第 17 組 B09902128 黃宏鈺、B09902017 李安傑

[DVE] project #1: High Dynamic Range Imaging

Data preprocess

- **Raw data**
  - camara : Canon PowerShot SX50 HS
  - file tpye : CR2
  - ISO : ISO-80
  - Aperture : f/8
  - size : 4000 × 2664
  - shutter speed : 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256
- **CR2 to PPM**
  we use DCRAW (https://github.com/ncruces/dcraw) to covert cr2 to ppm.
  `./dcraw -4 -w file_path.cr2`
  - -4
    It generates a linear 16-bit file instead of an 8-bit gamma corrected file which is the default.
  - -w
    If DCRAW manages to find it, it will use the white balance that was adjusted in the camera at shooting.
- below is the demonstration of photos by jpg transfromed from CR2.
  we use online transformer, iloveimg, here. (https://reurl.cc/nEGk16)
  Barrel distortion has been fix by iloveimg automatically.

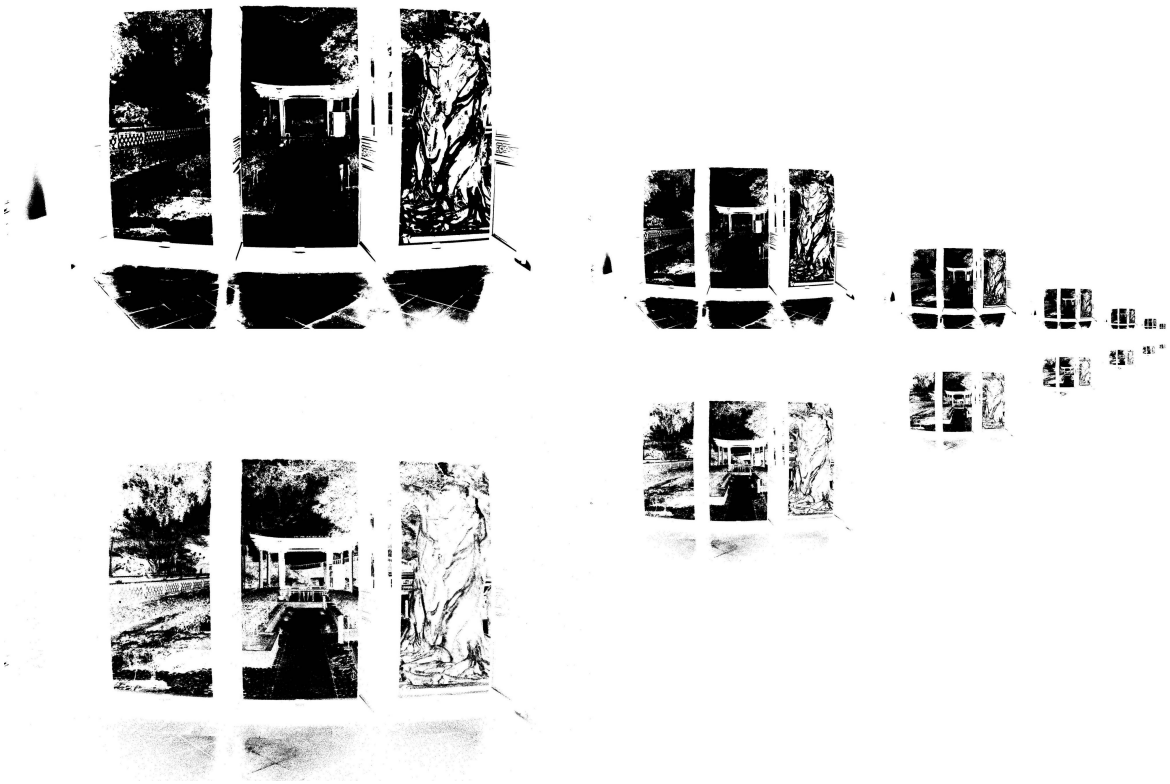
MTB

To deal with the image alignment issue, we implement the MTP algorithm with python.

