Project 1 Report (Team 3)

- 電機四 B05502012 許馨文

How to run my code

Environment and Dependencies

- Operating system: Mac OS Big Sur
- Programming language: Python 3.9.2
- Python modules: OpenCV, NumPy

HDR Image

- To run the code, the following files must be provided.
  - Images of different expossures. eg. `img01.jpg`
  - A csv file of shutter speeds in second.
  - A csv file for file names of images of different exposures.

  ```
  Note:
  - In the csv file, one line corresponds to one shutter speed/file name.
  - The order of the shutter speed and file name should match.
  ```

- File tree

  ```
  .
  |-- result.png
  |-- README.md
  |-- report.pdf
  |-- code
  |    |--hdr.py
  |    |--tonemapping.py
  |    |--utils.py
  |    |--filenames.csv
  |    |--speed.csv
  `-- data
       |--img01.JPG ~ img09.JPG
       |--hdr.hdr
       `--result.jpg
  ```

- To recover the hdr image, run the program with the following command. **(Run this in the `code/` directory)**

```
$ python3 hdr.py $DIR
```

  - Where `$DIR` is the directory of the imagers and csv files. (Here it is `../data`)
- The actual command:

```
$ python3 hdr.py ../data
```

  - The resulting hdr ing will be in the data folder

Tone Mapping (Implemented from the listed paper)

- To run the tonemapping program, run the following command: **(Run this in the `code/` directory)**

```
$ python3 tonemapping.py $path2image
```

- The actual command:

```
$ python3 tonemapping.py ../data/hdr.hdr
```

  - The output image will be in the data folder as well
- Note:
  - One can specify different parameters to customize the ldr image.

```
if __name__=='__main__':
    filename = sys.argv[1]
    hdr = cv2.imread(filename, cv2.IMREAD_ANYDEPTH)
    #please specify the desired parameters here
    bb = 2.7; bg = 2.7; br = 2.7;
    gb = 2.6; gg = 2.6; gr = 2.6;
    savename = filename.split('/')[-1].split('.')[0] + '.jpg'
    ldr = get_ldr(hdr, bb, bg, br, gb, gg, gr)
    cv2.imwrite(savename, ldr)
```

  - Basically, a larger b value will result in a darker image.
  - `gb`, `gg`, `gr` are gammas for the gamma correction described in the following sections.
  - If the parameter each channel is very different from each other, there will be a weird color shift.

The Algorithm

HDR Image

- I chose to recover the hdr image with Debevec's SIGGRAPH 1997 paper
- I implemented the program with Python. To avoid accessing important data such as number of images, and shutter speeds for multiple times, I implemented a class `exposure_set`, all the parameters required to recover the HDR images are initialized when the program first starts.

```
class exposure_set:
    def __init__(self, filedir, N, N_sqrt, lamb):
        #read the csv files
        self.filedir = filedir
        self.filenames = [line.rstrip() for line in
open(self.filedir+'filenames.csv')]
        self.speeds = [float(line.rstrip()) for line in
open(self.filedir+'speed.csv')]
        self.img_list = []
        self.N = N #number of sample points
        self.N_sqrt = N_sqrt
        self.lamb = lamb #smoothing factor

        #read images
        for name in self.filenames:
            print(self.filedir+name)
            img = cv2.imread(filedir+name)
            self.img_list.append(img)

        self.p = len(self.img_list) #number of images
        self.height, self.width = self.img_list[0].shape[:2]
```

- In `main.py`, we can specify some parameters for better performances.

```
# parameters
N = 900
N_sqrt = 30
lamb = 5 #smoothing factor
```

  - `N` is the number of sampled points in each image. I found it is better if there are more points
  - `N_sqrt` is just the square root of N, this is a handy parameter for data sampling
  - `lamb` is the smoothing parameter for the second order terms
- How I sampled my the data points:
  - With the decided number of data points to sample, I sample the points uniformly and store the 1-D sample array of each image inside an array. E.g. the $j-th$ image's $i-th$ sample: `Z[j, i]`
  - The three color channels are processed separately.

