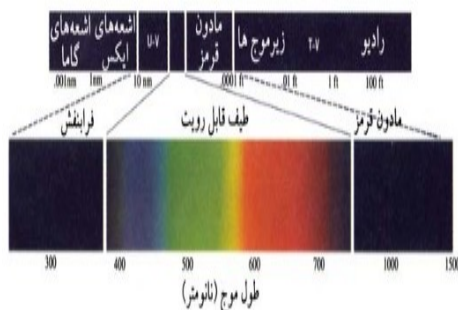


فاطمه باغخانی

اطلاعات گزارش	Color چکیده
تاریخ: 99/2/25	استفاده از رنگ در پردازش تصویر ناشی از دو عامل است اولاً که رنگ توصیف گر قدرتمندی است که غالباً شناسایی و استخراج اشیاء را از صحنه آسان میسازد و دوماً انسان میتواند در مقتیسه با 24 سطح خاکستری هزاران سایه رنگ و شدت را تشخیص میدهد. و پردازش تصویر رنگی به دو ناحیه رنگی و شبه رنگی تقسیم می شود و در مورد اول تصاویر معمولاً توسط حسگر تمام رنگی دریافت می شود اما در مورد دوم مساله تخصیص رنگ به شدت تک رنگ خاص یا بازه ای از شدت هاست و در کل اکثر الگوریتم های سطح خاکستری در این بخش با کمی تغییر قابل استفاده اند و کاربرد این پردازش در زمینه چاپ و نشر استفاده میشود
واژگان کلیدی: توصیف منبع رنگ تمایز رنگ ها در مغز انسان مدل های رنگ (فضاهای رنگ) RGB, HSI, HSV CMY, CMYK پردازش تصویر رنگی	

1- مقدمه

است و نور بی رنگ در واقع همان نوری که در تلویزیون های سیاه و سفید دیده می شود در ادامه به بررسی فضاهای رنگ و پردازش در این فضا می پردازیم.



گرچه فرآیندی که مغز انسان در دریافت و تفسیر رنگ اجرا میکند یک پدیده جسمی-روانی است که کاملاً درک نشده است ماهیت فیزیکی رنگ میتواند به طور رسمی بیان شود که توسط آزمایش ها به اثبات رسیده است.

در سال 1666 نیوتون کشف کرد که وقتی پرتوی نور از منشور عبور میکند نور خروجی نور سفید نیست بلکه طیف پیوسته ای از رنگ ها است.

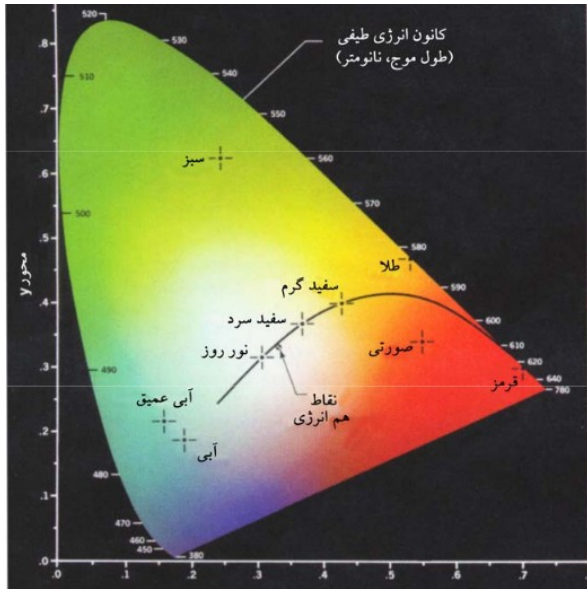
اساساً رنگ هایی که انسان و حیوانات از اشیاء دریافت میکنند به وسیله نور منعکس شده از آن ها توصیف می شود و همان طور که در شکل ملاحظه می شود نوری قابل رویت است که مرکب از باند نسبتاً باریکی از فرکانس هاست

یکی دیگر از نکات توصیف نور هسته علم رنگ است اگر نور فاقد رنگ باشد اگر نور بی رنگ باشد تنها صفت آن شدت و مقدار

2-شرح تکنیکال

• ضرایب تری کروماتیک:

ضرایب Tri-stimulus:
میزان قرمز سبز و آبی برای تشکیل یک رنگ خاص (با X و Y و Z) نمایش داده میشود



رنگ های حاشیه دیاگرام اشباع خالص و 100 درصد هستند
نقاط میانی رنگ های ترکیبی هستند و هرچه به نقطه تعادل نزدیک شویم اشباع آن صفر میشود.

مدل های رنگ

هدف مدل رنگ تسهیل تشخیص مشخصات رنگ ها در یک فضای استاندارد است
درواقع مدل رنگ زیر فضایی از یک سیستم است که در آن رنگ با مختصات نمایش داده میشود
اغلب مدل های رنگ امروزه به سمت سخت افزار و چاپگر و پرینت رنگی برای مثال می رود
مانند RGB که برای مانیتورهای رنگی و دسته وسیعی از دوربین های ویدیویی رنگی
مدل CMYK و CMY که برای چاپ رنگی استفاده میشوند و مدل بعدی مدل HSI که خیلی نزدیک به تفسیر و توصیف انسان از رنگ هاست .

