Color

شیوا رادمنش

اطلاعات گزارش

تاریخ: ۱۳۹۹/۱۰/۵

واژگان کلیدی:
quantization
saturation
hue
فضای رنگی

چکیده

در این گزارش به معرفی و بررسی فضاهای رنگی مختلف مانند HSI و HCL و YUV و YIQ و نحوه‌ی تبدیلات آنها از/به RGB ، کاهش تعداد رنگ‌های تصویر پرداخته شده است.

۱- مقدمه

رنگ یک توصیف‌گر قدرتمند است که تشخیص شی و استخراج آن از صحنه را آسان می‌کند. انسان می‌تواند هزاران شدت و سایه رنگی را از یکدیگر تمیز دهد در حالی که فقط تعداد محدودی سطح خاکستری را از هم تشخیص می‌دهد. در این تمرین به بررسی فضاهای رنگی و روش‌هایی برای کاهش تعداد رنگ‌های تصویر پرداخته شده است. تمامی پیاده سازی های این تمرین با استفاده از زبان پایتون انجام شده است.

۲- شرح تکنیکال

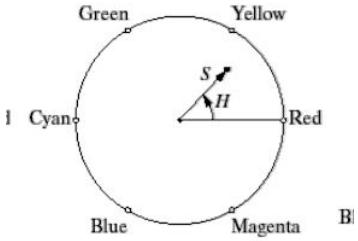
۱-۲- سوال ۵.۱.۱

در این بخش به فضای رنگی HSI پرداخته شده است. این فضا نزدیک‌ترین تناظر را با درک و تفسیر رنگ توسط انسان دارد.

در این فضا هر رنگ با سه مولفه‌ی intensity و saturation و hue شناخته می‌شود.

- Intensity: این مولفه نشان‌دهنده‌ی شدت نور می‌باشد.
- Hue (فام): نشان‌دهنده‌ی طول موج رنگ غالب که توسط بیننده دریافت می‌شود، می‌باشد.(زاویه‌ی رنگ با رنگ قرمز در دایره‌ی رنگ)

- Saturation: این مولفه میزان خلوص رنگ را مشخص ی‌کند(میزان ترکیب نور سفید با فام یک رنگ)، هرچه رنگ خالص تر باشد میزان رنگ سفید در آن کمتر می‌باشد.(فاصله از مرکز در دایره‌ی رنگ)



شکل۱- دایره‌ی رنگ

برای تبدیل رنگ از فضای RGB به فضای HSI می‌توان از روابط زیر استفاده کرد.

- $I = \frac{1}{3}(R + G + B)$
- $S = 1 - \frac{3}{(R+G+B)}\,[min(R, G, B)]$
- $\theta = cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{\left[(R-G)^2+(R-B)(G-B)\right]^{\frac{1}{2}}}\right)$

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

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

مقادیر به دست آمده برای S و θ ممکن است NaN شود.(به دلیل امکان تقسیم بر صفر) برای جلوگیری از این مسئله، هنگام محاسبه مقدار عبارت مخرج با عدد ۰.۰۰۰۰۱ جمع شده است.

۲-۲- سوال ۵.۱.۲

در این بخش سه فضای رنگی HCL و YIQ و YUV بررسی شده اند.

۱-۲-۲- فضای رنگی HCL

این فضای رنگی بر اساس عملکرد ادراک انسان است. در این فضا هر رنگ با سه مولفه‌ی hue و chroma و luminance شناخته می‌شود.

- Luminance: مقدار نوری که گیرنده از منبع نور دریافت می‌کند.
- Chroma: میزان رنگی‌بودن را مشخص می‌کند.

با استفاده از فضای رنگی HCL می‌توان به طور مستقیم hue و chroma و luminance را کنترل کرد.

