VFX Project #1: High Dynamic Range Imaging Report

B05902062 王子銘 b05902066 蔡翔陞

In this project, we implement a HDR image generating algorithm based on the paper “Recovering High Dynamic Range Radiance Maps from Photographs”, and also implement MTB and tone mapping for bonus.

Part One: Generate HDR image

How to:

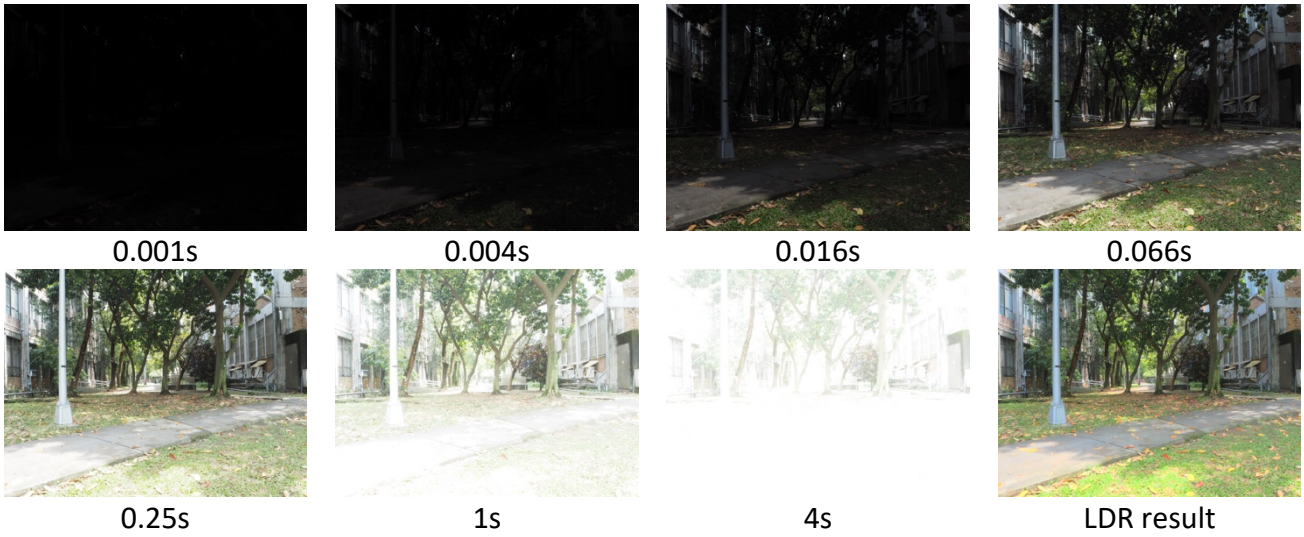
Generate the HDR image need following steps:

1. Gian the images of same scene under different shutter speed
2. Reconstruct the response curve of camera
3. Use the response curve and shutter speed to count for the energy

Out implementation detail:

1. Gian the images of same scene under different shutter speed

   We use the build-in HDR shooting mode from the Olympus camera, which could shoot seven images under different shutter speed in one press. The main scenes are in campus, both at daytime and night.



0.001s | 0.004s | 0.016s | 0.066s

0.25s | 1s | 4s | LDR result

T1-1. Images and corresponding exposure time

2. Reconstruct the response curve of camera

   We use the method mention in “Recovering High Dynamic Range Radiance Maps from Photographs”, which using the intensity value of RGB

channel between images under different shutter speed to reconstruct the response curve.

a. Sampling:

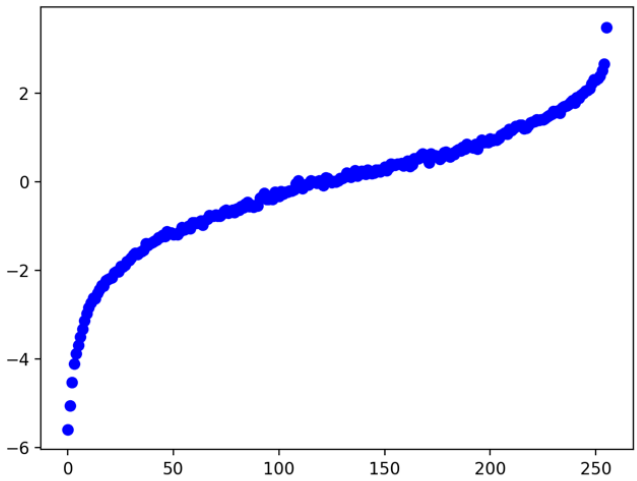
In our implementation, we sample N point in the image randomly and record the illumination. In order to refine our response curve, we use the "More is Better" strategy, which sampling more points if the range of the intensity value is too narrow. In addition, we use a single curve to reconstruct the RGB channels simultaneously. Implement as follow:

```
record = np.zeros(256)
i = 0
while i < N or ( np.sum( record ) < h and i < 2 * N ):
   p1, p2 = RandomPickPoint()
   for k in range( len ( imgs ) ):
      for c in range( 3 ):
         record[ imgs[k][p1][p2][c] ] = 1
   i += 1
```