• کمیت های اصلی برای

توصیف کیفیت منبع نور

تشدید یا Radiance:

مقدار کل انرژی ای که از منبع نور ساطع می شود

لومینانس یا Luminance:

مقیاسی برای اندازه گیری میزان انرژی ای که بیننده از منبع نور دریافت میکند

درخشندگی قابل درک یا Brightness:

بیانی کیفی و توصیفی از شدت است و قابل اندازه گیری نیست.

• تمایز رنگ ها در مغز انسان:

مشخصاتی که به طور عمده برای شناسایی یک رنگ از دیگری استفاده می شود عبارتند از:

Hue و Saturation و Brightness

فام یا Hue: نشان دهنده ی طول موج رنگ غالب که توسط بیننده دریافت می شود

Saturation: میزان ترکیب نور سفید با یک فام میزان خلوص رنگ درواقع

Brightness یا شدت روشنایی

ترکیب Hue و Saturation

باهم کروماتیسیته آن رنگ می نامند و بنابراین یک رنگ را میتوان با کروماتیسیته و Brightness شناسایی کرد.

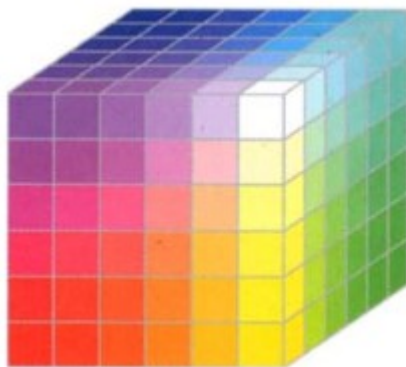
نکته قابل توجه آنکه آن رنگی که توس حسگرهای مخروطی چشم دریافت می شود RGB و آن رنگی که توسط مغز درک می شود با ویژگی های کروماتیسیته و Brightness قابل درک است

همان طور که در شکل ملاحظه میکنید ما 216 رنگ امن داریم در قسمت الف تصویر و قسمت ب خاکستری های 256 رنگ را نشان میدهد.

مدل رنگ CMY و CMYK

فیروزه ای بنفش و زرد رنگ های ثانویه نوری اند یا به عبارتی رنگ های اولیه مادی اند . اغلب دستگاه هایی که مواد رنگی روی کاغذ میپاشند مانند دستگاه های کپی و چاپگرهای رنگی نیاز به این ورودی و سپس تبدیل آن به RGB دارند که از فرمول زیر به دست می آید.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



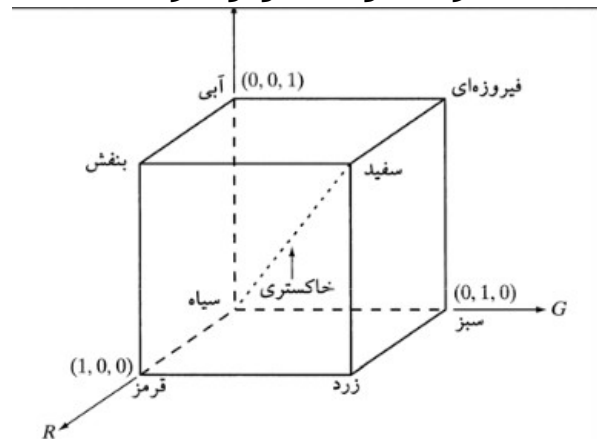
مکعب امن:

مدل HSI:

این فضا براساس نحوه درک رنگ براساس بینایی انسان ایجاد شده است. مدل های قبلی برای چشم انسان مناسب بودند چون عملاً انسان رنگ های R, G, B را به خوبی میتواند تشخیص دهد اما برای مثال نمیتواند از روی درصد رنگ ها رنگ به جود آمده را تجسم کند

