تمرین ۶ بینایی رنگ

سارا قوام پور

اطلاعات گزارش

تاریخ: ۱۴۰۱/۱۱/۱۷

واژگان کلیدی:
رنگ
rgb
hsi
quantization
فضا های رنگی
kmeans

چکیده

در این تمرین با حل مسائل مطرح شده با مفهوم پـردازش تصـویر رنگی و فضـاهای رنگی آشنا می‌شویم و همچنین با مفاهیمی مانند color image quantization و کم کردن تعداد رنگ‌ها در یک تصویر با clustering و انتقال از یک فضای رنگی بـه فضـای ر نگی دیگر آشنا میشویم.

۱-مقدمه

کد های این تمرین با اسـتفاده از کتابخانـه cv2 وscikit learn و زبان python پیاده‌سازی شده است.

۲- توضیحات فنی تمرینات و تحلیل نتایج

۲-۱ تمرین ۱-۱-۶

در این تمرین خواسته شده است تـا تصـویر رنگی لنـا را از فضای rgb به فضای hsi منتقل کنیم و کامپوننت های h و s و I نمایش داده شوند.

به این منظور تابع HSI_converter نوشـته شـده اسـت که به دریافت عکس رنگی, ابتـدا کانال‌هـای رنگی قرمـز و سبز و آبی آن را جدا میکند و سپس مقـدار هـر کانـال بـا تقسیم بر ۲۵۵ نرمال می‌شوند تا بتوان از آن‌ها در فرمـول هایی که بیان می‌شوند استفاده کرد.

این تـابع طبـق فرمـول هـای زیـر کـامپوننت هـای hsi را محاسبه میکند:

فرمول محاسبه فام :

$$denumerator = [1/2[R-G+R-B]]$$

$$numerator = [(R-G)^2+(R-B)(G-B)]$$

$$\theta = arc\,cosine\,[denumerator/numerator]$$

$$H = \theta \quad if \quad B \le G$$

$$H = 360 - \theta \quad if \quad B > G$$

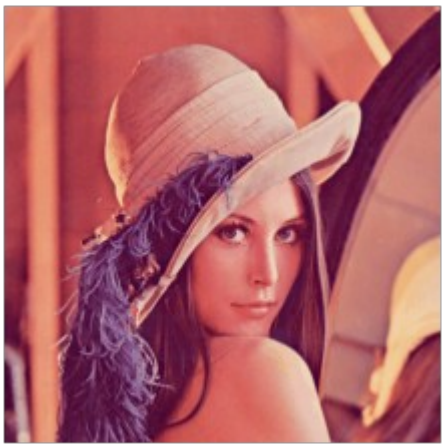
فرمول محاسبه خلوص رنگ:

$$S = 1 - ((3/(R+G+B)) * (min(R,G,B)))$$

فرمول محاسبه شدت نور:

$$I = (1/3) * (R+G+B)$$

این تـابع پس از محاسـبه سـه مقـدار H,S,I آن‌هـا را برمیگرداند.



شکل ۱) تصویر RGB لنا

نتایج خروجی تابع به صورت زیر است:

* ghavams2001@gmail.com

شکل ۲) فام رنگH تصویر لنا در فضای hsi

شکل ۳) خلوص رنگ S تصویر لنا در فضای hsi

شکل ۴) شدت روشنایی I تصویر لنا در فضای hsi

طبق شکل ۱ پشت زمینه لنا و خود صورت و شانه های لنا شامل ش=مقدار زیادی رنگ قرمز هستند و چون رنگ قرمز در فضای hsi با ۰ درجه مشخص می‌شود, مشاهده می‌شود که در شکل ۲ تمام این نواحی سیاه رنگ یا بسیار تیره میباشند که به علت همان ۰ یا همان رنگ قرمز میباشد. همچنین در شکل ۱ مشاهده می‌شود که مقدار قرمزی در موها کمتر است و پر روی کلاه کاملاً رنگی شبیه به آبی تیره دارد و میدانیم در سیستم hsi رنگ‌های دیگر از قرمز درجه بیشتری دارند در نتیجه این نواحی در شکل ۲ همانطور که دیده می‌شوند روشن و هستند به علت درجه حدود ۲۴۰ برای رنگ آبی که به همین دلیل نسبت به نواحی قرمز بسیار روشن‌تر شده است.