برای تبدیل رنگ از فضای RGB به فضای HCL می‌توان از روابط زیر استفاده کرد.(H در بخش ۲-۱ محاسبه شده است)

$$C = Q \times \frac{|R-G|+|B-R|+|G-R|}{3}$$

$$L = \frac{Q \times max(R,G,B) + (1-Q) \times min(R,G,B)}{2}$$

$$\alpha = \frac{1}{100} \times \frac{min(R, G, B)}{man(R, G, B)}$$

$\gamma = 3$

$Q = e^{\alpha\gamma}$

۲-۲-۲- فضای رنگی YUV

سیستم رمزگذاری رنگی که برای تلویزیون های آنالوگ در سراسر جهان استفاده می شود (، NTSC PAL و SECAM). فضای رنگی YUV (مدل رنگی) با RGB متفاوت است ، و همان چیزی است که دوربین ضبط می کند و انسانها مشاهده می کنند.

در این فضای رنگی Y بیانگر درخشندگی یا میزان روشنایی رنگ است(luminance) و مولفه‌های U و Vرنگ را مشخص می‌کنند. محدوده Y از 0 تا 1 (یا 0 تا 255 در قالب های دیجیتال) است ، در حالی که U و V از -0.5 تا 0.5 (یا -128 تا 127 در فرم دیجیتال امضا شده ، یا 0 تا 255 به صورت بدون امضا) است. برخی از استانداردها محدوده ها را بیشتر محدود می کنند بنابراین مقادیر خارج از مرزها اطلاعات خاصی مانند همگام سازی را نشان می دهند.

یک جنبه جالب از YUV این است که می توان اجزای U و V را جدا کرد و یک تصویر در مقیاس خاکستری بدست آورد. بدون اینکه کیفیت تصویر تخریب شود. از آنجا که چشم انسان بیش از آنکه به رنگش واکنش نشان دهد ، به روشنایی واکنش نشان می دهد ، بسیاری از فرمت های فشرده سازی تصویر ضعیف ، نیمی یا بیشتر از نمونه های موجود در کانال های رنگی را دور می ریزند تا میزان داده های مورد نظر را کاهش دهند.

برای تبدیل رنگ از فضای رنگی RGB به YUV می‌توان از روابط زیر استفاده کرد.

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.168R - 0.331G + 0.5B + 128$$
$$V = 0.5R - 0.418G - 0.813B + 128$$

۲-۲-۳- فضای رنگی YIQ

YIQ فضای رنگی است که توسط سیستم تلویزیون رنگی NTSC استفاده می شود و عمدتا در آمریکای شمالی و مرکزی و ژاپن استفاده می شود.

در این فضای رنگی Y بیانگر اطلاعات مربوط به luminance و مولفه‌های I و Q بیانگر اطلاعات مربوط به chrominance می‌باشند.

در این سیستم از ویژگی‌های واکنش چشم انسان به رنگ استفاده می‌شود.

برای تبدیل رنگ از فضای RGB به فضای YIQ می‌توان از روابط زیر استفاده کرد.

$$Y = 0.299R + 0.587G + 0.114B$$
$$I = 0.5959R - 0.2746G - 0.3213B$$
$$Q = 0.2115R - 0.5227G + 0.3112B$$

۳-۲- سوال ۵.۲.۱

در این بخش به quantization یکنواخت پرداخته شده است.

یکی از راه‌ها برای کاهش تعداد رنگ‌ها اعمال quantization می‌باشد. اعمال quantization در فضای رنگی شامل کاهش تعداد سطوح رنگ در هر یک از کانال های R و G و B می‌باشد. در ابتدا هر کانال رنگی در بازه‌ی ۰ تا ۲۵۶ می‌باشند.

با استفاده از رابطه ی زیر این بازه به ۰ تا level مورد نظر کاهش یافته است.

$$c = 256 / level$$

$$result = floor(\frac{pixel\ value}{c}) \times c$$

در این بخش تعداد سطوح هر یک از کانال‌های R و G و B پس از quantization برابر می‌باشند. نتایج برای تعداد سطوح ۸ و ۱۶ و ۳۲ و ۶۴ محاسبه و بررسی شده است.

۴-۲- سوال ۵.۲.۲

در این بخش مانند بخش قبلی عمل quantization بر تصویر pepper اعمال شده با این تفاوت که در این قسمت تعداد سطوح کانال‌های R و G و B برابر نمی‌باشد. برای کانال قرمز و سبز ۳ بیت (۸ سطح) و برای رنگ آبی ۲ بیت (۴ سطح) در نظر گرفته شده است.