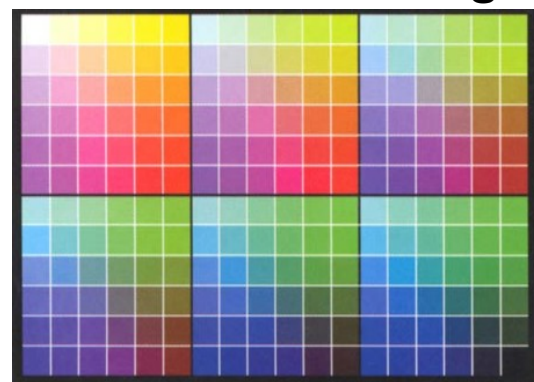
مدل رنگ RGB

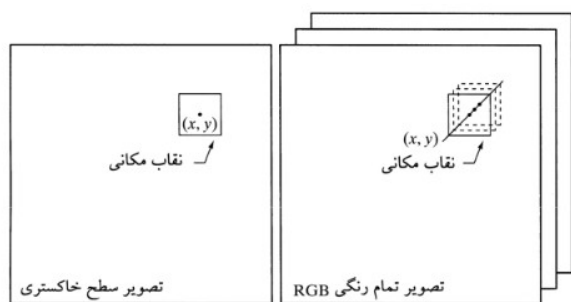
در این مدل هر رنگ مولفه ای از طیف های اولیه قرمز سبز و آبی است. این مدل مبتنی بر سیستم مختصات دکارتی است و زیر فضای مورد علاقه مکعبی است که رنگ های اصلی در سه گوشه آن وجود دارد.



و تمام مقادیر RGB در این مکعب در بازه صفر و یک هستند.

رنگ امن:





شکل ۶-۲۹ نقاب‌های مکانی برای تصاویر رنگی RGB و سطح خاکستری.

و این تصویر به ما نشان می‌دهد که برای پردازش با یک بردار روبرو هستیم و اعمال باید روی آن انجام شود.

3- نتیجه گیری:

تمرین اول: برای تبدیل تصویر از فرمول‌های که گفته شد استفاده کرده و به HSI تبدیل میکنیم که در ادامه میبینید:

HUE



که تا:

$$\theta = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \right\}$$

و مولفه اشباع به شکل زیر به دست می‌آید.

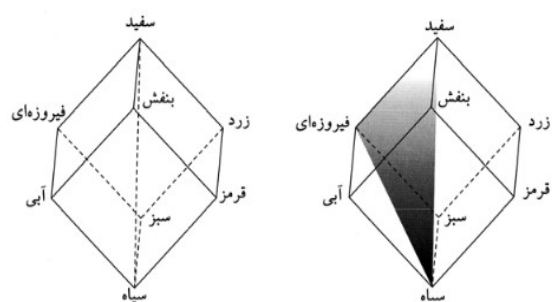
$$S = 1 - \frac{3}{(R + G + B)} [\min(R, G, B)]$$

و مولفه شدت نیز:

$$I = \frac{1}{3}(R + G + B)$$

● پردازش تصویر در فضای رنگی:

میدانیم چون تصاویر تمام رنگی عملاً حداقل سه مولفه دارند پس به بیانی دیگر میتوان تمام تبدیلات را روی هر سطح جدا (برای مثال R,G,B) انجام داد و باهم ترکیب کرد



این تصویر رابطه بین فضای RGB (سمت چپ) و فضای HSI را نشان میدهد و در این شکل فضای HSI توسط محور شدت عمودی و LOC و US نقاط رنگی که در صفحات عمود بر این محور قرار دارند نمایش داده میشود.

تبدیل RGB به HSI:

5.1.2:

استاندارد تلویزیون NTSC یک کدگذاری می باشد که از مقدار لومینانس Y و دو مقدار رنگینگی I و Q استفاده می نماید. در تلویزیون های سیاه و سفید فقط درخشندگی به کار برده می شود، در حالی که هر سه مقدار در تلویزیون های رنگی به کار برده می شود.

مدل های رنگی YUV و YIQ پتانسیل بهتری را برای فشرده سازی تصاویر و ویدیوی دیجیتال نسبت به سایر روش های کد گذاری فراهم می نمایند، زیرا در این دو مدل رنگی بر خلاف مدل RGB تفکیک درخشندگی و رنگینگی از همدیگر امکان پذیر می باشد.

مدل رنگی NTSC در سیستم تلویزیونی آمریکا استفاده می شود
اطلاعات روشنایی مستقل از مولفه های رنگی است.
هریک از مولفه ها توسط کانالی مستقل فرستاده میشوند



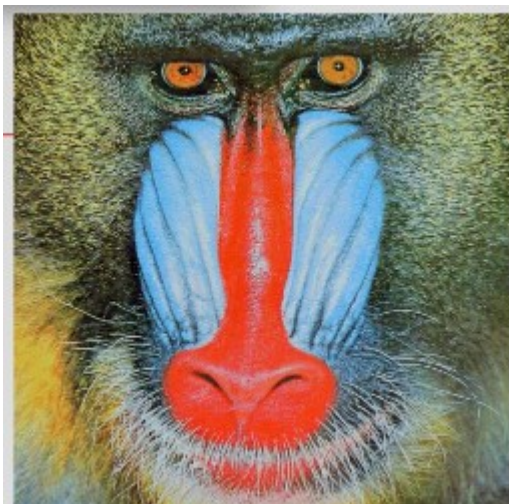
SATURATION

INTENSITY



$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

روشنایی
فام
فلوئس



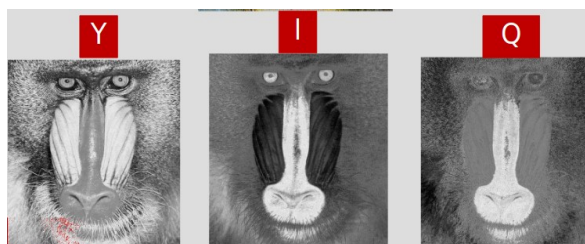
5.2.1:

