TÓM TẮT NỘI DUNG TUẦN 1-2

I. Các khái niệm

1) Xử lí ảnh là gì?

- Là 1 phần của lĩnh vực xử lí tín hiệu số
- Tăng cường chất lượng thông tin hình ảnh đối với quá trình tri giác của con người và biểu diễn trên máy tính

2) Các giai đoạn của xử lí ảnh

- Thu nhận ảnh
- Số hóa ảnh
- Phân tích ảnh
- Đối sánh nhận dạng

3) Các thiết bị cơ bản trong xử lí ảnh

- Camera
- Bộ xử lí tương tự
- Bộ xử lí ảnh số
- Máy chủ
- Bộ nhớ ngoài

4) Điểm ảnh (Pixel)

- Mỗi điểm ảnh là một cặp tọa độ (x,y) và màu sắc.
- Các điểm ảnh tạo nên độ phân giải.

5) Mức xám (Gray)

- Mức xám là kết quả của việc mã hóa ứng với một cường độ sáng của mỗi điểm ảnh với một giá trị số.

6) Ảnh (Image)

- Tập hợp các điểm ảnh được biểu diễn thông qua mảng hai chiều I(n,p) : n dòng và p cột. Kí hiệu I(x,y) để chỉ giá trị mức xám của điểm ảnh tại vị trí tọa độ (x,y).
- Một hàm hai biến f(x,y), trong đó (x,y) là tọa độ trong không gian hai chiều và f là độ lớn tại tọa độ (x,y) được gọi là mức xám của ảnh tại điểm đó.
- Khi mỗi điểm (x,y) trong không gian hai chiều biểu diễn cấp xám có độ lớn f hữu hạn, xác định và được lượng hóa rời rạc, ta gọi dó là ảnh số.
- Một ảnh màu có thể được viết dưới dạng : $f(x,y)=\begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$
- Xử lí ảnh tập trung vào 2 nhiệm vụ chính :
  - Cải thiện thông tin hình ảnh
  - Xử lí dữ liệu hình ảnh để lưu trữ, truyền dẫn…

7) Biểu diễn ảnh (Image Representation)

- Các phần tử đặc trưng cơ bản của ảnh là điểm ảnh
- Các mô hình thường sử dụng:

- Mô hình toán học – biểu diễn ảnh thông qua các hàm hai biến trực giao
- Mô hình thống kê – biểu diễn thông qua các đại lượng kỳ vọng, phương sai, moment

8) Tăng cường ảnh (Image Enhancement)

- Làm nổi bật các đặc trưng đã chọn
- Các kỹ thuật được chọn:
  - Lọc độ tương phản
  - Khử nhiễu
  - Nổi màu
  - Nổi biên
  - Giãn độ tương phản

9) Khôi phục ảnh (Image Restoration)

- Loại bỏ hay tối thiểu hóa các ảnh hưởng của môi trường bên ngoài hay hệ thống thu nhận ảnh gây ra
- Kết quả thu được là ảnh gần giống với ảnh gốc

10) Biến đổi ảnh (Image Transform)

- Sử dụng một lớp các ma trận đơn vị
- Các kĩ thuật thường sử dụng : Biến đổi Fourier, Sin, Cos, Karhumen Loeve,…

11) Phân tích ảnh (Image Analyze)

- Xác định độ đo định lượng của ảnh để đưa ra mô tả đầy đủ về ảnh
- Các kỹ thuật thường sử dụng :
  - Kỹ thuật lọc
  - Kỹ thuật tách
  - Kỹ thuật hợp dựa trên các tiêu chuẩn đánh giá về màu sắc, cường độ, kết cấu,…
  - Các kỹ thuật phân lớp dựa trên cấu trúc khác

12) Nén ảnh (Image Compression)

- Tìm cách loại bỏ thông tin dư thừa trong ảnh gốc để làm giảm dung lượng lưu trữ, thuận lợi cho việc truyền dữ liệu
- Có 2 phương pháp nén ảnh :
  - Nén ảnh không mất mát thông tin - ảnh sau khi nén được khôi phục giống hệt ảnh gốc
  - Nén ảnh có mất mát thông tin - ảnh sau khi nén được khôi phục gần giống ảnh gốc