- More functions of class `exposure_set`:
  - `get_zsamps(self)`
    - Description: Split each image into three channels, sample the points of each channel and store them in an array.
    - Returns the sampled array of each color channel.
    - Usage:

```
z_sample_b, z_sample_g, z_sample_r = class_instance.get_zsamps()
```

  - `get_g(self, zsamps)`
    - Description: Compute g of a single channel with the sampled pixels of the corresponding channels.
    - Returns an array g, which is of length 256
    - Usage:

```
g_b = class_instance.get_g(z_sample_b) # recoverd curve for the blue channel
g_g = class_instance.get_g(z_sample_g) # recoverd curve for the blue channel
g_r = class_instance.get_g(z_sample_r) # recoverd curve for the blue channel
```

  - `get_E(self, g, channel)`
    - Input:
      - `g`: The recoverd g of a certain channel
      - `channel`: An integer used to specify the channel. Blue: 0, Green: 1, Red: 2
    - Description:
      - Compute the E for each channel.
      - E of each pixel in the image is recovered through this formula

$$lnE_i = \frac{\Sigma_{j=1}^{P} w(Z_{ij}) * (g(Z_{ij} - ln\Delta t_j))}{\Sigma_{j=1}^{P} w(Z_{ij})}$$

      - What's different with my implementation is that I did not flatten the image to a 1-D array. I iterate thorugh the whole image, compute E for that pixel and store it in a numpy array.
    - Returns E for the specified channel.
    - Note: The E for each color channel will be merged to form the final hdr image.

Tone Mapping

- I chose to implement the tone mapping method described in this paper: **Drago, Frédéric, et al. "Adaptive logarithmic mapping for displaying high contrast scenes."** ***Computer graphics forum*****. Vol. 22. No. 3. Oxford, UK: Blackwell Publishing, Inc, 2003.**
- Steps
    - Load the hdr image and split the image into differnt color channels
    - Compute Ld with the following formula described in the paper:

$$L_d = \frac{L_{dmax} * 0.01}{log_{10}(L_{Wmax} + 1)} \cdot \frac{Log(L_w + 1)}{log(2 + ((\frac{L_W}{L_{Wmax}})^{\frac{log(b)}{log(0.5)}})) * 8}$$

        - In the paper $L_d max = 100cd/m^2$. I chaned this value from 100 to 10000 and it doesn't make a great difference as I scaled L_d to 0~255 with this formula:

$$L_d = L_d * \frac{L_{dmax} - L_{dmin}}{255}$$

- L_{Wmax} can be optained by `np.max(hdr[:, :, channel])`, where `channel` could be 0, 1, 2
- Gamma correction for L_d, with this formula

$$L'_d = L_d^{\frac{1}{\gamma}}$$

    - Once the L_d for each channels are computed, they are merged and written as a .jpg file.

```
ld_b = adaptive_tone_mapping(hdr_b, gamma_b, bb) # bb is the parameter
b in the above equation for channel b
ld_g = adaptive_tone_mapping(hdr_g, gamma_g, bg) # bg is the parameter
b in the above equation for channel g
ld_r = adaptive_tone_mapping(hdr_r, gamma_r, br) # br is the parameter
b in the above equation for channel r

ldr = np.zeros(hdr.shape, dtype=np.int8)
ldr[:, :, 0] = ld_b/(np.max(ld_b)-np.min(ld_b))*255
ldr[:, :, 1] = ld_g/(np.max(ld_g)-np.min(ld_g))*255
ldr[:, :, 2] = ld_r/(np.max(ld_r)-np.min(ld_r))*255
ldr = ldr.astype('uint8')
#convert the ldr image into hsv space
cv2.imwrite('ldr.jpg', ldr)
```

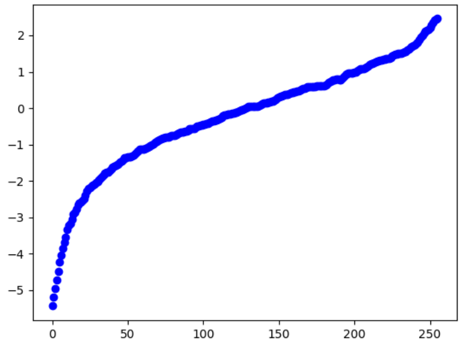
Image taking

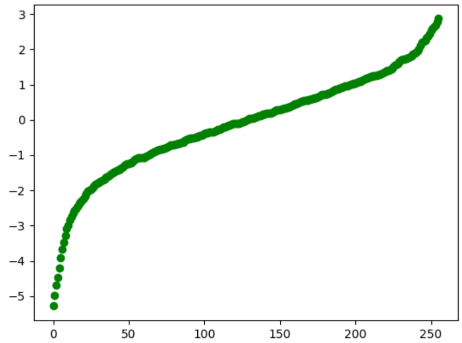
- Camera: Pentax KP
- Aperture: Fixed at f11
- Number of images: 9

