Korea Advanced Institute of Science and Technology

School of Electrical Engineering

EE838B

Special Topic in Image Engineering

Advanced Image Restoration and Quality Enhancement

Student's Name: Dinh Vu

Student's ID: 20184187

Homework 2

1. Dataset

In the given dataset, the Low Dynamic Range (LDR) images are inconsistent with the High Dynamic Range (HDR) images because the LDR images were not generated from the HDR images by using the virtual camera function in the paper [1]. Therefore, the given LDR images are not used in the training process. The dataset must be divided into training set and validation set.

There are 46 given HDR images, named as "Cxx_HDR.hdr", where xx is the index of the images from 01 to 46. The images having the indices from 01 to 43 are used for training while the rest becomes the validation set. As mention above, the given LDR images are not used but the LDR images for validation must be created from the camera function [1]. To generate the validation LDR images, the method is presented in the paper [1]. Because some given HDR images consists several zero-pixel rows, these pixel-rows must be removed first before doing the following step.

- ***Step 1:*** The pixel values in 3 channels are clamped from 10^{-5} to 10^{5}.
- ***Step 2:*** The luminance of each pixel is approximately determined by average of 3 channels.
- ***Step 3:*** All the luminance values are sorted in the 1-D ascending order vector. The threshold is normally random chosen from 0.85 to 0.95. The element of the 1-D vector, corresponding with this threshold, separates the vector into 2 parts: ones include 85÷95 percentage of the vector having lower values, the other consists 5÷15 percent with higher values.
- ***Step 4:*** The pixel values in 3 channels of the HDR images are divided to the value of the above element called H_{th}

$$H(i,j,k) = \frac{H(i,j,k)}{H_{th}}$$

Where $H(i,j,k)$ is the value of pixel (i,j) of the HDR image at the channel k.

- ***Step 5:*** The raw LDR image is generated by the below camera function:

$$D(i,j,k) = f(H) = (1+\sigma)\frac{[H(i,j,k)]^n}{[H(i,j,k)]^n + \sigma}$$

Where:

- $D(i,j,k)$ is the value of pixel (i,j) of the LDR image at the channel k.
- $\sigma \sim \mathcal{N}(0.9, 0.1)$ and $n \sim \mathcal{N}(0.6, 0.1)$

- ***Step 6:*** Because after step 4, the pixel values of the HDR images are from 0 to greater than 1, the pixel values of the raw LDR image also have the same range. Hence, these LDR pixel values must be normalized to become integer from 0 to 255. Then the normalized LDR images are saved in PNG format.

$$D(i,j,k) \coloneqq \text{uint8}[255 \times D(i,j,k)]$$

2. Training Process

First, the HDR image is read by using function `imageio.imread()`. Next, all zero-pixel rows in the HDR image are removed. After that, a crop, covering uniformly from 20 to 60 percentage of the HDR image, is randomly cropped from the image by using function `tf.random_crop()`. However, notice that after removing all zero-pixel rows, the images have various resolution from 800×1920 to 1080×1920, so the crop size requires an upper bound to avoid cropping larger size than the size of the images.

$$\min cropsize = \text{int}\left(\sqrt{800 \times 1920 \times 0.2}\right) = 554$$
$$\max cropsize = \min\left\{\text{int}\left(\sqrt{1080 \times 1920 \times 0.6}\right), 1080\right\} = 1080$$

The crop size always larger than 320. Hence, the crop must be bilinear down-sampling to 320×320 by using function `tf.image.resize_images()`. So, the HDR patch is created. The LDR patch is generated from the HDR patch by following step 1 to 5 in Section 1, then using the below equations, respectively:

$$D(i,j,k) = \frac{255 \times \min\{1, D(i,j,k)\} + 0.5}{255}$$

Where $X(i,j,k)$ is the value of pixel (i,j) at channel k in the LDR patch.

