color

فهیم جعفری

اطلاعات گزارش

تاریخ:

واژگان کلیدی:
Color space
Xyz
YIQ
YUV
Hsi(hue,saturation,luminance)
Mmse
PSNR
histogram
quantization
k-means
per color

چکیده

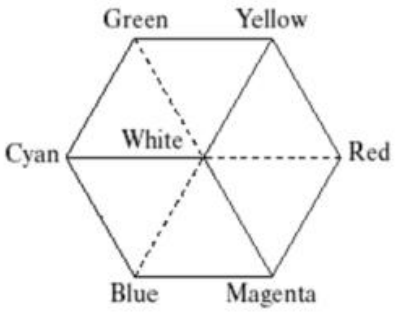
استفاده از رنگ در پردازش تصویر ،ناشی از دو عامل است. اولا ، رنگ توصیفگر قدرتمندی است که غالبا شناسایی و استخراج اشیا را از صحنه آسان میسازد. ثانیا ،انسان میتواند در مقایسه با فقط 24 سایه خاکستری ، هزاران سایه رنگ و شدت را تشخیص دهد. این عامل دوم ، مخصوصا در تحلیل تصویر دستی(یعنی وقتی که توسط انسان انجام میگیرد) مهم است.پردازش تصویر رنگی به دو ناحیه تقسیم میشود: پردازش تمام رنگی و شبه رنگی. در دسته اول ، تصاویر معمولا توسط حسگر تمام رنگی دریافت میشوند، مثل دوربین Tv رنگی یا اسکنر رنگی. در دسته دوم ،مسئله تخصیص رنگ به شدت تک رنگ خاص یا بازه ای از شدت ها است.فضاهای رنگی زیادی وجود دارد که در این صفحه hsi استفاده شده است که با درک انسان مناسب تر میباشد.

1-مقدمه

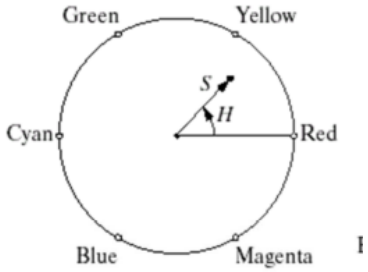
2-شرح تکنیکال

- **Color space**

اگر مکعب rgb را به شکل زیر دوران دهیم



شش ضلعی مشاهده میشود که اگر مرکز را به عنوان مرکز دایره در نظر بگیریم فاصله از مرکز را که شعاع میباشد را S در نظر میگیریم و زاویه ان را H و ارتفاع مقطع برش زده از مبدا را I در نظر بگیرم فضای hsi را مدل کرده ایم که این مدل با چشم و درک انسان قابل مفهوم تر میباشد



محاسبات از rgb به hsi و برعکس ،بر حسب بیت ها انجام میگیرد. برای سهولت ، وابستگی (x,y) را در معادلات تبدیلی در نظر نگرفتم.اگر تصویری با فرمت rgb داشته باشیم ، مولفه h هر پیکسل rgb با استفاده ار معادله زیر به دست می اید:

$$H=\begin{cases}\theta & \text{if } B\le G\\ 360-\theta & \text{if } B>G\end{cases}$$

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

مولفه اشباع به صورت زیر به دست می اید:

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

سرانجام ،مولفه شدت به صورت زیر به دست می اید:

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

فرض میشود که مقادیر rgb به بازه ی {0و1} نرمال سازی شدند و زاویه theta نسبت به محور قرمز در فضای hsi محاسبه میشود پرده رنگ میتواند به بازه {0و1}نرمال شود،که برای این کار باید تمام مقادیر حاصل از h,را بر 360 درجه تقسیم گردد. اگر مقادیر rgb در فاصله {0,1} باشند دو مولفه دیگر نیز در این بازه بوده اند

- **فضای رنگی XYZ**

در سال CIE، 1931 اقدام به معرفی یک مجموعه رنگ پایه ی جدید برای برطرف کردن مشکلات فضای رنگ RGB نمود:XYZ.مشکل فضای رنگ قبلی در این بود که برای مشاهده گرهای مختلف نظیر انسان ها،سنسورها و... تعریف واحدی برای دریافت اطلاعات رنگی موجود نبود.یعنی در فضای RGB،مشخصات رنگی نیمه شهودی و وابسته به وسایل اندازه گیری است. در این فضای رنگ پایه جدید اجزای رنگ های پایه رنگ های واقعی نیستند.میزان درخشندگی رنگ در این فضا بر خلاف فضای قبل فقط به یک پارامتر (Y) بستگی دارد و بقیه ی پارامتر ها ،ویژگی های رنگی را پوشش میدهند.

