گزارش تمرین Color

مریم واقعی

اطلاعات گزارش

تاریخ:
۱۳ بهمن ۱۴۰۱

واژگان کلیدی:
فضای رنگی
RGB
HIS
مکعب رنگ
کوانتزاسیون یکنواخت
کوانتیزاسیون انطباقی
کاهش تعداد رنگ
خوشه بندی
k-means

چکیده

در فضای رنگی، بینایی انسان قدرت تفکیک پذیری بیشتری دارد. بنا به کاربردهای مختلف فضاهای رنگی متفاوتی به وجود آمده اند. فضای رنگی RGB در چشم انسان، و فضای رنگی HSI در مغز انسان، دو نمونه از شاخص ترین فضاهای رنگی هستند. می‌توان با استفاده از روابط ریاضی، یک رنگ را در فضای رنگی به فضای رنگی دیگری منتقل نمود. از آنجا که در ذخیره سازی تصاویر دیجیتال با محودیت هایی مواجه هستیم، با چندی سازی مقادیر پیکسل های تصویر و کاهش تعداد رنگ ها، این مسئله را مدیریت می‌کنیم. روش های متفاوتی برای چندی سازی وجود دارد. روش چندی سازی یکنواخت با نسبتی یکسان پیکسل های تصویر را چندی سازی می‌کند. چندی سازی انطباقی براساس رنگ های پرتکرار در تصویر، رنگ های با فراوانی کم را با نزدیک ترین رنگ پرتکرار جایگزین می‌کند. از الگوریتم k-means برای یافتن نزدیکترین رنگ ها استفاده می‌شود.

فهرست مطالب

۱-مقدمه

رنگ را میتوان یک توصیفگر قدرتمند دانست؛ زیرا چشم انسان در فضای رنگی قدرت تفکیک بیشتری دارد. بینایی انسان در تفکیک شدت رنگ هاو تشخیص اشیا و استخراج آن ها از یک صحنه، توانایی بیشتری دارد. در حلی که این توانایی در تفکیک سطوح خاکستری بسیار محدودتر است.

نواحی ای از تصویر که شدت نور یکسانی دارند سطح خاکستری یکسانی تولید می‌کنند در حالی که در فضای رنگی با هم متفاوت بوده و قابل تمایز هستند.

به طور کلی، در صورتی که فضای خاکستری پاسخگوی نیاز های ما باشد به سراغ فضای رنگی نمی‌رویم. چرا که اطلاعات در فضای رنگی برای ذخیره سازی، به حجم بیشتری نیاز دارند.

اکثر الگوریتم های مطرح شده در فضای خاکستری با تغییرات جزئی قابل اعمال به تصویر رنگی هستند. در این گزارش، به بررسی فضاهای رنگی در چشم و مغز انسان و بازنمایی های آن ها می‌پردازیم.

۲- توضیحات فنی

۲-۱- Color space

۲-۱-۱- بخش اول

در این بخش ابتدا هر کدام از کانال های رنگی تصویر lena را به صورت جداگانه نمایش داده ایم. سپس با استفاده از متد rgb_to_hsi فضای رنگی rgb را به hsi تبدیل کردیم. به این صورت که ابتدا بازه ی مقادیر پیکسل ها را از ۰ تا ۲۵۵ به ۰ تا ۱ می بریم، سپس هر کدام از مقادیر فام رنگ، اشباع و شدت روشنایی را به روش زیر بدست می‌آوریم:

فام رنگ: نشان دهنده طول موج رنگ غالب که بیننده دریافت می‌کند. فام رنگ را با متد calculate_hue محاسبه میکنیم. و به این صورت محاسبه میشود که ابتدا زاویه θ را از فرمول زیر بدست می‌آوریم:

فرمول(۱)

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

سپس مقدار فام رنگ را از رابطه زیر محاسبه میکنیم:

فرمول(۲)

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

پس با استفاده از روابط بالا فام رنگ را به ازای تک تک پیکسل ها بر اساس مقادیر رنگ های قرمز، سبز و آبی بدست می‌آوریم.

اشباع: میزان خلوص رنگ یا میزان ترکیب نور سفید با یک فام.

مقدار اشباع را با استفاده از متد calculate_saturation() با استفاده از فرمول زیر بدست می‌آوریم:

فرمول(۳)

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

شدت روشنایی: در نهایت شدت روشنایی را با استفاده از فرمول زیر محاسبه می‌کنیم:

فرمول(۴)

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

در نهایت فام رنگ، اشباع و شدت روشنایی را نمایش میدهیم که در بخش بررسی نتایج میتوانید خروجی را مشاهده کنید.

۲-۱-۲- بخش دوم

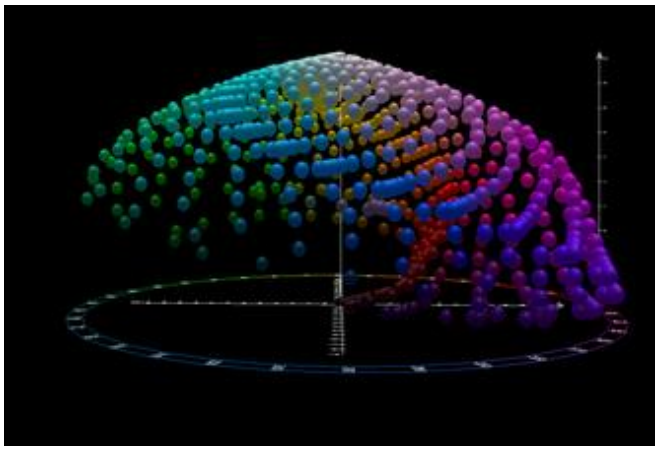
فضای رنگی CIELAB که با نام L*a*b*نیز شناخته می‌شود، یک فضای رنگی است که توسط کمیسیون بین‌المللی روشنایی به اختصار (CIE) در سال ۱۹۷۶ تعریف شد.