Finally, the value of pixel in the HDR and LDR patch must be clamped to avoid outliner. For the LDR patch, the pixel value from 0 to 1 while for the HDR patch, from 0 to 10.

$$D(i,j,k) = \min\left\{1, \max\{0, D(i,j,k)\}\right\}$$

$$H(i,j,k) = \min\left\{10, \max\{0, H(i,j,k)\}\right\}$$

Two patches are push into the network to learn. The number of epochs is 1000. The initial learning rate is 5×10^{-5} and unchanged until epoch 400^{th}. The learning rate decay is applied after finish first 400 epochs, since that time, after 100 epochs, the learning rate is decreased by half. The model is trained from scratch. All the configurations is in the file `./source/config.py`.

Because the number of examples in the training set is small, just 43 images. Therefore, the data augmentation is applied using horizontal and vertical flip images with 0.5 probability by two functions `tf.image.random_flip_left_right()` and `tf.image.random_flip_up_down()`.

3. Network Architecture

The architecture of the U-net autoencoder is depicted in Figure 3.1. The authors wrote that all layers of the network use ReLU activation function [1], which is not quite right. In fact, following each transpose convolution layer and batch normalization layer in the decoder, there is a leaky ReLU activation function. Moreover, in the skip connection layers, there is not any activation function.

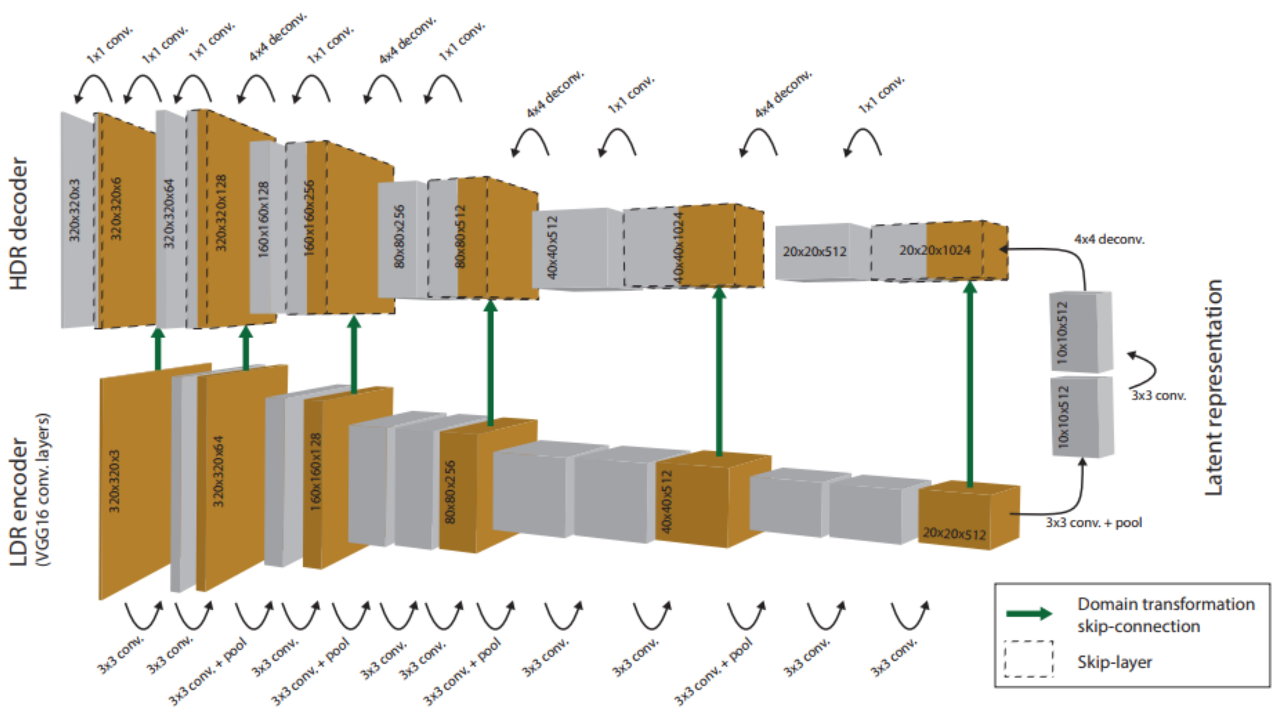


Figure 3.1. The structure of the network [1]

Because the range of pixel values in LDR patches in the author's reference code is from 0 to 255, so in the skip connection before logarithm transformation, the input of skip connection must be divided to 255. However, my input LDR patches already were normalized from 0 to 1, hence it is logarithm transformed directly without dividing.

