گروه كامپيوتر - دانشكده مهندسي - دانشگاه فردوسي مشهد

مستندات گزارش تمرین ششم

درس مبانی بینایی کامپیوتر

**ارائه ‌دهنده:**

**صالح شیروانی**

**استاد درس:**

**دکتر امیرحسین طاهری نیا**

**پاییز1401**

چکیده

این گزارش به مباحث مربوط به پردازش تصویر رنگی و فضا های رنگی می‌پردازد. در این تمرین فضا های رنگی مختلف را بررسی می‌کنیم مثل فضای رنگی HSI که بیشترین هماهنگی را با درک مغز انسان از رنگ دارد و به همین دلیل بررسی آن می‌تواند مفید باشد. در این تمرین همچنین به بررسی فظای رنگی RGB می‌پردازیم و در نهایت با پیاده سازی و بررسی Quantization روی کانال های مختلف تصاویر رنگی بحث را به پایان می‌بریم.

Technical Description

**6- Color**

6-1- Color space

6-1-1- در این تمرین از ما خواسته شده تا تصویرLena را در فضای رنگی HSI نمایش دهیم. همچنین باید هر یک از مولفه های H، S و I را به طور جداگانه به صورت یک تصویر grayscale نمایش دهیم. برای تبدیل یک تصویر در فضای RGB به فضای HSI به صورت زیر عمل می‌کنیم.

ابتدا کانال های R، G و B را به دست می‌آوریم و ذخیره می‌کنیم.

برای محاسبه ی I مطابق فرمول زیر عمل می‌کنیم:

یعنی بین سه کانال R، G و B که در گام قبل به دست آوردیم میانگین می‌گیریم.

همچنین برای محاسبه S از فرمول زیر بهره می‌گیریم:

که این فرمول دقیقا همان فرمول مطرح شده در کتاب گونزالس است.

در نهایت برای محاسبه ی H به صورت زیر عمل می‌کنیم:

حال که هریک از کانال های HSI را به طور جداگانه حساب کرده ایم با ترکیب این کانال ها تصویر مورد نظر در فضای رنگی HSI به دست می‌آید.

6-1-2- در این تمرین به معرفی سه فضای رنگی LCH وXYZ وYIQ می‌پردازیم و کاربرد ها و روابط مربوط به این فضاهای رنگی را مورد بررسی قرار می‌دهیم.

* LCH مخفف "Lightness، Chroma، Hue" است. فضای LCH بر اساس فضای رنگی CIELAB است، اما LCH اطلاعات رنگ را به سه کانال جداگانه تقسیم می کند: Lightness، Chroma و Hue. در فضای رنگی LCH، Lightness نشان دهنده روشنایی یا تیرگی یک رنگ است. همچنین کروما نشان دهنده شدت یا اشباع یک رنگ می‌باشد. Hue نشان دهنده طول موج غالب یک رنگ است. فضای رنگی LCH اغلب در رنگ سنجی(colorimetry) و علم رنگ استفاده می شود زیرا اطلاعات رنگ را به کانال های مجزا جدا می کند و از نظر ادراکی یکنواخت است، به این معنی که تغییرات در مقادیر رنگ توسط چشم انسان در تمام قسمت های فضای رنگی یکسان درک می شود. این باعث می شود که ابزار مفیدی برای تطبیق رنگ و تصحیح رنگ باشد.

کاربرد های فضای رنگی LCH به صورت زیر است:

Translation is too long to be saved

* + - * تطبیق رنگ: فضای رنگی LCH اغلب در برنامه های تطبیق رنگ(color matching) استفاده می شود زیرا اطلاعات رنگ را به کانال های مجزا جدا می کند و مقایسه و مطابقت رنگ ها را آسان تر می کند. این قابلیت آن را به ابزاری مفید برای تصحیح رنگ در عکاسی، چاپ و سایر کاربردها تبدیل می کند که در آن بازتولید دقیق رنگ مهم است.
      * طراحی گرافیکی: فضای رنگی LCH در طراحی گرافیکی برای کمک به انتخاب و هماهنگی رنگ ها و همچنین ایجاد هماهنگی رنگ و طرح های رنگی استفاده می شود.
* تصویربرداری پزشکی: فضای رنگی LCH در تصویربرداری پزشکی برای کمک به تمایز بین انواع مختلف بافت مانند تومورها، عروق خونی و سایر ساختارها استفاده می شود.
* دسترسی به رنگ(color accessibility): از فضای رنگی LCH در طراحی دسترس پذیری استفاده می شود تا اطمینان حاصل شود که انتخاب رنگ برای افراد دارای نقص دید رنگ قابل تشخیص است.