b. Reconstruct response curve

This problem could solve by the less square solution of the over-determined linear system. We use numpy.linalg.lstaq in python to help us.

c. Result



3. Use the response curve and shutter speed to count for the energy

We use the formula

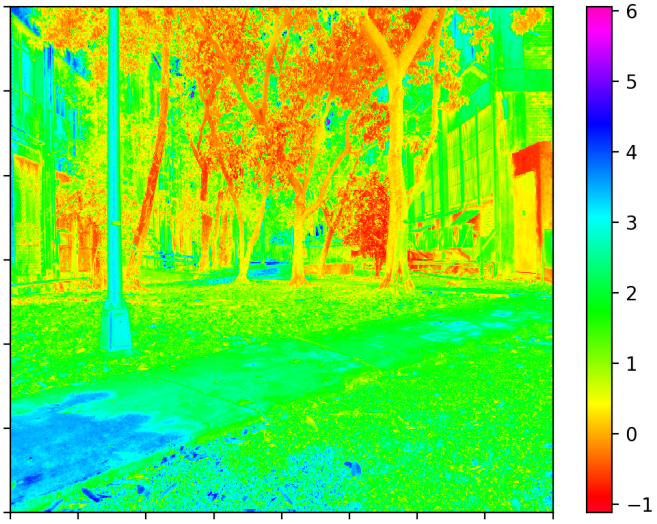
$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})\left(g(Z_{ij}) - \ln \Delta t_j\right)}{\sum_{j=1}^{P} w(Z_{ij})}$$

to revert the origin energy in the scene.

To prevent the black holes appearing on the HDR image at some extreme bright point, we modify the formula when counting w to ensure the reasonable energy distribution.

```
def weight(a):
    if(a > 128):
        if(a == 255): return 1
        return 255-a
    return a
```

Log Radiance Map:



Part Two: MTB

MTB(Median Threshold Bitmap) is a technique to align images. In this project, MTB is mainly used to pre-processing on the images of same scene under different shutter speed.

Implementation:

We would take a series of images as input and find the shift between each pair of consecutive images. Then choose one of the image as benchmark, count the shift on other images, set the region out of the boundary as BGR=(255, 0, 0) and output.

More details on MTB:

MTB(A, B):

1. Read images A, B and convert into gray scale
2. Built image pyramid I_{A0}~I_{AN}, I_{B0}~I_{BN} $\left(size(I_{Ak}) = 2 * size\left(I_{A(k+1)}\right)\right)$
3. Set initial shift value as (0, 0)
4. From layer N to layer 0, repeat a. - f.

a. Find the median M_{Ak}, M_{Bk} of image I_{Ak}, I_{Bk}
b. generate $bit_{Ak} = I_{Ak} > M_{Ak}, bit_{Bk} = I_{Bk} > M_{Bk}$
c. generate $mask_{Ak} = |I_{Ak} - M_{Ak}| > threshold$
d. generate $mask_{Bk} = |I_{Bk} - M_{Bk}| > threshold$
e. shift $bit_{Bk}, mask_{Bk}$ according to shift value
f. fix $bit_{Ak}, mask_{Ak}$, move $bit_{Bk}, mask_{Bk}$ in eight direction for one pixel and count for $Loss = \left((bit_{Ak} \hat{} bit_{Bk}) \& mask_{Ak}\right) \& mask_{Bk} == True$
g. find the shift value to minimize Loss and update