13) Nhận dạng ảnh (Image Recognition)

- Là quá trình phân loại đối tượng được biểu diễn theo một mô hình nào đó và gán chúng vào một lớp dựa theo những quy luật và các mẫu chuẩn
- Nhận dạng áp dụng trong việc bảo mật, an ninh, nhận dạng chữ viết,…
- Các phương pháp :
  - Nhận dạng dựa vào phân hoạch không gian
  - Nhận dạng theo cấu trúc
  - Nhận dạng dựa theo mạng nơron
  - Mô hình Markov ẩn

14) Tra cứu ảnh (Image Retrieval)

- Tìm các ảnh thỏa mãn các yêu cầu cho trước trong một cơ sở dữ liệu lớn
- Tra cứu ảnh được áp dụng trong thư viện, y học, hệ thống an ninh, bảo mật,…
- Có 2 kỹ thuật :
  - Dựa trên từ khóa
  - Dựa trên nội dung

15) Một số phép biến đổi cấp xám cơ bản
- Phủ định ảnh - ảnh âm bản
- Phép biến đổi log
- Phép biến đổi lũy thừa
- Các phép biến đổi tuyến tính từng phần

16) Các khái niệm cơ bản về lọc không gian
- Các bộ lọc không gian làm trơn :
  - Lọc tuyến tính
  - Lọc thống kê thứ tự
- Các bộ lọc không gian làm nét :
  - Sử dụng đạo hàm bậc hai – toán tử Laplacian
  - Sử dụng đạo hàm bật nhất – toán tử Gradient
- Miền không gian :
  - Là tập hợp các điểm ảnh trong ảnh
  - Phương pháp miền không gian là một thủ tục tác động lên điểm ảnh trong miền không gian đó
- Miền tần số :
  - Được biểu diễn theo tần số thông qua các phép biến đổi
  - Phương pháp miền tần số dựa trên phép biến đổi Fourier của ảnh
  - Tương ứng với các miền này $\rightarrow$ các phương pháp tăng cường ảnh

17) Xử lí trong miền không gian
- Các phép xử lí trong miền không gian tác động trực tiếp lên điểm ảnh được ký hiệu là : g(x,y)=T[f(x,y)]
  - f(x,y) là ảnh đầu vào
  - g(x,y) là ảnh đầu ra
  - T là toán tử tác động lên f trong lân cận của điểm (x,y)
- Lân cận của điểm (x,y) là hình vuông hoặc hình chữ nhật có tâm là (x,y)
- Tâm của lân cận (x,y) di chuyển theo từng điểm ảnh bắt đầu từ gốc trái phía trên (gốc tọa độ)
- Toán tử T hoạt động tại mỗi vùng lân cận của vị trí điểm ảnh (x,y) trong ảnh f để cho ảnh đầu ra g tương ảnh
- T tác động lên vùng lân cận có kích thường 1x1 (tác động lên điểm đơn) $\rightarrow$ g chỉ phụ thuộc vào giá trị của f tại điểm (x,y), T trở thành hàm biến đổi cấp xám có dạng : s=T(r)
  - r=f(x,y)
  - s=g(x,y)

→ Kỹ thuật này được gọi là kỹ thuật xử lí điểm (Point Processing)

- Đối với những lân cận lớn hơn 1x1 việc xử lí điểm ảnh phức tạp hơn nhiều
- Một lân cận có kích thước lớn hơn 1x1 được gọi là một mặt nạ, hoặc bộ lọc, hoặc mẫu, hoặc cửa sổ
- Các giá trị trong mặt nạ được gọi là các hệ số của mặt nạ

→Kỹ thuật này được gọi là kỹ thuật xử lí mặt nạ hay kỹ thuật lọc (Mask Processing)