۵-۲- سوال ۵.۲.۳

در این بخش به کاهش تعداد رنگ‌های به کار رفته در تصویر پرداخته شده است.

۱-۵-۲- کاهش تعداد سطوح هر کانال رنگی

یکی از روش‌های کاهش تعداد رنگ‌های به کار رفته در تصویر کاهش تعداد سطوح رنگ‌های هر کانال رنگی می‌باشد.(مانند بخش قبل) برای کاهش تعداد رنگ‌ها به ۸، ۱۶ و ۳۲ رنگ حالت های مختلف کاهش تعداد سطوح رنگ‌های هر یک از کانال‌های رنگی در نظر گرفته شده است که در بخش نتایج قابل مشاهده می‌باشد.

۲-۵-۲- quantization با استفاده از الگوریتم kmeans

الگوریتم kmeans یک الگوریتم بدون ناظر یادگیری ماشین است. هدف این الگوریتم آن است که داده‌های ورودی را به k خوشه، خوشه‌بندی کند به طوری که داده‌های هر خوشه بیشترین شباهت را نسبت به هم داشته باشند و داده‌هایی در خوشه‌های متفاوت با هم تفاوت قابل توجهی داشته باشند. یکی از ساده ترین راه‌های محاسبه‌ی شباهت داده‌ها محاسبه‌ی فاصله‌ی اقلیدسی آنها می‌باشد.

یکی از روش‌های مقایسه دو رنگ نیز می‌تواند فاصله‌ی اقلیدسی آنها باشد که از طریق رابطه‌ی زیر قابل محاسبه می‌باشد.

$$\sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$

بنابراین با اعمال الگوریتم kmeans به پیکسل‌های تصویر(دیتای ورودی) می‌توان تعداد رنگ‌های تصویر را به k رنگ کاهش داد، بدین صورت که وقتی الگوریتم kmeans پیکسل‌ها را بر اساس مولفه‌های رنگی آنها خوشه‌بندی کرد، رنگ مرکز هر خوشه را به جای رنگ اعضای آن خوشه قرار می‌دهیم.

برای پیاده سازی این روش از الگوریتم kmeans کتابخانه‌ی opencv استفاده شده است.

۳- شرح نتایج

۱-۳- سوال ۵.۱.۱

hue و saturation و intensity تصویر pepper در ادامه قابل مشاهده است.

شکل۲- تصویر pepper

شکل۳- تصویر pepper بخش hue

مشاهده می‌شود که قسمت‌هایی از تصویر که به رنگ قرمز می‌باشد، فام آنها سفید و یا سیاه است. دلیل آن امر این است که در فضای HSI در محل تلاقی ۰ و ۳۶۰ درجه در H، ناپیوستگی وجود دارد و ۰ و

۳۶۰ از نظر بصری تقریبا مشابه یکدیگر هستند. و مقدار فام رنگ قرمز می‌تواند نزدیک به ۰ و یا نزدیک به ۳۶۰ درجه باشد. این مسئله از مشکلات فضای رنگی HSI می‌باشد.

همچنین مشاهده می‌شود که قسمت‌های سبز رنگ در تصویر اصلی، فام آنها مقداری میانی دارد و خاکستری می‌باشد. همانطور که در شکل یک مشاهده می‌شود رنگ سبز با رنگ قرمز زاویه‌ی ۱۲۰ درجه دارد. به همین دلیل در تصویر مربوط به فام آن بخش‌های سبز رنگ به رنگ خاکستری تیره می‌باشند.

شکل۴- تصویر pepper بخش saturation

مشاهده می‌شود که بخش‌هایی از تصویر اصلی که روشن تر هستند و میزان رنگ سفید ترکیب شده با آنها بیشتر است(خلوص کمتری دارند) مقدار saturation آنها نیز کمتر بوده و در شکل۴ رنگ تیره تری دارند.