5. Output shift value

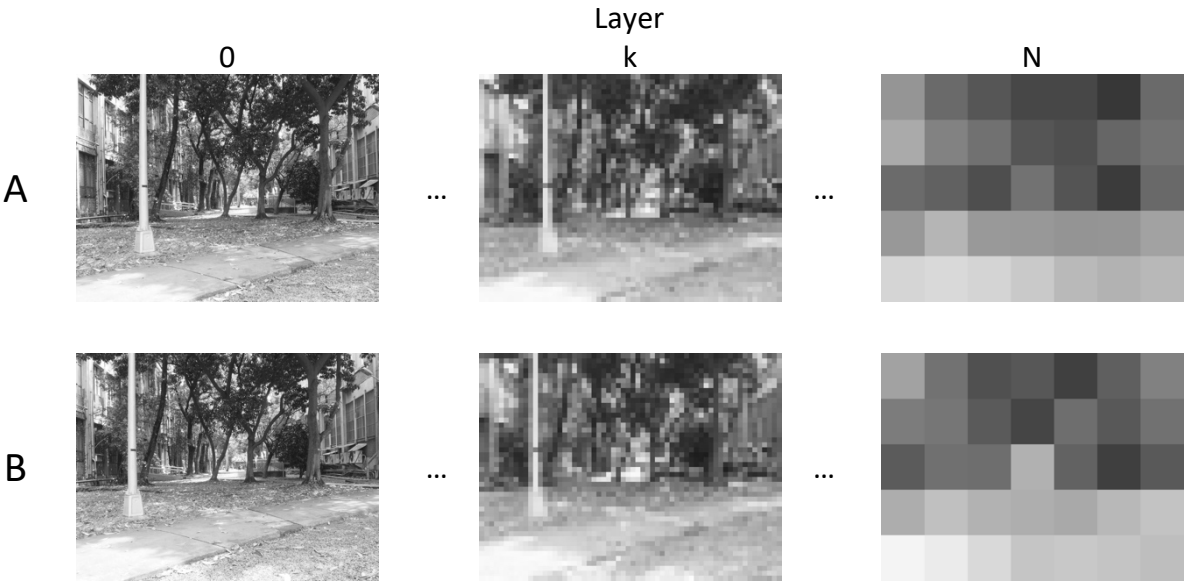
Example:

Read images A, B (shift manually)