این فضای رنگی، رنگ را به صورت سه مقدار بیان می کند: L* برای روشنایی دریافتی و a* و b* برای چهار رنگ منحصر به فرد بینایی انسان: قرمز، سبز، آبی و زرد.

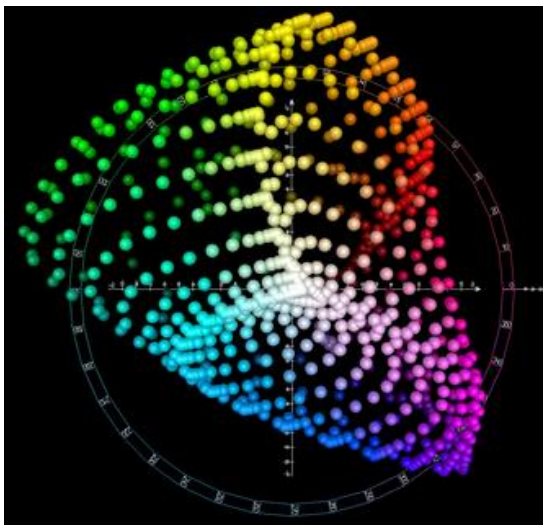
L* = 0 نشان دهنده سیاه و L * = 100 نشان دهنده سفید پراکنده است؛ رنگ سفید ممکن است بالاتر باشد.

درa* مقادیر منفی نشان دهنده سبز و مقادیر مثبت نشان دهنده قرمز است و b* که مقادیر منفی نشان دهنده آبی و مقادیر مثبت نشان دهنده زرد هستند.

از آنجایی که مدل L*a*b* دارای سه محور است، نیاز به یک فضای سه بعدی برای نمایش کامل دارد، همچنین به دلیل غیر خطی بودن هر محور، امکان ایجاد نمودار رنگی دوبعدی وجود ندارد.



تصویر ۱- فضای CIELAB از نمای روبرو



تصویر ۲- فضای CIELAB از نمای بالا

CIELAB به عنوان یک فضای ادراکی یکنواخت در نظر گرفته شده بود، که در آن یک تغییر عددی داده شده مربوط به یک تغییر درک شده مشابه در رنگ است.

برخلاف مدل های رنگی RGB و CMYK، CIELAB برای تقریب بینایی انسان طراحی شده است. مولفه *L کاملاً با درک انسان از روشنایی مطابقت دارد. CIELAB در محورهای رنگی کمتر یکنواخت است، اما برای پیش بینی تفاوت های کوچک در رنگ مفید است.

تبدیل RGB به CIELAB:

برای این کار ابتدا لازم است که از RGB به XYZ تبدیل کنیم.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

که ماتریس M از روابط زیر بدست می‌آید:

$$[M] = \begin{bmatrix} S_r X_r & S_g X_g & S_b X_b \\ S_r Y_r & S_g Y_g & S_b Y_b \\ S_r Z_r & S_g Z_g & S_b Z_b \end{bmatrix}$$

که هر یک از متغیر های بالا از روابط زیر بدست می‌آیند (اندیس i، تمام اندیس های r و g و b را در بر میگیرد):

$$X_i = {}^{x_i}/_{y_i}$$
$$Y_i = 1$$
$$Z_i = {}^{(1 - x_i - y_i)}/_{y_i}$$

$$\begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix}$$

حال با فرمول های زیر از XYZ به LAB تبدیل میکنیم:

برای این تبدیل ما نیاز به رنگ سفید مرجع داریم $(X_r . Y_r . Z_r)$

$$L = 116 f_y - 16$$
$$a = 500(f_x - f_y)$$
$$b = 200(f_y -)$$

که در آن f_x و f_y و f_z برابر است با:

$$f_x = \begin{cases} \sqrt[3]{x_r} & if\ x_r > \epsilon \\ \dfrac{k x_r + 16}{116} & otherwise \end{cases}$$

$$f_y = \begin{cases} \sqrt[3]{y_r} & if\ y_r > \epsilon \\ \dfrac{k y_r + 16}{116} & otherwise \end{cases}$$

$$f_z = \begin{cases} \sqrt[3]{z_r} & if\ z_r > \epsilon \\ \dfrac{k z_r + 16}{116} & otherwise \end{cases}$$

$$x_r = \frac{X}{X_r}$$
$$y_r = \frac{Y}{Y_r}$$
$$z_r = \frac{Z}{Z_r}$$
$$\epsilon = \begin{cases} 0.008856 & actual\ CIE\ standard \\ 216/24389 & intent\ of\ the\ CIE\ standard \end{cases}$$
$$k = \begin{cases} 903.3 & actual\ CIE\ standard \\ 24389/27 & intent\ of\ the\ CIE\ standard \end{cases}$$

فضای مختصات CIELAB کل وسعت دید فتوپیک انسان (نور روز) را نشان می دهد و بسیار بیشتر از وسعت sRGB یا CMYK است. در پیاده سازی عدد صحیح مانند TIFF، ICC یا Photoshop، فضای مختصات بزرگ به دلیل استفاده از مقادیر کد استفاده نشده منجر به ناکارآمدی قابل توجه داده می شود. فقط حدود ۳۵ درصد از مقادیر کد مختصات موجود در وسعت CIELAB با فرمت عدد صحیح است.

