Homework 1

Problem 0

- *Original image*

a

- *result1*

b

- *result2*

Problem 1

- *Original Image*

a

- *Motivation and Approach*

    - *Using cv2.imread() with flag v2.IMREAD_GRAYSCALE to read sample2.png as sample2*
    - *Simply divide all the pixel value in sample2 by 2 as result3*
- *result3*

- *Discussion*
    - *The image becomes darker but still can tell apart the mountains and clouds*
    - *Actually, after divided by 2, some of the pixel values in result3 become floating points, but they will be round to integers when written to result3.png*

---

b

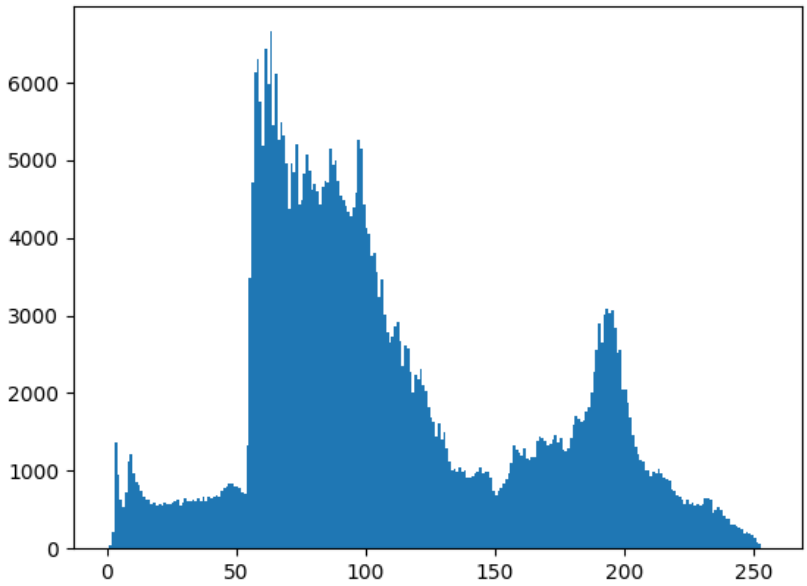
- *Motivation and Approach*
    - *Unlike problem (a), since if we simply multiply all the value in result3 by 3 as result4, some pixel values will become larger than 255*
    - *Instead, we use for loop to check all the pixel values after multiplication, if larger than 255, we'll set it to 255*
- *result4*

- *Discussion*
  - *The image becomes quite brighter · the mountians look more appealing but it becomes harder to tell the details of the clouds*
  - *Again, after muitiplied by 3 and modifying the pixel values larger than 255, some of the values in result4 become floating points, but they will round to integers when written to result4.png*