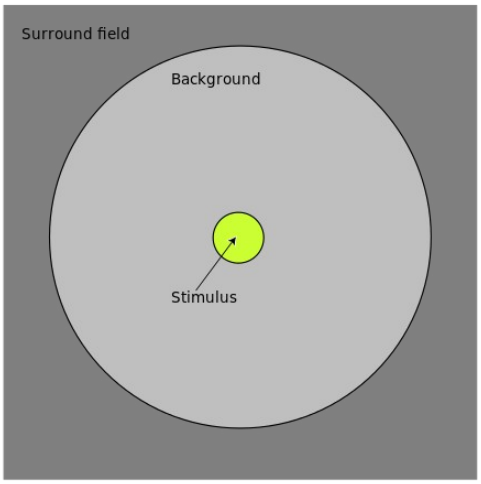
در نواحی مانند پوست و پشت زمینه که واضحا رنگ سفید زیادی وجود دارد دنتیجه خلوص کم است باعث شده است تا در شکل ۳ که مربوط به خلوص است این مناطق رنگی تیره داشته باشند. و نواحی مانند موها و قسمت‌هایی از پر کلاه که رنگ‌ها خالص تر هستند و کمتر با سفید ترکیب شده‌اند در شکل ۳ مربوط به خلوص روشن هستند چون این نواحی خلوص بیشتری دارند.

شکل ۴ هم که مربوط به شدت رنگ یا intensity میباشد درواقع نسخه gray scale تصویر میباشد.

۲-۲ تمرین ۲-۱-۶

در این تمرین خواسته شده استد تا سه سیستم رنگ جدید معرفی شود.

۱- مدل CIECAM02 : این مدل یک color appearance model است که در سال ۲۰۰۲ ارائه شد. دو قسمت اصلی از این مدل chromatic adaption transform و معادله های ریاضی این مدل برای محاسبه correlation بین ۶ محور تعریف شده brightness, lightness, colorfulness, chroma, saturation , hue میباشد.



شکل ۵) مدل CIECAM02

این الگوریتم برای ورودی خـود مقـادیر tristmulus از adapting white point و adapting background و surround luminance را در یافت میکند.

CIECAM02 سـه surrounding تعریـف میکنـد: dark , average,dim که پارامتر های آن‌هـا در عکس پایین مشخص شده است:

Surround condition	Surround ratio	*F*	*c*	N_c	Application
Average	$S_R > 0.15$	1.0	0.69	1.0	Viewing surface colors
Dim	$0 < S_R < 0.15$	0.9	0.59	0.9	Viewing television
Dark	$S_R = 0$	0.8	0.525	0.8	Using a projector in a dark room

شکل ۶) surrounding conditions

مقدار مطلق luminance به صورت زیر است:

$$LA = Ew/\pi \ \ Yb/Yw$$

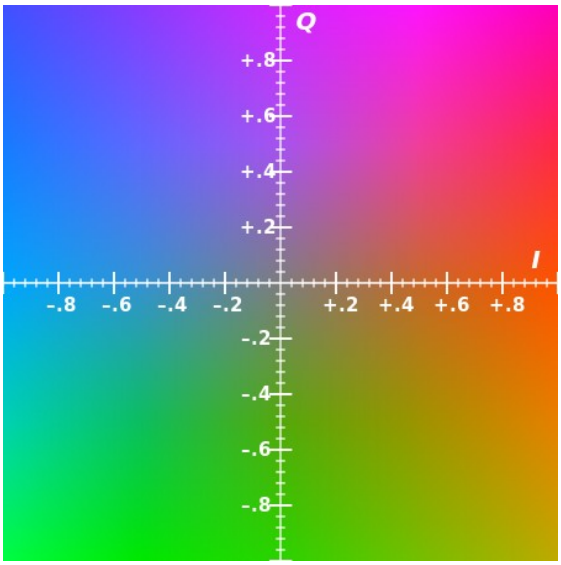
کـاربرد هـای این مـدل هم در شـکل ۶ در سـتون Application بیان شده است.