استفاده از CIELAB در قالب عدد صحیح ۸ بیتی در هر کانال معمولاً منجر به خطاهای کوانتیزاسیون قابل توجهی می شود. حتی ۱۶ بیت در هر کانال می تواند منجر به برش شود، زیرا وسعت کامل از فضای مختصات مرزی امتداد می یابد. در حالت ایده آل، CIELAB باید با داده های ممیز شناور برای به حداقل رساندن خطاهای کوانتیزاسیون آشکار استفاده شود.

کاربرد های فضای رنگی CIELAB:

برخی از سیستم ها و نرم افزارهای کاربردی که از CIELAB پشتیبانی می کنند عبارتند از:

CIELAB توسط کتابخانه PantoneLive استفاده می شود.

CIELAB به طور گسترده توسط XRite به عنوان فضای رنگی با سیستم های اندازه گیری رنگ سخت افزاری و نرم افزاری آنها استفاده می شود.

CIELAB D50 در Adobe Photoshop موجود است، جایی که به آن Lab mode گفته می شود.

CIELAB با تغییر فرمت رنگ سند به "Lab(16bit)" در Affinity Photo در دسترس است. نقطه سفید، که به طور پیش فرض روی D50 است، می تواند توسط نمایه ICC تغییر یابد.

CIELAB توسط GIMP برای فیلتر تنظیم رنگ، انتخاب فازی و سطل رنگ استفاده می شود. یک انتخابگر رنگ LCh(ab) نیز وجود دارد.

CIELAB D50 در پروفایل های ICC به عنوان فضای اتصال پروفایل به نام Lab color space موجود است.

CIELAB یک گزینه در رنگ سنج دیجیتال در macOS است که با عنوان L*a*b* توصیف شده است.

CIELAB در ویرایشگر عکس RawTherapee موجود است.

Quantization -۲-۲

۲-۲-۱- بخش اول

یک چندی ساز به طور کلی مقادیر دامنه را به مجموعه ای از مقادیر گسسته kQ نگاشت میکند که Q فاصله چندی سازی[1]یا اندازه گام هاٌست.

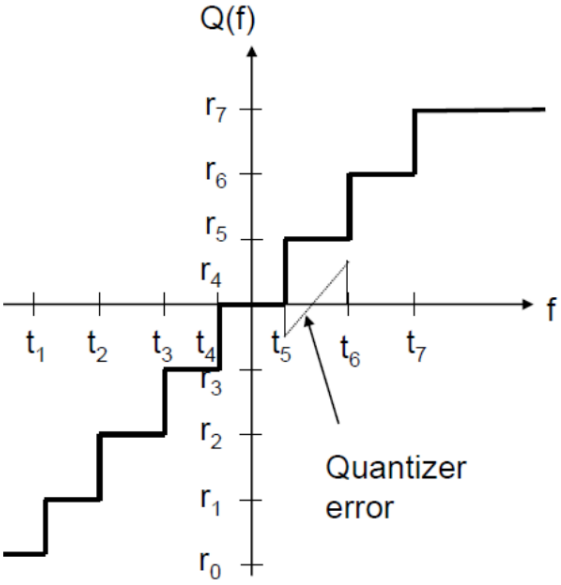
مقدار Q وابسته به محدوده پویا[3](D) دامنه سیگنال و حساسیت ادراکی است. با توجه به رابطه

فرمول (۵)

$$2^R = \frac{D}{Q}$$

که R تعداد بیت ذخیره به ازای هر نمونه است. برای صوت و تصویر، R معمولا برابر ۸ بیت و برای موسیقی ۱۶ بیت است.

شکل زیر عملکرد یک چندی ساز را نشان می‌دهد:



تصویر ۳- نحوه عملکرد یک چندی ساز

در تصویر ۳، t_k ها (k=0,...,L) سطوح تصمیم گیری و r_k ها (k=0, ..., L-1) سطوح بازسازی تصویر هستند. با توجه به تصویر ۳:

$$if\ f\ \in [t_k . t_{k+1}) . then\ Q(f) = r_k$$

L سطح به $R = \lceil log_2 L \rceil$ بیت نیاز دارند.