II. Thực hiện một số phép toán trên ảnh

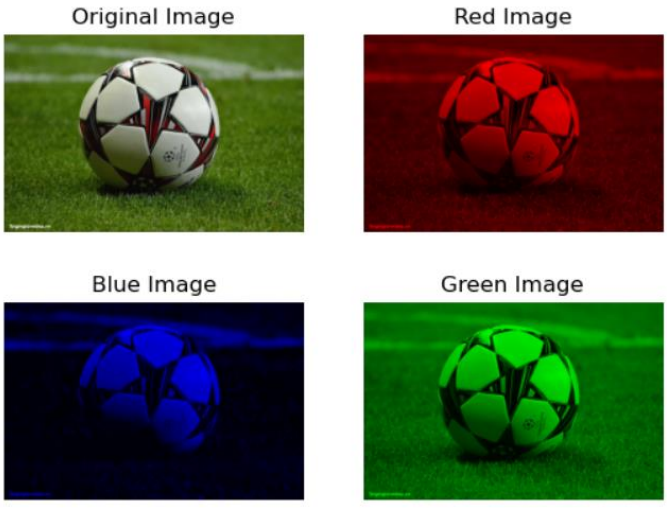
0) import matplotlib.pyplot as plt
   import numpy as np

1) Open Image (Mở ảnh)

```
im = plt.imread('bong.jpg')
imlist=[im]
plt.imshow(im);
```

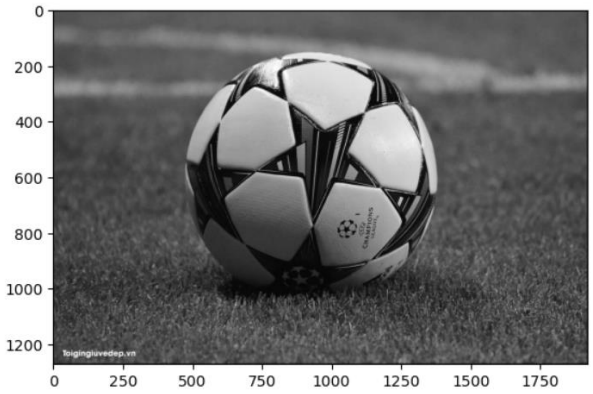
2) Color Image (Ảnh màu)

```
imR=np.zeros_like(im)
imR[:,:,0]=im[:,:,0]
imG=np.zeros_like(im)
imG[:,:,1]=im[:,:,1]
imB=np.zeros_like(im)
imB[:,:,2]=im[:,:,2]
imlist=[im,imR,imB,imG]
mapping={0:'Original Image',1:'Red
Image',2:'Blue Image',3:'Green Image'}
fig,ax=plt.subplots(2,2)
ax=ax.flatten()
for i in range(4):
ax[i].imshow(imlist[i])
ax[i].set_title(mapping[i])
ax[i].axis('off')
```



3) Gray Image (Ảnh xám)

```
from PIL import Image
gray=np.array(Image.open
('bong.jpg').convert('L'))
print(gray)
plt.imshow(gray, cmap='gray');
```



4) Negative Image (Phép biến đổi âm bản)

```
def negative(gray):
  return 255-gray
plt.subplot(1,2,1)
plt.imshow(gray,cmap='gray')
plt.title("Original Image")
plt.axis('off')
neg=negative(gray)
plt.subplot(1,2,2)
plt.imshow(neg, cmap='gray')
plt.title("Negative Image")
plt.axis('off');
```

5) Thresholding (Phân ngưỡng)

```
plt.subplot(1,2,1)
plt.imshow(gray,cmap='gray')
plt.title("Before")
plt.axis('off')
m=150
gray_ = gray.copy()
for i in range(gray.shape[0]):
  for j in range(gray.shape[1]):
```

```
    if gray_[i,j] > m:
        gray_[i,j]=255
    else:
        gray_[i,j]=0
plt.subplot(1,2,2)
plt.imshow(gray_,cmap='gray')
plt.title("After")
plt.axis('off');
```

6) Log Transformation (Phép biến đổi log)