برای ورود به فضای رنگی XYZ از RGB و بالعکس میتوان از ماتریس های تبدیل زیر استفاده کرد

$$\begin{bmatrix}X\\Y\\Z\end{bmatrix}=\begin{bmatrix}0.412453 & 0.357580 & 0.180423\\0.212671 & 0.715160 & 0.072169\\0.019334 & 0.119193 & 0.950227\end{bmatrix}*\begin{bmatrix}R\\G\\B\end{bmatrix}$$

$$\begin{bmatrix}R\\G\\B\end{bmatrix}=\begin{bmatrix}3.240479 & -1.537150 & -0.498535\\-0.969256 & 1.875992 & 0.041556\\0.055648 & -0.204043 & 1.057311\end{bmatrix}*\begin{bmatrix}X\\Y\\Z\end{bmatrix}$$

- **فضاهای رنگی YUV,YIQ**

این فضا ها با تغییرات کوچکی از روی مدل R`G`B` بدست امده است. این تغییرات برای ان است که مدل جدید برای انتقال ،نسبت به مدل RGB کارامدتر شود و همینطور با سیستم تلویزیون های سیاه و سفید ،سازگار گردد. در حقیقت جزءY از این فضاها اطلاعات ویدیویی مورد نیاز یک سیستم تلویزیون سیاه و سفید را بطور کامل تامین میکند.برای تبدیل فضای RGB به فضاهای YIQ , YUV و بالعکس از روابط زیر استفاده میشود

$$\begin{bmatrix}Y\\I\\Q\end{bmatrix}=\begin{bmatrix}0.299 & 0.587 & 0.114\\0.596 & -0.275 & -0.321\\0.212 & -0.523 & 0.311\end{bmatrix}\begin{bmatrix}R'\\G'\\B'\end{bmatrix}\qquad \begin{bmatrix}R'\\G'\\B'\end{bmatrix}=\begin{bmatrix}1 & 0.9557 & 0.6199\\1 & -0.2716 & -0.6469\\1 & -1.1082 & 1.7051\end{bmatrix}\begin{bmatrix}Y\\I\\Q\end{bmatrix}$$

تبدیل از RGB به YIQ و بالعکس

$$\begin{bmatrix}Y\\U\\V\end{bmatrix}=\begin{bmatrix}0.299 & 0.587 & 0.114\\-0.147 & -0.289 & 0.436\\0.615 & -0.515 & -0.100\end{bmatrix}\begin{bmatrix}R'\\G'\\B'\end{bmatrix}\qquad \begin{bmatrix}R'\\G'\\B'\end{bmatrix}=\begin{bmatrix}1 & 0 & 1.1398\\1 & -0.3946 & -0.5805\\1 & 2.0320 & -0.0005\end{bmatrix}\begin{bmatrix}Y\\U\\V\end{bmatrix}$$

تبدیل از RGB به YUV و بالعکس

حساسیت سیستم بینایی انسان به تغییرات روشنایی بیشتر از حساسیت ان به ته رنگ و اشباع است و فضاهای YIQ,YUV به گونه ای طراحی شده اند که از این خاصیت استفاده کنند

.به همین دلیل در استانداردهای منطبق بر این فضاها ، بیشتر پهنای باند (تعداد بیت در رنگ دیجیتال) در دسترس را ، به پارامتر Y اختصاص میدهند و پهنای باند کمتری را برای اجزای کرومینانس نگه میدارند.