4. Validation Process

Unlike training, the LDR patches are produced from the corresponding HDR patches, in validation, the validation LDR images, already created in Section 1, are converted to `float32` and normalized from 0 to 1 by using function `tf.image.convert_image_dtype`(). The given validation HDR images are normalized following step 1 to 4 then clamped from 0 to 10 as Section 2. The output of the network is linearized by the inverse camera function then alpha blending.

Inverse camera function: $f^{-1}(D) = \left(\frac{\sigma D}{1 + \sigma - D}\right)^{\frac{1}{n}}$

Alpha blending: $\hat{H} = (1 - \alpha) \times f^{-1}(D) + \alpha \times \exp(\hat{Y})$

The final prediction HDR image is multiplied with H_{th}: $\hat{H} = \hat{H} \times H_{th}$. Where:

- D is the input normalized LDR image of the network
- $\hat{Y}$ is the output of the network
- $\hat{H}$ is the prediction HDR image

The validation process are in the file `valid.py`.

5. The experimental results

The neural network is trained and validated in the computer with an Intel(R) Core(TM) i7 @ 4.20GHz and a GPU NVIDIA GTX GeForce 960Ti. Total training time is about 1.5 hours and total validation time is about 26 seconds.

There are two different loss in the paper [1]: illuminance/reflectance loss (I/R loss) and direct loss. The network is optimized with the I/R loss. The diagram of the I/R loss and direct loss are shown in Figure 5.1 and 5.2, respectively. The direct loss was not applied for optimizer but calculated to looking for the relation between two kinds of loss. It is noticeable to see that both values of loss are covariate. When I/R loss is optimized, the direct loss also is converged.