شکل۵- تصویر pepper بخش intensity

۲-۳- سوال ۵.۲.۱

نتایج حاصل از اعمال uniform quantization بر تصویر pepper در ادامه قابل مشاهده می‌باشد.

شکل۶- تصویر حاصل از uniform quantization با ۶۴ سطح

شکل۸- تصویر حاصل از uniform quantization با ۱۶ سطح

شکل۷- تصویر حاصل از uniform quantization با ۳۲ سطح

شکل۹- تصویر حاصل از uniform quantization با ۸ سطح

همانطور که مشاهده می‌شود کاهش تعداد سطوح هر کانال رنگی سبب کاهش کیفیت از نظر بصری می‌شود. این کاهش کیفیت در تصویر کوانتیزه شده به ۸ سطح(برای هر کانال رنگی) بیشتر قابل مشاهده می‌باشد.

برای مقایسه‌ی تصاویر حاصل از معیاری به نام خطای میانگین مربعات(MSE) و همچنین معیار PSNR استفاده شده است.

معیار PSNR نسبت اوج سیگنال به نیز را در تصویر محاسبه می‌کند. این نسبت به عنوان اندازه گیری کیفیت بین تصویر اصلی و تصویر تغییر یافته استفاده می‌شود. هرچه این مقدار بیشتر باشد، کیفیت بیشتر است.

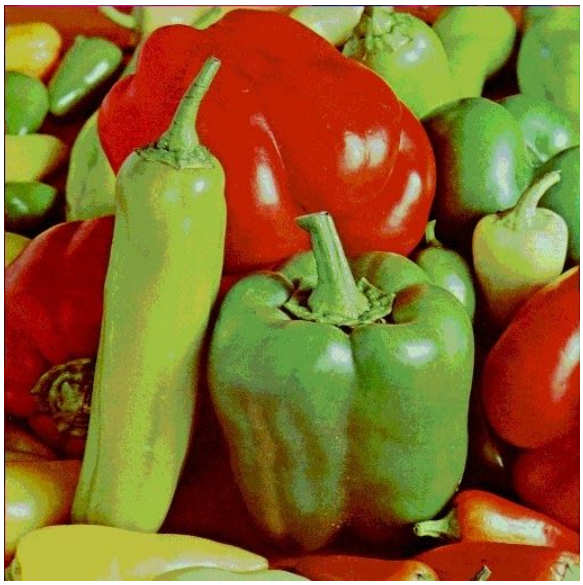
نتایج این مقایسه در جدول زیر قابل مشاهده است.

	64	32	16	8
MSE	3.35	16.67	73.30	88.96
PSNR	42.87	35.91	29.47	28.63

همانطور که از بررسی تصاویر به صورت بصری نتیجه گرفته شد، با کاهش تعداد سطوح رنگ در کانال‌های رنگی(R و G و B) کیفیت تصویر کاهش می‌یابد(MSE افزایش و NSPR کاهش می‌یابد)، این کاهش کیفیت در حالتی که تعداد سطوح رنگی ۸ و ۱۶ می‌باشد، بیشتر محسوس است.

۳-۳- سوال ۵.۲.۲

در ادامه تصویر حاصل از quantization بر تصویر pepper با تعداد سطوح ۸(۳بیت) و ۸ و ۴(۲بیت) به ترتیب برای کانال‌های قرمز و سبز و آبی نمایش داده شده است.



شکل۱۰- تصویر حاصل از quantization با در نظر گرفتن ۳ بیت برای کانال‌های قرمز و سبز و ۲ بیت برای کانال آبی

با کاهش تعداد بیت‌های هر کانال، از مقدار ۸ به ۲ یا ۳ کیفیت از نظر بصری به طور کلی کاهش می‌یابد. در اینجا به کانال رنگی قرمز و سبز ۳ بیت