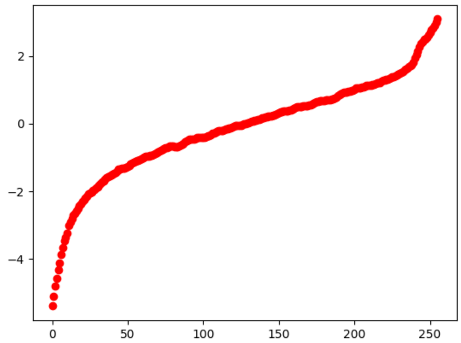
Result

HDR

- g curves

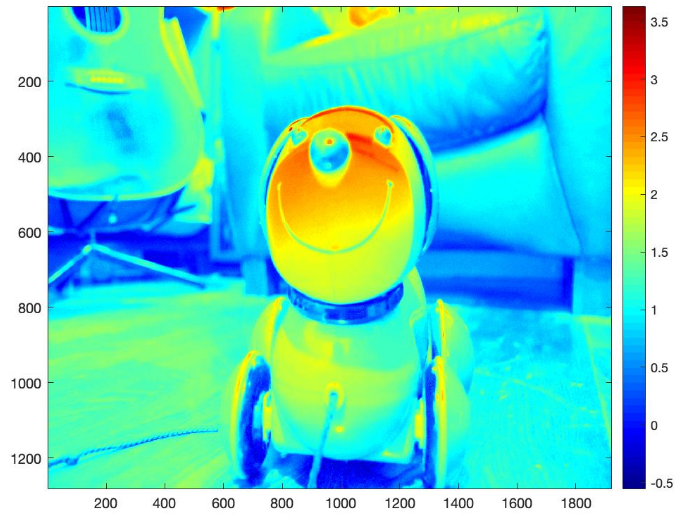




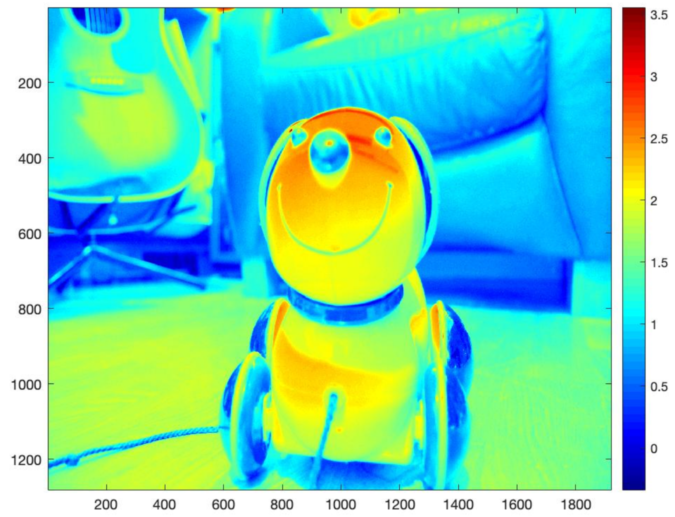


- From left to right, are the recovered g curves of channel blue, green and red.

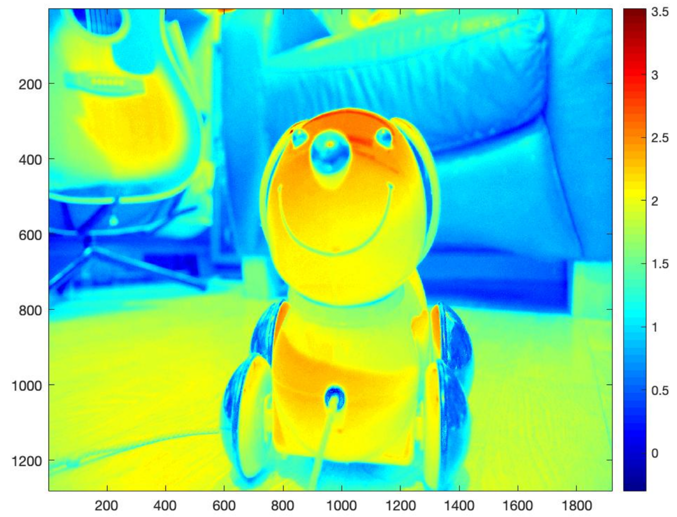
- HDR Image. The following HDR images are displayed with Matlab
    - Channel Blue