چندی سازی یکنواخت:

چنانچه فاصله مساوی بین سطوح تصمیم گیری مجاور و بین سطوح بازسازی مجاور باشد، چندی‌سازی را یکنواخت می‌نامند.

$$t_i - t_{i-1} = r_i - r_{i-1} = q$$

پارامتر های تابع چندی‌سازی یکنواخت عبارتند از:

۱. L: سطوح $(L = 2^R)$

۲. D: محدوده پویا $(D = f_{max} - f_{min})$

۳. q: فاصله چندی سازی $(q = \frac{D}{L})$

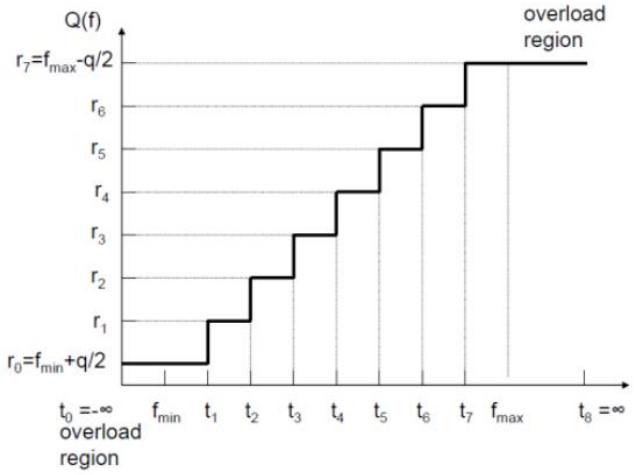
---

[1] Quantizer
[2] Quantization Interval
[3] Dynamic range

در نهایت رابطه چندی‌سازی یکنواخت، به صورت زیر است:

فرمول (۶)

$$Q(f) = \left\lfloor \frac{f - f_{min}}{q} \right\rfloor * q + \frac{q}{2} + f_{min}$$



تصویر ۴- بازنمایی تابعی یک چندی ساز یکنواخت

به طور معمول تصاویر سطح خاکستری از ۸ بیت برای ذخیره سازی اطلاعات هر پیکسل استفاده می‌کنند. بنابراین $L = 2^8 = 256$. یعنی ۲۵۶ سطح خاکستری در هر پیکسل وجود دارد. کاهش یا افزایش تعداد این بیت ها، منجر به کاهش یا افزایش تعداد سطوح خاکستری و به تبع آن تغییر کیفیت تصویر می‌شود.

برای چندی سازی یکنواخت یک تصویر رنگی در قالب RGB، ابتدا مولفه های R، G و B تصویر را جدا کرده و به صورت جداگانه، چندی‌ساز یکنواخت را که براساس توضیحات داده شده در متد uniform_quantize پیاده کردیم روی هر یک از مولفه ها اعمال میکنیم. خروجی کد و گزارش خطای MSE و PSNR در بخش بررسی نتایج آورده شده است.

۲-۲-۲- بخش دوم

تعداد بیت ها برای هر یک از مولفه های R، G و B به ترتیب ۳، ۳ و ۲ بیت است. بنابراین، مطابق با توضیحات داده شده در بخش قبلی، L برای این مولفه ها به ترتیب ۸، ۸ و ۴ خواهد بود. مولفه های R، G و B را جدا نموده و سپس چندی سازی یکنواخت را بر هر یک از مولفه ها اعمال می‌نماییم.

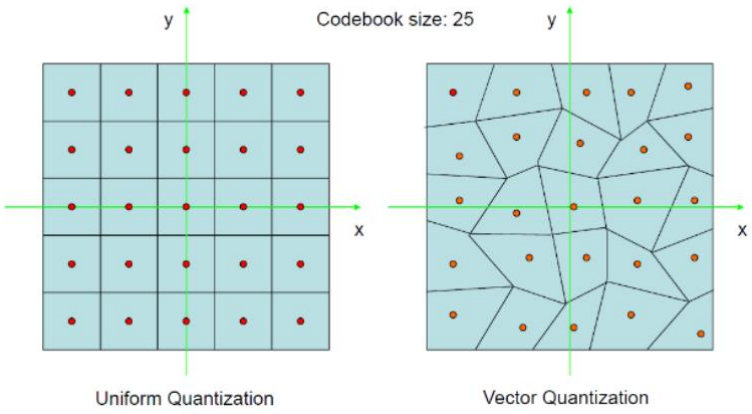
۲-۲-۳- بخش سوم

چندی سازی یکنواخت (اسکالر) هر مولفه رنگی را به طور یکنواخت چندی سازی می‌کند. برای مثال چنانچه برای نمایش مولفه های رنگی به جای ۲۴ بیت، از ۸ بیت استفاده کنیم، می‌توان ۳ بیت برای مولفه R، ۳ بیت برای مولفه G و ۲ بیت برای مولفه B در نظر گرفت. با این حال، این روال نتیجه چندان مطلوبی نخواهد داشت.

بنابراین ایده چندی سازی انطباقی[1](چندی سازی برداری[2]) می‌تواند به عنوان یک راه حل برای این چالش استفاده شد. ایده کلی به این صورت است که با هر رنگ (در اصل یک بردار ۳ گانه) به عنوان یک موجودیت مستقل برخورد میکنیم. N رنگ (بردار) که در تصویر بیشتر از همه ظاهر شده اند را پیدا میکنیم و آنها را در یک پالت رنگ[3]ذخیره می‌نماییم. سپس رنگ را در هر

[1] Adaptive quantization
[2] Vector quantization
[3] Color palette/ codebook

پیکسل با نزدیکترین رنگ در پالت رنگ، جایگزین میکنیم. با توجه به اینکه پالت رنگ از روی محتوای تصویر بدست می‌آید، بنابراین برای هر تصویر متفاوت است.