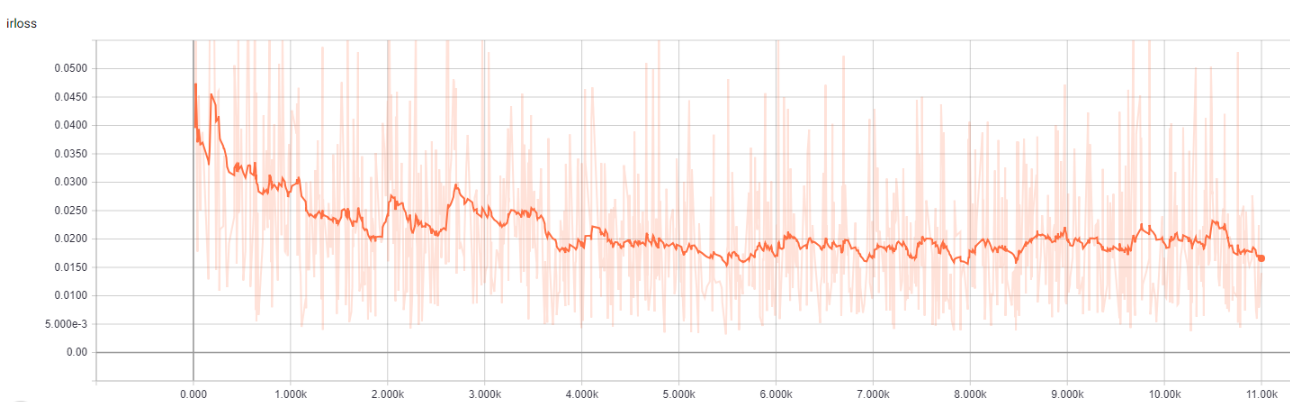


Figure 5.1. The I/R loss with respect to (w.r.t.) iterations

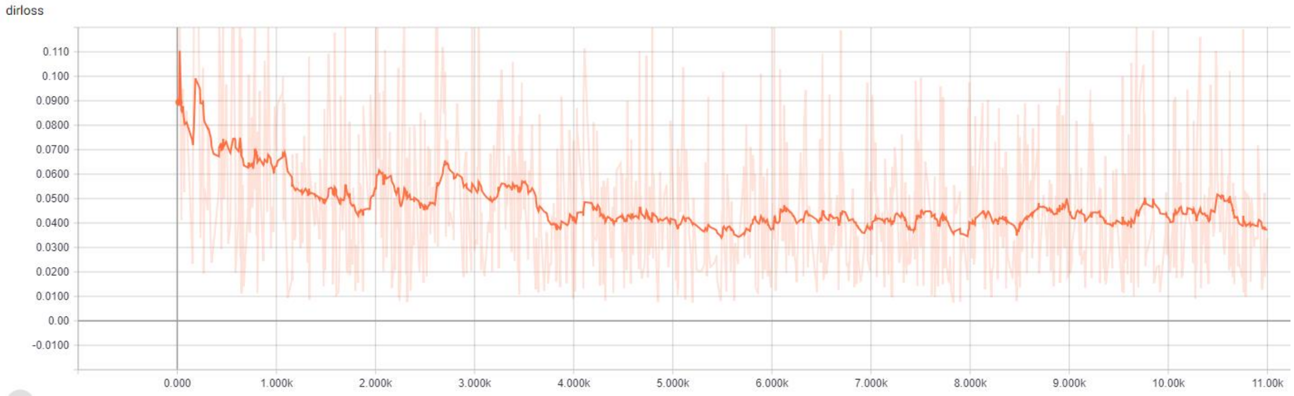


Figure 5.2. The direct loss w.r.t. iterations

The alpha map, displayed in Figure 5.3, separates the image into two parts: one includes 85÷95% of lower pixel values and other is the most brightness.

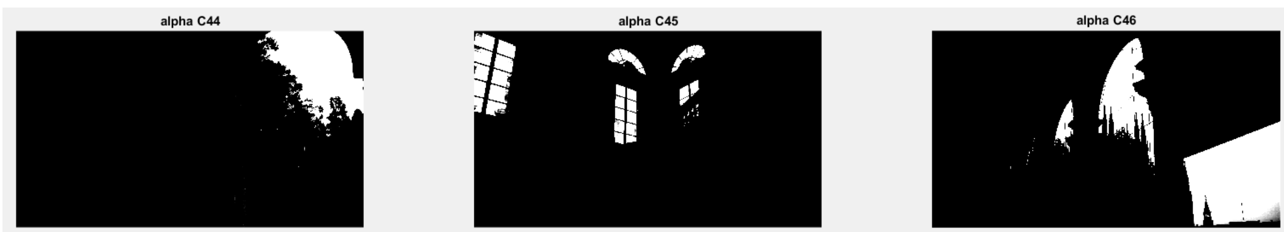
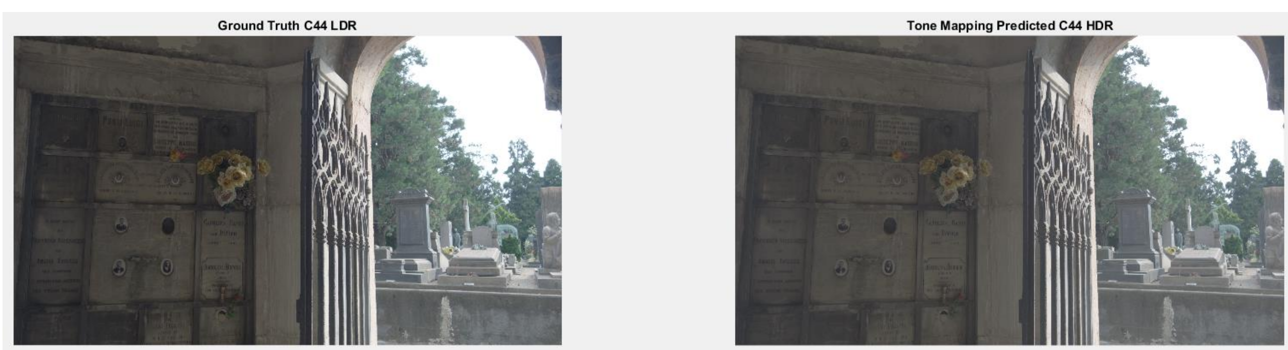
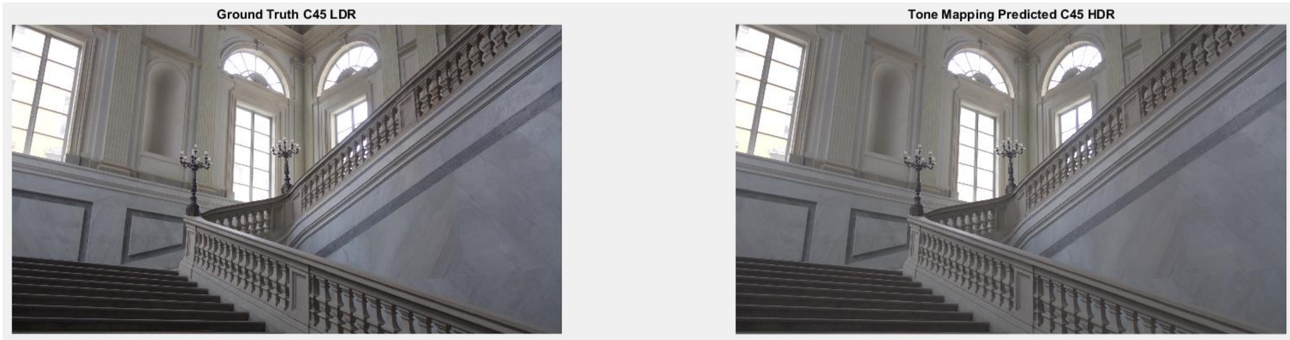


