

# بسم الله الرحمن الرحيم



محمد عرفان زارع زردینی

۹۸۴۱۱۴۳۲

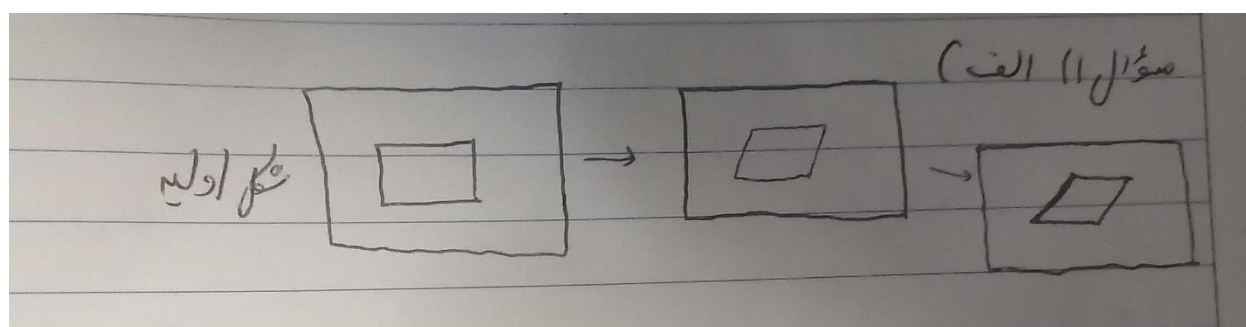
تمرین سری یک درس بینایی ماشین

نکته : تصاویر خروجی کد ها در کد ها هست ولی توضیحات کد ها بیشتر در **همینجا** موجود است

(1)

(الف)

اگر از بالای تصویر شکل را بررسی کنیم و به سمت پایین حرکت نماییم ، میبینیم که تصویر به سمت چپ متمایل می شود. به این دلیل که در این سبک دوربین، حسگر عکس برداری مان قطعی می باشد. و از بالا شروع به ثبت تصاویر می نماید برای وضوح بیشتر مفهوم موضوع و مبث بیان شده ، تصویر زیر کمک میکند.

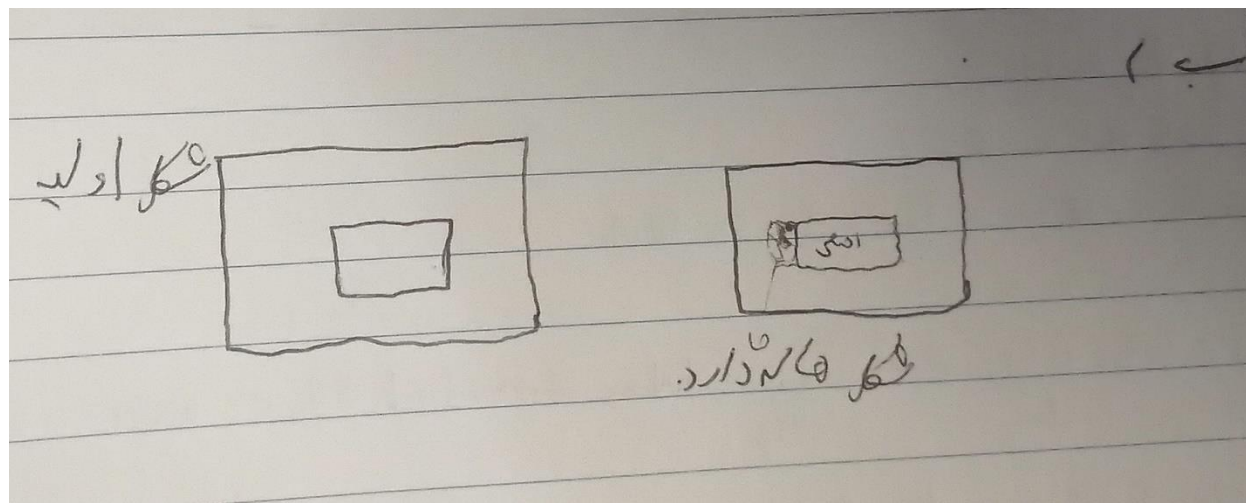


همانطور که مشخص هست وقتی جسم به چپ میرود، ما میبینیم که در عکس نهایی قسمت پایین تر تصویر جسم مایل به چپ شده است.