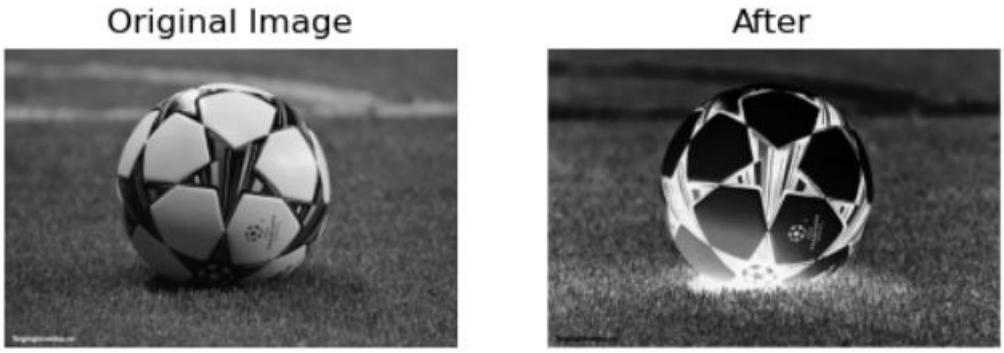
```
def log(gray,c=1):
    return np.log10(1+gray)*c
log1=log(gray,c=1)
log2=log(gray,c=2)
log3=log(gray,c=3)
log_list=[gray,log1,log2,log3]
fig,ax=plt.subplots(2,2)
ax=ax.flatten()
mapping={0:'Before',1:'c=1',2:'c=2',3:'c=3'}
for i in range(4):
    ax[i].imshow(log_list[i], cmap='gray')
    ax[i].set_title(mapping[i])
    ax[i].axis('off') ;
```

7) Power-law Transformation (Phép biến đổi lũy thừa)

```
def power(gray,c=1,w=0.5):
   return c*gray**w
powergray=power(gray/255,w=0.2)
plt.subplot(1,2,1)
plt.imshow(gray,cmap='gray')
plt.title("Original Image")
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(powergray,cmap='gray')
plt.title ('After')
plt.axis('off');
```

8) Another constrast stretching function (Một chức năng khác để kéo dài độ tương phản)

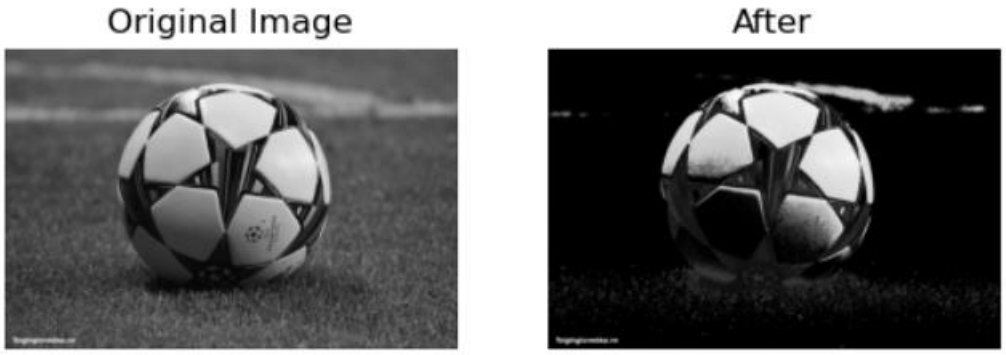
```
def s(gray,m=0.2,E=0.9):
   return 1/(1+(m/gray)**E)
sgray=s(gray/255,E=-2)
plt.subplot(1,2,1)
plt.imshow(gray,cmap='gray')
plt.title("Original Image")
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(sgray,cmap='gray')
plt.title('After')
plt.axis('off') ;
```



9) Piece-wise Linear Transformation (Giãn độ tương phản)