۲- مـدل RYB: یـک مـدل بـه معنـای قرمـز-زرد-آبی میباشد. یک مـدل رنگی تفـریقی میباشـد کـه در هـنر و design applied که در آن‌هـا این سـه رنـگ رنگ‌هـای اصلی حساب میشوند, کاربرد دارد.

شکل ۷) مدل رنگ RYB

در این مدل سه رنگ قرمـز و زرد و آبی رنگ‌هـای اصـلی حساب می‌شوند و رنگ‌های ثانوی در این مـدل نـارنجی و سبز و بنفش هستند و از ترکیب تمان آن‌ها رنگ سیاه بـه دست می‌آید. برای تبدیل RGB بـه RYB هم تنهـا لازم هست تا رنگ‌های قرمز و سبز و آبی بـه قرمـز و زرد و آبی تبدیل شوند.
هنرمندان از این مدل رنگی استفاده میکنند

۳- مـدل YIQ: این فضـای رنگی در در سیسـتم‌های تلویزیـونی آنـالوگ NTSC بـه صـورت عمـده در ژاپن و آمریکای شمالی و مرکزی استفاده میشده است.
Y کامپوننتی است برای luma information. کامپوننت های I و Q نشان دهنـده chrominance information هستند.
این سیستم از human color-response بهره میگیرد. چشم بیشت ر بـه تغیـرات در رنج آبی-نـارنجی حسـاس است تا رنج سبز-بنفش.



شکل ۸) YIQ color space

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \approx \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.5959 & -0.2746 & -0.3213 \\ 0.2115 & -0.5227 & 0.3112 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

شکل ۹) روابط تبدیل RGB به YIQ

۲-۳ تمرین ۱-۲-۶

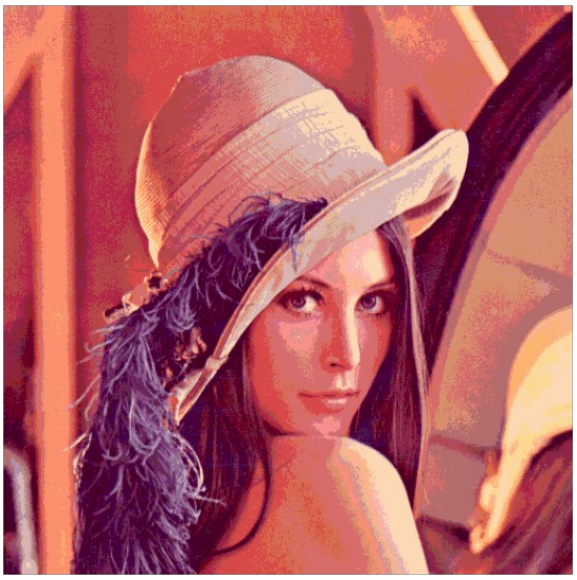
در این تمرین باید colored image quantization را بر روی تصویر لنا به ترتیب با سطوح quantization , به صورت ۸ و ۱۶ و ۳۲ و۶۴ ایجاد کرد و بعد بر وری هر کدام از این تصاویر mse و PSNR آن هارا با تصویر اصلی سنجید. برای PSNR از کتابخانه cv2 استفاده می‌شود و برای mse از تابع mse که در تمرین سری اول مربوط به فصل ۲ بود استفاده شده است.

برای کوانتایز کردن تصاویر ر نگی کافیست تا هر چنل آن‌ها جداگانه کوانتایز شود.

برای این منظور تابع Quantize_colored پیاده‌سازی شده است که عکس ورودی به همراه سطح کوانتایز خواسته شده برای هر سه چنل قرمز و سبز و آبی را دریافت میکند.

هر چنل رنگی به صورت جداگانه توسط تابع Quantize که در تمرین سری اول برای کوانتایز کردن تصاویر خاکستری استفاده شده بود, کوانتایز میشود. در انتها هر کدام از این مولفه ها در جای مخصوص خود در تصویر خروجی کنار هم قرار میگیرند و تصویر به همراه mse و PSNR تصویر با تصویر اولیه به عنوان خروجی داده میشوند.