برای تبدیل فضای رنگی CLH به CIELAB می توان از فرمول های زیر بهره برد:

در این فرمول ها ، و مقادیر مربوط به فضای CIELAB هستند

* YIQ یک مدل رنگی است که در encoding تلویزیون آنالوگ برای جداسازی سیگنال‌های کرومینانس (اطلاعات رنگ) و روشنایی (اطلاعات روشنایی) استفاده می‌شود. این فضای رنگی در سیستم تلویزیون NTSC استفاده می شود.

کاربرد های این فضای رنگی به صورت زیر است:

* Analog TV encoding
* Image and video processing
* Color correction and balancing
* Color detection in computer vision

برای تبدیل RGB به YIQ می‌توان از فرمول زیر بهره برد:

همچنین برای تبدیل YIQ به RGB از فرمول های زیر استفاده می‌شود:

* LMS (Long-Medium-Short) یک فضای رنگی است که نشان دهنده واکنش سه نوع مخروط در چشم انسان است. فضای رنگی LMS در رنگ سنجی استفاده می شود، زمینه ای که با اندازه گیری رنگ سروکار دارد تا درک رنگ در بینایی انسان را تعریف و کمی کند. این فضا بر این اصل استوار است که انسان، رنگ ها را در نتیجه تحریک سه نوع مخروط در شبکیه می بیند که حساسیت طیفی متفاوتی دارند. فضای رنگی LMS راهی برای اندازه گیری و توصیف رنگ از نظر پاسخ این مخروط ها فراهم می کند و اغلب در برنامه های تصحیح رنگ و تطبیق رنگ استفاده می شود.

فضای رنگی LMS معمولاً از طریق یک ماتریس تبدیل رنگ به فضای رنگی RGB تبدیل می شود.

فرمول زیر برای تبدیل RGB به LMS به کار می‌رود:

که ، و ضرایب ماتریس تبدیل رنگ هستند.

از طرف دیگر برای تبدیل LMS به RGB می‌توان فرمول زیر را به کار برد:

که در آن ، ، و ضرایب ماتریس تبدیل رنگ معکوس هستند.

مقادیر ضرایب ، ، ، ، و به فضای رنگی RGB خاصی که استفاده می‌شود بستگی دارد و می‌تواند بر اساس سیستم خاص و نمایشگر مورد استفاده متفاوت باشد.

برخی از کاربرد های این فضای رنگی به صورت زیر است:

* تصحیح رنگ (Color Correction): فضای رنگی LMS برای تصحیح رنگ ها در تصاویر، فیلم ها و سایر رسانه های بصری برای مطابقت با درک انسان از رنگ استفاده می شود.
* شبیه‌سازی کوررنگی (Color Blindness Simulation): فضای رنگی LMS را می‌توان برای شبیه‌سازی اثرات کوررنگی و ایجاد راه‌حل‌هایی برای افراد مبتلا به کمبود دید رنگ استفاده کرد.
* تشخیص پزشکی (Medical Diagnostics): فضای رنگی LMS در تصویربرداری پزشکی برای تشخیص و نظارت بر شرایط مربوط به دید رنگی مانند آب مروارید و کمبود دید رنگ استفاده می شود.

***6-2- Quantization***

*6-2-1- در این سوال می‌خواهیم uniform quantization را روی یک تصویر پیاده کنیم و تاثیر آن را بررسی کنیم. برای این سوال باید کارهای زیر را انجام دهیم:*

* *پیاده سازی quantization: برای این کار باید شدت تمام کانال های هر پیکسل را در بازه مورد نظر سوال قرار دهیم. به عنوان مثال: در این سوال شدت پیکسل ها در بازه ی (256-0) قرار دارد. اگر L=64 را در نظر بگیریم ابتدا 256 را بر 64 تقسیم می کنیم تا تعداد bin ها به دست آید. در این مثال این تعداد برابر است. سپس برای تمام کانال های هر پیکسل مقدار intensity آن را بر تعداد bin ها تقسیم می‌کنیم تا intensity جدید آن که در بازه مطلوب ما است به دست آید. سپس تصویر quantize شده را ذخیره می‌کنیم. این کار را به ازای L=32,16,8 نیز تکرار می‌کنیم.*
* *محاسبه MSE: برای این کار به صورت زیر عمل می‌کنیم:*
* *محاسبه PSNR: برای این کار به صورت زیر عمل می‌کنیم:*