(ب)

با توجه به اینکه در **global shutter** ، حسگر عکس برداری به شکل آرایه ای می باشد (حسگر های عکس همزمان تصویر را ثبت می کنند، نه هر خط را از بالا به ترتیب)، و

اینکه در این بخش سرعت شاتر کم می باشد (یعنی طول مدت باز بودن زیاد است و نور زیادی را جذب می کند)، پس گویی تصویر کشیده تر می شود و گویی نویز و هاله دارد. همچنین تصویر کشیده می شود در این که جابجا شده به سمت راست ولی پشت آن (سمت پپ شکل) هاله هست. شکل آن مانند زیر است :



(2)

(الف)

با توجه به تصویر موجود در پوشه تمرین و کشیدن آن روی برگه که در عکس و با توجه به داده های صورت سوال و فرمول های موجود در عکس فرمول ها، و متغیر بودن فاصله کانونی به حل سوال می پردازیم و با جایگذاری به یافتن پاسخ ها می پردازیم.

$$\frac{y}{Y} = \frac{v}{u}$$

$$\frac{y}{Y} = \frac{v - f}{f}$$

$$\frac{1}{f} = \frac{1}{v} + \frac{1}{u}$$

نکته: توپ بیس بال در عکس جا مونده مناسبش (که فواسته هم نشده) ولی در زیر نوشته

$$\frac{1}{10} + \frac{1}{70} = \frac{1}{f} \Rightarrow \frac{1}{f} = \frac{8}{70} \Rightarrow f = 8.75 \text{ شده.}$$

سوال ۲

الف) لنز قابلیت تغییر فاصله کانونی (f): فرض را دارای باشد.

فاصله ممتد فیلیم تا لنز 10 cm

فاصله توپ به تا لنز 70 cm

بیس بال

فرض:  $\frac{1}{f} = \frac{1}{v} + \frac{1}{u}$

ب) شخصی شده توپ بکبال با فاصله 100 از صفحه فیلیم باشد:

فاصله توپ = 100 - 10 = 90 cm

نکته: 1) فاصله ممتد فیلیم تا لنز هست 10 فاصله لنز تا جسم هست. فاصله کانونی باشد.

2)  $\frac{1}{f} = \frac{1}{10} + \frac{1}{90} = \frac{10}{90} = \frac{1}{9} \Rightarrow f = 9$

ب) شخصی شده توپ بکبال و در فاصله 90 cm از توپ بکبال باشد.

فاصله توپ = فاصله توپ + فاصله توپ بکبال = 100 cm

فاصله توپ بکبال = 100 - 90 = 10 cm

فاصله توپ بکبال تا لنز = 100 - 10 = 90 cm

از نکته 1 و 2:  $\frac{1}{f} = \frac{1}{10} + \frac{1}{90} = \frac{10}{90} = \frac{1}{9} \Rightarrow f = 9$

فاصله توپ بکبال تا لنز = 100 - 10 = 90 cm

## References:

<https://noornegar.com/blog/aperture-and-depth-of-field-principles-of-focusing/>

<https://noornegar.com/blog/diaphragm-and-dof/>

<https://www.didnegar.com/mag/what-is-aperture-in-photography/>

<https://www.didnegar.com/mag/what-is-aperture-in-photography/>

<https://amoozal.com/mag/depth-of-field-photography/#lwptoc2>

دیافراگم (دریچه در دوربین) علاوه بر تار کردن یک گراند و ایجاد ابعاد و حجم جدید ، نوردهی تصاویر را نیز تنظیم و تغییر می دهد و میتواند تصاویر روشنتر یا تیره تر شود( می شود گفت از دریچه نور عبور و وارد دوربین می شود ). همچنین با تنظیم دیافراگم ، می توان عمق میدان را در تصویر تنظیم نمود و تغییر داد .