بررسی نتایج عکس‌های کوانتایز شده با mse و PSNR آن‌ها :



شکل ۱۰) تصویر لنا کوانتایز شده با ۸ سطح

شکل ۱۱) تصویر لنا کوانتایز شده با ۱۶ سطح

شکل ۱۲) تصویر لنا کوانتایز شده با ۳۲ سطح

شکل ۱۳) تصویر لنا کوانتایز شده با ۶۴ سطح

جدول ۱) جدول بررسی نتایج mse و PSNR برای تصاویر کوانتایز شده لنا با سطوح مختلف

سطح کوانتایز	mse	PSNR
۸	۷۴.۵۴	۲۶.۱۱
۱۶	۱۳.۳۱	۳۳.۱۷
۳۲	۱۳.۳۲	۳۶.۸۸
۶۴	۱.۵۵	۴۶.۲۰

با توجه به شکل ۱۰ که تصویر کوانتایز شده با سطح ۸ میباشد, کیفیت تصویر به طور محسوسی کاهش یافته است.

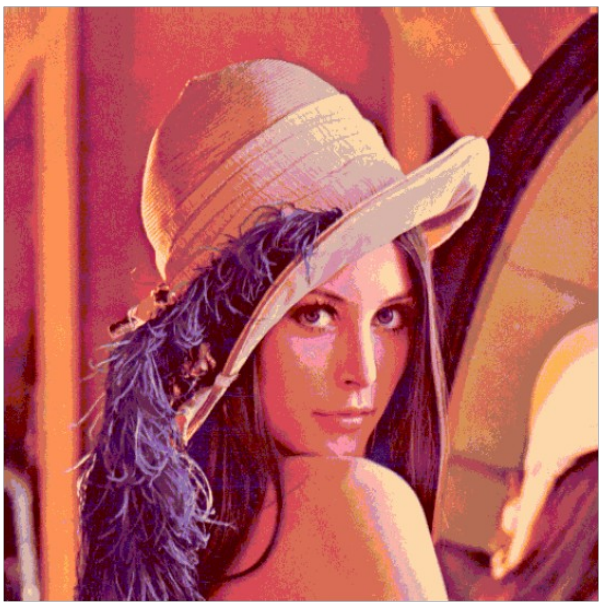
با توجه به جدول ۱ مشاهده می‌شود که با اضافه کردن سطوح کوانتایز کیفیت تصویر افزایش پیدا کرده است چون از تعداد سطوح ۸ و ۱۶ و ۳۲ و۶۴ , مقدار PSNR به صورت صعودی افزایش پیدا کرده و مقدار MSE نیز که ارور میباشد با عکس اصلی کاهش پیدا کرده است. به این معنا که با افزایش سطوح کوانتایز عکس به عمس اصلی نزدیک‌تر میباشد و اگر تعداد سطوح خیلی کم باشد مانند شکل ۱۰ کیفیت تصویر خراب می‌شود.

شکل‌های ۱۱و ۱۲ و۱۳ که با سطوح بیشتری موانتایز شده‌اند از شکل ۱۰ کیفیت بیشتری دارند.

۲-۴ تمرین ۲-۲-۶

در این تمرین خواسته شده است تا دوباره تصویر رنگی لنا کوانتایز شود و نتایج mse و PSNR بررسی شوند اما تفاوتی که این سؤال با سؤال ۱-۲-۶ دارد در این است که در سؤال قبل تعداد سطوح کوانتایز برای هر سه چنل رنگی با هم برابر بود اما در این سؤال خواسته شده است تا چنل قرمز و سبز با ۳ بیت و چنل آبی با ۲ بیت کوانتایز شوند که معنای ۸ سطح کوانتیزیشن برای چنل قرمز و سبز و ۴ سطح کوانتیزیشن برای چنل آبی میباشد.

برای این سؤال از همان تابع Quantize_colored که در سؤال قبل توضیح داده شود استفاده می‌شود با ورودی های عکس رنگی لنا و اعداد ۸ و ۸ و ۴ به ترتیب برا سطوح کوانتیزیشن چنل های قرمز و سبز و ابی.