در اینجا ابتدا تصاویر جدا جدا در سه سطح R,G,B کوانتایز میکنیم و سپس سه سطح را با یکدیگر مرج تا تصویر حاصل به دست آید

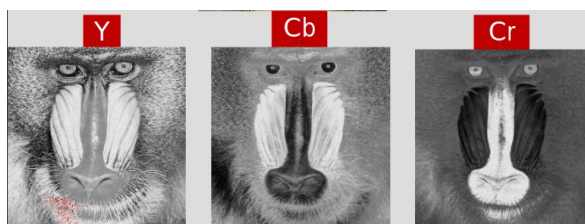
تصویر حاصل با $q=8$



تصویر حاصل $q=16$



واین هم نمونه این عکس در این فضا
مدل بعدی میتوان به YcbCr اشاره کرد
که دراین اطلاعات روشنایی مستقل از
مولفه های رنگی است
استفاده در کدینگ تصاویر جهت ارسال

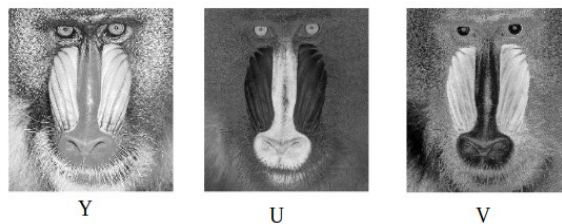


مدل رنگ YUV
برای رنگ خاکستری $R=G=B$
درخشندگی یا luminance یا Y میزان خاکستری
است.
برای یک تصویر خاکستری تابع U, V صفر است

$$U = B - Y \quad V = R - Y$$

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

این هم مثال و نمونه این فضا:



q=32



q=64



حال به گزارش mse و psnr توجه کنید:

	q=8	q=16	q=32	q=64
mse	79.727	22.375	5.6463	1.4857
psnr	29.114	34.633	40.613	46.411

و همان طور که در جدول مشاهده میکنید:
در q های بالا تصویر بهتر شده و با تصویر اصلی
با توجه به mse ها mse آن کمتر و همین
طور psnr زیاد تر و تصویر بهتر شده است .

5.2.2:



در اینجا mse, psnr این تصویر نسبت به

تصویر اصلی

mse :88.24010721842448

psnr 28.67414333734713

نسبت به حالت که هر سه سطح 3 بیت بود:

mse 79.72724914550781

psnr 29.114735811159107

و همین طور که میبینید اینکه هر ۳ سطح ۳ بیت بوده mse کمتر و بهتر است و در اون حالت درواقع شدت آبی و میزان آن نسبت به بقیه کمتر خواهد بود.

8رنگ و ssim=0.7042225081979808

:5.2.3

در این سوال از ما خواسته شده تعداد رنگ های تصویر را کاهش دهیم من روش های متفاوتی را مانند dithering را امتحان کردم و اما آن اما با کمترین میزان از دست دادن اطلاعات روبرو نشدم و سپس clustering رو امتحان کردم.

البته یک نکته که وجود دارد در این سوال استثنا uniform quantization هم میتواند کارساز باشد اما در کل خوب نیست و چون به همه رنگ ها وزن و سهم یکسانی میدهد زیاد خوب نیست در حالیکه باید به اندازه فراوانی رنگ ها وزن بدهد .



16 رنگ و ssim=0.76

در این شیوه درواقع خوشه بندی گفته میشود که براساس ویژگی های مشترک دیتاها را دسته بندی میکنند ما در اینجا میخواهیم تعداد رنگ های تصویر کم کنیم باید به هر سطح رنگ (R,G,B) در این الگوریتم تعدادی رنگ اختصاص داده میشود به گونه ای که کمترین میزان Lost اطلاعات را داشته باشیم که به تصویرهای زیرتوجه کنید و ssim به دست آمده