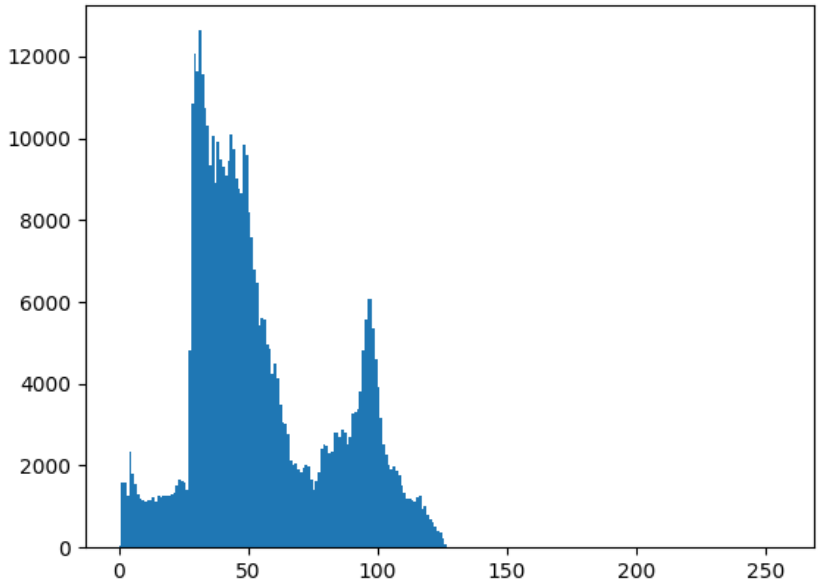
---

c

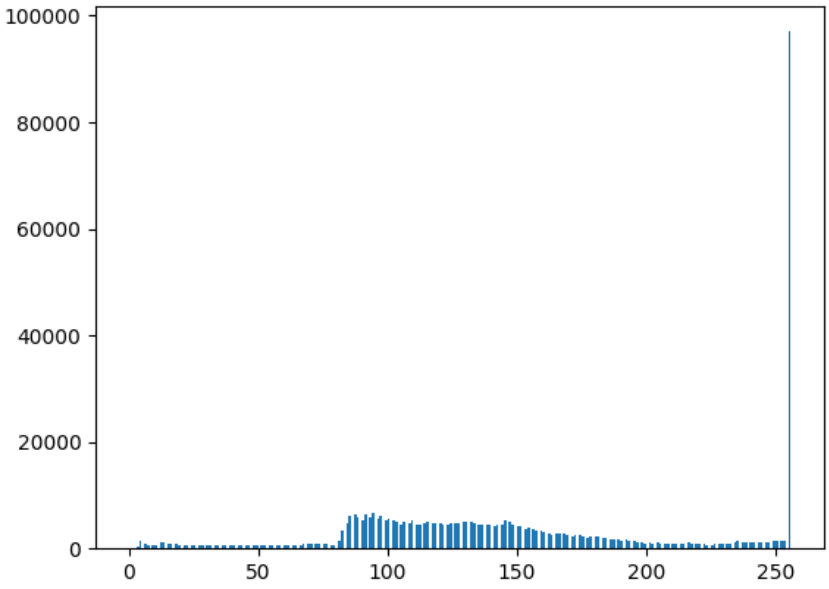
- *sample2*



- *result3*



- *result4*



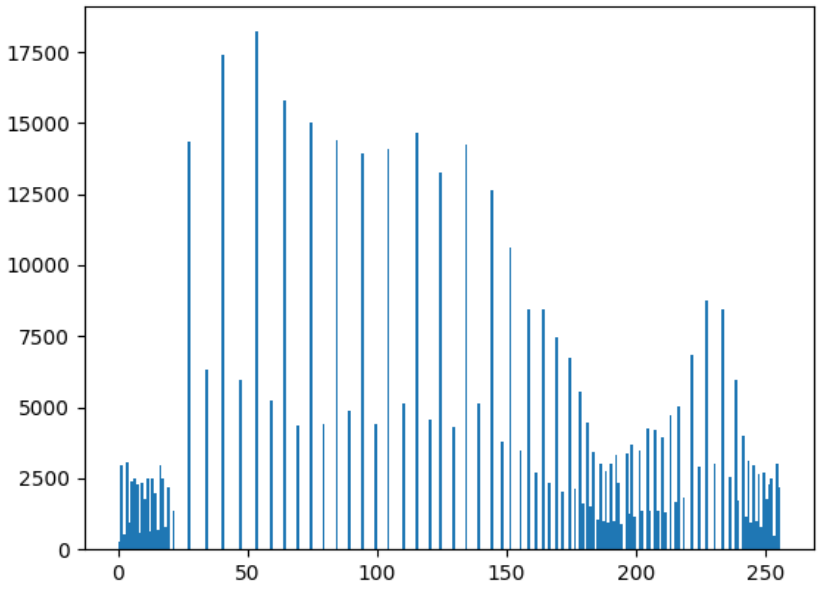
- *Discussion*
  - *The histogram of result3 looks like the histogram of sample2 be compressed horizontally and the pixel values seperate between 0 ~ 128*
  - *Ideally, the histogram of result4 should look like the histogram of result3 expands horizontally. However, since the pixel values should in range 0 ~ 255, lots of pixel values accumulate at 255*