شکل ۱۴) تصویر کوانتایز شده لنا با سطوح کوانتیزیشن ۸ و ۸ و ۴ به ترتیب برای چنل های قرمز و سبز و آبی

مقدار mse برای شکل ۱۴ با توجه بـه تصـویر لنـا اصـلی مقدار ۷۵.۹۷ و مقدار PSNR برابر با ۲۲.۵۶ میباشد.
با توجه به مقادیر جدول ۱ مشاهده می‌شود که با سـطوح کوانتیزیشن ۸ و ۸ و ۴ برای چنل های قرمز و سـبز و آبی بیشترین mse و کمترین PSNR نسبت به دیگر مقـادیر جدول ۱, دارد. همچـنین دیـده می‌شـود کـه نتیجـه این کوانتیزیشن که در شکل ۱۴ آمده است به شدت با تصویر اصلی تفـاوت دارد و بی کیفیت اسـت. شـکل ۱۴ بـدترین کیفیت را در بین شکل‌های کوانتایز شده ۱۰تا ۱۴ دارد.

۲-۵ تمرین ۳-۲-۶

در این تمرین خواسته شده است تا تصویر رنگی baboon را با ۳۲ و ۱۶ و ۸ رنگ نشـان دهیم و mse را بـرای این تصاویر محاسبه کنیم.
درواقع عملی که این سؤال میخواهد این اسـت کـه تعـداد کل رنگ‌ها را به طور مثال به ۳۲ رنـگ کـاهش دهـد. این مسأله color quantizaion را میتوان با کمک گرفتن از روش‌های خوشه بندی حـل کـرد بـه این صـورت کـه بـا آموزش دادن یک مدل خوشه بنـدی ماننـد kmeans بـر روی عکس, الگـوریتم عکس را بخش بنـدی و segment میکند و چنانچه تعداد خوشه هارا بـه الگـوریتم kmeans ۳۲ و ۱۶ و ۸ رد کنیم, این الگوریتم هر بـار ۳۲ و ۱۶ و ۸ کلاستر را پیدا مکیند یا به عبارتی تصویر را به ۳۲ و ۱۶ و ۸ بخش سگمنت میکند که به معنای ۳۲ و ۱۶ و ۸ رنـگ شدن تصویر است که خواسته این مسأله است.

تابع safe_rgb مراحل توضیح داده شده را انجام میدهـد. این تابع برای ورودی عکس بابون رنگی و تعداد رنگ‌هـایی که میخواهیم در انتها تصویر شامل این تعداد رنگ (تعـداد خوشه ها) باشد را دریافت میکند.
این تـابع ابتـدا یـک اینسـتنس از kmeans موجـود در کتابخانـه scikit learn میسـازد. چـون kmeans این کتابخانه باید داده که میخواهد خوشه بندی کند ۲ بعـدی باشد و عکس ۳ بعدی است, عکس را reshape میکنیم به گونه‌ای کـه دایمنشـن اول آن حاصلضـرب طـول و عـرض تصویر باشد و دایمنشن دوم آن به تعداد چنل هـای رنگی که ۳ است باشد.
Kmeans را بـرروی این عکس تغیـیر سـایز یافتـه اجـرا میکنیم و با تابع های آماده این کتابخانه لیبل ها ی نتـایج برای هر یک از پیکسل ها و centroid های نتیجه خوشه بندی را دریافت می کنیم.

با یـک حلقـه روی labels پیمـایش میکـنیم و بـرای هـر پیکسل با استفاده از لیبل آن, centroid را که به آن تلق دارد را پیدا کرده و مقدار این centroid را به جـای رنـگ سابق این پیکسل قرار میدهیم.

در انتها عکس سگمنت شده و mse آن و عکس اصلی بـه عنوان خروجی این تابع برگردانده میشوند.