عمق میدان به فاصله ای گفته می شود که عناصر تصویر واضح و فوکوس روی عناصر در اون فاصله که مشخص شده خواهند بود (به فاصله که میان نزدیکترین و دورترین اشیا یک عکس که هر دو شارپ هستند) .



دیافراگم به عکاس اجازه میدهد که مقدار بزرگی و کوچکی بازه فاصله را مشخص کند و با بزرگ و کوچک کردن دریچه لنز روی میزان نور وارده به عکس تاثیر میگذارد (هرچه دریچه بزرگتر عکس روشنتره و بالعکس). همچنین هرچه دریچه **بزرگتر**، فاصله ای که عکس های واضح تولید می نماید، **کوچکتر** است و همچنین هرچه دریچه کوچکتر شود، بازه فاصله ای که عکس تولید میکند بزرگتر می شود و همچنین اشیایی که از سوژه مان دورتر اند، فوکوس نخواهند شد.

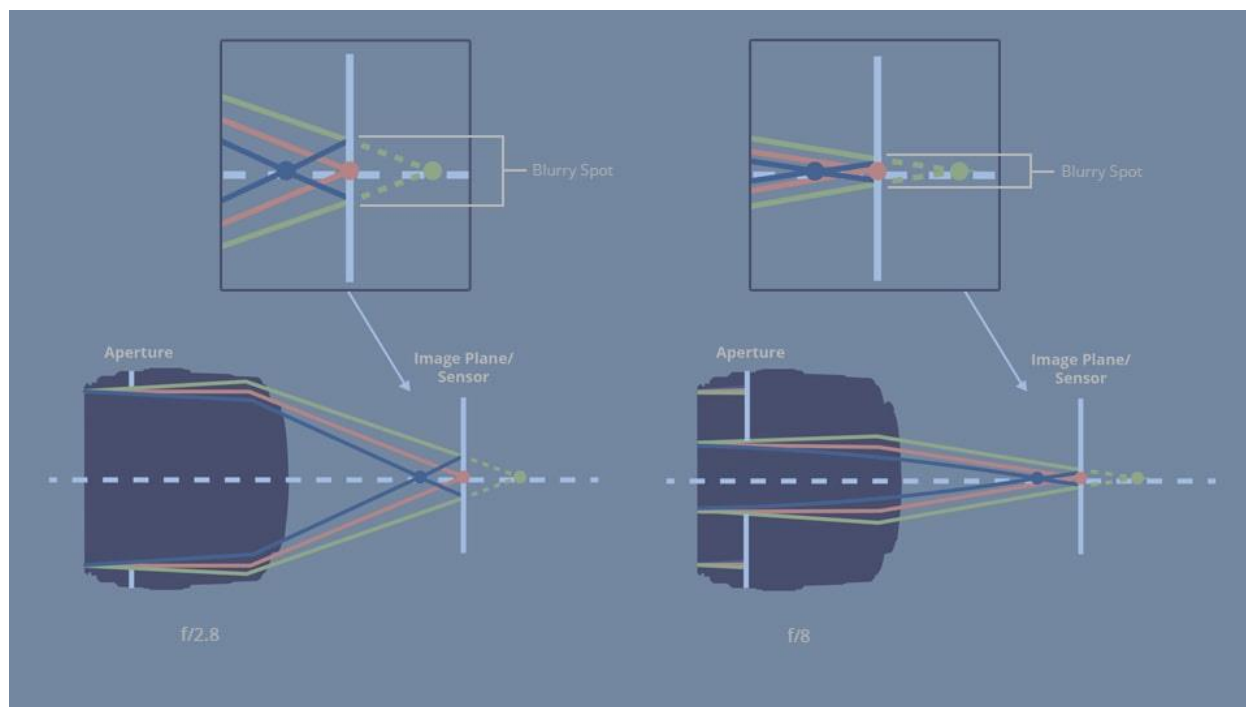
در واقع دیافراگم باز (واپد) سبب عمق میدان کم و آنچه در پس زمینه سوژه (شی مورد نظر) هست خارج از فوکوس هست (در واقع بازه کوچکی از فاصله که در فوکوس می افتد). دریچه باریک (کوچک) عمق میدان زیاد حاصل می شود و سبب میشه پس زمینه در فوکوس باشه و اصطلاحاً عکس **sharp** بشه. (یعنی بازه بزرگی از فاصله تا شی در فوکوس می افتد) (دیافراگم و عمق میدان با هم رابطه عکس دارند) در این مثال پایین به ترتیب از چپ به راست دو تا عکس با عمق میدان کم و زیاد را مشاهده می کنید.