تصویر ۵- مقایسه نحوه عملکرد چندی سازی برداری و چندی سازی یکنواخت

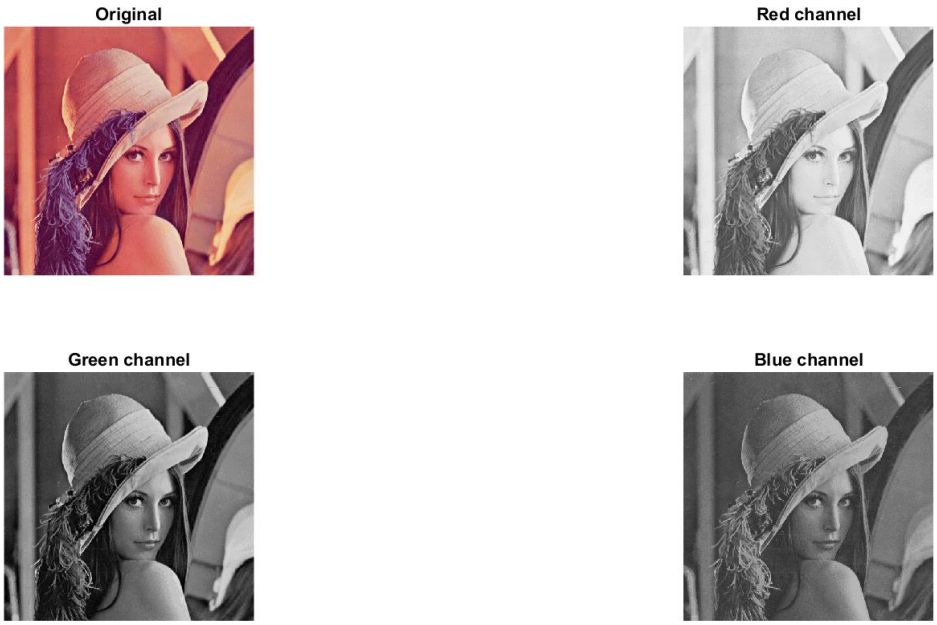
یکی از راه حل های دیگر برای این کار، استفاده از الگوریتم های خوشه بندی است. خوشه بندی به بیان ساده، گروه بندی مجموعه ای از اشیا انجام می‌شود. این کار به این صورت است که اشیا در یک خوشه در مقایسه با دیگر خوشه ها، مشابه تر هستند. این روش، در بسیاری از زمینه ها از جمله یادگیری ماشین، تشخیص الگو؛ تجزیه و تحلیل تصویر و... استفاده می‌شود. روش مورد استفاده در حل این سوال، استفاده از الگوریتم خوشه بندی k-means است.

ما میتوانیم ابتدا طول و عرض تصویر را ادغام کنیم و تصاویر را از ۳ بعد (ارتفاع، عرض و کانال) به ۲ بعد (تعداد پیکسل ها و کانال) تغییر دهیم. سپس الگوریتم k-means را روی داده های جدید با تعداد ۸ خوشه آموزش می دهیم. پس از آنکه الگوریتم توانست پیکسل های تصویر را در ۳۲/۱۶/۸ خوشه قرار دهد، ما مراکز این خوشه ها که هر مرکز دارای ۳ مقدار (برای هر کانال رنگ) است را به همراه برچسب پیکسل ها دریافت میکنیم. برچسب پیکسل ها نشان دهنده این است که هر پیکسل مربوط به کدام خوشه است. پس از این برای هر داده، مرکز خوشه مربوط به آن را با مقدار خودش جایگزین میکنیم. به عبارتی اکنون ما ۳۲/۱۶/۸ رنگ داریم و برای هر پیکسل، مرکز خوشه نزدیک به آن یعنی نزدیکترین رنگ از بین این رنگ ها را جایگزین میکنیم. به این صورت تعداد رنگ های تصویر به تعداد مدنظر کاهش میابد.

۳- بررسی نتایج

۳-۱- Color space

۳-۱-۱- بخش اول



تصویر ۶- تصویر رنگی و کانال های قرمز، سبز و آبی

فام رنگ، اشباع و شدت روشنایی تصویر lena به صورت زیر می باشد:

تصویر ۷- فام رنگ، اشباع و شدت روشنایی تصویر lena

۳-۱-۲- بخش دوم

این بخش دارای کد و خروجی نمی باشد. توضیحات این بخش در توضیحات فنی آمده است.