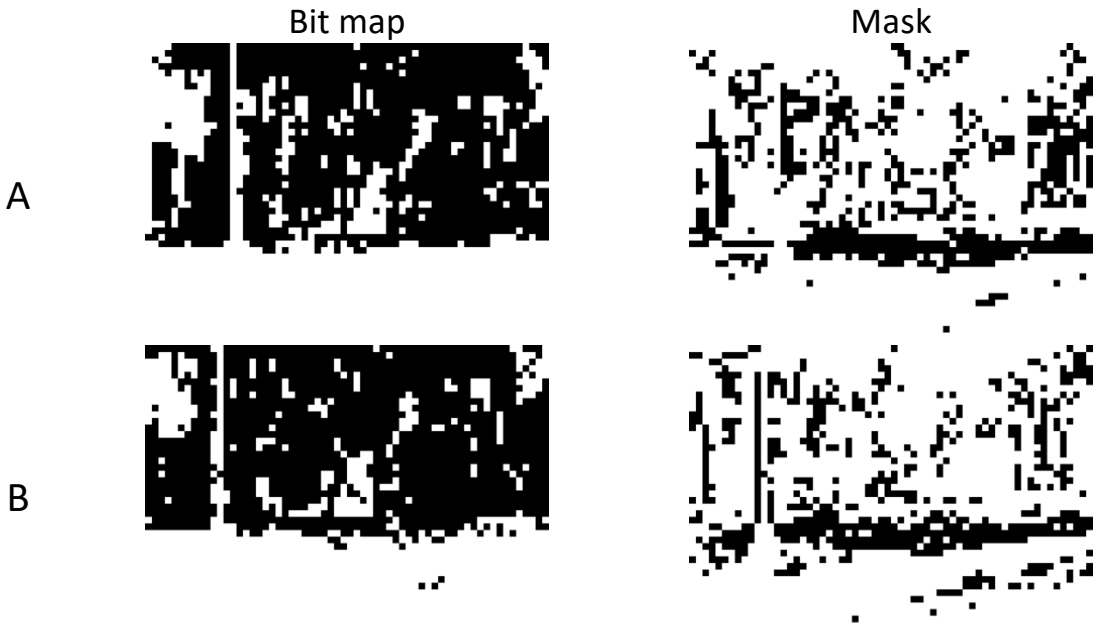
A

B

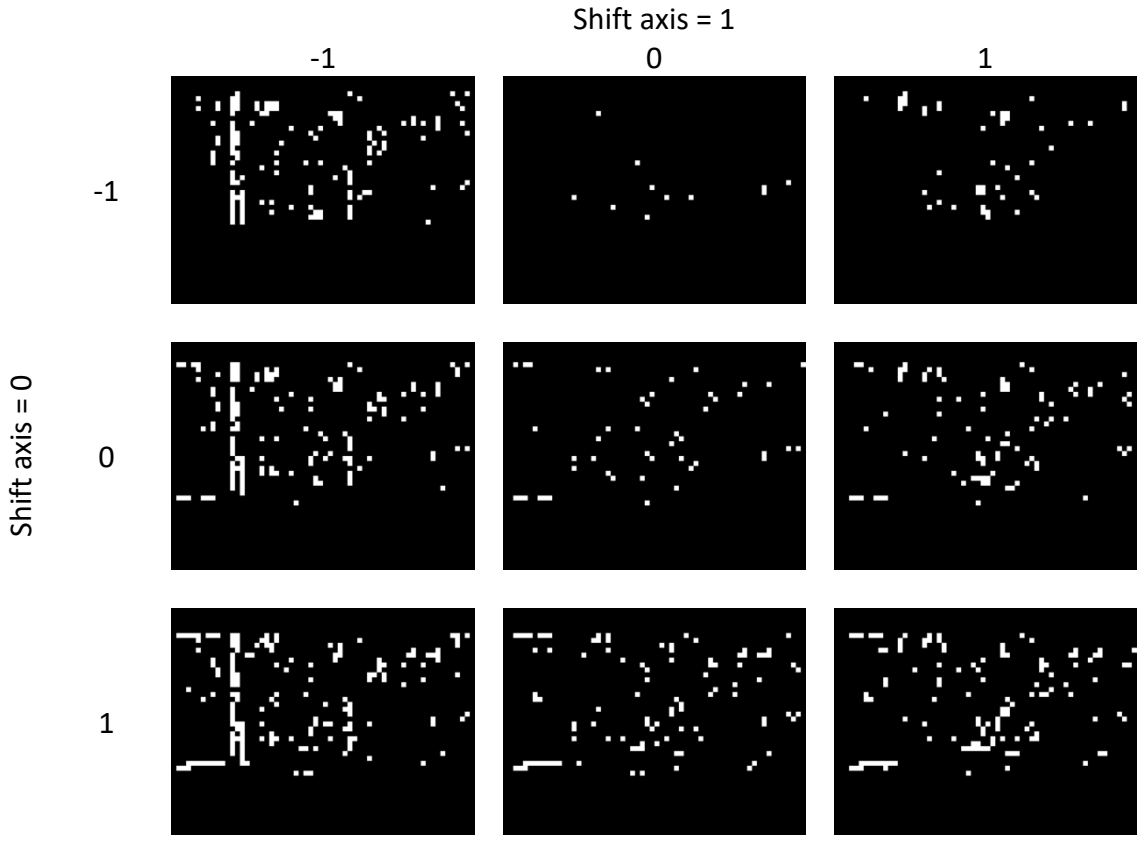
Generate image pyramid



Generate bit and mask (Layer = k)



Loss map



Result:

	With MTB	Without MTB
A		
B		
Result		

Part Three: Tone Mapping

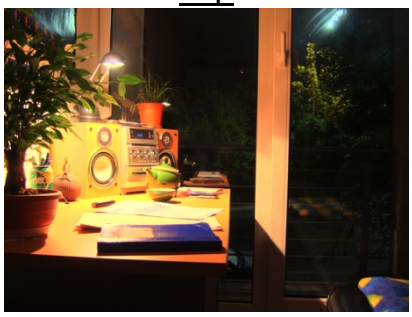
We have also implemented a slightly simplified photographic tonemapping algorithm taught in class. It includes a global operator and a local burning and dodging. But for the local burning and dodging, instead of setting a threshold and let it decide the size of a gaussian filter itself, we make the size a constant value which we can pass as an argument, for we think this can obtain the goal of sharpening images already and make our program faster. Our current program runs roughly an hour on a 4608×3456 image on a macbook pro.