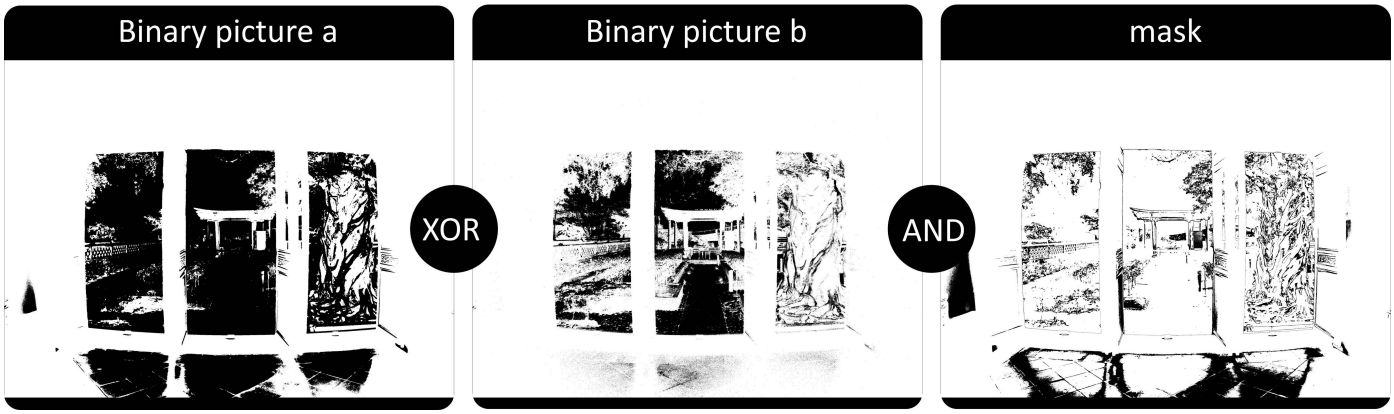
We compress the image to 7 size, and run the process to estimate the best offset 7 times from the smallest image to the biggest one. In each process we chosse the best offset by the method below:

1. We first change the image to grayscale with function : $Y = (54R + 183G + 19B)/256$
   (Y represent the light intensity of the pixel, and the red, green, blue of each pixels is denoted as R, G, B)

2. Then, we calculate $\bar{Y}$ (the average of Y). And turn the image to binary by comparison with $\bar{Y}$ (If greater than $\bar{Y}$ we give it 0, else we give it 1).



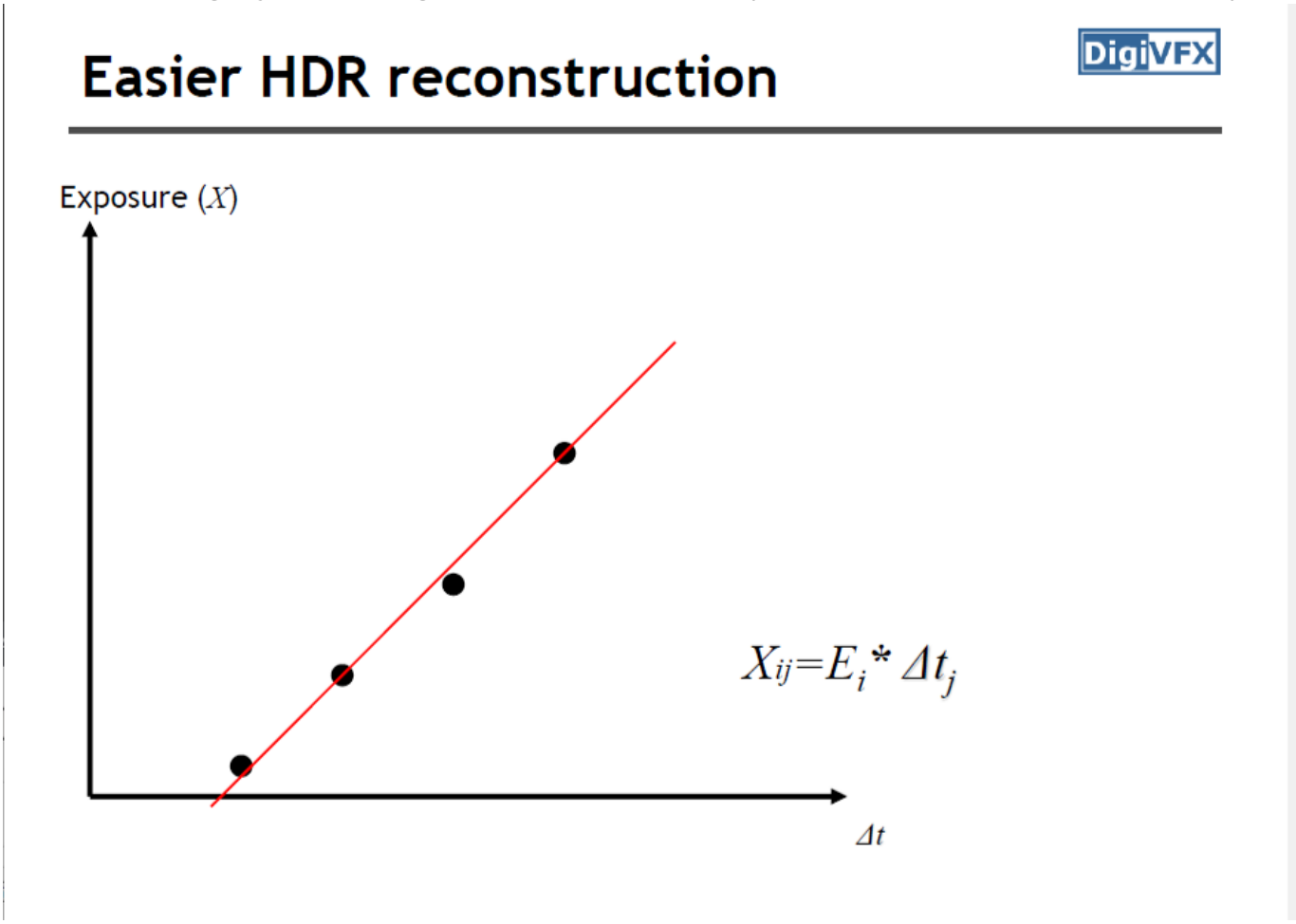
3. Also we compute the mask to filter out the pixel with intenseti in $\bar{Y}$ +- 5%.
4. Then we evaluate the score of each offset by the function $\sum_{x,y}(\mathrm{p1}[x,y] \oplus \mathrm{p2\_offset}[x,y] \& \mathrm{mask}[x,y])$, and choose offset with the min vlue of score.