در واقع اگر بخواهیم دقیق بگوییم نوری که از دریچه وارد لنز میشود باید با زاویه بیشتری شکسته شود تا روی سنسور ، تصویر ایجاد کند. همچنین پرتوهایی که وارد لنز میشوند ولی قبل یا بعد از سنسور به هم میرسند (نقاط تار تصویر را می سازند) زاویه بیشتری دارند که سبب ایجاد نواحی محو بیشتر و عمق میدان کمتر می شود .

حال اگر دریچه بسته تر باشد، نور هنگام عبور از دیافراگم شکست کمتری دارد و پرتوهای نوری که قبل و بعد سنسور بهم میخورند و نقاط تار تصویر را میسازند، نیز شکست کمتری دارد و بهم نزدیک ترند و به همین علت ناحیه تار کوچک تری ایجاد میشود که سبب پیدایش عمق میدان بیشتر در عکس میشود.





(3)

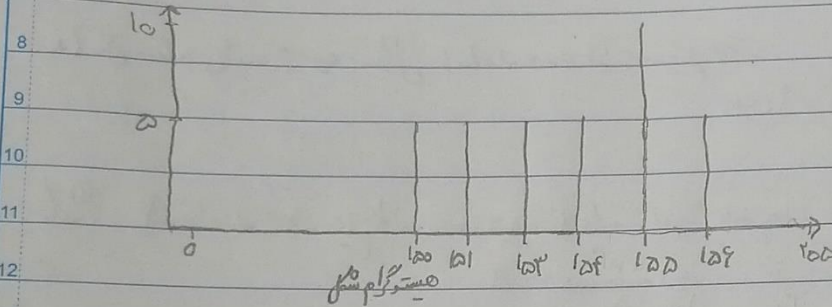
(الف)

در حل این بخش از سوال ابتدا هیستوگرام را رسم می کنیم (یعنی به ازای هر کدام از اعداد نظیر تصاویر و تعداد آن، روی نمودار نمایش می دهیم). سپس با توجه به این که بازه سطوح روشنایی اعداد محدود و کوچک است ، با استفاده از کشش هیستوگرام محدوده وسیعی را تلاش می کنیم که پوشش دهیم که سبب افزایش کنتراست تصاویر می شود. فرمول کشش هیستوگرام به شکل زیر می باشد که در حل سوال استفاده شده است:

$$g(x, y) = stretch[f(x, y)] = \left( \frac{f(x, y) - f_{min}}{f_{max} - f_{min}} \right) (MAX - MIN) + MIN$$



سوال (۳ الف)



14  $g(x) = \text{stretch}[f(x)] = \left( \frac{f(x) - f_{\min}}{f_{\max} - f_{\min}} \right) (MAX - MIN) + MIN$

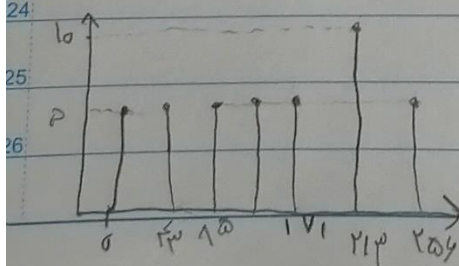
16  $MAX = 105$   $MIN = 0$   $f_{\min} = 100$   $f_{\max} = 105$

17 for:

18  $100: \frac{100 - 100}{105 - 100} \times 105 + 0 = 0$   $101: \frac{101 - 100}{105 - 100} \times 105 = \frac{105}{5} = 21$

20  $102: \frac{102 - 100}{105 - 100} \times 105 = 42$   $103: \frac{103 - 100}{105 - 100} \times 105 = 63$