*که در اینجا منظور از MAX حداکثر Intensity است که در این سوال 255 می‌باشد.*

*6-2-2- در این سوال می‌خواهیم هر یک از کانال های RGB یک تصویر را quantize کنیم. برای اینکار ابتدا تصویر را به سه کانال R، G و B تجزیه می کنیم و از تابع نوشته شده در سوال قبل استفاده می‌کنیم تا این کانال ها را quantize کنیم. در نهایت با کنار هم قرار دادن این کانال ها، تصویر رنگی خروجی را می‌سازیم.*

*6-2-3- در این سوال می‌خواهیم تعداد رنگ های یک تصویر را به گونه ای کاهش بدهیم که حداقل کاهش کیفیت تصویر را داشته باشیم. مطابق صورت سوال باید سه حالت در نظر بگیریم که در هریک به ترتیب تصویر می‌تواند 32، 16 یا 8 رنگ مختلف داشته باشد. در هرکدام از این حالت ها باید مشخص کنیم که هر مولفه ی RGB چند مقدار مختلف می‌تواند اختیار کند تا در نهایت تعداد رنگ مورد نظر حاصل شود. در انجام این کار حالات مختلفی پیش می‌اید که ما تمام آن ها را بررسی می‌کنیم تا ببینیم در کدام یک کمترین کاهش در کیفیت تصویر حاصل می‌شود. به عنوان مثال برای حالتی که می‌خواهیم 32 رنگ در تصویر داشته باشیم حالات زیر پیش می‌آید:*

|  |  |
| --- | --- |
| *کانال* | *تعداد حالات مختلف برای کانال* |
| *R* | *4* |
| *G* | *4* |
| *B* | *2* |

|  |  |
| --- | --- |
| *کانال* | *تعداد حالات مختلف برای کانال* |
| *R* | *4* |
| *G* | *2* |
| *B* | *4* |

|  |  |
| --- | --- |
| *کانال* | *تعداد حالات مختلف برای کانال* |
| *R* | *2* |
| *G* | *4* |
| *B* | *4* |

*همین کار را برای حالاتی که تصویر 16 و 8 رنگ باید داشته باشد نیز تکرار می‌کنیم.*

***Results***

*برخی توابع مهم و کلی که در این تمرین استفاده شده است به صورت زیر هستند:*

def MSE(*image1*, *image2*):

    return np.square(np.subtract(*image1*, *image2*)).mean()

def PSNR(*reference*, *reconstructed*):

*# calculate MSE*

    mse = MSE(*reference*, *reconstructed*)

*# calculate PSNR*

    if mse == 0:

        return float('inf')

    else:

        return 10 \* np.log10(255.0 \*\* 2 / mse)

***6- Color***

*6-1- Color space*

*6-1-1-*

def RGB\_TO\_HSI(*img*):

    with np.errstate(*divide*='ignore', *invalid*='ignore'):

*#Load image with 32 bit floats as variable type*

        bgr = np.float32(*img*)/255

*#Separate color channels*

        blue = bgr[:,:,0]

        green = bgr[:,:,1]

        red = bgr[:,:,2]

*#Calculate Intensity*

        def calc\_intensity(*red*, *blue*, *green*):

            return np.divide(*blue* + *green* + *red*, 3)

*#Calculate Saturation*

        def calc\_saturation(*red*, *blue*, *green*):

            minimum = np.minimum(np.minimum(*red*, *green*), *blue*)

            saturation = 1 - (3 / (*red* + *green* + *blue* + 0.001) \* minimum)

            return saturation

*#Calculate Hue*

        def calc\_hue(*red*, *blue*, *green*):

            hue = np.copy(*red*)

            for i in range(0, *blue*.shape[0]):

                for j in range(0, *blue*.shape[1]):

                    hue[i][j] = 0.5 \* ((*red*[i][j] - *green*[i][j]) + (*red*[i][j] - *blue*[i][j])) / \

                                math.sqrt((*red*[i][j] - *green*[i][j])\*\*2 +

                                        ((*red*[i][j] - *blue*[i][j]) \* (*green*[i][j] - *blue*[i][j])))

                    hue[i][j] = math.acos(hue[i][j])

                    if *blue*[i][j] <= *green*[i][j]:

                        hue[i][j] = hue[i][j]

                    else:

                        hue[i][j] = ((360 \* math.pi) / 180.0) - hue[i][j]

            return hue

*#Merge channels into picture and return image*

        hsi = cv2.merge((calc\_hue(red, blue, green), calc\_saturation(red, blue, green), calc\_intensity(red, blue, green)))

        return hsi

*# convert Lena image from RGB to HSI*

lena\_in\_HSI = RGB\_TO\_HSI(lena)

*# find each H, S and I component of lena image*

lena\_hue = lena\_in\_HSI[:, :, 0]

lena\_saturation = lena\_in\_HSI[:, :, 1]

lena\_intensity = lena\_in\_HSI[:, :, 2]

plot\_image(lena\_in\_HSI, 'Lena in HSI')

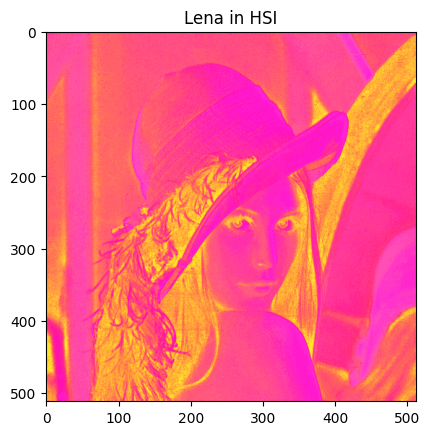
plt.figure(*figsize*=(11, 4))

subplot(1, 3,

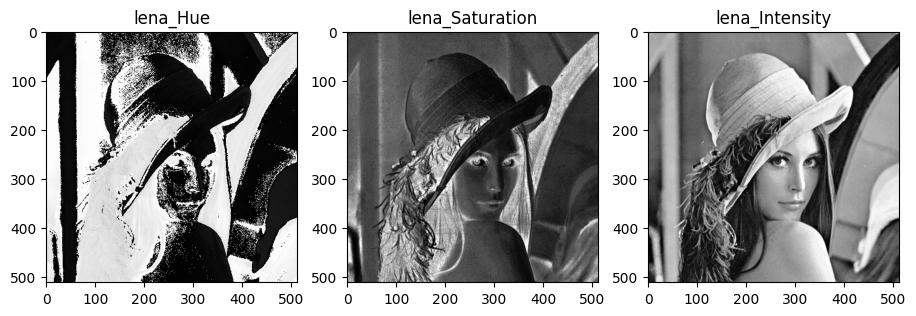
        [lena\_hue, lena\_saturation, lena\_intensity],

        ['lena\_Hue', 'lena\_Saturation', 'lena\_Intensity'])

*تصویر زیر تصویر Lena در فضای رنگی HSI را نشان می‌دهد.*

**

*تصویر زیر هر یک از مولفه های H، S و I تصویر بالا را نشان می‌دهد:*

**

*می‌دانیم H اطلاعات مربوط به رنگ غالب تصویر را نشان می‌دهد. بنابراین مقادیر مختلف H به مقادیر grayscale نگاشت شده است. بنابراین نقاطی از تصویر که مقدار graylevel آن ها بیشتر است (یعنی روشن تر هستند) مقدار H بیشتری داشته اند و نقاطی که مقدار graylevel آن ها کمتر است (تیره تر هستند) مقدار H آن ها کمتر بوده است. مثلا اگر دقت کنیم نواحی که رنگ غالب آن ها در تصویر اصلی قرمز بوده است در تصویر grayscale تیره تر هستند.*

*همچنین می‌دانیم که مقدار S خلوص رنگ ها را نشان می‌دهد. بنابراین در تصویر grayscale مربوط به S نواحی تیره پیکسل هایی را نشان می‌دهند که خلوص رنگ در آن ها کمتر بوده و نواحی روشن تر مربوط به نواحی هستند که خلوص رنگ در تصویر اصلی در آن ها بیشتر بوده است.*

*مقدار I روشنایی تصویر را نشان می‌دهد مانند دو قسمت قبل می‌توان این طور استنباط کرد که نواحی روشن تر در تصویر grayscale نواحی را نشان می‌دهد که مقدار I آن ها در تصویر اصلی بیشتر بوده و نواحی تیره تر نواحی را نشان می‌دهند که مقدار I آن‌ها در تصویر اصلی کمتر بوده(یعنی تاریک تر بوده اند.)*