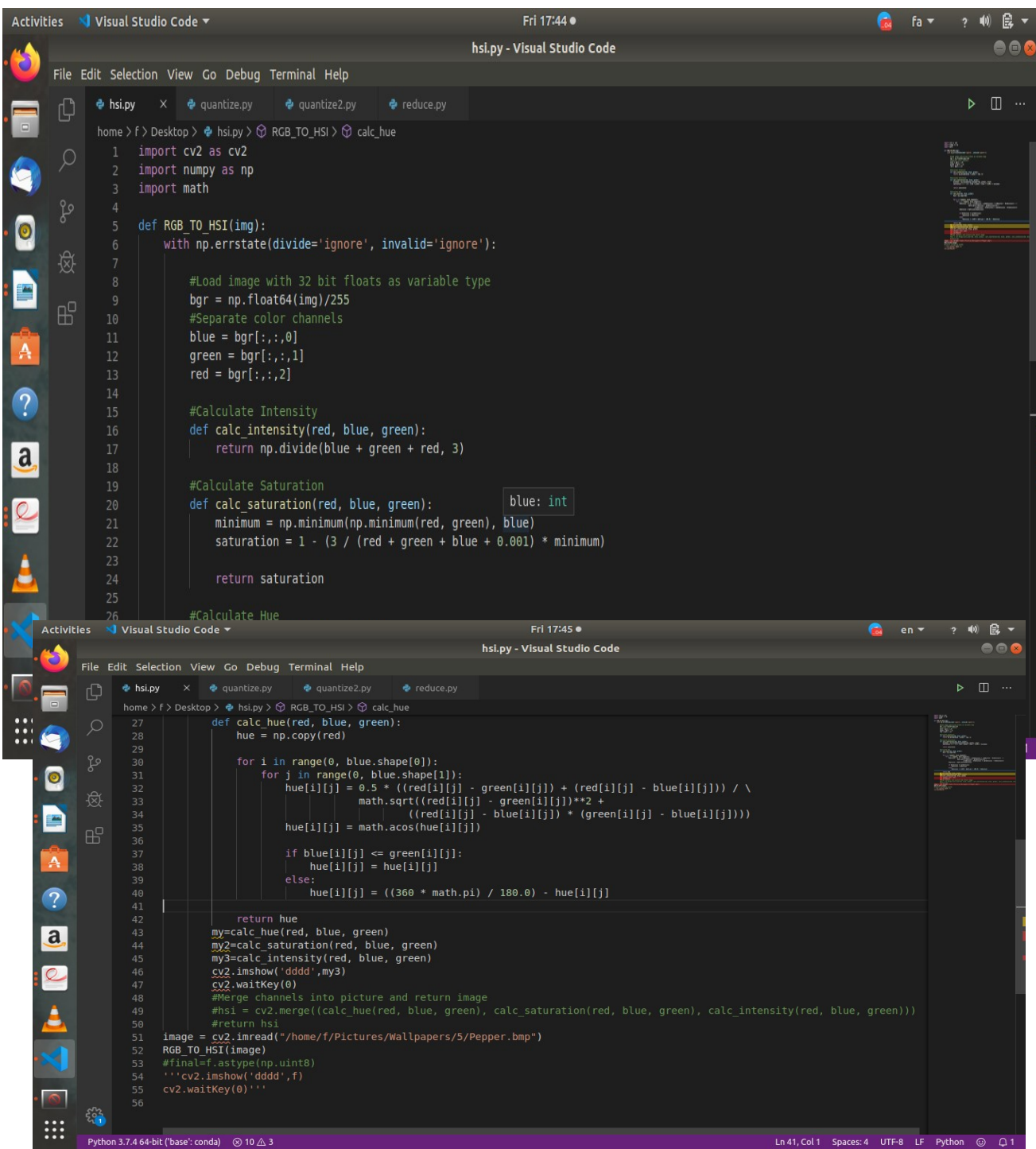
و همان طور که ملاحظه میکنید هر چه
تعداد رنگ ها هم بیشتر ssim بهتری دارد.

4- پیوست کد:

تمرین اول که درواقع طبق فرمول گفته شده

32رنگ و $ssim=0.8333949364178022$

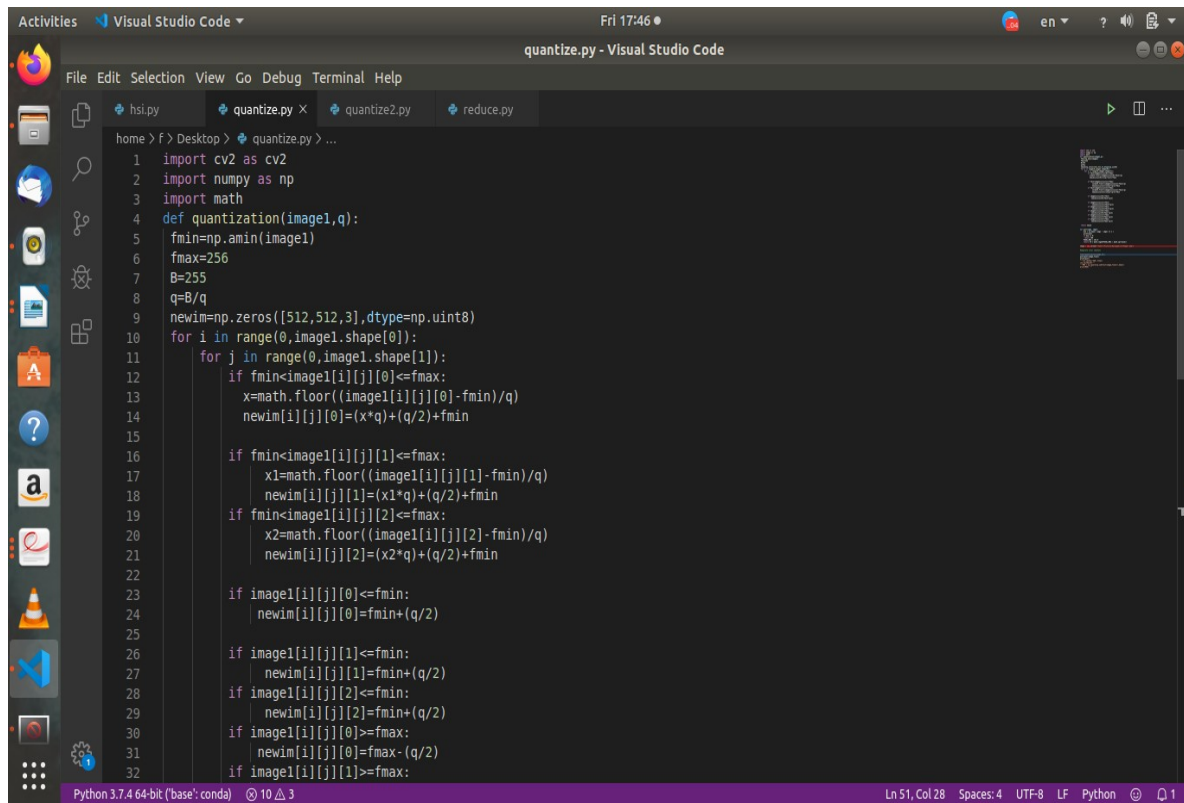




تمرین دوم که درواقع طبق فرمول
uniform quantize عمل میکنیم:

Quantization function

$$Q(f) = \begin{cases} \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min} & f_{\min} < f \leq f_{\max} \\ f_{\min} + q/2 & f \leq f_{\min} \\ f_{\max} - q/2 & f > f_{\max} \end{cases}$$



```

1  import cv2 as cv2
2  import numpy as np
3  import math
4  def quantization(image1,q):
5      fmin=np.amin(image1)
6      fmax=255
7      B=255
8      q=B/q
9      newim=np.zeros([512,512,3],dtype=np.uint8)
10     for i in range(0,image1.shape[0]):
11         for j in range(0,image1.shape[1]):
12             if fmin<image1[i][j][0]<=fmax:
13                 x=math.floor((image1[i][j][0]-fmin)/q)
14                 newim[i][j][0]=(x*q)+(q/2)+fmin
15
16             if fmin<image1[i][j][1]<=fmax:
17                 x1=math.floor((image1[i][j][1]-fmin)/q)
18                 newim[i][j][1]=(x1*q)+(q/2)+fmin
19             if fmin<image1[i][j][2]<=fmax:
20                 x2=math.floor((image1[i][j][2]-fmin)/q)
21                 newim[i][j][2]=(x2*q)+(q/2)+fmin
22
23             if image1[i][j][0]<=fmin:
24                 newim[i][j][0]=fmin+(q/2)
25
26             if image1[i][j][1]<=fmin:
27                 newim[i][j][1]=fmin+(q/2)
28             if image1[i][j][2]<=fmin:
29                 newim[i][j][2]=fmin+(q/2)
30             if image1[i][j][0]>=fmax:
31                 newim[i][j][0]=fmax-(q/2)
32             if image1[i][j][1]>=fmax:

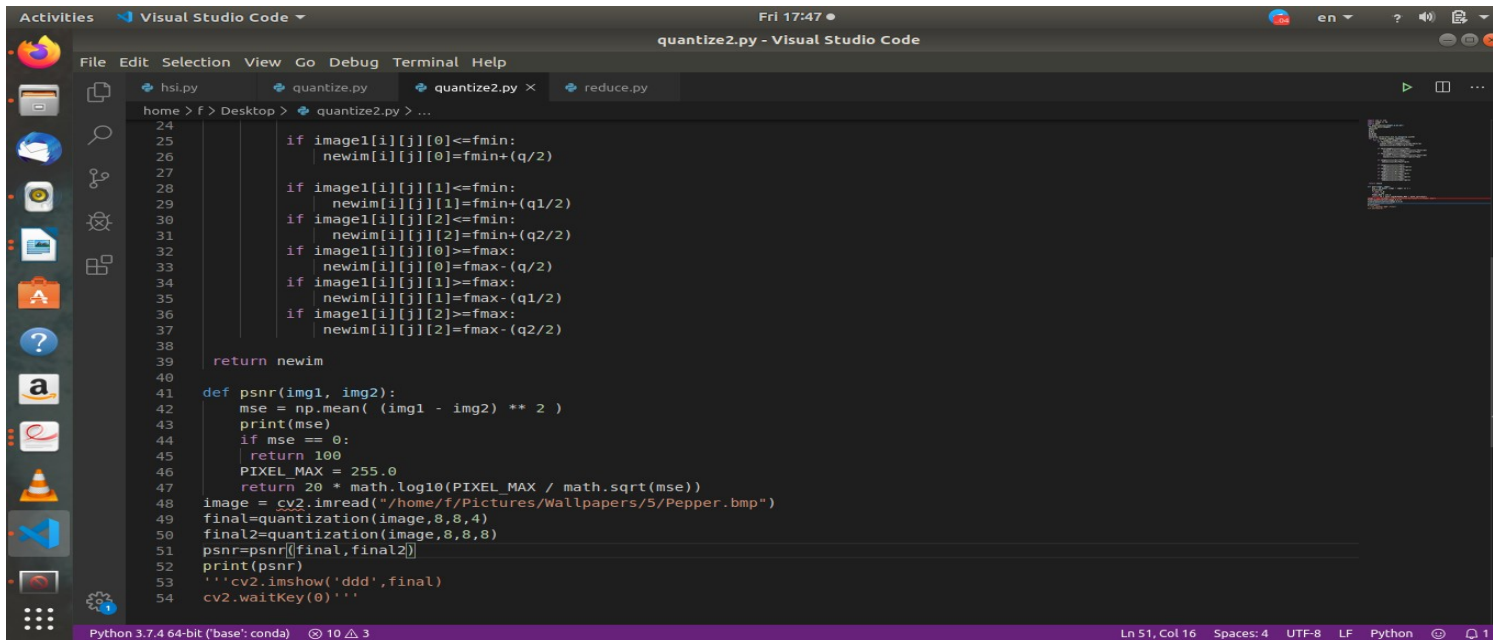
```



```
27         newim[i][j][1]=fmin+(q/2)
28     if image1[i][j][2]<=fmin:
29         newim[i][j][2]=fmin+(q/2)
30     if image1[i][j][0]>=fmax:
31         newim[i][j][0]=fmax-(q/2)
32     if image1[i][j][1]>=fmax:
33         newim[i][j][1]=fmax-(q/2)
34     if image1[i][j][2]>=fmax:
35         newim[i][j][2]=fmax-(q/2)
36
37     return newim
38
39 def psnr(img1, img2):
40     mse = np.mean( (img1 - img2) ** 2 )
41     print(mse)
42     if mse == 0:
43         return 100
44     PIXEL_MAX = 255.0
45     return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
46
47 image = cv2.imread("/home/f/Pictures/Wallpapers/S/Pepper.bmp")
48
49 #Separate color channels
50
51 final=quantization(image,64)
52 psnr=psnr(image,final)
53 print(psnr)
54 '''cv2.imshow('ddd',final)
55 cv2.waitKey(0)'''
56 '''MSE = np.square(np.subtract(image,final)).mean()
57 print(MSE) '''
```