---

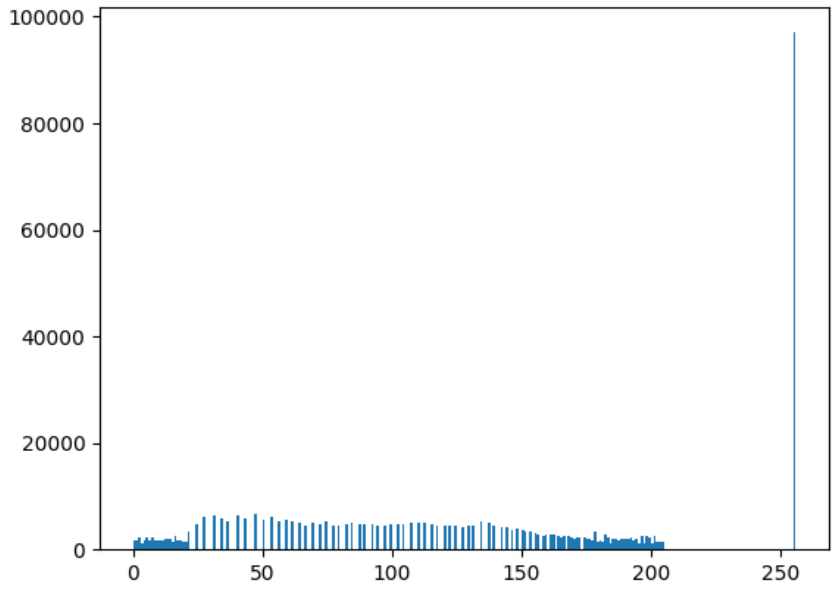
d

- *Motivation and Approach*
  - *Build $unif$ which stores the cdf of uniform distribution from 0 to 255*
  - *Construct a function call $global\_histogram\_equalization()$*
    - *Take a two-dimension ndarray $source$ as input and construct its cdf called cdf*
    - *Construct a $lookup$ table which map every pixel values (0~255) to another pixel values (0~255) according to the relationship between cdf and $unif$, following is the pseudo code*
      - $lookup[i] = unif^{-1}(cdf[i])$
    - *construct a two-dimension ndarray $output$ by mappping all the pixel values in $source$ according to $lookup$ table*
    - *return $output$*
  - *Use $global\_histogram\_equalization()$ to transform result3 and result4 to result5 and result6*

- *result5*



- *result6*



- *Discussion*
  - *Image*
    - *I think that result5 is more appealing since we can easily tell apart the layers of sky compare to the original one*
    - *In result6, the sky and mountains look not bad. However, the clouds is even brighter than result4 so that we can't tell apart any details*
  - *Histogram*
    - *Since result3's pixel value scatter in range 0~128, the global equalization map them to range 0~255 and make the layer of sky become more obviously.*
    - *However, result4's pixel values scatter in range 0 ~ 255 initially thus a lot of pixel value still accumulated at 255 after modification.*