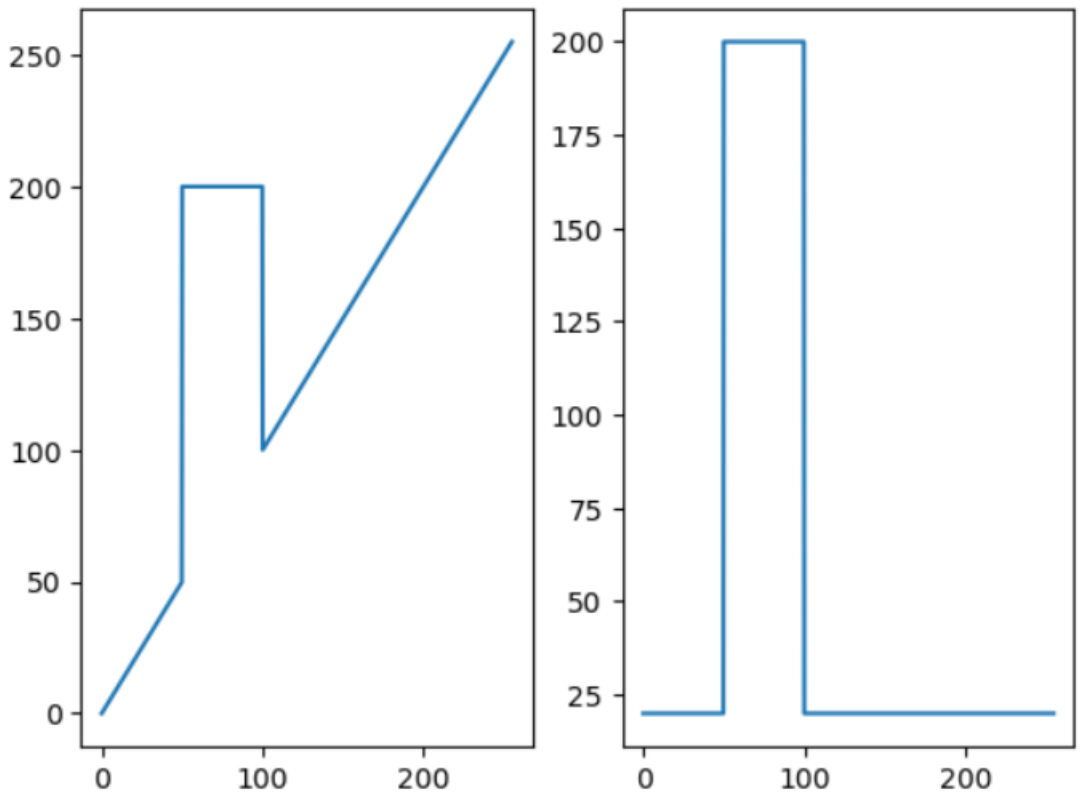
```
# nếu r1=s1,r2=s2 => phép biến đổi đồng nhất
# nếu s1=0, s2=255 => phân ngưỡng
# thông thường ta xét r1 <= r2 và s1 <= s2
# r1=40,r2=100,s1=80,s2=160
# d1:y=2x
# d2: y=(4x+80)/3
# d3: y=(19x+3060)/31
grays=np.zeros_like(gray)
for i in range(gray.shape[0]):
  for j in range(gray.shape[1]):
    if gray[i,j] < 40:
       grays[i,j]=2*gray[i,j]
    if 40 <= gray_[i,j] <= 100:
       grays[i,j]=(4*gray[i,j]+80)/3
    if 100 < gray_[i,j] <=255:
       grays[i,j]=(19*gray[i,j]+3060)/31
plt.subplot(1,2,1)
plt.imshow(gray, cmap='gray')
plt.title("Original Image")
plt.axis('off')
```

```
plt.subplot(1,2,2)
plt.imshow(grays,cmap='gray')
plt.title('After')
plt.axis('off');
```



10) Intensity Level Slicing (Làm mỏng mức xám)

```
def intensity_level_slicing(r,high,A=50,B=100,low=None,reduce=False):
    arr=r.copy()
    if reduce == True and low != None:
        arr=np.ones_like(arr)*low
    arr[np.where((r>=A)&(r<=B))]=high
    return arr
x=np.linspace(0,255,1000)
fig,ax=plt.subplots(1,2)
ax=ax.flatten()
ax[0].plot(x,intensity_level_slicing(x,high=200))
ax[1].plot(x,intensity_level_slicing(x,high=200,low=20,reduce=True));
```