کوانتایز سوال بعد:

```
1 import cv2 as cv2
2 import numpy as np
3 import math
4 def quantization(image1,q,q1,q2):
5     fmin=np.amin(image1)
6     fmax=256
7     B=255
8     q=B/q
9     q1=B/q1
10    q2=B/q2
11    newim=np.zeros([512,512,3],dtype=np.uint8)
12    for i in range(0,image1.shape[0]):
13        for j in range(0,image1.shape[1]):
14            if fmin<image1[i][j][0]<=fmax:
15                x=math.floor((image1[i][j][0]-fmin)/q)
16                newim[i][j][0]=(x*q)+(q/2)+fmin
17
18            if fmin<image1[i][j][1]<=fmax:
19                x1=math.floor((image1[i][j][1]-fmin)/q1)
20                newim[i][j][1]=(x1*q1)+(q1/2)+fmin
21
22            if fmin<image1[i][j][2]<=fmax:
23                x2=math.floor((image1[i][j][2]-fmin)/q2)
24                newim[i][j][2]=(x2*q2)+(q2/2)+fmin
25
26            if image1[i][j][0]<=fmin:
27                newim[i][j][0]=fmin+(q/2)
28
29            if image1[i][j][1]<=fmin:
30                newim[i][j][1]=fmin+(q1/2)
31
32            if image1[i][j][2]<=fmin:
33                newim[i][j][2]=fmin+(q2/2)
34
35            if image1[i][j][0]>=fmax:
```