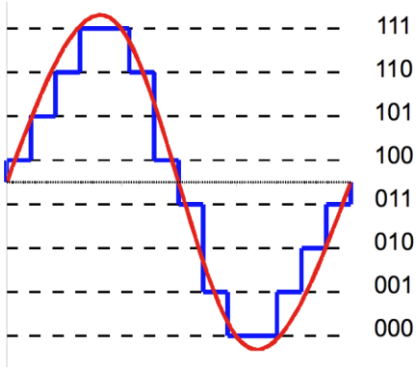
مزیت عمده دیگری که این فضاها نسبت به مدل RGB دارند ان است که پارامتر مشخص کننده روشنایی Y از دو پارامتر مشخص کننده رنگ جدا است با توجه به ان که متغیر روشنایی متناسب با مقدار نوری است که چشم بیننده دریافت میکند .با استفاده از این خاصیت میتوان روشنایی یک تصویر را تغییر داد بدون انکه در ترکیب رنگ ان تغییری ایجاد شود. برای مثال میتوان تکنیک یکسان سازی هیستوگرام را تنها در مورد پارامتر Y از تصویری با فرمت YIQ یا YUV اعمال کرد. در این حالت رنگ تصویر تغییری نخواهد کرد.

- **Quantization**

گسسته سازی مپ کردن یک مجموعه بزرگ به یک مجموعه کوچک میباشد در این تمرین مجموعه اعداد از رنج پیوسته نامحدود 0تا 255 که بینهایت اعداد در ان وجود دارد به مجموعه اعداد محدود با تعداد دلخواه از 0 تا 255 مپ شده است .به این صورت که هر رنج خاصی از مجموعه نامحدود به عدد مورد نظر مپ میشود. که در این صورت اطلاعات تصویر از بین میرود و با تصویر اصلی تفاوت دارد.برای تصاویر رنگی به این صورت میباشد که هر 3 کانال rgb جدا جدا (per color) گسسته سازی انجام میدهیم و این عمل برگشت ناپذیر میباشد که فرمول ان به صورت زیر میباشد

$$d = \frac{\max - m_i n}{L}$$
$$f = [f/d] * d$$

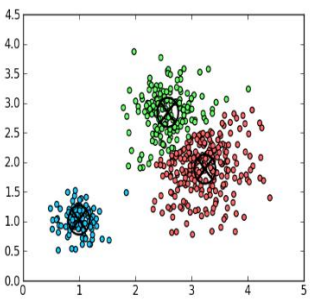
که L=level میباشد تصویر زیر برای درک بیشتر میباشد



- **قسمت 5.2.3**

برای این قسمت از الگوریتم زیر که در یادگیری ماشین برای دسته بندی unsupervised استفاده میشود بهره گرفته شده است

(1)تمام نقاط در تصویر را به صورت نقطه در فضای rgb میبریم(2) سپس به اندازه نقاطی که میخواهیم به ان کاهش دهیم به صورت رندوم در این فضا میگزاریم برای مثال اگر به 32 رنگ میخواهیم کاهش دهیم به صورت رندوم 3 مقدار برای هر یک از 32 نقطه برای 3 کانال ان انتخاب میکنیم و کنار سایر نقاط موجود در فضا قرار میدهیم(3) سپس تمام نقاط موجود در تصویر را که در فضای rgb میباشد به نزدیک ترین از این 32 نقطه ها نسبت میدهیم برای مثال برای 3 نقطه(3 دسته) به شکل زیر میباشد



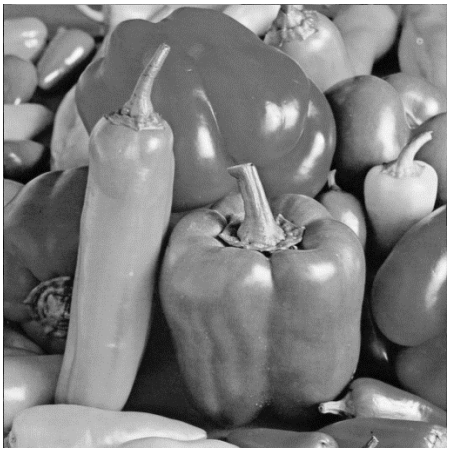
(4)سپس مکان این 32 نقطه را به مرکز داده هایی که به انها نسبت داده شده است بروزرسانی میکنیم حال دوباره مراحل 3 و4 تکرار میکنیم این مراحل را به اندازه ای تکرار میکنیم تا به یک حالت پایدار برسیم و داده ای بعد از تکرار دسته بندی اش عوض نشود بدین صورت با کمترین خطا ما 32 رنگ انتخاب کردیم برای اینکه دسته ها روی هم قرار نگیرند برای اطمینان بیشتر میتوان این الگوریتم را دوباره یا 3باره تکرار کرد تا از نتیجه ان مطمئن بود