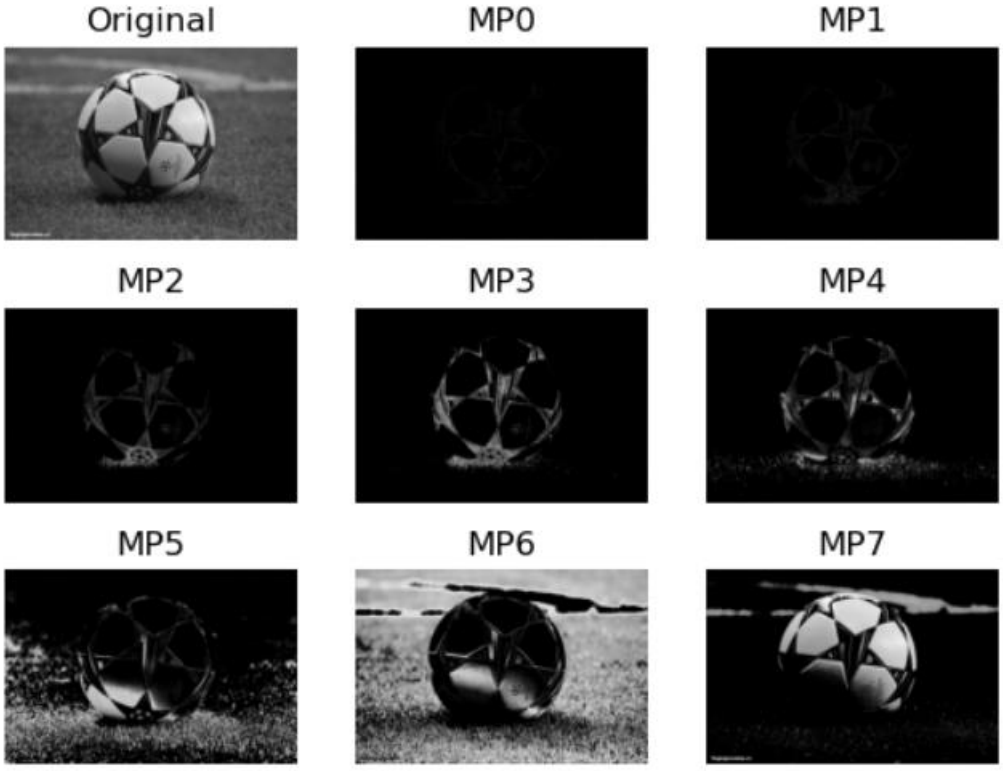



```
mapping={0:'original',1:'reduce=True',2:'reduce=False'}
slicing_list=[gray,intensity_level_slicing(gray,high=200,reduce=True),intensity_level_slicing(g
ray,high=200,reduce=False)]
fig1,ax1=plt.subplots(1,3)
ax1=ax1.flatten()
for i in range(3):
    ax1[i].imshow(slicing_list[i], cmap='gray')
    ax1[i].set_title(mapping[i])
    ax1[i].axis('off');
```

11) Bit-plane Slicing (Làm mỏng mặt phẳng bit)