    - Channel Green

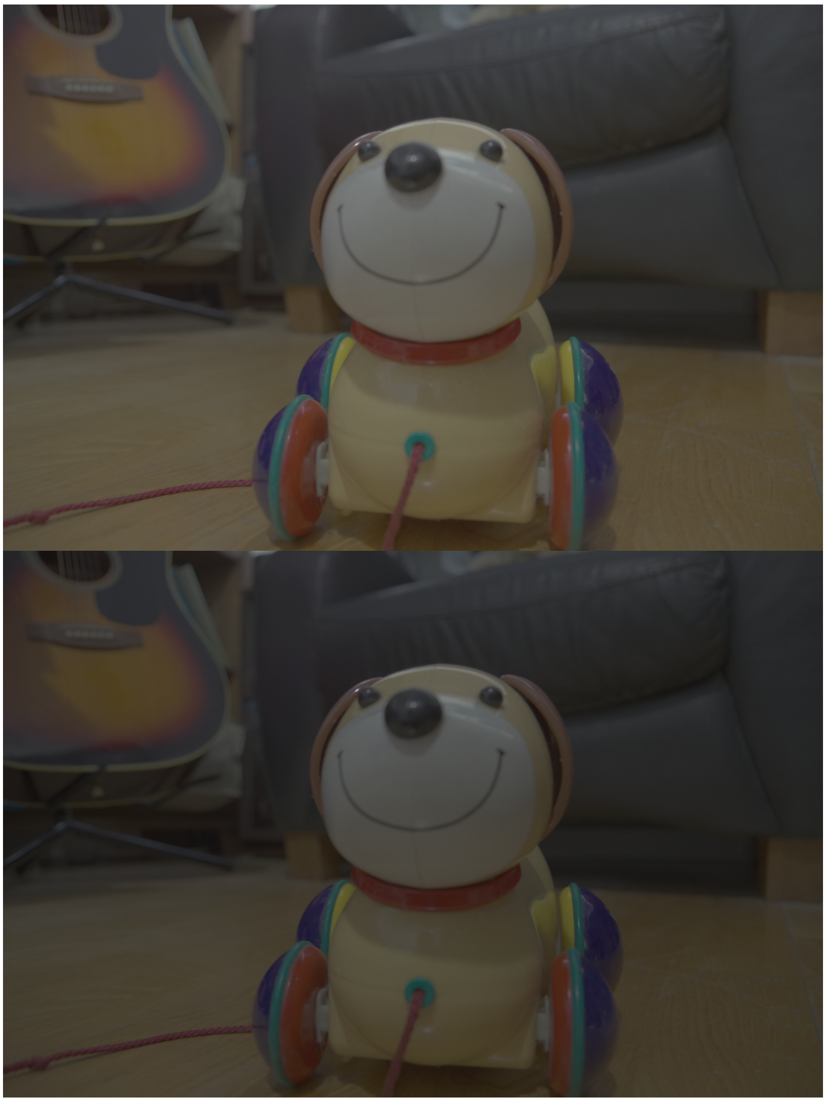


    - Channel Red.



Tone Mapping

- When parameter b is set to a higher value, the image looks darker.
  - Left: b =5, Right: b = 20 (gamma = 2)



- We can try to enhance a certain color by adjusting the parameters
  - `gb = 2; gg = 2; gr = 3;` (The larger gamma value for the red channel gives a redder image)

- Note for the following images: All the parameters are set to :

  ```
  bb = 10; bg = 10; br = 30;
  gb = 1.8; gg = 2.2; gr = 2.5;
  ```

  - I found out that if a devide L_{Wxax} by a certain factor, the image wouldn't be so washed up:
    - Before

    - After (devide L_{Wmax} by 40)

- I found out by adjusting the $log(0.5)$ in the tone mapping formula to $log(0.7)$, the image will give a contrasty look.
  -
  - Note: The last image is the one I will be submitting.

Reference

- Paul E. Debevec, Jitendra Malik, Recovering High Dynamic Range Radiance Maps from Photographs, SIGGRAPH 1997.
- Drago, Frédéric, et al. "Adaptive logarithmic mapping for displaying high contrast scenes." *Computer graphics forum*. Vol. 22. No. 3. Oxford, UK: Blackwell Publishing, Inc, 2003
- Matlab code for false color display: https://danialchitnis.com/hdr/index.html