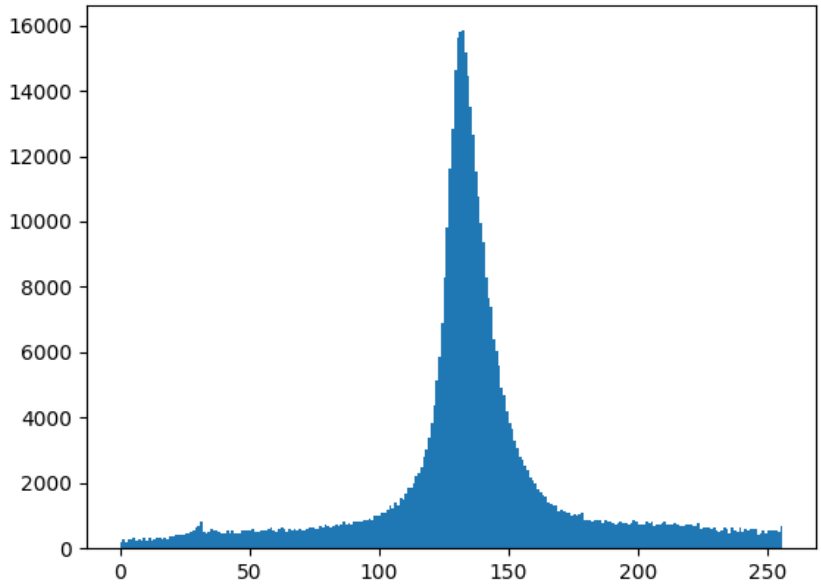
---

e

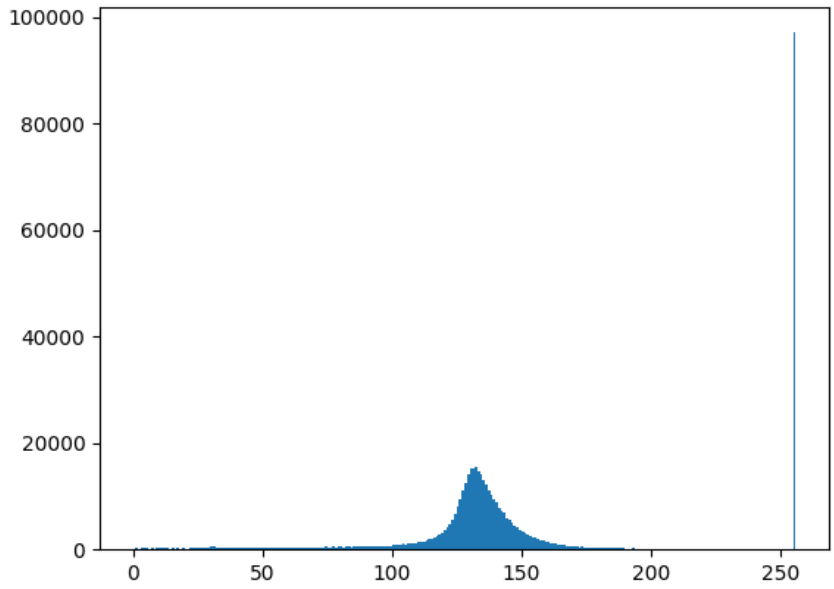
- *Motivation and Approach*
  - *Unlike problem d, we only consider the nearby pixel values when doing the histogram equalization.*

- *First, we padding the result3 by flipping the boundary pixel value by row and col according to window size*
- *Start from the left-top corner, calculate the cdf the center only according to the pixel values within the window*
- *Then, set center's pixel value according to the $unif$ in problem d*
  - $output[i][j] = unif^{-1}(paddind[floor(i + \frac{window}{2})][floor(j + \frac{window}{2})]'s\ cdf\ within\ padding[i : i + window][j : j + windo$
- *In my approach, I set window size to 101 since small window sizes will make the whole image looks wierd.*

- *result7*



- *result8*



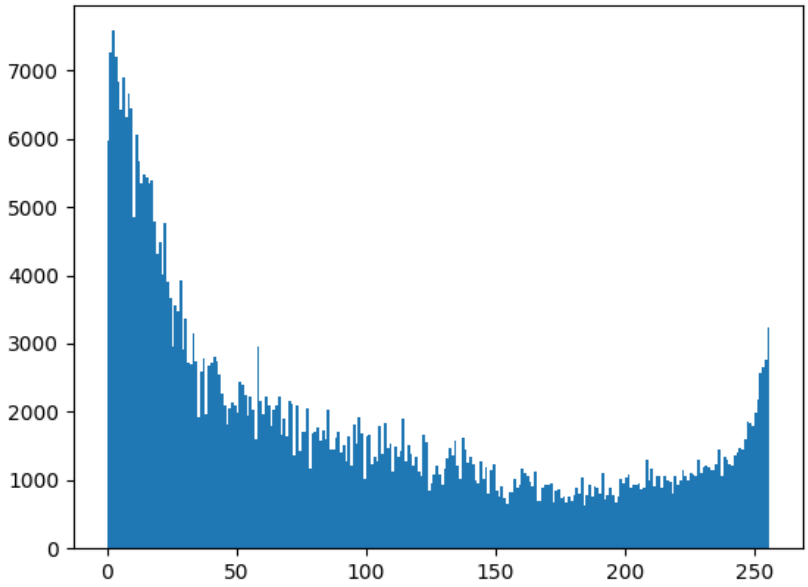
- *Discussion*
  - *Image*
    - *Both result7 and result8 look more stereoscopic after modification especially the clouds in result7. Nevertheless, the clouds in the result8 still to bright*
  - *Histogram*
    - *The histogram is quite interesting, most of the pixel values are accumulating at the half of max pixel value and the number decreasing as the pixel value.*
    - *That's because in average the nearby pixels will have close values, thus lots of pixel will be mapped to half of the largest value after histogram equalization*
    - *Though we can also see the trend in the histogram of result8. Since there are too many pixel values accumulate at 255 since no matter how large the window size is, the pixel value 255 will remain 255 after histogram equalization.*