3-شرح نتایج

- **Color space**

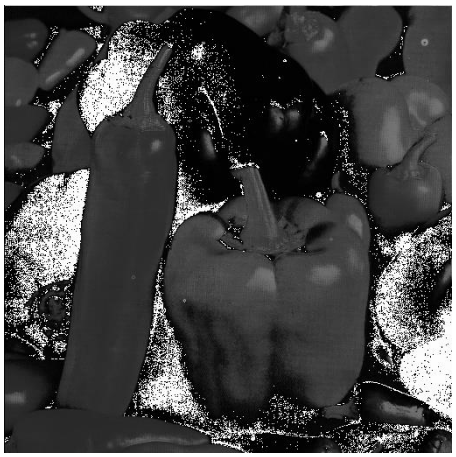
تصویر زیر تصویر رنگی میباشد که مولفه های hsi ان را جدا کردیم

تصویر زیر مولفه I تصویر اصلی میباشد



همانطور که مشاهد میشود این مولفه میزان روشنایی در تصویر را نشان میده جاهای که براق در تصویر اصلی میباند در در مولفه I مقدار نزدیک به ماکس گرفته اند و جاهایی که تاریک میباشد و روشنایی کم است مقدار نزدیک به صفر گرفته اند این مولفه تصویر gray scale تصویر هم گفته میشود در این مولفه نویز هایی که در تصویر اصلی میباشد از بین رفته است چرا که طبق فرمول ان که در شرح تکنیکال بحث شد میانگیری سه مولفه میباشد که خود همانند یک فیلتر پایین گذر عمل میکند

تصویر زیر مولفه H تصویر اصلی میباشد



همانطور که مشاهده میشود این تصویر بر اساس تغییرات رنگ تغییرات محسوس تری دارد و رنگ تصویر را نشان میدهد و همینطور گسستگی هایی در رنگ قرمز وجود دارد که علت ان گسستگی فضای رنگ his میباشد که در 360 درجه H رخ میدهد و همینطور در نقاطی که I شدت روشنایی به صفر یا یک نزدیک است تغییر H که رنگ میباشد احساس نمیشود که از خصوصیات این فضای رنگی میباشد

تصویر زیر مولفه S تصویر اصلی میباشد

همان طور که مشاهده میشود این مولفه میزان سفیدی در رنگ را نشان میدهد و قسمت هایی که رنگ های ان میزان سفیدی در انها بیشتر است تاریک یا صفر شده اند و رنگ هایی که میزان سفیدی در انها کم میباشد در تصویر مولفه S به مقدار ماکس یا نزدیک ماکس (255) گرفته اند همینطور در قسمت هایی که I ماکس هست و یا مین (صفر) است S معنی ندارد و تاثیری در ان ندارد