و به آبی ۲ بیت اختصاص داده شده است که باعث می‌شود تاثیر رنگ آبی در تصویر کمتر از قرمز و سبز باشد. مقدار psnr و mse برای این تصویر به ترتیب برابر ۲۸.۳۶ و ۹۴.۷ می‌باشد. این مقادیر در حالتی که به هر کانال رنگی ۳ بیت اختصاص دهیم به ترتیب ۲۸.۶۳ و ۸۸.۹۶ می‌باشند(با توجه به بخش۲-۳). مشاهده می‌شود که این مقادیر تغییر زیادی نکرده اند و همچنین کیفیت این تصویر از نظر بصری نسبت به شکل۹ به طور محسوس کاهش نیافته است. دلیل این امر آن است که این تصویر به طور کلی بیشتر شامل رنگ‌های قرمز و سبز است و سهم رنگ آبی در این تصویر کم می‌باشد بنابراین کمتر بودن تعداد بیت‌های رنگ آبی نسبت به وضعیتی که به هر کانال رنگی ۳ بیت اختصاص یابد، تغییر زیادی ایجاد نکرده است.

۳-۴- سوال ۵.۲.۳

در این بخش در ابتدا نتایج مربوط کاهش تعداد سطوح هر کانال رنگی نمایش داده شده و بررسی شده است. سپس به بررسی نتایج الگوریتم kmeans پرداخته شده است.

شکل۱۱- تصویر girl

در ادامه تصویر girl با ۸ رنگ قابل مشاهده است.

شکل۱۲- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای هر کانال رنگی

در ادامه حالات مختلفی از تصویر girl با ۱۶ رنگ قابل مشاهده است.

شکل۱۳- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای قرمز و سبز و ۴ سطح(۲ بیت) برای آبی

شکل۱۴- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای قرمز و آبی و ۴ سطح(۲ بیت) برای سبز

شکل۱۵- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای آبی و سبز و ۴ سطح(۲ بیت) برای قرمز

در بین تصاویر شکل۱۳ و ۱۴ و ۱۵، تصویر ۱۴ و ۱۵ کیفیت تصویر را از نظر بصری بهتر حفظ کرده اند. مثلا جزئیات روی دیوار را بیشتر نشان داده اند. با این حال هر سه تصویر کیفیت مناسبی ندارند و رنگ‌ها از نظر بصری طبیعی نیستند.

در ادامه حالات مختلفی از تصویر girl با ۳۲ رنگ قابل مشاهده است.

شکل۱۶- تصویر حاصل از quantization با در نظر گرفتن ۴ سطح رنگی(۲ بیت) برای آبی و سبز و ۲ سطح(۱ بیت) برای قرمز

شکل۱۷- تصویر حاصل از quantization با در نظر گرفتن ۴ سطح رنگی(۲ بیت) برای آبی و قرمز و ۲ سطح(۱ بیت) برای سبز

شکل۱۸- تصویر حاصل از quantization با در نظر گرفتن ۴ سطح رنگی(۲ بیت) برای قرمز و سبز و ۲ سطح(۱ بیت) برای آبی

مشاهده می‌شود تصویری که در آن ۲ بیت برای سبز و قرمز و ۱ بیت برای آبی در نظر گرفته شده است(شکل ۱۸)، جزئیات بیشتری از نکر بصری حفظ کرده و کیفیت بهتری نسبت به شکل‌های ۱۶ و ۱۷ دارد.

شکل۱۹- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای قرمز و سبز و ۸ سطح(۳ بیت) برای آبی

شکل۲۱- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای آبی و سبز و ۸ سطح(۳ بیت) برای قرمز

شکل۲۰- تصویر حاصل از quantization با در نظر گرفتن ۲ سطح رنگی(۱ بیت) برای قرمز و آبی و ۸ سطح(۳ بیت) برای سبز

همانطور که مشاهده شد، کاهش تعداد رنگ‌ها با استفاده از روش کاهش تعداد بیت‌های هر رنگ کیفیت را از نظر بصری کاهش می‌دهد. به همین دلیل از روش‌های دیگری مانند الگوریتم kmeans استفاده می‌شود. در ادامه نتایج این روش بررسی شده است.