Figure 5.3. The alpha map of each validation image

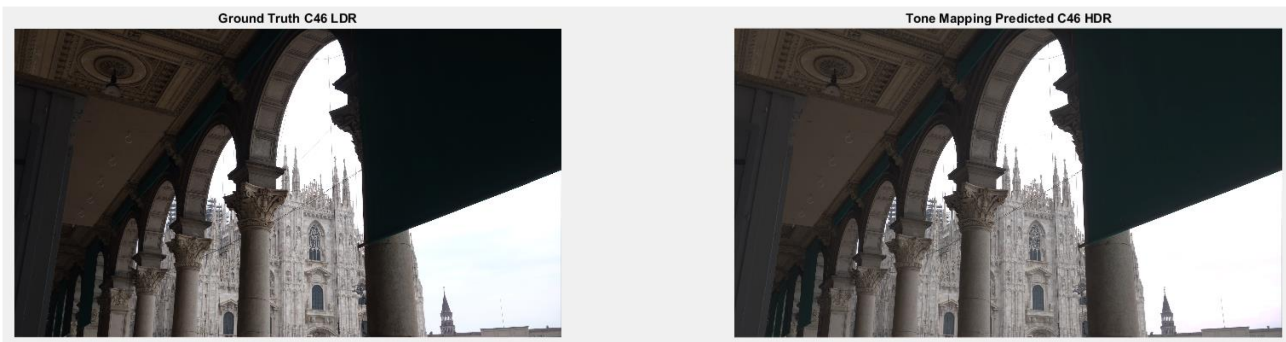
Unfortunately, due to lack of HDR monitor, the prediction HDR images cannot be viewed directly, which causes inconvenient to know what happens during and after training. In order to solve that problem and compare with the ground truth indirectly, the prediction HDR images are tone mapping to LDR then compare with the ground truth LDR images. It can be seen in Figure 5.4, the tone mapping prediction look similar with the ground truth LDR images.