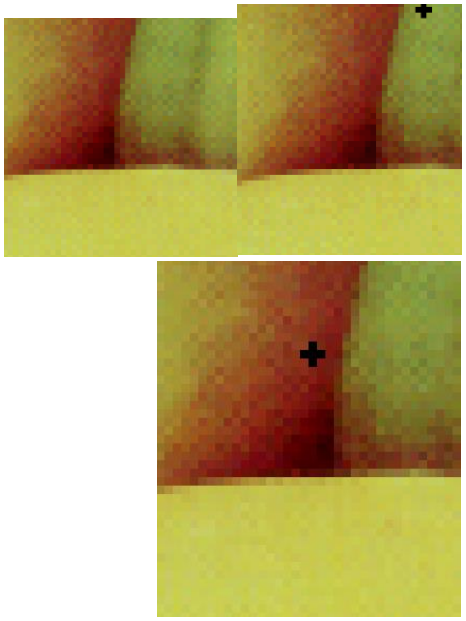
- **Quantization**

تصویر زیر تصویر اصلی میباشد که عمل کاهش سطوح خاکستری بر روی انها انجام میشود

تصویر زیر کاهش یافته به 64 سطح میباشد

تصویر زیر کاهش یافته به 32 سطح میباشد

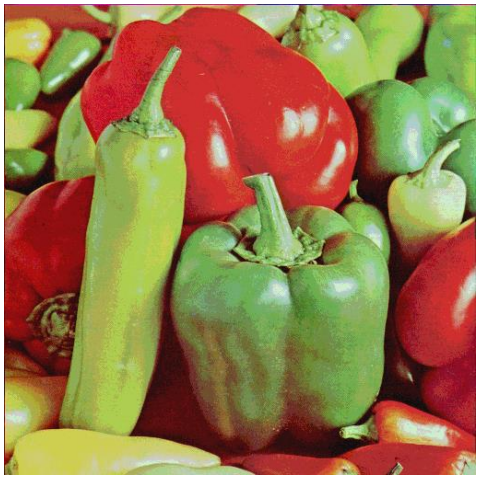
همانطور که مشاهده میشود با چشم انسان این 3 تصویر با هم قابل تمایز نیستند و تفاوتی در انها احساس نمیشود به تصاویر زیر که 8برابر بزرگ شده اند نگاه کنید



مشاهده میشود که در این تصاویر فرق خیلی خیلی کمی وجود دارد که در صورت بزرگ نمایی بیشتر دیده میشود.

تصویر زیر کاهش یافته به 16 سطح میباشد

تصویر زیر تصویر کاهش یافته به 8 سطح میباشد



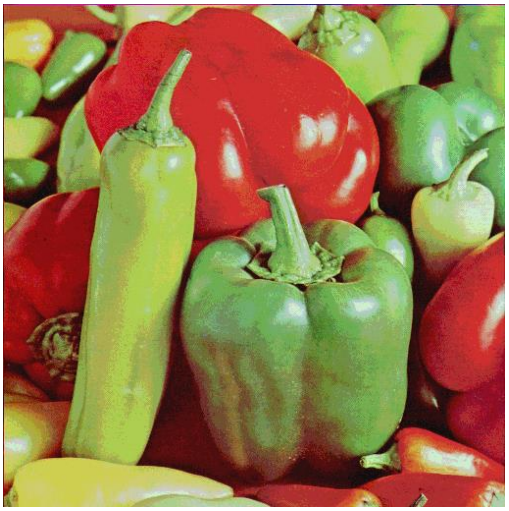
همانطور که مشاهده میشود این تصاویر با چشم انسان با تصویر اصلی قابل تمایز است و کیفیت تصویر با کاهش سطوح پایین امده است و نیازی به زوم کزدن نمیباشد

جدول زیر نمایش mmse و psnr تصویر اصلی و کاهش یافته میباشد

	64	32	16	8
immse	2.1404674910008907	6.244435252000888	22.818275943398476	86.39989860852559
psnr	44.82571724626365	40.17587194063335	34.54797533079939	28.765671280392911

همانطور که مشاهده میشود mmse از 32 به پایین افزایش چشمگیری دارد که این هم به صورت شهودی مشاهده شد و همینطور psnr که مقاومت تصویر به نویز و تخریب تصویر را نشان میدهد کاهش چشمگیری داشته است که البته با برای 8 سطح از 20 بیشتر میباشد پس نتیجه مطلوبی است

تصویر زیر تصویر کاهش یافته کانالr,g,b به3 میباشد



وتصویر زیر تصویر کاهش یافته کانال r,g به 3 بیت و کانال b به 2 بیت میباشد

همانطور که مشاهده میشود در تصویر دوم در مقایسه با تصویر اول در بعضی از نقاط رنگهای جدیدی به وجود امده است و در بعضی جاها ترکیب رنگی به هم خورده است چرا که بازه تغییرات رنگ ابی بیشتر است از سایر رنگ های اصلی است که موجب تغییر رنگ مکمل ان زرد نیز میشود و خطای بیشتری را بوجود می اورد که در جدول زیر که mmse و psnr حالت های مختلف به ترتیبbgr از چپ به راست میباشد این موضوع را نشان میدهد