---

f

- *Motivation and Approach*

    - *I tried to combined both global equalization and local equalization. Thus I do the global equalization window by window and overlap the area*
    - *The window is set to 100 without padding and slide the window every 10 pixels*
- *result9*



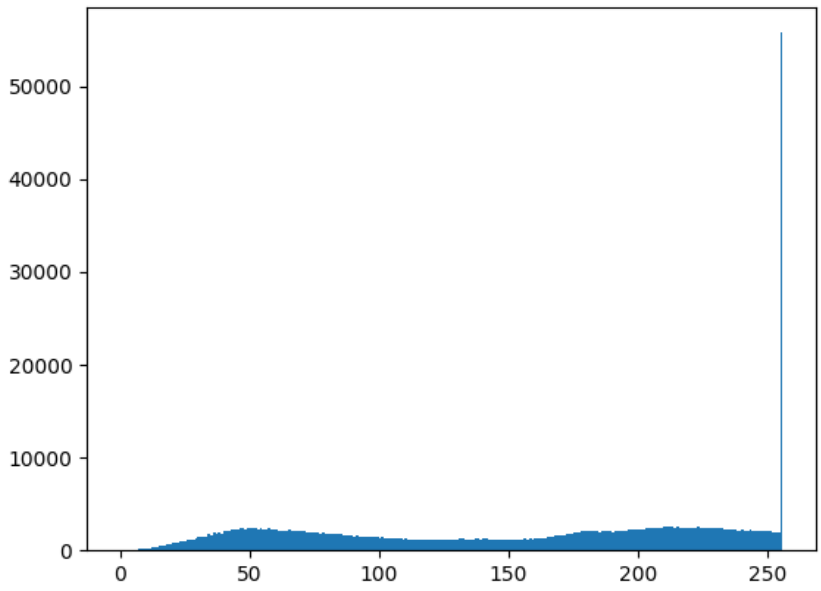
- *Discussion*
    - *Image*
        - *Unlike the local equalization in problem g · this time I mapping all the pixel value in the window. Thus there must 0 and 255 pixel value in every window. In the end the whole image looks like add a mosaic effect on it*
    - *Histogram*
        - *Different from local equalization, in these approach, more pixels value accumulate at 0 and 255 since I do the global equalization area by area.*