*6-1-2- سه فضای رنگی به همراه کاربرد ها و معادلات مربوطه در بخش شرح فنی (technical description) مربوط به این مساله در همین گزارش معرفی شده است.*

*6-2- Quantization*

*یک تابع کلی که در سوال 2-6 زیاد از آن استفاده می‌شود به صورت زیر است:*

def quantize\_image(*image*, *levels\_per\_channel*):

*# Convert the image to a NumPy array*

  image\_array = np.array(*image*)

*# Find the size of the bins for each channel*

  bin\_size = 256 // *levels\_per\_channel*

*# Iterate over each pixel in the image*

  for i in range(image\_array.shape[0]):

    for j in range(image\_array.shape[1]):

*# Divide the intensity values by the size of the bins*

*# to find which bin the intensity falls into*

      binned\_values = image\_array[i, j] // bin\_size

*# Replace the intensity values with the center value of the bins*

      image\_array[i, j] = bin\_size \* (binned\_values + 0.5)

  return image\_array

***6-2-1-***

L\_values = [64, 32, 16, 8, 4, 2]

quantized\_images = []

MSE\_values = []

PSNR\_values = []

for L in L\_values:

    quantized\_lena = quantize\_image(lena, L)

    quantized\_images.append(quantized\_lena)

    mse = MSE(lena, quantized\_lena)

    MSE\_values.append(mse)

    psnr = PSNR(lena, quantized\_lena)

    PSNR\_values.append(psnr)

plt.figure(*figsize*=(8, 12))

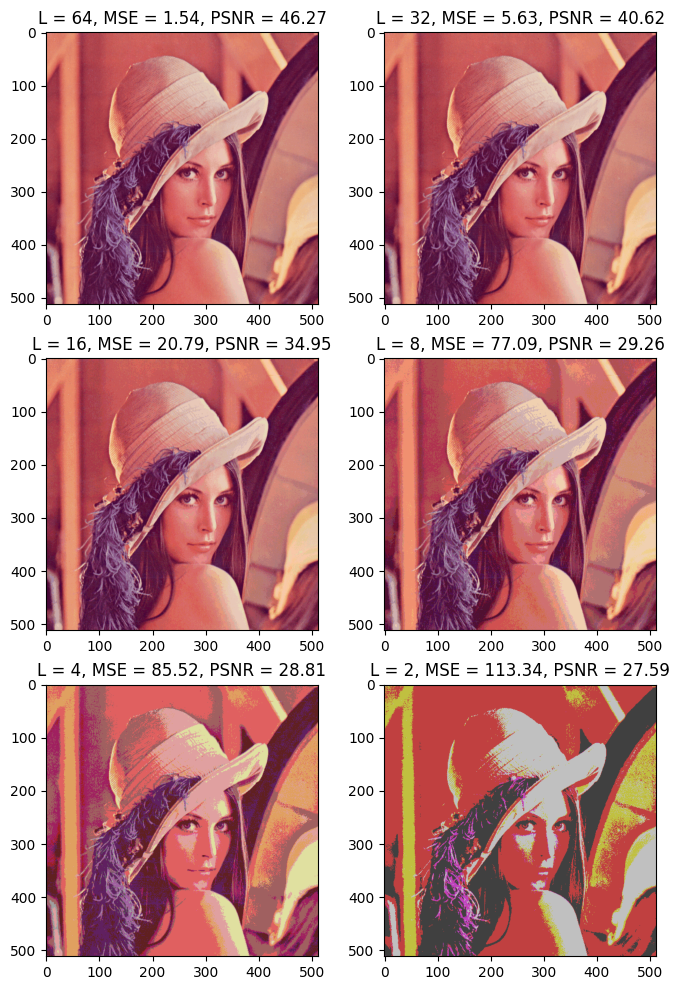
titles = []

for i in range(len(L\_values)):

    titles.append('L = {}, MSE = {:.2f}, PSNR = {:.2f}'

                  .format(L\_values[i], MSE\_values[i], PSNR\_values[i]))

subplot(3, 2, quantized\_images, titles)

**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *2* | *4* | *8* | *16* | *32* | *64* |  |
| *113.34* | *85.52* | *77.09* | *20.79* | *5.63* | *1.54* | *MSE* |
| *27.59* | *28.81* | *29.26* | *34.95* | *4.62* | *46.27* | *PSNR* |