	2,2,2	2,3,3	3,3,3
mmse	366.62299942970276	162.3023874759674	86.39989860852559
psnr	22.488606547095404	26.02755452501863	28.76567128039291

- **کاهش تعداد رنگها 5.2.3**

تصویر زیر تصویر اصلی میباشد

تصویر زیر به 32 رنگ کاهش یافته است که با 50 بار تکرار الگوریتم به دست امده است

مشاهده میشود که نزدیک ترین رنگ در تصویر اصلی که خطای کمتری تولید کرده است به انها نظیر شده است ولی رنگ ها عوض شده است

تصویر زیر به 16 رنگ کاهش یافته است

این تصویر به تصویر 32 رنگ کاهش یافته نزدیک تر است

تصویر زیر هم به 8 رنگ کاهش یافته است

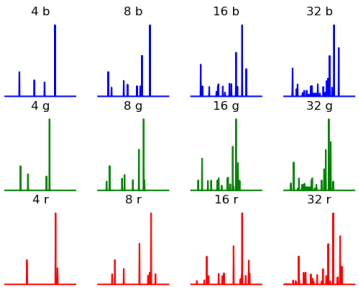
که به دو تصویر قبلی نزدیک است همانطور که در این تصاویر دیده شده تفاوت بین کاهش یافته ها و تصویر اصلی به و اضحی با چشم انسان مشاهده میوشد چرا که رنگ را عوض کردیم

جدول زیر mmse بین تصویر اصلی و کاهش یافته ها را نشان میدهد

	8	16	32
mmse	91.55585225423177	84.42915852864583	81.20977401733398
psnr	28.513942512199474	28.865879000109746	29.034720588885335

همانطور که مشاهده میشود mmse در این تصاویر بر خلاف سوال قبلی که uniform quantization بود و mmse بالا بود در اینجا mmse کم میباشد

و هیستوگرام انها به شکل زیر میباشد



که مشاهده میشود همان تعداد رنگ خواسته شده وجود دارد

همینطور ssim انها گرفته شد که برای 8 رنگ مقدار 0.4642 بدست امد که در متلب اندازه گرفته شده است

چون این الگوریتم یعنی k-means به مینیومم های محلی میرسد برای اینکه به مینیمم های بهتری برسیم از کتابخانه cv2 تابع k-means استفاده شد که نتایج ان در زیر امده است

برای 32 رنگ

برای 16 رنگ

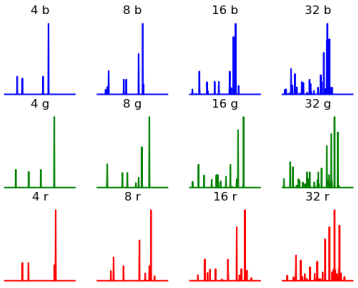
برای 8 رنگ

هیستوگرام انها

```
cv2.imwrite('../image/hw5.1.1/H.png',H)
min = np.min(S)
max = np.max(S)
S = normalizeimg(S,min,max)
cv2.imwrite('../image/hw5.1.1/S.png',S)
min = np.min(I)
max = np.max(I)
I = normalizeimg(I,min,max)
cv2.imwrite('../image/hw5.1.1/I.png',I)
```

کد قسمت 5.2.1

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


def PSNR(original, compressed):
    mse = np.mean((original -
compressed) ** 2)
    if(mse == 0):  # MSE is zero means
no noise is present in the signal .
        # Therefore PSNR have no
importance.
        return 100
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel /
np.sqrt(mse))
    return psnr,mse


def unifrom_quantization(img,level):
    divider = 255/level
    quantizedimg = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            for k in
range(img.shape[2]):
                quantizedimg[i,j,k] =
(int(img[i,j,k]/divider)*divider)+int(di
vider/2)
    return quantizedimg


def run():
    img =
cv2.imread('../image/Pepper.bmp')
    levels = [64,32,16,8]
    immse_list = ["immse"]
    psnr list = ["psnr"]
    for level in levels:
        quantizedimg =
unifrom_quantization(img,level)

cv2.imwrite('../image/hw5.2.1/quantizedi
mag{}.png'.format(level),quantizedimg)
        psnr,immse =
PSNR(img,quantizedimg)
        immse_list.append(immse)
        psnr_list.append(psnr)