۳-۲- Quantization

۳-۲-۱- بخش اول

گزارش معیار MSE و PSNR برای تصاویر چندی سازی شده به صورت زیر می باشد:

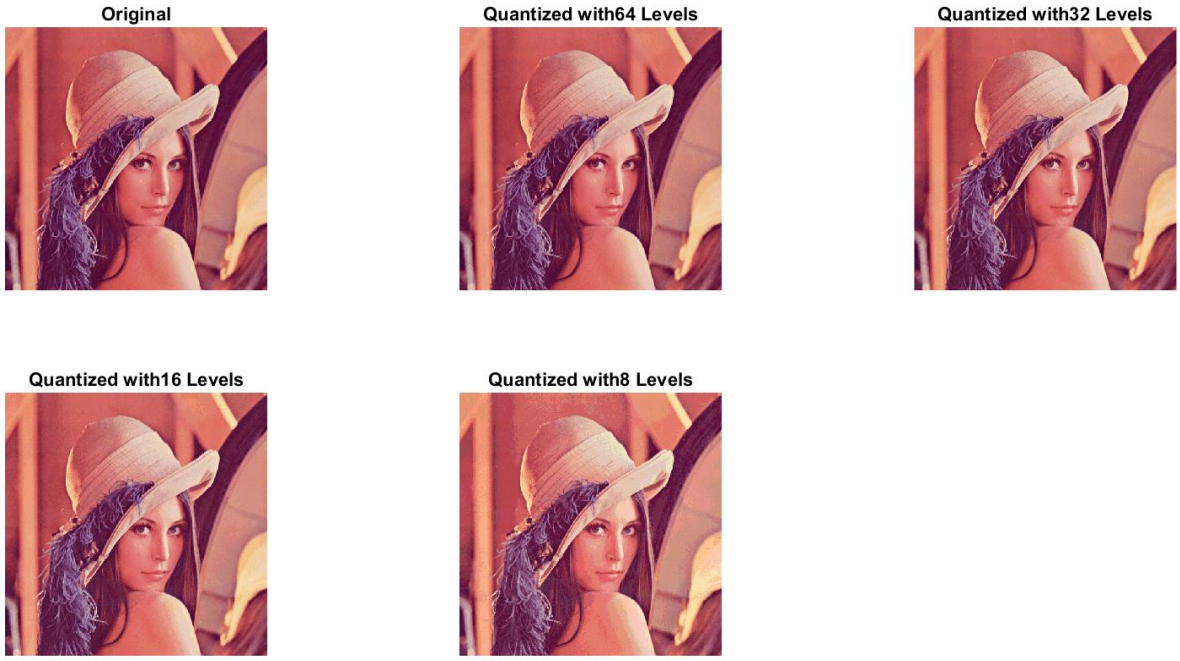
جدول ۱- گزارش معیار MSE و PSNR برای تصاویر چندی سازی شده با تعداد سطوح ۶۴ و ۳۲ و ۱۶ و۸

	MSE	PSNR

42.2999	3.8290	**L=64**
37.4491	11.6995	**L=32**
31.2296	48.9920	**L=16**
25.9141	166.5978	**L=8**

همانطور که میبینیم هر چه تعداد سطوح کاهش می‌یابد، مقدار معیار MSE کاهش و معیار PSNR افزایش می‌یابد، درنتیجه هرچه تعداد سطوح خاکستری کاهش می‌یابد، با تعداد رنگ کمتری می‌توان تصویر اصلی را بازسازی کرد در نتیجه کیفیت تصویر کمتر می‌شود.

تصویر زیر نیز خروجی چندی سازی یکنواخت تصویر lena با تعداد متفاوت سطوح خاکستری می باشد:



تصویر ۸- تصویر lena با تعداد سطوح خاکستری مختلف

۳-۲-۲- بخش دوم

تصویر ۹- تصویر با تعداد سطوح خاکستری ۸-۸-۴ برای قرمز، سبز و آبی و همچنین تعداد سطوح خاکستری ۸ برای هر ۳ مولفه

۳-۲-۳- بخش سوم

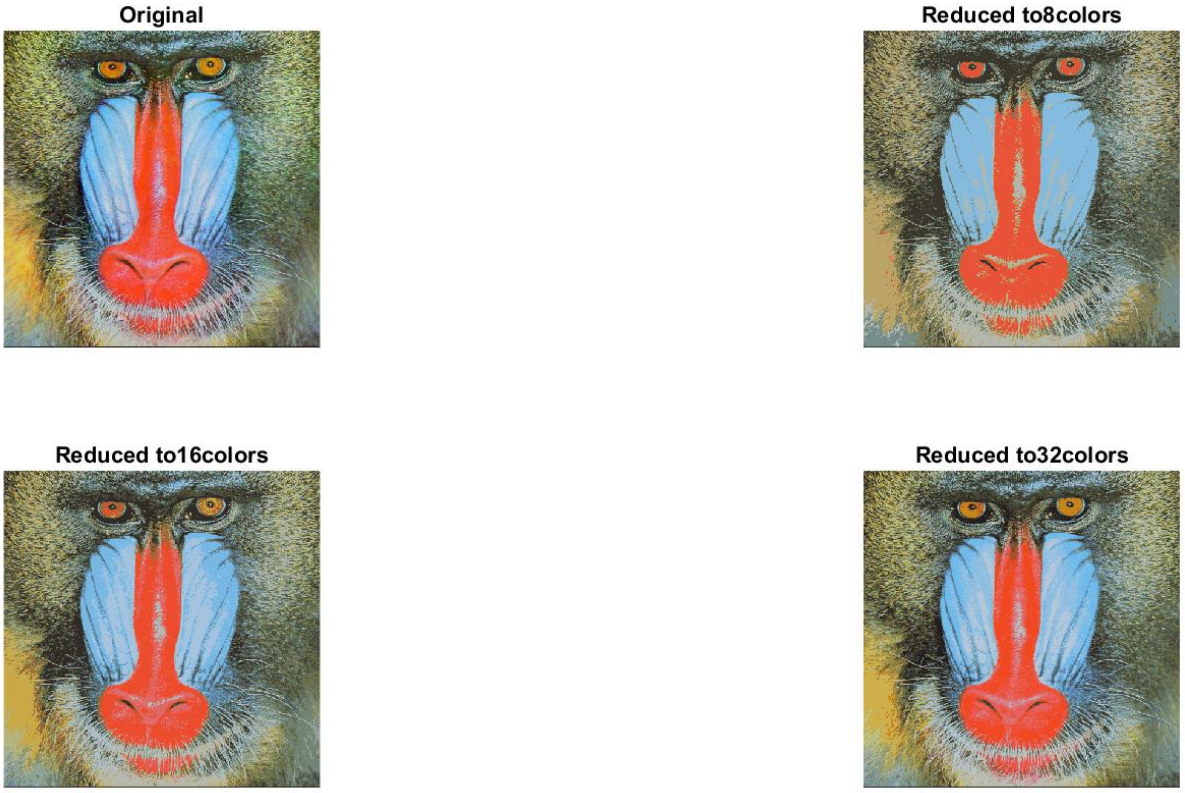
در جدول زیر گزارش معیار MSE و PSNR برای تصاویر کاهش داده شده به صورت زیر می باشد:

جدول ۲- گزارش معیار MSE و PSNR برای تصاویر با تعداد رنگ ۸ و ۱۶ و ۳۲

PSNR	MSE	
22.4251	372.0273	Num_colors=8
24.9137	209.7534	Num_colors=16
27.1709	124.7360	Num_colors=32

همانطور که مشاهده میکنید هر چه تعداد رنگ ها از بالا به پایین افزایش می‌یابد، مقدار معیار MSE کاهش و PSNR افزایش می‌یابد که به این دلیل است که هر چه تعداد رنگ برای تصویر بیشتر شود، تصویر اصلی با دقت بیشتری بازسازی می‌شود.

در زیر نیز تصویر خروجی برای هر یک از این تعداد رنگ را مشاهده می‌کنید:



جدول ۳- تصاویر با تعداد رنگ ۸ و ۱۶ و ۳۲

۴- پیوست

۴-۱- Color space

۴-۱-۱- بخش اول

کد main:

```
lena = imread("D:\Dars\Masters\digital image processing\Homeworks\Images\6\Lena.bmp");
redChannel = lena(:, :, 1);
greenChannel = lena(:, :, 2);
blueChannel = lena(:, :, 3);
% plot red green and blue channels seperately
figure;
```

```
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
subplot(2,2,1);colormap gray;imshow(rgbImage);title('Original');
subplot(2,2,2);colormap gray;imshow(redChannel); title('Red channel');
subplot(2,2,3);colormap gray;imshow(greenChannel); title('Green channel');
subplot(2,2,4);colormap gray;imshow(blueChannel); title('Blue channel');
[hue, saturation, intensity] = rgb_to_hsi(rgbImage);
figure;
% plot Hue and saturation and intensity
subplot(1,3,1);colormap gray;imshow(hue,[]); title('Hue');
subplot(1,3,2);colormap gray;imshow(saturation,[]); title('Saturation');
subplot(1,3,3);colormap gray;imshow(intensity,[]); title('Intensity');
```

كد متد calculate_hue:

```
function hue = calculate_hue(rgb_image)
redChannel = rgb_image(:, :, 1);
greenChannel = rgb_image(:, :, 2);
blueChannel = rgb_image(:, :, 3);
[height, width] = size(redChannel);
hue = zeros(height, width);
for x=1:height
   for y=1:width
      temp = ((redChannel(x, y) - greenChannel(x, y)) + (redChannel(x, y) - blueChannel(x, y))); %(R-
G)+(R-B)
      temp2 = power(redChannel(x, y) - greenChannel(x, y), 2); % (R-G).^2
      temp3 = (redChannel(x, y) - blueChannel(x, y)) * (greenChannel(x, y) - blueChannel(x, y)); % (R-
B)(G-B)
      arg = (0.5 * temp) / sqrt(temp2 + temp3);
      theta = acos(arg);
      if blueChannel(x, y) > greenChannel(x, y)
         hue(x,y) = 2.0 * pi - theta;
      else
         hue(x,y) = theta;
      end
   end
end
hue = hue / (2.0 * pi);
hue = hue * 255.0;
end
```

كد متد calculate_intensity:

```
function intensity = calculate_intensity(rgb_image)
redChannel = rgb_image(:, :, 1);
greenChannel = rgb_image(:, :, 2);
blueChannel = rgb_image(:, :, 3);
intensity = (redChannel + greenChannel + blueChannel) / 3.0;
intensity = intensity * 255.0;
```

كد متد calculate_saturation:

```
function saturation = calculate_saturation(rgb_image)
redChannel = rgb_image(:, :, 1);
greenChannel = rgb_image(:, :, 2);
blueChannel = rgb_image(:, :, 3);
x = min(min(redChannel,greenChannel),blueChannel);
```

```
y = (redChannel + greenChannel + blueChannel + 1e-4);
saturation = 1.0 - ((3.0 * x) ./ y);
saturation = saturation * 255.0;
```

کد متد rgb_to_hsi:

```
function [hue, saturation, intensity] = rgb_to_hsi(rgb_image)
rgb_image = double(rgb_image) / 255.0;
hue = calculate_hue(rgb_image);
saturation = calculate_saturation(rgb_image);
intensity = calculate_intensity(rgb_image);
end
```

۴-۱-۲- بخش دوم

این بخش دارای کد و خروجی نمی باشد. توضیحات این بخش در توضیحات فنی آمده است.