22  $104: \frac{104 - 100}{105 - 100} \times 105 = 84$   $105: \frac{105 - 100}{105 - 100} \times 105 = 105$



## References:

```
##references used:
https://docs.opencv.org/4.x/d8/dbc/tutorial\_histogram\_calculation.html
```

با توجه به خواسته سوال که پیاده سازی بخش الف در کد بود، ابتدا برای هندل کردن و ذخیره آرایه و پردازش آن از کتابخانه **numpy** استفاده کردیم. و سپس در بخش دوم آن جنس نوع داده را مشخص نمودیم تا بتوان در قسمت های دیگر هیستوگرام آن را رسم نمود. (با توجه به بررسی های انجام شده در صورت عدم تعریف آرایه به صورت زیر، هیستوگرام رسم نمی شود).

```
image1=np.array([
    [150, 151, 153, 155, 156, 155, 154],
    [150, 151, 153, 155, 156, 155, 154],
    [150, 151, 153, 155, 156, 155, 154],
    [150, 151, 153, 155, 156, 155, 154],
    [150, 151, 153, 155, 156, 155, 154],
    ],dtype=np.uint8)
```

در تابع **calc\_hist** ما آمدیم ورودی گرفته شده را که شدت روشنایی تصویر تعریفی را می نماییم. همچنین بازه روشنایی و سایز آن و همچنین **accumulate** را مشخص می کنیم که باشد یا نه.

```
def calc_hist(image):
    flag = False
    h_range=(0,255)
    #hist=[0]*3
    hist = cv2.calcHist(image,[0], None, [256], h_range, accumulate=flag)
    return(hist)
```

حال با دستورات زیر ابتدا تصویر و سپس هیستوگرام را رسم و نمایش می دهیم.

```
plt.imshow(image1,cmap='gray',vmin=0,vmax=255)
plt.figure()
plt.plot(calc_hist(image1))
plt.title('histogram')
```

سپس با استفاده از دستور **stretch\_hist** ، کشش هیستوگرام را پیاده سازیم یکنیم و با استفاده از فرمولی که قسمت قبل بدان اشاره شد، ابتدا بیشترین مقدار روشنایی و کمترین مقدار را میابیم. سپس با جایگزینی در فرمول ، مقادیر جدید را در کپی ایجاد شده از ورودی تابع جایگزین می نماییم و در نهایت خروجی را بر میگردانیم.

```
def stretch_hist(image):
    ...
    don't use libraries
    input(s):
        image (ndarray): input image

    output(s):
        output_image (ndarray): enhanced image with
        ...

    output_image = image.copy()
    # Start
    s=image.shape
    MAX=255
    MIN=0
    fmax=0
    fmin=255
    for i in range(s[0]):
        for j in range(s[1]):
            temp=image[i][j]
            if temp < fmin:
                fmin=temp
            if fmax < temp :
                fmax = temp
    print(fmax,fmin)
    for i in range(s[0]):
        for j in range(s[1]):
            a=image[i][j]-fmin
            b=(fmax-fmin)
            c=MAX-MIN
            d=(a/b)*c + MIN
            output_image[i][j]= d
    # End
    return output_image
```

حال با فراخوانی تابع قسمت قبل ، رسم تصویر و هیستوگرام را انجام میدهم .

```
#dont change this cell
plt.imshow(stretch_hist(image1),cmap='gray',vmin=0,vmax=255)
plt.figure()
plt.plot(calc_hist(stretch_hist(image1)))
plt.title('new histogram')
```

ج)