*با توجه به تصاویر بالا و مقادیر MSE و PSNR واضح است که با کاهش مقدار L کیفیت و کنتراست تصویر کاهش می‌یابد که می‌تواند منجر به حذف برخی جزئیات تصویر شود. بنابراین هر چه مقدار L بیشتر باشد تصویر quantize شده به تصویر اصلی نزدیکتر است. همچنین کاهش L منجر به کاهش تعداد رنگ های تصویر می‌شود.*

*با توجه به تصاویر، از لحاظ بصری تمایز زیادی نمی‌توان برای تصویر متناظر با L=64 و L=32 و حتی L=16 قائل شد و این کمک می‌کند که بتوانیم با حذف برخی جزئیات کم اهمیت حجم تصویر را کمتر کنیم.*

*6-2-2-*

def quantize\_RGB\_channels(*img*, *R\_levels*, *G\_levels*, *B\_levels*):

    R, G, B = *img*[:, :, 0], *img*[:, :, 1], *img*[:, :, 2]

    R\_quantized = quantize\_image(R, *R\_levels*)

    G\_quantized = quantize\_image(G, *G\_levels*)

    B\_quantized = quantize\_image(B, *B\_levels*)

    return np.dstack((R\_quantized, G\_quantized, B\_quantized))

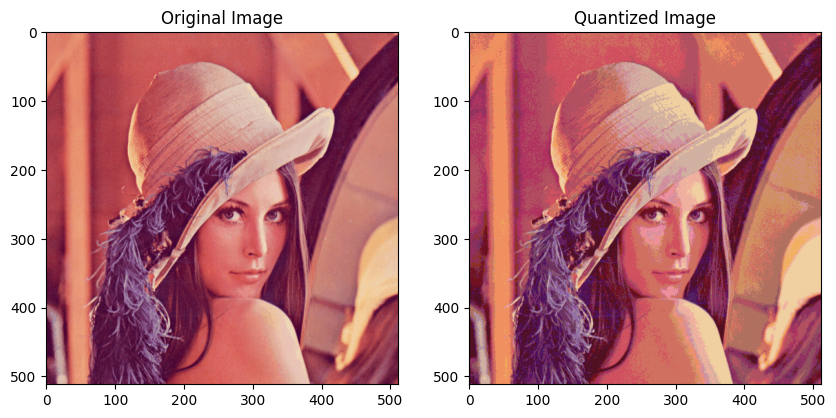
lena\_RGB\_quantized = quantize\_RGB\_channels(lena, 2\*\*3, 2\*\*3, 2\*\*2)

plt.figure(*figsize*=(10, 12))

subplot(1, 2, [lena, lena\_RGB\_quantized],

        ['Original Image', 'Quantized Image'])

*تصاویر خروجی به صورت زیر هستند:*

**

*با توجه به تصاویر بالا، می‌توان دید که کیفیت تصویر در تصویر quantize شده کاهش پیدا کرده است. در واقع ما تعداد سطوح شدتی که هر مولفه رنگ می‌تواند داشته باشد را کاهش داده ایم و این یعنی تعداد رنگ های تصویر quantize شده نسبت به تصویر اصلی کمتر است. در تصویر quntize شده طبق صورت سوال برای مولفه های R، G و B، به ترتیب 3 بیت، 2 بیت و 2 بیت در نظر گرفته شده است و این یعنی تعداد کل رنگ هایی که می‌تواند در این تصویر وجود داشته باشد به رنگ کاهش یافته است. بنابراین واضح است که دقت رنگ در تصویر کاهش می‌یابد. همچنین کاهش تعداد رنگ های تصویر منجر به کاهش کنتراست تصویر نیز شده است.*

*6-2-3-*

*# Baboon image with 8 colors.*

*# (R <- 2 & G <- 2 & B <- 2) ==> 2 \* 2 \* 2 = 8*

baboon\_8\_colors = quantize\_RGB\_channels(baboon, 2, 2, 2)

mse = MSE(baboon, baboon\_8\_colors)

psnr = PSNR(baboon, baboon\_8\_colors)

plot\_image(baboon\_8\_colors, 'R=2, G=2, B=2 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