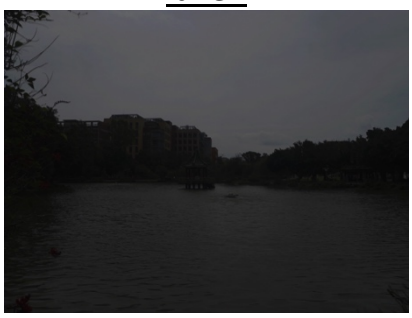
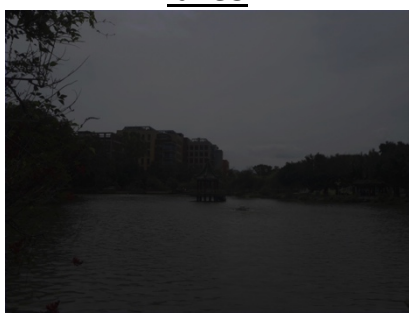
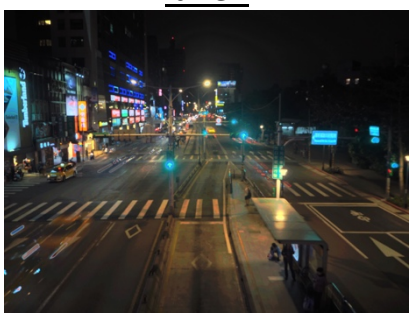
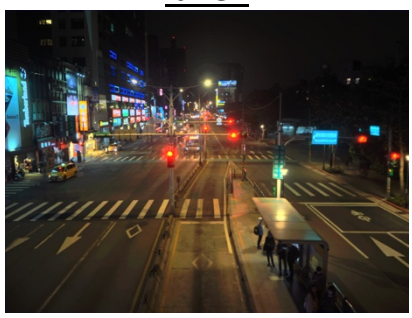
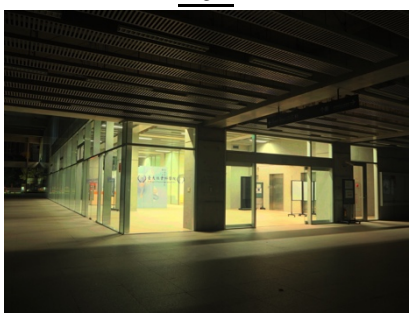
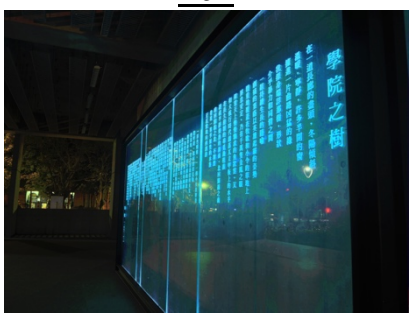
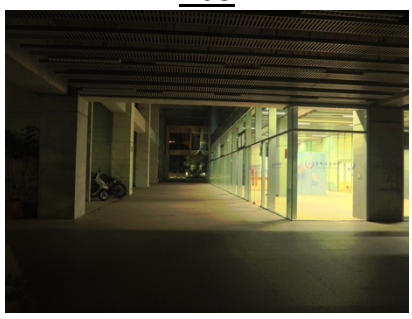
The paper didn't mention how to calculate the values for the RGB channels after we tonemap an image. We eventually chose scaling color channels up according to the ratio

between the luminance of the pixel of the original HDR image and that of the tonemapped image linearly.

We have 2 different sets of images for scenes in both daytime and at night, including one from our course website. Our first time of photographic tonemapping uses only global operator with a key value of 0.18 for all HDR images to produce LDR images. After that trial, we find out that we have two sets, tree1 and phy1, that have incorrect shutter speed settings, so for the rest of tonemappings we would only run the algorithm on the other 19 sets.