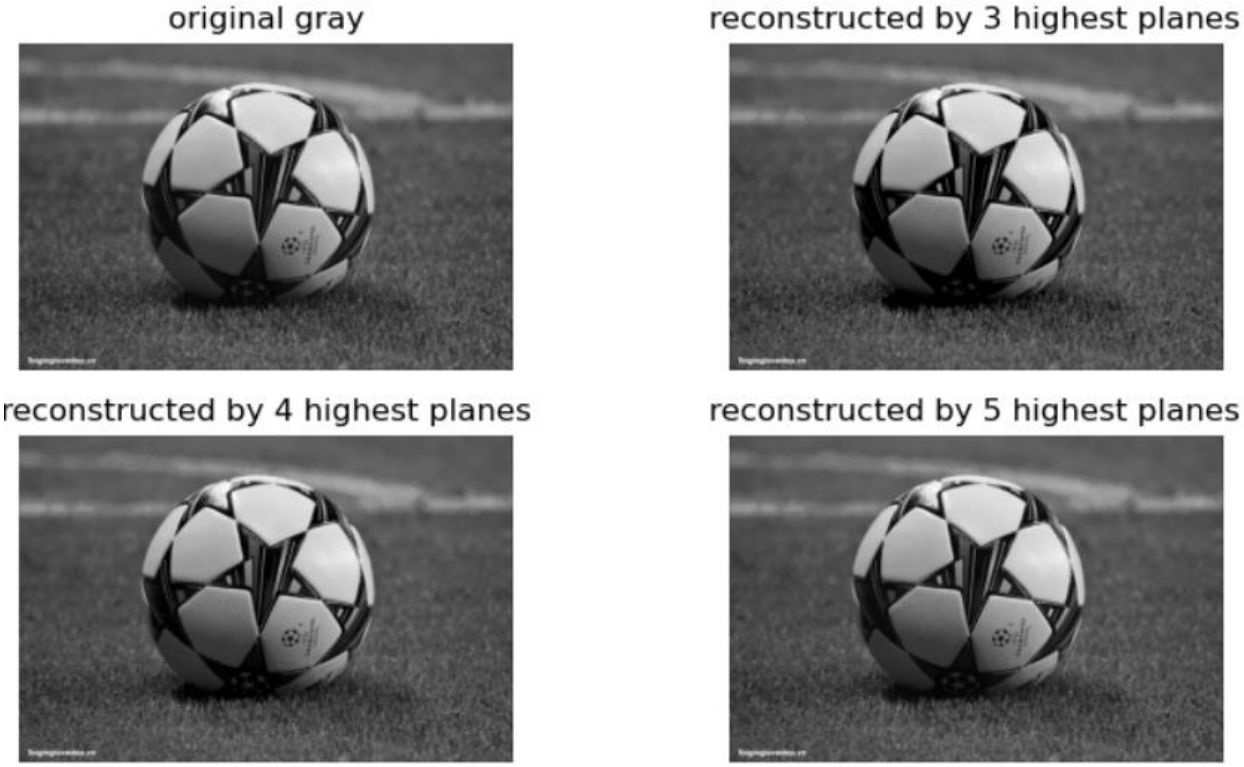
```
def bit_plane_slicing(r):
    bit_list=[]
    for i in range(8):
        arr=np.zeros_like(r)
        arr[np.where((r>=2**i)&(r<=2**(i+1)))]=r[np.where((r>=2**i)&(r<=2**(i+1)))]
```

```
        bit_list.append(arr)
    return bit_list
bit_list=[gray]
bit_list.extend(bit_plane_slicing(gray))
mapping={0:'Original',1:'MP0',2:'MP1',3:'MP2',4:'MP3',5:'MP4',6:'MP5',7:'MP6',8:'MP7'}
fig,ax=plt.subplots(3,3)
ax = ax.flatten()
for i in range(9):
    ax[i].imshow(bit_list[i],cmap='gray')
    ax[i].set_title(mapping[i])
    ax[i].axis('off');
```



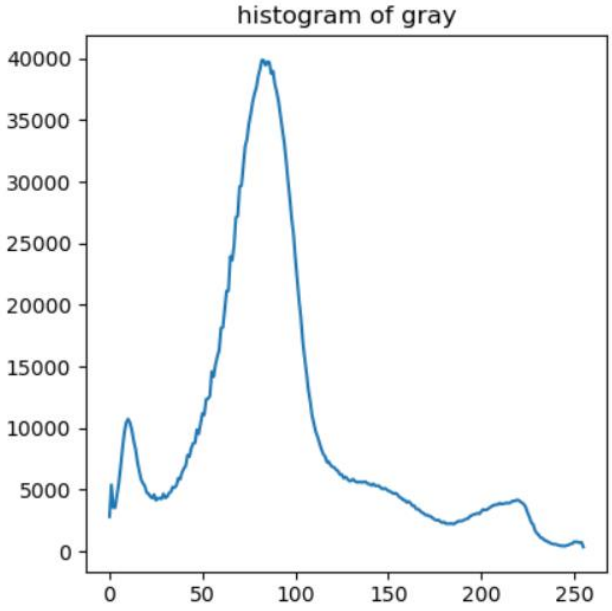

```
#3,4 and 5 highest bitplanes
mp678=bit_list[8]+bit_list[7]+bit_list[6]
mp5678=bit_list[8]+bit_list[7]+bit_list[6]+bit_list[5]
mp45678=bit_list[8]+bit_list[7]+bit_list[6]+bit_list[5]+bit_list[4]
mp_list=[gray,mp678,mp5678,mp45678]
mp_mapping={0:'original gray',1:'reconstructed by 3 highest planes',2:'reconstructed by 4
highest planes',3:'reconstructed by 5 highest planes'}
fig1,ax1=plt.subplots(2,2)
fig1.set_size_inches(9, 5)
ax1=ax1.flatten()
for i in range(4):
```