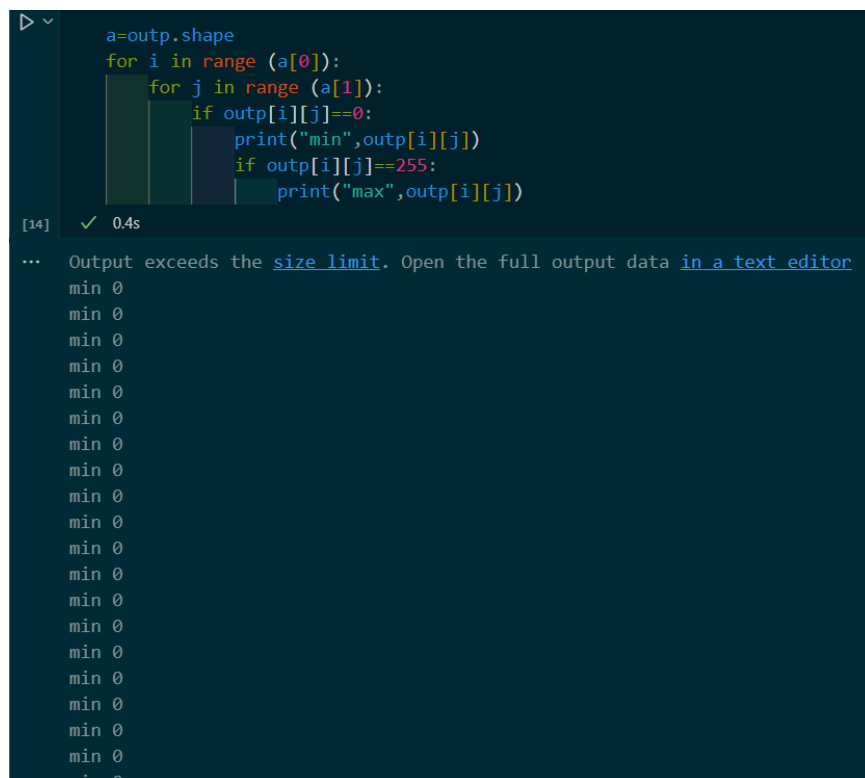
ابتدا عکس را میخوانیم .برای راحتی طے ، می آییم و تصویر را تک کاناله می کنیم تا راحت تر بتوان آن را پردازش کرد و در توابع با آن کار کرد. سپس در قسمت بعد کشش هیستوگرام را روی تصویر انجام میدهم و تصویر و هیستوگرام آن را نمایش می دهیم.

با توجه به هیستوگرام، تصویر بهبودی نیافت، دلیل آن هم وجود مقادیر روشنایی در حد 0 در تصویر است که با کد عکس دوم پایینی ، چک شده است که سبب میشود که فرمول به درستی کار نکند و تصویر اپدیت نشود.(چون مقدار مینیمم روشنایی صفر موجود است).(حتی کم))

```
image_1=cv2.imread('Q3_image/image2.jpg',cv2.IMREAD_GRAYSCALE)
print(image1.shape)
cv2.imshow('image',image_1)
cv2.waitKey()

5, 7)
1

#use stretch function to improve quality of the image and show
outp=stretch_hist(image_1)
plt.imshow(outp,cmap='gray',vmin=0,vmax=255)
plt.figure()
plt.plot(calc_hist(outp))
plt.title('histogram')
```



## References:

# Slides(for class)

ابتدا یک کپی از ورودی گرفته می شود، سپس با توجه به فرمول موجود در اسلاید درس برای برش هیستوگرام و استفاده از نکات و هیستوگرام شکل در قسمت های قبل یک متغیر برای تعیین محدودیت و بررسی روی آن محدوده در نظر میگیریم. و من اومدم با دیکشنری پیاده سازی و بررسی کردم. سپس اومدم با توجه به فرمول برش هیستوگرام، و نکته اشاره شده، به اپدیت مقادیر پرداختم. سپس در سلول بعد به چاپ و نمایش آن می پردازیم. (نکته: از لحاظ منطقی و سافتواری کدام اوکی هست برای این بخش و هرچه بررسی کردم مشکل منطقی پیدا ننمودم. ولی هنگام ران طول میکشد که خروجی بدهد.)

```

def modified_stretch_hist(image):
    #we use (boresh histogram) for improve quality of image and co
    # so we need threshold and and max=255 ,min =0 and ignore data
    output_image = image.copy()
    a=image[0]
    size_outp=len(image)
    fmax=255
    fmin=0
    #calc histogram and then create stretch histogram and convert
    #set min and max
    histogram=calc_hist(image)
    hist_file=dict(enumerate(histogram,0))
    strch_his=np.zeros(256)
    for i in hist_file:
        if hist_file[255-i] >500:
            fmax=255-i
            break
    for i in hist_file:
        if hist_file[i] >500:
            fmin=i
            break
    #calculate domain and parts for gistogram
    #and update image with algorithm (boresh histogram)
    d=int(255/(fmax-fmin))
    for i in range(fmax-fmin+1):
        t=i+fmin
        tq=i*d
        strch_his[tq]=hist_file[t]
        for j in range(size_outp):
            for k in range(len(a)):
                if output_image[j][k] == t :
                    output_image[j][k]= tq
    return output_image