شکل ۱۵) تصویر Baboon رنگی

شکل ۱۶) تصویر بابون نشان داده شده با ۸ رنگ

شکل ۱۷) تصویر بابون نشان داده شده با ۱۶ رنگ

شکل ۱۸) تصویر بابون نشان داده شده با ۳۲ رنگ

جدول ۲) جدول نتایج mse برای نشان دادن تصویر بابون با تعداد رنگ های متفاوت

mse	تعداد رنگ
۸۶.۴۸	۸
۷۷.۶۶	۱۶
۶۶.۹۲	۳۲

شکل ۱۶ که تصویر سگمنت شده به ۸ رنگ شکل ۱۵ هست دیده می‌شود که در ناحیه آبی رنگ تنوع رنگی کمتری دارد و با یگ نشان داده شده است و در ناحیه موهای پایین سمت چپ همه موها با یه رنگ مشخص شده در حالی که این موها پایین سمت چپ تصویر در شکل ۱۵ چند رنگ هستند. شکل‌های ۱۷ و ۱۸ تعداد رنگ‌های بیشتری از شکل ۱۶ دارند.

با توجه به جدول ۲ نیز مشاهده می‌شود که شکل ۱۸ که بیشترین تعداد رنگ را دارد کمترین ارور را دارد و از همه بیشتر به شکل اصلی شماره ۱۵ نزدیک است که نشان میدهد با کمتر شدن تعداد رنگ‌ها در تصویر ارور افزایش میابد.