شکل۲۲-تصویر ۸ رنگ حاصل از الگوریتم kmeans

شکل۲۴-تصویر ۳۲ رنگ حاصل از الگوریتم kmeans

شکل۲۳-تصویر ۱۶ رنگ حاصل از الگوریتم kmeans

همانطور که مشاهده می‌شود، تصاویر بدست آمده از الگوریتم kmeans تقریبا رنگ و کیفیت و جزئیات تصویر اصلی را حفظ کرده اند. و حتی شکل ۲۴ که ۳۲ رنگ دارد، شباهت بسیار زیادی به تصویر اصلی دارد.

منابع

[1] Digital Image Processing, Rafael C. Gonzalez, Rechard E. Wood

[2] http://hclwizard.org/why-hcl/

[3] https://softpixel.com/~cwright/programming/colorspace/yuv/

[4] https://www.pcmag.com/encyclopedia/term/yuv

[5] https://en.wikipedia.org/wiki/YIQ

[6] http://datahacker.rs/007-color-quantization-using-k-means-clustering/

[7] https://docs.opencv.org/master/d1/d5c/tutorial_py_kmeans_opencv.html

Appendix

```
import cv2
import numpy as np
import math
from sklearn.cluster import KMeans


def quantize(img, level):
    coef = 256 / level
    quantized = np.floor(np.true_divide(img, coef)) * coef
    quantized = quantized.astype("uint8")
    return quantized

def rgb_quantize(img, rlevel, glevel, blevel):
    blue = img[:,:, 0]
    green = img[:,:, 1]
    red = img[:,:, 2]

    q_red = quantize(red, rlevel)
    q_blue = quantize(blue, blevel)
    q_green = quantize(green, glevel)
    quantized_image = np.zeros_like(img)
    quantized_image[:,:,0] = q_blue
    quantized_image[:,:,1] = q_green
    quantized_image[:,:,2] = q_red
    return quantized_image

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

```
def normal(img):
   min_val = np.amin(img)
   max_val = np.amax(img)
   output = np.zeros_like(img)

   for i in range(np.size(img, 0)):
       for j in range(np.size(img, 1)):
           new_val = math.floor(((img[i, j] - min_val)*255) / (max_val -
min_val))
           output[i, j] = new_val
   return output
#-----------------------------------------------------------------
def get_intensity(img):
   img = np.float32(img)/255
   img_blue = img[:,:, 0]
   img_green = img[:,:, 1]
   img_red = img[:,:, 2]
   intensity = (img_blue + img_green + img_red) / 3
   intensity = intensity * 255
   return(normal(intensity))
#-----------------------------------------------------------------
def get_saturation(img):
   img = np.float32(img)/255
   img_blue = img[:,:, 0]
   img_green = img[:,:, 1]
   img_red = img[:,:, 2]

   saturation = 1 - (3 / (img_red + img_green + img_blue + 0.001) *
np.minimum(np.minimum(img_red, img_green), img_blue))
   return normal(saturation)
#-----------------------------------------------------------------
def get_hue(img):
   img = np.float32(img)/255
   hue = np.zeros((img.shape[0], img.shape[1]))
   for i in range(img.shape[0]):
       for j in range(img.shape[1]):
           b = img[i, j, 0]
           g = img[i, j, 1]
           r = img[i, j, 2]


           teta = np.arccos((0.5*(2*r-b-g)) / ((((r-g)**2) + (r-b)*(g-b))**0.5)
+ 0.000001)
           teta = np.degrees(teta)
```

```
            if(b <= g):
                hue[i, j] = teta
            else:
                hue[i,j] = 360 - teta

            if(math.isnan(hue[i, j])):
                hue[i, j] = 0
    return normal(hue)
#-----------------------------------------------------------------
def color_quantization(img, k):
# Defining input data for clustering
 data = np.float32(img).reshape((-1, 3))
# Defining criteria
 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
# Applying cv2.kmeans function
 ret, label, center = cv2.kmeans(data, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)
 center = np.uint8(center)
 result = center[label.flatten()]
 result = result.reshape(img.shape)
 return result

pepper = cv2.imread("image/Pepper.bmp")
cv2.imwrite("output/pepper.jpg", pepper)
girl = cv2.imread("image/Girl.bmp")
cv2.imwrite("output/girl.jpg", girl)