    #display mmse table
    data = [immse_list,psnr_list]
    col_labels = tuple(" ")+
tuple(levels)
    fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
    ax.axis('off')
    ax.table(cellText=data,
colLabels=col_labels, loc='center')
```



و جدول مربوط به mmse,psnr

	8	16	32
mmse	77.71824010213216	64.65484364827473	49.441773737250326
psnr	29.225574032503356	30.024792950556574	31.189863181152123

4-کد برنامه

کد قسمت 5.1.1

```
import cv2
import numpy as np


def normalizeimg(img,min,max):
    img = (img-min)/(max-min)*255
    return img


def toHSI(img):
    B = np.float32(img[:,:,0])
    G = np.float32(img[:,:,1])
    R = np.float32(img[:,:,2])
    B+=1
    G+=1
    R+=1
    sorat = (1/2)*((R-G)+(R-B))
    makhraj = np.power( np.power(R-G,2)
+ ((R-B)*(G-B)) , 1/2)
    makhraj = makhraj+1
    casr = np.divide(sorat , makhraj)
    theta = np.degrees(np.arccos(casr))
    H = theta
    S = np.zeros(H.shape,dtype=float)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if B[i,j] > G[i,j]:
                H[i,j] = 360 -
theta[i,j]
            S[i,j] = 1 -
(3/(R[i,j]+G[i,j]+B[i,j])*min([R[i,j],G[
i,j],B[i,j]]))
    I = (1/3)*(R+G+B)
    return H,S,I

img = cv2.imread('../image/Pepper.bmp')
img = img.astype(np.float)
H,S,I = toHSI(img)
min = np.min(H)
max = np.max(H)
H = normalizeimg(H,min,max)
```

```
fig.savefig('../image/hw5.2.1/resTable.png')


# run()
```

کد قسمت 5.2.2

```
import cv2
import numpy as np
from src.hw5_2_1 import PSNR
import matplotlib.pyplot as plt


def unifrom_quantization_per_color(img,levelB,levelG,levelR):
    dividerB = 255/levelB
    dividerG = 255/levelG
    dividerR = 255/levelR
    quantizedimg = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            quantizedimg[i,j,0] = (int(img[i,j,0]/dividerB)*dividerB)+int(dividerB/2)
            quantizedimg[i,j,1] = (int(img[i,j,1]/dividerG)*dividerG)+int(dividerG/2)
            quantizedimg[i,j,2] = (int(img[i,j,2]/dividerR)*dividerR)+int(dividerR/2)
    return quantizedimg

def run():
    img = cv2.imread('../image/Pepper.bmp')
    levelB = [4,4,8]
    levelG = [4,8,8]
    levelR = [4,8,8]
    mmseL = ["mmse"]
    psmrL = ["psnr"]
    for i in range(3):
        quantized_img = unifrom_quantization_per_color(img,levelB[i],levelG[i],levelR[i])
        B = img[:,:,0]
        G = img[:,:,1]
        R = img[:,:,2]

cv2.imwrite('../image/hw5.2.2/quantized_img{}.png'.format(str(levelB[i])+str(levelG[i])+str(levelR[i])),quantized_img)
        psnr,mmse = PSNR(img,quantized_img)
        mmseL.append(mmse)
        psmrL.append(psnr)
    #display mmse table
    data = [mmseL,psmrL]
    col_labels = tuple(["","2,2,2","2,3,3","3,3,3"])
    fig, ax = plt.subplots(dpi=300, figsize=(5,1))
    ax.axis('off')
    ax.table(cellText=data, colLabels=col_labels, loc='center')

fig.savefig('../image/hw5.2.2/resTable.png')
run()
```

کد قسمت 5.2.3

```
import cv2
import numpy as np
import random
from src.hw5_2_1 import PSNR
import matplotlib.pyplot as plt