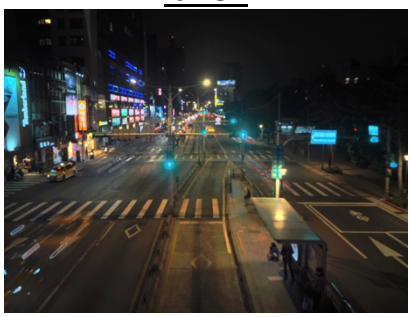
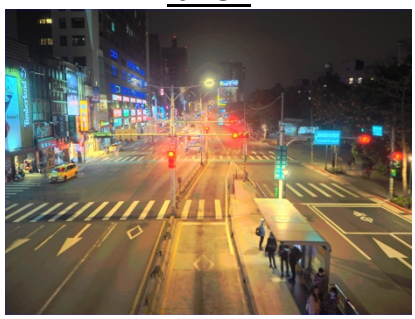
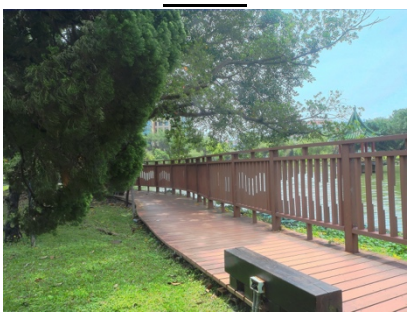
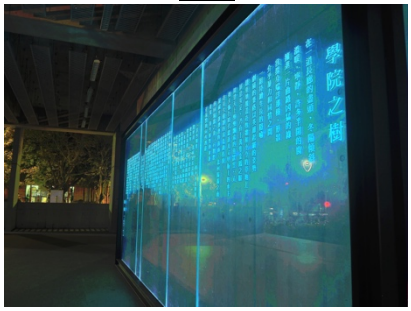
The result of the first time of tonemapping (key is for the overall brightness. g size means gaussian filter size):

Set of images (7 images for each set)	**Set of images** (7 images for each set)	**Set of images** (7 images for each set)
arg1  Key = 0.18 gaussian filter size: X(no used)	bank1  Key = 0.18 gaussian filter size: X(no used)	bank2  Key = 0.18 gaussian filter size: X(no used)
bike  Key = 0.18 gaussian filter size: X(no used)	exp  Key = 0.18 gaussian filter size: X(no used)	house1  Key = 0.18 gaussian filter size: X(no used)

lake1  Key = 0.18 gaussian filter size: X(no used)	lake2  Key = 0.18 gaussian filter size: X(no used)	lake3  Key = 0.18 gaussian filter size: X(no used)
lake4  Key = 0.18 gaussian filter size: X(no used)	lane1  Key = 0.18 gaussian filter size: X(no used)	lane2  Key = 0.18 gaussian filter size: X(no used)
wallway1  Key = 0.18 gaussian filter size: X(no used)	lane4  Key = 0.18 gaussian filter size: X(no used)	lane5  Key = 0.18 gaussian filter size: X(no used)
lib1  Key = 0.18 gaussian filter size: X(no used)	lib2  Key = 0.18 gaussian filter size: X(no used)	lib3  Key = 0.18 gaussian filter size: X(no used)

Below is the final parameters we choose for the our tonemapping

Set of images (7 images for each set)	**Set of images** (7 images for each set)	**Set of images** (7 images for each set)
arg1  Key: 0.36 gaussian filter size: 1.5	bank1  Key: 0.72 gaussian filter size: 3	bank2  Key: 0.72 gaussian filter size: 3
Bike  Key: 0.54 gaussian filter size: 4	exp  Key: 0.54 gaussian filter size: 1.5	house1  Key: 0.54 gaussian filter size: 3
lake1  Key: 2 gaussian filter size: 1.5	lake2  Key: 1.1 gaussian filter size: 1.5 (It is pretty good already at the second time. Lower it a bit to let the sky be clearer.)	lake3  Key: 2 gaussian filter size: 1.5

lake4  Key: 1.1 (be clearer) gaussian filter size: 1.5 (It is pretty good already at the second time. Lower it a bit to let the sky	lane1  Key: 0.18 gaussian filter size: X(no used) (hard to decide how to make adjustment)	lane2  Key: 0.54 gaussian filter size: 3.0
wallway1  Key: 0.9 gaussian filter size: 1.5	lane4  Key: 0.7 gaussian filter size: 1.5	lane5  Key: 0.9 gaussian filter size: 3.0
lib1  Key: 0.16 gaussian filter size: 1.5	lib2  Key: 0.36 gaussian filter size: 3.0 (It is pretty good already at the second time.)	lib3  Key: 0.1 gaussian filter size: 1.5 (It is pretty good already at the second time.)

Below are some examples of how our HDR images outperform original images





A, exposure time = 3.2S

B, exposure time = 1S

HDR image

The tree at the back of the image in our HDR image is as clear as A. But we can see it preserve the detail we see in B, which disappear in A.



C.shutter speed 0.0625S

D.shutter speed 0.25S

HDR image

Our HDR image displays the tree in the middle with the same quality as C. But we can see the yellow leaf on the track in HDR image clearly, which mix up with the background in C. So the HDR image is better than C and D, let alone images with higher or lower shutter speed than C and D respectively.