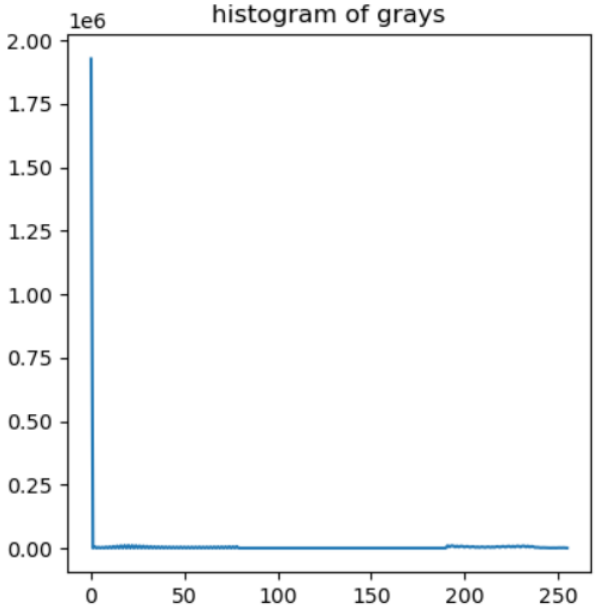
```
    ax1[i].imshow(mp_list[i],cmap='gray')
    ax1[i].set_title(mp_mapping[i])
    ax1[i].axis('off');
```

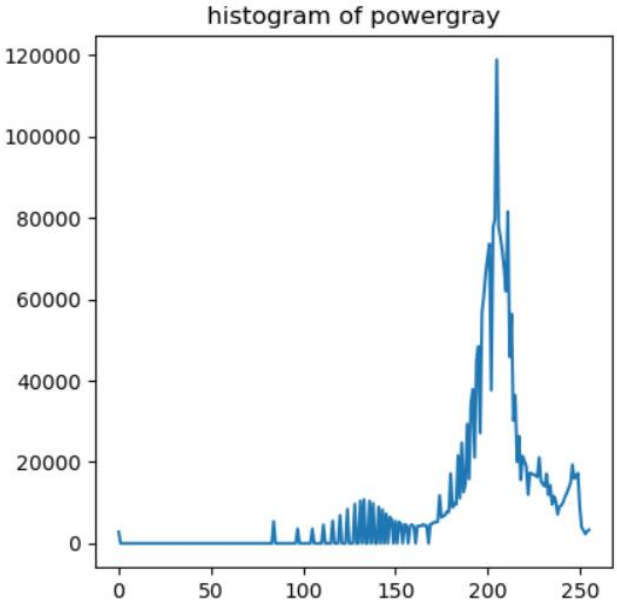


12) Histogram (Lược đồ xám)

```
hist_list=[gray,grays,powergray*255]
mapping={0:'histogram of gray',1:'histogram of grays',2:'histogram of powergray'}
fig, ax = plt.subplots(3,2, figsize = (10,16))
for i in range(3):
    hist = np.histogram(hist_list[i].flatten(), bins=256, range=[0,255])
    ax[i,0].plot(hist[0])
    ax[i,0].set_title(mapping[i])
    ax[i,1].imshow(hist_list[i], cmap = 'gray')
    ax[i,1].axis('off');
```


histogram of gray
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250


histogram of grays
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
250


histogram of powergray
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250