*# (R <- 8 & G <- 1 & B <- 1) ==> 8 \* 1 \* 1 = 8*

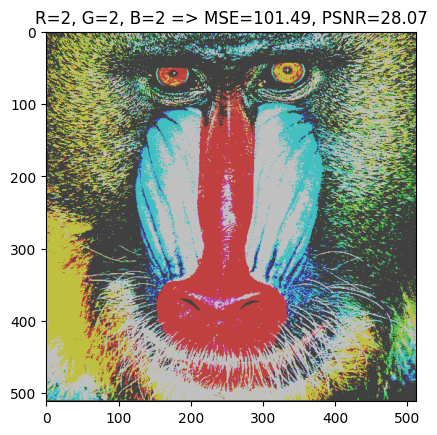
baboon\_8\_colors = quantize\_RGB\_channels(baboon, 8, 1, 1)

mse = MSE(baboon, baboon\_8\_colors)

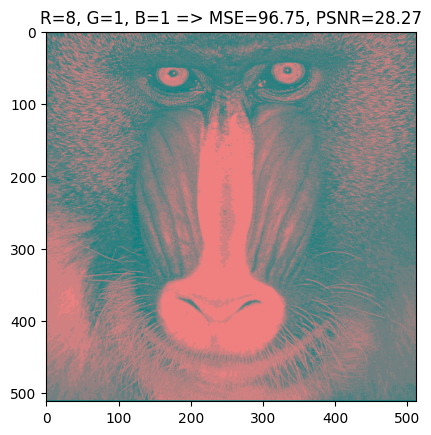
psnr = PSNR(baboon, baboon\_8\_colors)

plot\_image(baboon\_8\_colors, 'R=8, G=1, B=1 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

*تصاویر خروجی به صورت زیر است:*

**

1-تصویر با 8 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

**

2-تصویر با 8 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

*# (R <- 4 & G <- 2 & B <- 2) ==> 4 \* 2 \* 2 = 16*

baboon\_16\_colors = quantize\_RGB\_channels(baboon, 4, 2, 2)

mse = MSE(baboon, baboon\_16\_colors)

psnr = PSNR(baboon, baboon\_16\_colors)

plot\_image(baboon\_16\_colors, 'R=4, G=2, B=2 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

*# (R <- 2 & G <- 4 & B <- 2) ==> 2 \* 4 \* 2 = 16*

baboon\_16\_colors = quantize\_RGB\_channels(baboon, 2, 4, 2)

mse = MSE(baboon, baboon\_16\_colors)

psnr = PSNR(baboon, baboon\_16\_colors)

plot\_image(baboon\_16\_colors, 'R=2, G=4, B=2 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

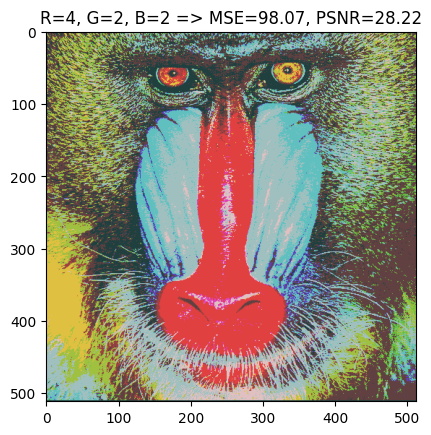
*# (R <- 2 & G <- 2 & B <- 4) ==> 2 \* 2 \* 4 = 16*

baboon\_16\_colors = quantize\_RGB\_channels(baboon, 2, 2, 4)

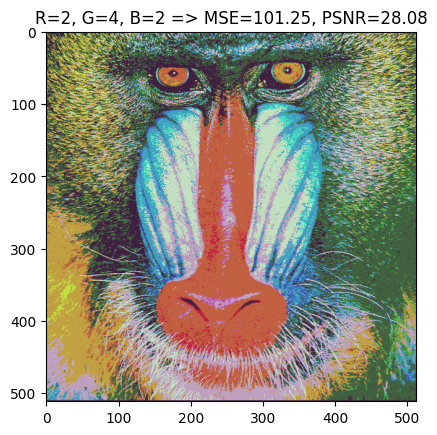
mse = MSE(baboon, baboon\_16\_colors)

psnr = PSNR(baboon, baboon\_16\_colors)