def classify(mat,num_of_class,num_of_iteration):
    kernel_mat = []
    class_members = []
    for i in range(num_of_class):
        t = (random.randint(0,255),random.randint(0,255),random.randint(0,255))
        kernel_mat.append(t)
    for iterate in range(num_of_iteration):
        for i in range(num_of_class):
            class_members.append([])
        for i in range(mat.shape[0]):
            for j in range(mat.shape[1]):
                min_distance = 450
                num_class = 0
                for cls in range(len(kernel_mat)):
                    distance = np.sqrt(((mat[i,j,0] - kernel_mat[cls][0])**2)+((mat[i,j,1] - kernel_mat[cls][1])**2)+((mat[i,j,2] - kernel_mat[cls][2])**2) )
                    if distance < min_distance :
                        min_distance = distance
                        num_class = cls
                mat[i,j,3] = min_distance
                mat[i,j,4] = num_class

                class_members[num_class].append(mat[i,j,0:4])
        for cls in range(len(class_members)):
            sumg = 0
            sumb = 0
            sumr = 0
            for member in class_members[cls]:
                sumg += member[0]
                sumb += member[1]
                sumr += member[2]
            if len(class_members[cls]) > 0:
                newB = int(sumb/len(class_members[cls]))
                newG = int(sumg/len(class_members[cls]))
                newR = int(sumr/len(class_members[cls]))
                kernel_mat[cls] = (newB,newG,newR)
        print('iter:{}'.format(iterate))
    return kernel_mat,mat
```

```
cv2.imwrite('../image/hw5.2.3/stack_over
_flow/res{}.png'.format(level), result)


def save_mmse_psnr():
    # for compute mmse and psnr
uncomment
    l = [8,16,32]
    mmseL = ["mmse"]
    psmrL = ["psnr"]
    img =
cv2.imread('../image/Girl.bmp')
    for i in range(len(l)):
        res =
cv2.imread('../image/hw5.2.3/stack_over_
flow/res{}.png'.format(l[i]))
        psnr,mmse = PSNR(img,res)
        mmseL.append(mmse)
        psmrL.append(psnr)
    #display mmse table
    data = [mmseL,psmrL]
    col_labels =
tuple(["","8","16","32"])
    fig, ax = plt.subplots(dpi=300,
figsize=(5,1))
    ax.axis('off')
    ax.table(cellText=data,
colLabels=col_labels, loc='center')

fig.savefig('../image/hw5.2.3/stack_over
_flow/resTable.png')
    plt.hist(img.ravel(),256,[0,256])


def saveHistogram():
    l = [4,8,16,32]
    fig,axs = plt.subplots(3,4)
    for i in range(len(l)):
        img =
cv2.imread('../image/hw5.2.3/stack_over_
flow/res{}.png'.format(l[i]))
        color = ('b','g','r')
        for j,col in enumerate(color):
            histr =
cv2.calcHist([img],[j],None,[256],[0,256
])
            axs[j,i].plot(histr,color =
col)
            axs[j,i].set_title(str(l[i])+
" "+str(col))
            axs[j,i].axis('off')

fig.savefig('../image/hw5.2.3/stack_over
_flow/hist.png')


# l = [4,8,16,32]
# for i in l:
#     run(i,10)
saveHistogram()
```

```
#this segment is copied from stack over
flow for comparsion to my result
def classify_stack_over_flow(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))


def run(level,iterate):
    # for my classifier uncomment below
    img =
cv2.imread('../image/Girl.bmp')
    leny = img.shape[0]
    lenx = img.shape[1]
    lenC = img.shape[2]
    #in third dimention 0,1,2 are bgr
value and 4 is distance 5 is class
    mat = np.zeros((leny,lenx,lenC+2))
    mat[:,:,0:3] = img
    kernel_mat,res_mat =
classify(mat,level,iterate)
    for i in range(res_mat.shape[0]):
        for j in
range(res_mat.shape[1]):
            res_mat[i,j,0] =
kernel mat[int(res mat[i,j,4])][0]
            res_mat[i,j,1] =
kernel_mat[int(res_mat[i,j,4])][1]
            res_mat[i,j,2] =
kernel_mat[int(res_mat[i,j,4])][2]

    for cls in kernel_mat:
        print(cls)
    res = res_mat[:,:,0:3]

cv2.imwrite('../image/hw5.2.3/res{}.png'
.format(level),res)


    #for classify stack over flow
uncomment below
    # image =
cv2.imread('../image/Girl.bmp')
    # result =
classify_stack_over_flow(image,
clusters=level)
    #
```

مراجع

کتاب گنزالس رافائل