۴-۲- Quantization

۴-۲-۱- بخش اول

کد main:

```
lena = imread("D:\Dars\Masters\digital image processing\Homeworks\Images\6\Lena.bmp");
redChannel = lena(:, :, 1);
greenChannel = lena(:, :, 2);
blueChannel = lena(:, :, 3);
rgbImage = cat(3, redChannel, greenChannel, blueChannel);

levels = [64; 32; 16; 8];
quantized_images = zeros(size(redChannel));
figure;
subplot(2,3,1); imshow(rgbImage,[]); title('Original');
for i=1:size(levels)
   current_level = levels(i);
   red_quantized = uniform_quantize(redChannel, current_level);
   green_quantized = uniform_quantize(greenChannel, current_level);
   blue_quantized = uniform_quantize(blueChannel, current_level);
   quantized = cat(3, red_quantized, green_quantized, blue_quantized);

   mse_quantized = immse(lena, uint8(quantized))
   psnr_quantized = psnr(uint8(quantized), lena)
   quantized_to_show = uint8(cat(3, quantized(:,:,1), quantized(:,:,2), quantized(:,:,3)));
   subplot(2,3,i+1); imshow(quantized_to_show); title(strcat('Quantized with ',int2str(levels(i)),'
Levels'));
 end
```

کد متد uniform_quantizer:

```
function dst = uniform_quantize(img, num_levels)
[height, width] = size(img);
dst = zeros(height, width);
max_value = max(img(:));
min_value = min(img(:));
dynamic_range = max_value - min_value;
Q = zeros(256, 1);
step_size = dynamic_range / num_levels;
for i=1:256
   Q(i, 1) = floor((i - min_value) / step_size) * step_size + step_size / 2 + min_value;
end

for x=1:height
   for y=1:width
      dst(x, y) = Q(img(x,y),1);

   end
end
end
```

۴-۲-۲- بخش دوم

کد main:

```
lena = imread("D:\Dars\Masters\digital image processing\Homeworks\Images\6\Lena.bmp");
redChannel = lena(:, :, 1);
greenChannel = lena(:, :, 2);
blueChannel = lena(:, :, 3);
rgbImage = cat(3, redChannel, greenChannel, blueChannel);

red_quantized_8 = uniform_quantize(redChannel, 8);
green_quantized_8 = uniform_quantize(greenChannel, 8);
blue_quantized_4 = uniform_quantize(blueChannel, 4);
quantized_884_to_show = cat(3, red_quantized_8, green_quantized_8, blue_quantized_4);
blue_quantized_8 = uniform_quantize(blueChannel, 8);
quantized_8_to_show = cat(3, red_quantized_8, green_quantized_8, blue_quantized_8);

figure;
subplot(3,1,1); imshow(rgbImage); title('Original');
subplot(3,1,2); imshow(uint8(quantized_884_to_show)); title('Quantized with 8-8-4 Levels for RGB
Respectively');
subplot(3,1,3); imshow(uint8(quantized_8_to_show)); title('Quantized with 8 Levels');
```

۴-۲-۳- بخش سوم

کد main:

```
baboon = imread("D:\Dars\Masters\digital image processing\Homeworks\Images\6\Baboon.bmp");
redChannel = baboon(:, :, 1);
```

```
greenChannel = baboon(:, :, 2);
blueChannel = baboon(:, :, 3);
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
Levels = [8; 16; 32];
figure;
subplot(4,1,1); imshow(rgbImage); title('Original');
for i=1:size(Levels)
   img_reduced = kmeans_color_quantize(baboon, Levels(i));
   red_reduced = img_reduced(:, :, 1);
   green_reduced = img_reduced(:, :, 2);
   blue_reduced = img_reduced(:, :, 3);

   img_reduced_to_show = cat(3, red_reduced, green_reduced, blue_reduced);
   subplot(4,1,i+1); imshow(img_reduced_to_show); title(strcat('Reduced to
',int2str(Levels(i)),'colors'));

   mse_value = immse(baboon, uint8(img_reduced_to_show))
   psnr_value = psnr(uint8(img_reduced_to_show), baboon)
end
```

کد kmeans_color_quantize:

```
function image_reduced = kmeans_color_quantize(img, clusters)
[height, width, ~] = size(img);
samples = zeros(height * width, 3);
count = 1;
for x=1:height
   for y=1:width
      samples(count,1) = img(x, y,1);
      samples(count,2) = img(x, y,2);
      samples(count,3) = img(x, y,3);
      count = count + 1;
   end
end
[labels, centers] = kmeans(samples,clusters,'maxiter',10000,'Display','iter','OnlinePhase','off');
centers = uint8(centers);
labels = uint8(labels);
image_reduced = centers(labels,:);
image_reduced = reshape(image_reduced, size(img));
image_reduced = imrotate(image_reduced, -90);
image_reduced = flip(image_reduced, 2);
end
```