After each rounds, the offset will change to 2 times and be add as the initialize offset of the next round.

HDR

Recover HDR image by method taught on class. Minimize the square error between the best fit line and points.



$$arg \min_{E_i} Err(E_i) = \sum_{j=1}^{p} (X_{ij} - E_i \times \Delta t_j)^2$$

$$0 = \frac{\partial Err}{\partial E_i} = \sum_{j=1}^{p} -2\Delta t_j (X_{ij} - E_i \times \Delta t_j)$$

$$\sum_{j=1}^{p} \Delta t_j X_{ij} = E_i \sum_{j=1}^{p} (\Delta t_j)^2$$

$$E_i = \frac{\sum_{j=1}^{p} \Delta t_j X_{ij}}{\sum_{j=1}^{p} (\Delta t_j)^2}$$

We use the final function $E_i = \frac{\sum_{j=1}^{p} \Delta t_j X_{ij}}{\sum_{j=1}^{p} (\Delta t_j)^2}$ to calculate every pixel's RGB.

Tone Mapping (Bonus)

default global operator

(x, y) : the posision of a pixel
c : the color channel (r, g, b)

$$
\begin{aligned}
\bar{L_w}(c) &= exp\left(\frac{1}{N}\sum_{x,y} log(\delta + L_w(x,y,c))\right) \\
L_m(x,y,c) &= \frac{\alpha}{\bar{L_w}} L_w(x,y,c) \\
L_d(x,y,c) &= \frac{L_m(x,y,c)}{l + L_m(x,y,c)}
\end{aligned}
$$

Picture of $\delta = 10^{-6}, \alpha = 0.09$:

barrel distortion

To fix the barrel distortion, we take the method on **stackoverflow**(https://reurl.cc/dXZ5qg).
Use cv2.undistort().

L_{white}

Add L_{white} in L_d, making picture brighter and cut off the light brighter than L_{white}.

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

Picture of $L_{white} = 1.5,\ \delta = 10^{-6},\ \alpha = 0.09$:

color

To fix the color imbalance, we give a Coefficient to each color.

$$\begin{aligned} L'_d(x.y,g) &= 0.75 \times L_d(x,y,g) \\ L'_d(x,y,r) &= 1.1 \times L_d(x,y,r) \\ L'_d(x,y,b) &= 1 \times L_d(x,y,b) \end{aligned}$$

Picture of modifing the color :

β

The Luminous intensity contrast is still too big that we lose lots of detail. Therefore, we modify theL_m by adding the β:

$$
\begin{aligned}
L_m(x,y,c) &= \frac{\alpha}{\bar{L}_w} L_w(x,y,c) + \beta \\
&= \left(\frac{\alpha}{\bar{L}_w} + \frac{\beta}{L_w(x,y,c)} \right) L_w(x,y,c) \text{The smaller } L_w(x,y,c) \text{ comes the bigger } coif \\
&= coif \times L_w(x,y,c)
\end{aligned}
$$

Picture of $beta = 0.3,\ L_{white} = 1.5,\ \delta = 10^{-6},\ \alpha = 0.09$:

Final global operator

$$
\begin{aligned}
\bar{L_w}(c) &= exp\left(\frac{1}{N}\sum_{x,y} log(\delta + L_w(x,y,c))\right) \\
L_m(x,y,c) &= \frac{\alpha}{\bar{L_w}} L_w(x,y,c) + \beta \\
L_d(x,y,g) &= 0.75 \times \frac{L_m(x,y,g)}{l + L_m(x,y,g)} \\
L_d(x,y,r) &= 1.1 \times \frac{L_m(x,y,r)}{l + L_m(x,y,r)} \\
L_d(x,y,b) &= 1 \times \frac{L_m(x,y,b)}{l + L_m(x,y,b)}
\end{aligned}
$$

Compare to cv2.createTonemapDurand()

Our tone mapping (left), cv2.creatTonemapDurand() (right) :

Compare to the cv2, we still have a lot to do on detail. The roof, word on the wall, and the shadow of doors on our picture lose most of their color and light.
Because we modify the color manually, the colors may still have some implanced. cv2 doesn.t have to do that so it's more rubust.