#5.2.1
quantized = []
levels = [64, 32, 16, 8]
for i in range(len(levels)):
    level = levels[i]
    quantized.append(rgb_quantize(pepper, level, level, level))
    cv2.imwrite("output/5_2_1/pepper" + str(level) + ".jpg", quantized[i])

for i in range(len(quantized)):
    mse  = np.mean( (pepper - quantized[i]) ** 2 )
    psnr_val = psnr(pepper, quantized[i])
    print(str(levels[i]) + "mse: " + str(mse))
    print(str(levels[i]) + "psnr: " + str(psnr_val))

#5.2.2
quantized_522 = rgb_quantize(pepper, 8, 8, 4)
cv2.imwrite("output/5_2_2/quantized.jpg", quantized_522)
```

```
mse  = np.mean( (pepper - quantized_522) ** 2 )
psnr_val = psnr(pepper, quantized_522)
print("522 " + "mse: " + str(mse))
print("522 " + "psnr: " + str(psnr_val))
"""
#5.1.1
img = pepper

img_intensity = get_intensity(pepper)
cv2.imwrite("output/5_1_1/intensity.jpg", img_intensity)

img_saturation = get_saturation(pepper)
cv2.imwrite("output/5_1_1/saturation.jpg", img_saturation)

img_hue = get_hue(pepper)
cv2.imwrite("output/5_11/hue.jpg", img_hue)
"""
#5.2.3
quantized_8 = rgb_quantize(girl, 2, 2, 2)
cv2.imwrite("output/5_2_3/222.jpg", quantized_8)

levels_16 = [(2, 2, 4), (2, 4, 2), (4, 2, 2)]
quantized_16 = []
for i in range(len(levels_16)):
   l = levels_16[i]
   q = rgb_quantize(girl, l[0], l[1], l[2])
   quantized_16.append(q)
   cv2.imwrite("output/5_2_3/" + str(l[0]) + str(l[1]) + str(l[2]) + ".jpg", q)

levels_32 = [(2, 4, 4), (4, 2, 4), (4, 4, 2), (8, 2, 2), (2, 8, 2), (2, 2, 8)]
quantized_32 = []
for i in range(len(levels_32)):
   l = levels_32[i]
   q = rgb_quantize(girl, l[0], l[1], l[2])
   quantized_32.append(q)
   cv2.imwrite("output/5_2_3/" + str(l[0]) + str(l[1]) + str(l[2]) + ".jpg", q)


mse8  = np.mean( (pepper - quantized_8) ** 2 )
psnr_val8 = psnr(pepper, quantized_8)
print("(2, 2, 2)" + "mse: " + str(mse8))
print("(2, 2, 2)" + "psnr: " + str(psnr_val8))

for i in range(len(quantized_16)):
   mse  = np.mean( (pepper - quantized_16[i]) ** 2 )
```

```
    psnr_val = psnr(pepper, quantized_16[i])
    print(str(levels_16[i]) + "mse: " + str(mse))
    print(str(levels_16[i]) + "psnr: " + str(psnr_val))

for i in range(len(quantized_32)):
    mse  = np.mean( (pepper - quantized_32[i]) ** 2 )
    psnr_val = psnr(pepper, quantized_32[i])
    print(str(levels_32[i]) + "mse: " + str(mse))
    print(str(levels_32[i]) + "psnr: " + str(psnr_val))
# 5.2.3 second solution(kmeans)
colorNum = [3, 8, 16, 32]
kmeans_quantized = []
for i in range (len(colorNum)):
    q = color_quantization(girl, colorNum[i])
    kmeans_quantized.append(q)
    cv2.imwrite("output/5_2_3/kmeans" + str(colorNum[i]) + ".jpg", q)

for i in range(len(colorNum)):
    mse  = np.mean( (pepper - kmeans_quantized[i]) ** 2 )
    psnr_val = psnr(pepper, kmeans_quantized[i])
    print("kmeans" + str(colorNum[i]) + "  mse: " + str(mse))
    print("kmeans" + str(colorNum[i]) + "  psnr: " + str(psnr_val))
```