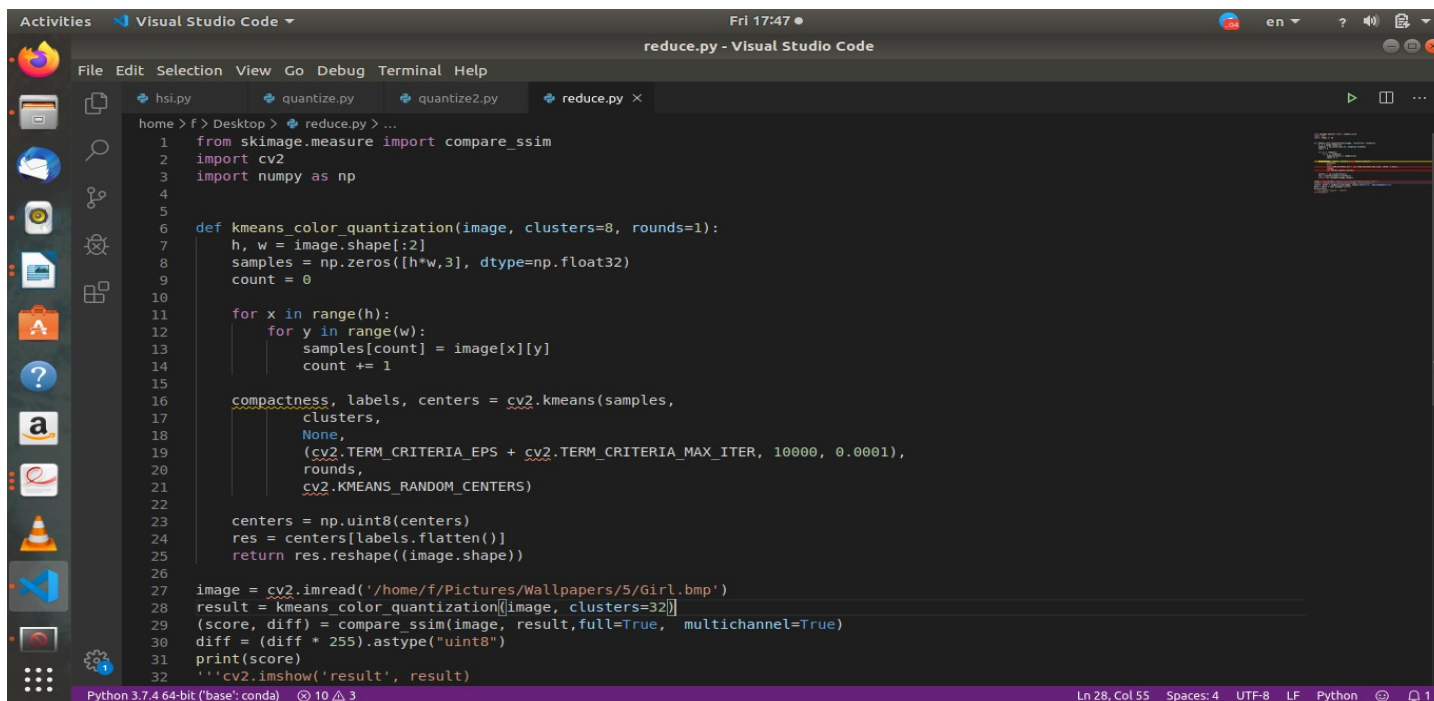


```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
def quantize2(image, clusters=8, rounds=4):
    newim = image.copy()
    for i in range(rounds):
        if image[i][j][0] <= fmin:
            newim[i][j][0] = fmin + (q/2)
        if image[i][j][1] <= fmin:
            newim[i][j][1] = fmin + (q1/2)
        if image[i][j][2] <= fmin:
            newim[i][j][2] = fmin + (q2/2)
        if image[i][j][0] >= fmax:
            newim[i][j][0] = fmax - (q/2)
        if image[i][j][1] >= fmax:
            newim[i][j][1] = fmax - (q1/2)
        if image[i][j][2] >= fmax:
            newim[i][j][2] = fmax - (q2/2)
    return newim

def psnr(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    print(mse)
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))

image = cv2.imread("/home/f/Pictures/Wallpapers/5/Pepper.bmp")
final1 = quantization(image, 8, 8, 4)
final2 = quantization(image, 8, 8, 8)
psnr = psnr(final1, final2)
print(psnr)
'''cv2.imshow('ddd', final)
cv2.waitKey(0)'''
```

و سوال آخر که از clustering استفاده میشود



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
def kmeans_color_quantization(image, clusters=8, rounds=1):
    h, w = image.shape[:2]
    samples = np.zeros([h*w, 3], dtype=np.float32)
    count = 0
    for x in range(h):
        for y in range(w):
            samples[count] = image[x][y]
            count += 1
    compactness, labels, centers = cv2.kmeans(samples,
        clusters,
        None,
        (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10000, 0.0001),
        rounds,
        cv2.KMEANS_RANDOM_CENTERS)
    centers = np.uint8(centers)
    res = centers[labels.flatten()]
    return res.reshape(image.shape)

image = cv2.imread('/home/f/Pictures/Wallpapers/5/Girl.bmp')
result = kmeans_color_quantization(image, clusters=32)
(score, diff) = compare_ssim(image, result, full=True, multichannel=True)
diff = (diff * 255).astype("uint8")
print(score)
'''cv2.imshow('result', result)
```


۴