Problem 2

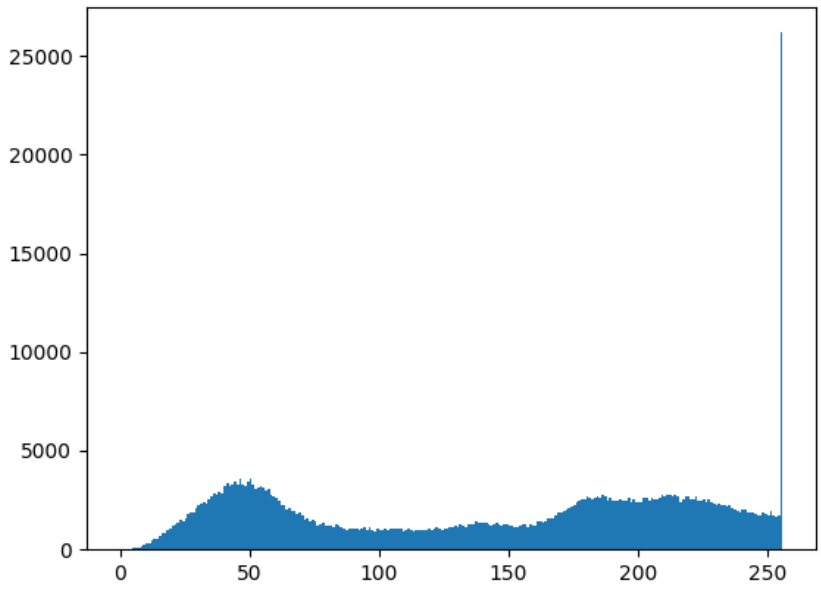
a

- *Motivation and Approach*

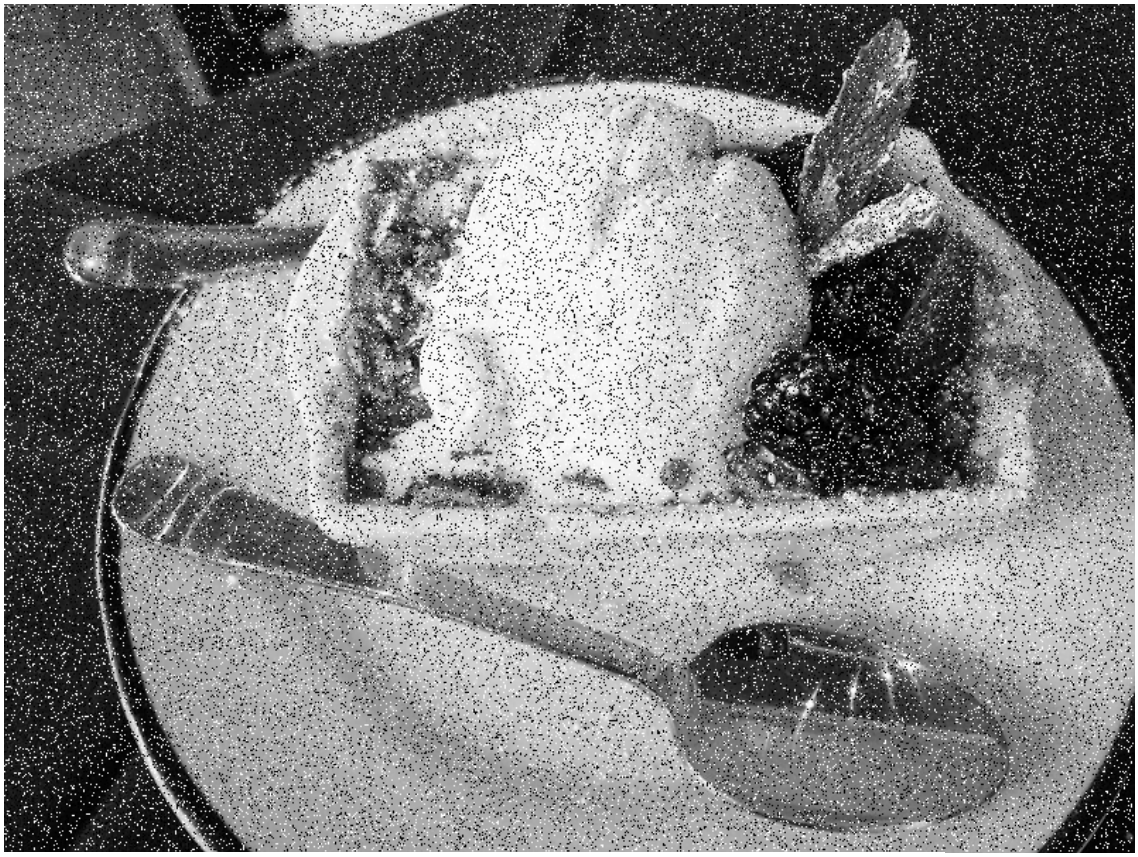
    - *I have tried all the noise removement approach in the lecture slide (low-pass, median, MAXMIN, MINMAX, PMED, MAXMIN of MINMAX) and aim to pick the most appealing result.*
    - *All the approaches are implementing with window sise 3 since I think any window size larger than 3 are too smooth*
- *sample4*