```

(4

References:

<https://numpy.org/doc/stable/reference/generated/numpy.histogram.html>

<https://www.geeksforgeeks.org/numpy-argwhere-in-python/>

<https://www.geeksforgeeks.org/numpy-ravel-python/>

[https://en.wikipedia.org/wiki/Histogram\\_matching](https://en.wikipedia.org/wiki/Histogram_matching)

slides(class)

(الف)

ابتدا با استفاده از تابع آماده هیستوگرام که ورودی هایش آرایه، بازه محدوده و سایز محدوده هست. ما در اینجا از **ravel** برای یک بعدی کردن و مسطح کردن آرایه استفاده نموده ایم که عناصر را همرو به صورت یک بعدی پشت هم میآورد بدون حذف داده.

```
hist,bins=np.histogram(image.ravel(),256,[0,255])
```

حال در تابع مناسبه توزیع تجمعی، ابتدا هیستوگرام آن را مناسبه و سایز کانال ورودی را مشخص می کنیم. با استفاده از فرمول ریاضی تابع توزیع تجمعی، جواب را میابیم.

```
histogram=calc_hist(channel)
cdf=histogram.copy()
ch_sh=channel.shape
all_p=ch_sh[0]*ch_sh[1]
sum=0
for i in range(len(histogram)):
    sum+=histogram[i]
    temp=(sum/all_p)*255
    cdf[i]=round(temp)
```

$$\frac{L-1}{n} \sum_{j=0}^k n_j$$

سپس در تابع **histogram matching**، ابتدا توزیع تجمعی هر کدوم از کانال های هر دو عکس منبع و رفرنس را مناسبه میکنیم با تابع نوشته شده در قبل و سپس در یک



ارایه میریزیم.انگاه میاییم با استفاده از دستور زیر ، داده های روی نمودار عکس منبع را در فرنس معادلش را پیدا و بدان assign می کنیم . بدین سان در نهایت خروجی و عمل خواسته شده حاصل می شود.

```
m=np.argwhere(allcdf[channel+3] >= allcdf[channel][src_image[i,j]][channel])
output_image[i,j,channel]=m[0][0]
```

(ب)

در سیستم های تشفیص دهنده یا پردازش هایی که روی طیف نوری خاص خوب عمل می کند و قابل مشاهده هست. بدین سان ما تصاویرمان را به آن طیف می بریم تا قابل مشاهده و پردازش شود. همچنین از دیگر کاربرد های آن مدرج کردن یک تصویر بر دیگری است.همچنین در متعادل کردن پاسخ آشکارساز به عنوان یک روش کالیبره کردن نسبی استفاده میشود.

(5)

## References:

<https://www.geeksforgeeks.org/histograms-equalization-opencv/>

<https://www.geeksforgeeks.org/python-opencv-cv2-copymakeborder-method/>

[https://docs.opencv.org/3.4/dc/da3/tutorial\\_copyMakeBorder.html](https://docs.opencv.org/3.4/dc/da3/tutorial_copyMakeBorder.html)

## slides(class)

(الف)

با دستور **equalizehist** ، تصویر را متعادل کرده ولی به دلیل مواجهه بودن با طیف های نوری متفاوت ، تصویر بهبود نمیابد. پشت تصویر را بهبود می بخشد ولی نور روی صورت مجسمه تقویت و بسیار زیاد می شود.