(a) C44



(b) C45



(c) C46

Figure 5.4. The ground truth LDR and tone mapping prediction HDR

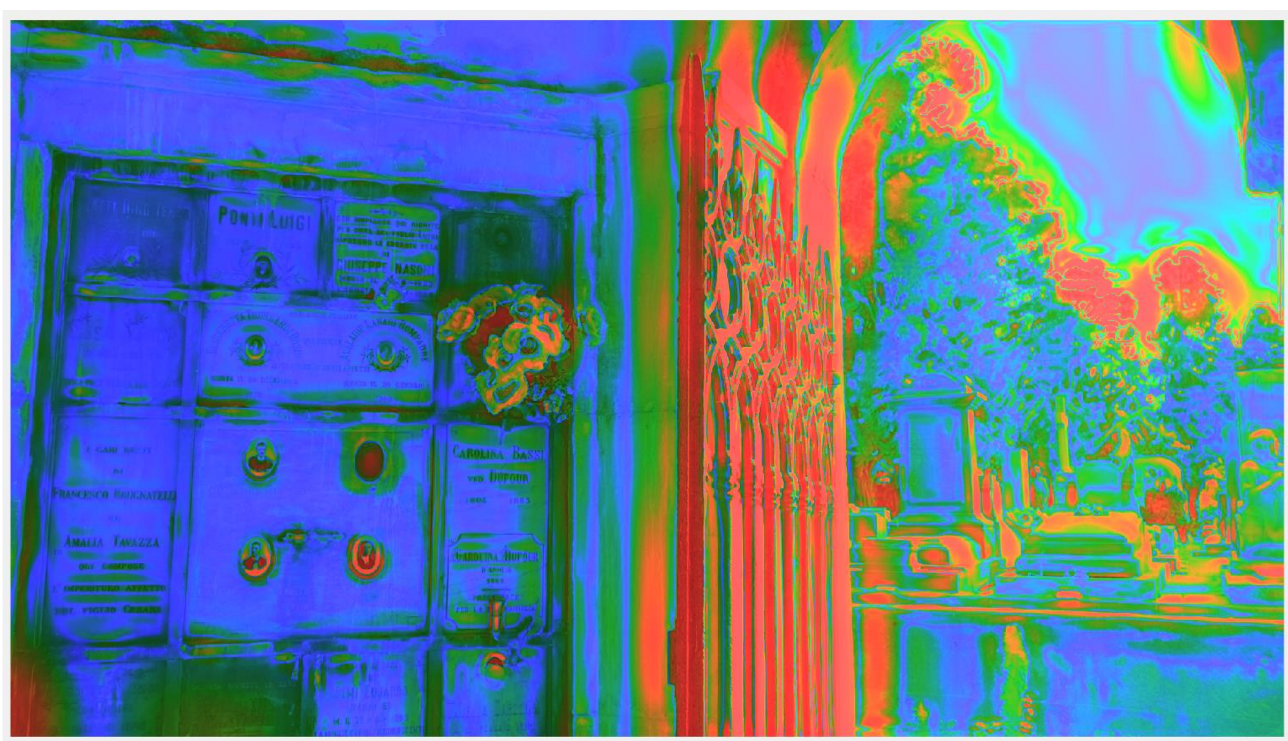


Figure 5.5. The visualization map of the validation image C44

Figure 5.6. The visualization map of the validation image C45

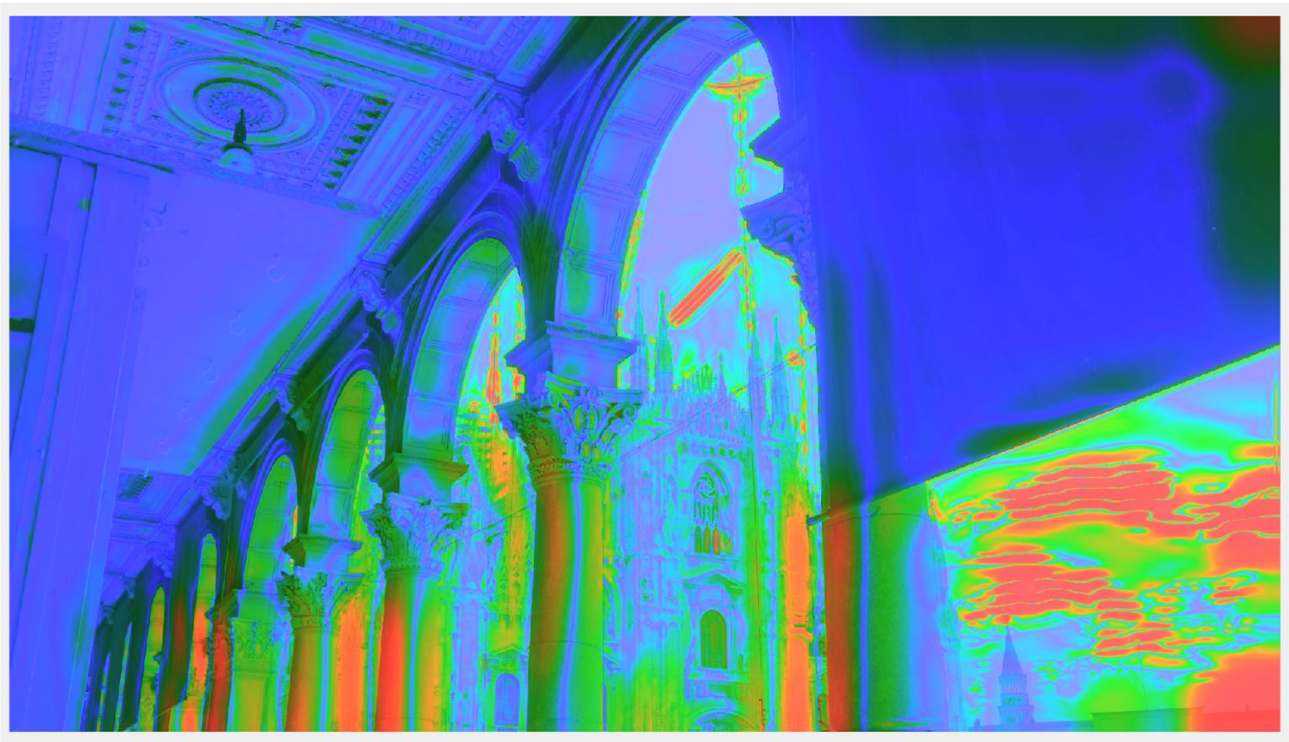


Figure 5.7. The visualization map of the validation image C46

Table 5.1. The quality of each validation images

	C44	C45	C46
I/R Loss	0.01589	0.00634	0.00902
Direct Loss	0.003328	0.01160	0.01520
mPSNR (dB)	20.82	30.43	21.03
Quality Score	81.91	82.48	82.83

The mPSNR is calculated by the function `mPSNR()` in HDR Toolbox [2]. The visualization maps and quality scores are determined by HDR-VDP 2.2.1 [3]. Before estimating the mPSNR, quality scores and visualization maps of each validation image, the pixel values of the ground truth and prediction HDR images are normalized from 0 to 1 to make the reference and distorted image have the same data range. The visualization maps are presented from Figure 5.3 to 5.5. Most of red regions, which contains artifact, locates at the most saturated areas such as sky or window. Table 5.1 above shows the quality of each validation image.

6. Testing Process

In the validation, the ground truth of HDR and LDR images are known. However, in the testing process in real application, the LDR images are only given and the HDR images have to be generated from these LDR images with unknown H_{th} values. For the given dataset in this homework, in the testing, after alpha blending, the prediction HDR pixels will be multiplied with 500 instead of H_{th}.

Reference

[1] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafal K. Mantiuk and Jonas Unger, "HDR image reconstruction from a single exposure using deep CNNs", *ACM Transactions on Graphics*, volume 36, no. 6, article 178, October 2017.

[2] Francesco Banterle, Alessandro Artusi, Kurt Debattisa and Alan Chalmers, "Advanced High Dynamic Range Imaging 2nd Edition", *AK Peters (CRC Press)*, July 2017.

[3] Rafal Matiuk, Kil Joong Kim, Allan G. Rempel and Wolfgang Heidrich, "HDR-VDP-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions", ACM Transactions on Graphics, volume 30, issue 4, article no. 40, July 2011.