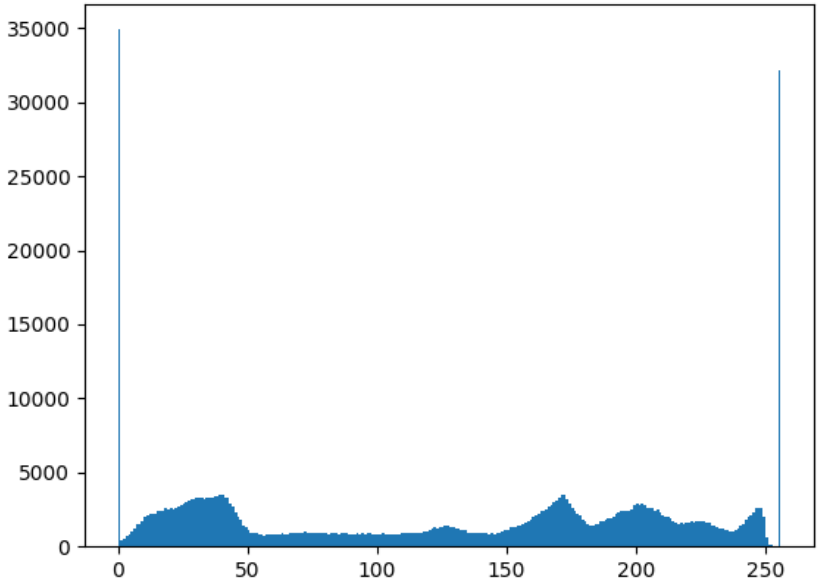


- *result10*

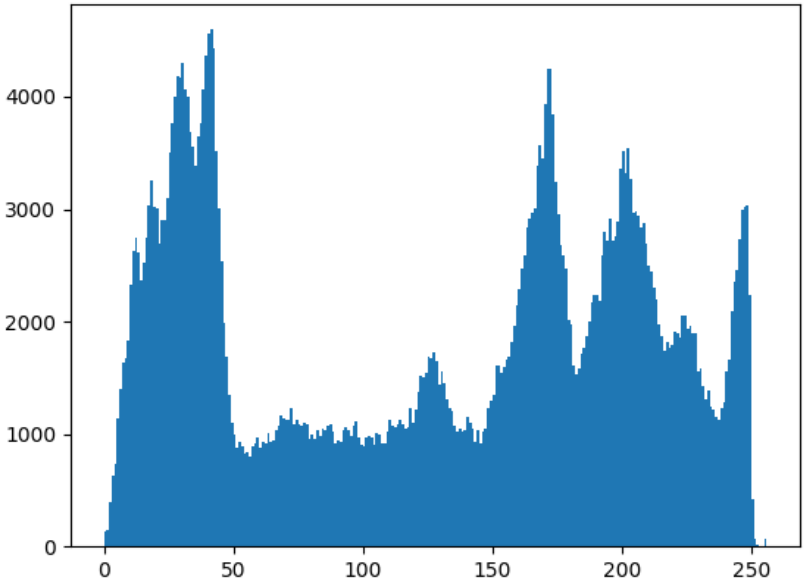


- *sample5*




35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250

- *result11*



- *Discussion*
  - *result10*
    - *In the histogram, the number of pixel value 255 decrease a lot. Thus, the whole picture looks more darker but I think that the noise removal doesn't work a lot*
    - *In the beginning, I use low-pass filtering to remove the high value pixel but the result doesn't look appealing for me. Thus, I use median filtering in the end although the noise still obvious.*
  - *result11*
    - *Obviously, it's impulse noise. Thus I try mean and median filtering at first but not satisfying with them. In the end, I choose MAXMIN after MINMAX to remove almost all the noise.*
    - *According to the histogram, the number of high pixel value and low pixel value actually drop a lot*

b

- *PSNR*
  - *sample4: 28.03843185645809*
  - *sample5: 36.21757688112266*
  - *result10: 28.23885244892262*

    - *result11: 36.252581121719174*
- *Discussion*
    - *In my implementation, both images' PSNR become a little higher after noise removal and they are actually become more appealing.*
    - *However, I think that PSNR is just a objective standard which reflects the overall performance. Since some of the output during the coding have higher PSNR but not that appealing to me. Thus, I think that it depends on viewer in the end.*

tags: `DIP`