۳-کد تمرینات

```
# -*- coding: utf-8 -*-
"""hw6_sara_ghavampour_9812762781.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ZuYvNYj7VqAQmEeUWVrUinwhV-Ff9F1g
"""

############## Sara Ghavampour 9812762781 ########

!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate

!wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import cv2
from math import *
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans
# %matplotlib inline

def show_img(*args, figsize=10, is_gray=True, title=None, fontsize=12):
    if isinstance(figsize, int):
        figsize = (figsize, figsize)
    images = args[0] if type(args[0]) is list else list(args)
    cmap=None
    if not is_gray:
        images = list(map(lambda x: cv2.cvtColor(x, cv2.COLOR_BGR2RGB), images))
    else:
        cmap = 'gray'
    plt.figure(figsize=figsize)
    for i in range(1, len(images)+1):
        plt.subplot(1, len(images), i)
        if title is not None:
            plt.title(title[i-1], fontsize=fontsize)

        plt.imshow(images[i-1], cmap=cmap)
        plt.axis('off')

lena_img=cv2.imread('Lena.bmp')
lena_img = cv2.cvtColor(lena_img,cv2.COLOR_BGR2RGB)
plt.imshow(lena_img)
plt.axis('off')
plt.show()
#(512, 512) uint8

baboon_img=cv2.imread('Baboon.bmp')
baboon_img = cv2.cvtColor(baboon_img,cv2.COLOR_BGR2RGB)
plt.imshow(baboon_img,cmap='gray')
plt.axis('off')
plt.show()
#(512, 512) uint8

def normalize(img):
  min = np.min(img)
  max = np.max(img)
  return ((img-min)/(max-min)*255).astype('uint8')

"""
6.1.1"""

############## 6.1.1 ########################

def HSI_converter(img):
  red,green,blue = img[:,:,0],img[:,:,1],img[:,:,2]
  # scale rgb between 0-1
  red =  red / 255.0
  green =green / 255.0
  blue =   blue /255.0

  H = np.zeros((img.shape[0],img.shape[1]))
  S = np.zeros((img.shape[0],img.shape[1]))

# theta formula in page 62 of slide 7
  hue_numerator = 1/2 * ((red-green)+(red-blue))
  hue_denumerator = np.sqrt((red-green)**2 + ((red-blue)*(green - blue)))
  hue = np.arccos(hue_numerator / hue_denumerator)

  # 2 conditions for  H
  for i in range (img.shape[0]):
    for j in range(img.shape[1]):
      if hue_denumerator[i,j]  != 0 :
        if blue[i,j]<= green[i,j] : H[i,j]=hue[i,j]
        if blue[i,j]> green[i,j] : H[i,j]=np.pi- hue[i,j]
      ###################################
      ## compute s
      mean = min(min(red[i,j],green[i,j]),blue[i,j])
      S[i,j]= 1-(3/(red[i,j]+green[i,j]+blue[i,j]))*mean

  # compute I
  I = (red + green + blue) / 3

  H = normalize(H)
  return H,S,I

H_lena,S_lena ,I_lena = HSI_converter(lena_img)

show_img(H_lena)

show_img(S_lena)

show_img(I_lena)

"""6.2.1"""

#################### 6.2.1 ##########

def mse(img1,img2):
  pic_1=img1.copy().ravel().astype('uint8')
  pic_2=img2.copy().ravel().astype('uint8')
  sqr_diff=(pic_1 - pic_2)**2
  size=pic_1.shape[0]
  err = np.sum(sqr_diff) / size
  return err

def Quantize_channel(img,n_final__levels):
```

```
  output= np.floor(img / (256/n_final__levels))
  output= output * np.floor(255/(n_final__levels -1 ))
  return output

def Quantize_colored(img,r_level,g_level,b_level):
  r_channel , g_channel , b_channel = img[:,:,0] , img[:,:,1],img[:,:,2]
  # quantise each color channel speratly
  r_channel = Quantize_channel(r_channel,r_level)
  g_channel = Quantize_channel(g_channel,g_level)
  b_channel = Quantize_channel(b_channel,b_level)

  quantized_img = np.zeros((img.shape))
  quantized_img[:,:,0] = r_channel
  quantized_img[:,:,1] = g_channel
  quantized_img[:,:,2] = b_channel

  quantized_img=quantized_img.astype('uint8')

  psnr  = cv2.PSNR(img,quantized_img)

  return quantized_img , mse(img.astype('uint8'),quantized_img) , psnr

quantized_lena_8,mse_8,psnr_8 = Quantize_colored(lena_img,8,8,8)
quantized_lena_16,mse_16,psnr_16 = Quantize_colored(lena_img,16,16,16)
quantized_lena_32,mse_32,psnr_32 = Quantize_colored(lena_img,32,32,32)
quantized_lena_64,mse_64,psnr_64 = Quantize_colored(lena_img,64,64,64)

print('mse_8: ',mse_8,' psnr_8: ',psnr_8)
show_img(quantized_lena_8)

print('mse_16: ',mse_16,' psnr_16: ',psnr_16)
show_img(quantized_lena_16)

print('mse_32: ',mse_32,' psnr_32: ',psnr_32)
show_img(quantized_lena_32)

print('mse_64: ',mse_64,' psnr_64: ',psnr_64)
show_img(quantized_lena_64)

"""6.2.2"""

################## 6.2.2 ######################

# 3,3,2 bits for r,g,b --> 8,8,4 levels for r,g,b
q_622_img,q_622_mse,q_622_psnr = Quantize_colored(lena_img,8,8,4)
print('q_622_mse: ',q_622_mse,' q_622_psnr: ',q_622_psnr)
show_img(q_622_img)

"""6.2.3"""

############# 6.2.3 ###################

def safe_rgb(img,colors_count):
  kmeans = KMeans(colors_count,random_state=0)
  clustered_img = np.reshape(img, (img.shape[0] * img.shape[1], img.shape[2]))
  kmeans.fit_predict(clustered_img)

  cluster_centroids = kmeans.cluster_centers_ .astype('uint8')
  labels = kmeans.labels_.reshape(img.shape[0],img.shape[1])

  segmented_pic = np.zeros((img.shape))

  for i in range(labels.shape[0]):
    for j in range(labels.shape[1]):
      segmented_pic[i,j,:]=cluster_centroids[labels[i,j]].astype('uint8')


  return segmented_pic.astype('uint8') , mse(img,segmented_pic.astype('uint8'))

segmented_baboon_8,mse_baboon_8 = safe_rgb(baboon_img,8)
segmented_baboon_16,mse_baboon_16 = safe_rgb(baboon_img,16)
segmented_baboon_32,mse_baboon_32 = safe_rgb(baboon_img,32)

print('mse_baboon_8 : ', mse_baboon_8)
show_img(segmented_baboon_8)

print('mse_baboon_16 : ', mse_baboon_16)
show_img(segmented_baboon_16)

print('mse_baboon_32 : ', mse_baboon_32)
show_img(segmented_baboon_32)

################ The End of hw6 ########
```