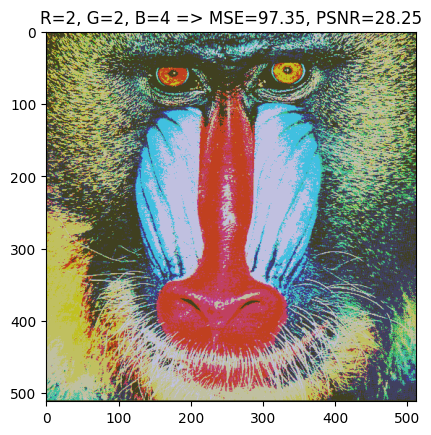
plot\_image(baboon\_16\_colors, 'R=2, G=2, B=4 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

**

3-تصویر با 16 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

**

4-تصویر با 16 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

**

5-تصویر با 16 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

*# (R <- 4 & G <- 4 & B <- 2) ==> 4 \* 4 \* 2 = 32*

baboon\_32\_colors = quantize\_RGB\_channels(baboon, 4, 4, 2)

mse = MSE(baboon, baboon\_32\_colors)

psnr = PSNR(baboon, baboon\_32\_colors)

plot\_image(baboon\_32\_colors, 'R=4, G=4, B=2 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

*# (R <- 4 & G <- 2 & B <- 4) ==> 4 \* 2 \* 4 = 32*

baboon\_32\_colors = quantize\_RGB\_channels(baboon, 4, 2, 4)

mse = MSE(baboon, baboon\_32\_colors)

psnr = PSNR(baboon, baboon\_32\_colors)

plot\_image(baboon\_32\_colors, 'R=4, G=2, B=4 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

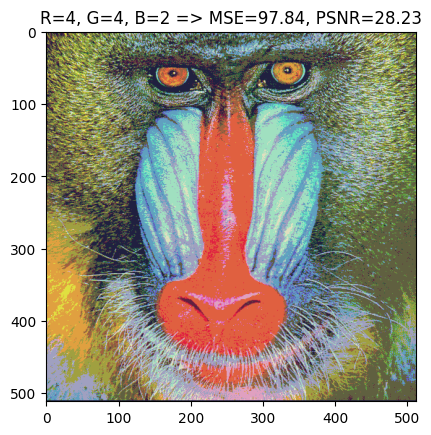
*# (R <- 2 & G <- 4 & B <- 4) ==> 2 \* 4 \* 4 = 32*

baboon\_32\_colors = quantize\_RGB\_channels(baboon, 2, 4, 4)

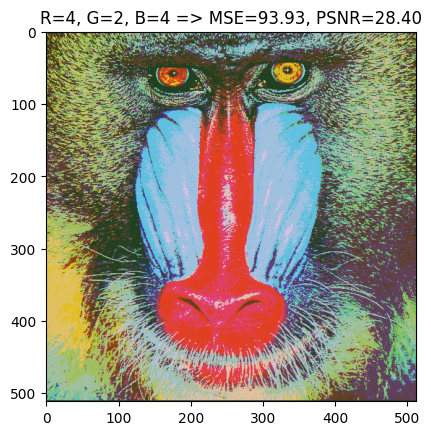
mse = MSE(baboon, baboon\_32\_colors)

psnr = PSNR(baboon, baboon\_32\_colors)

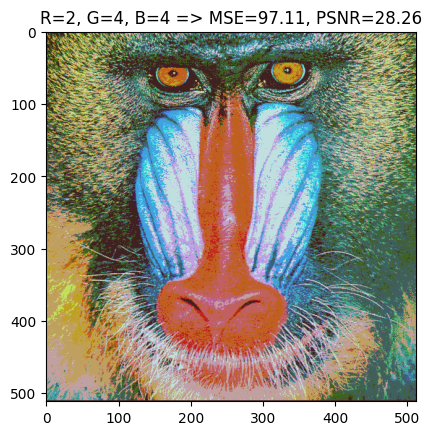
plot\_image(baboon\_32\_colors, 'R=2, G=4, B=4 => MSE={:.2f}, PSNR={:.2f}'.format(mse, psnr))

**

6-تصویر با 32 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

**

7--تصویر با 32 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

**

8-تصویر با 32 رنگ (تعدادمقادیر هر مولفه RGB در بالای تصویر نوشته شده است)

*تصویر اصلی به صورت زیر است:*

**

***پیوست***

* *کدها و تصاویر مربوط به این تمرین در*[*GitHub*](https://github.com/saleh-sh/ComputerVision_ColorImages_FUM) *در دسترس است.*