ب) در این بخش با حلقه های تو در تو به پیاده سازی قسمت اول روی قطعه عکس های 58 در 48 به شکل مستقل انجام میدهد. ساختار بدین گونه است که تصاویر در حلقه به شکل بازی ای با سایز مشخص جدا و روی آن عمل متعادل سازی رخ می دهد. خروجی شکل، تصویری است که به وضوح مربع های جدا شده مشخص شده اند. این روش هم بهبود خاصی بر روی تصویر نمی دهد. با اینکه شاید برخی جزئیات رو تقویت کنه ولی ایجاد نویز هم میکنه و برخی قسمت ها هم کیفیت را کاهش میدهد.

```
temp=image[i*48:(i+1)*48,j*48:(j+1)*48]  
output_image[i*48:(i+1)*48,j*48:(j+1)*48]=cv2.equalizeHist(temp)
```

(ج)


در این بخش در ورودی تابع ، یک تصویر و یک ورودی محدوده میگیرد. سپس اندازه پدینگ اضافی را براساس محدوده داده شده تعیین میکنیم که در واقع 4 بهت تصویر است. حال نوع برادر را تعریف و در نهایت تصویر را **resize** میکند و به صورت **zero padding** ایجاد میکند. سپس هر پیکسل را براساس پیکسل های همسایه مناسبه و مقدار جدید را ذخیره میکنیم و در نهایت خروجی را برمیگردانیم. با این روش تصویر بهتر

می شود اما مشکلی که دارد این است که در قسمت هایی که زنگ ها یکنواخته ،  
میزان نویز تقویت میشود و باعث کیفیت تصویر آن حد مطلوب نباشد.

```
output = image.copy()
a=gridSize[0]
b=gridSize[1]
left=int(b/2)
right=b - left - 1
top=int(a/2)
bottom=a-top-1
temp=np.zeros(3)
resized_image=cv2.copyMakeBorder(image,top,bottom,left,right,cv2.BORDER_CONSTANT,None,temp)
for i in range(len(output)):
    for j in range (len(output[0])):
        output[i][j]=cv2.equalizeHist(resized_image[i:i+gridSize[0],j:j+gridSize[1]])[top][left]
```

(د)

در این قسمت ، دو تابع کمکی برای متعادل سازی و مناسبه هیستوگرام استفاده نموده  
ایم که به دلیل عدم مجاز بودن به استفاده از کتابخانه ها می باشد. البته تابع مناسبه  
هیستوگرام ، قبلا توضیح داده شده است. در تابع متعادل ساز هم ابتدا تعداد تکرار ها مشخص  
و سپس با فرمول متعادل سازی **table** مپ شده را ساخته و **return** می کنیم.  
در ابتدای قسمت تابع کلاهه، هم مانند قسمت (ج) تصویر جدید با پدینگ مدنظر ایجاد  
میکنیم. سپس در حلقه کد به ازای هر پیکسل تصویر مان یک **grid** از تصویر با اندازه  
گفته شده و دارای پدینگ را به تابع مناسبه هیستوگرام فرستاده سپس بر اساس  
**clip\_limit** ورودی تابع کلاهه، با توجه به ماکزیمم مجاز تکرار در هیستوگرام، طیف زوش  
کلاهه، رنگ های اضافی را به همه اضافه نموده و هیستوگرام جدید را ایجاد میکنیم. سپس  
هیستوگرام مان را به تابع متعادل ساز فرستاده و با استفاده از کاربری تابع که بالا توضیح  
داده شده، مقدار مناسب روشنایی هر پیکسل تعیین میشود.



(نکته : در این کد هم منطق و ساختار هر بخش درست است ولی مشخص نیست که چرا بخش آخر در هنگا مران شدن (نقدر طول میکشد).

در این بخش با پایین آمدن محدوده و بیشتر شدن اندازه فیلتر، تصویر کنتراست کمتری دارد و همچنین لبه تصاویر صاف و نرم تر (smooth) و نویز کمتر میشود. بالعکس اگر محدوده بالا تر رود و فیلتر کمتر بشه، نویز ها وضوح پیدا میکنند و لبه ها شارپ و کنتراست بیشتر میشه.

برای نتیجه مناسب باید حد میانی از این دو عامل را در نظر گرفت.