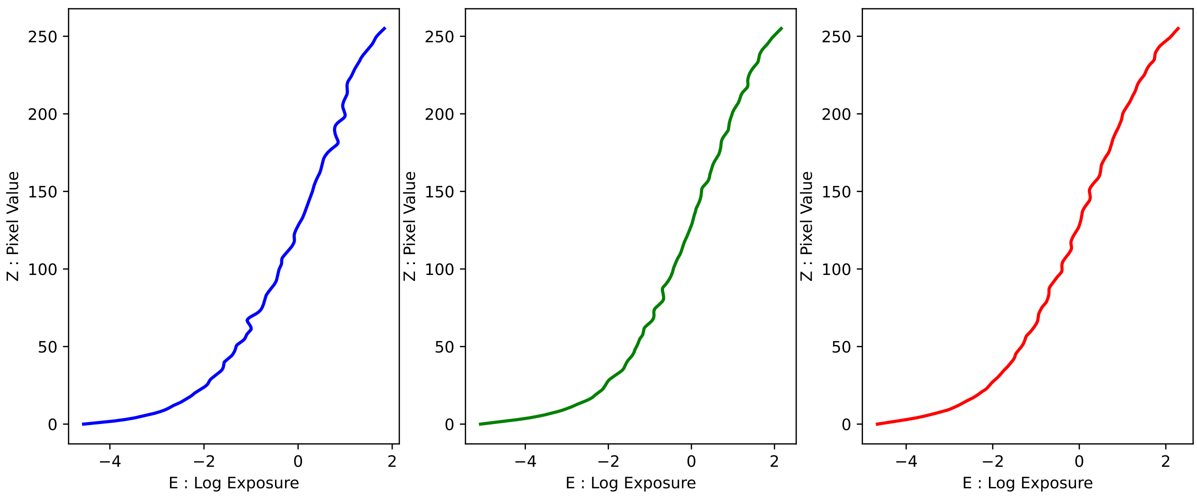
1. Programming language: python
2. Module used: opencv for python, matplotlib, numpy
3. When using the image alignment we implemented, if only a little part of the image is different with complex details, and the rest of the image is black, this algorithm sometimes fails while the number of layers is too large. Usually, setting 3 layers can get a better comparison result.
4. We used cv2.pyrdown to downsample images with a 1/4 scale when experimenting because the original images are too large.
5. Debevec's method's core concept is to solve a big linear system (Ax = b), and matrix A is overdetermined. In Matlab, we could code "$x = A/b$" to get vector x. However, python doesn't support this syntax, and it is suggested that using the package to solve SVD of A. Instead, we used x = np.dot(np.linalg.pinv(A), b) here to get x directly without importing any other packages.

Results

I. Image Alignment

			
Exposure time: 1/800	Exposure time: 1/640	Exposure time: 1/400	Exposure time: 1/320
			
Exposure time: 1/200	Exposure time: 1/160	Exposure time: 1/100	Exposure time: 1/80
			
Exposure time: 1/50	Exposure time: 1/40	Exposure time: 1/25	Exposure time: 1/20
			
Exposure time: 1/13	Exposure time: 1/10	Exposure time: 1/6	Exposure time: 1/5

II. Response Curve



III. Tone Mapping

Photographic Tone Reproduction

	
local, para: S_{max} = 25, a = 1.0, ϕ=8.0, ϵ=0.01	global, para: δ = 1e-6 , a = 0.5

Bonus

1. MTB algorithm
2. Tone Mapping

Reference

1. MTB algorithm: Greg Ward, **Fast, Robust Image Registration for Compositing High Dynamic Range Photographs from Handheld Exposures** , jgt 2003.
2. HDR: Paul E. Debevec, Jitendra Malik, **Fast, Recovering High Dynamic Range Radiance Maps from Photographs**
3. Tone Mapping: Erik Reinhard, Michael Stark, Peter Shirley, James Ferwerda, **Photographic Tone Reproduction for Digital Images** , SIGGRAPH 2002