تمرین ۵

هادی تمیمی
۹۶۲۲۷۶۲۴۰۸

اطلاعات گزارش
تاریخ: ۱۴۰۰/۳/۲۰

واژگان کلیدی:
تصاویر رنگی
چندی سازی
کاهش رنگ
خوشه بندی

چکیده

در ابتدا به معرفی رنگ پرداخته سپس فضاهای رنگی مختلف را معرفی و تبدیلات آنها به همدیگر را مورد بررسی می کنیم.

در مرحله بعد چندی سازی در فضای رنگی RGB را مورد بررسی قرار میدهیم.

در نهایت انواع روش های قطعه بندی به منظور کاهش تعداد رنگ های تصویر بررسی میشود.

۱-مقدمه

رنگ یک توصیف گر قدرتمند است که تشخیص یک شیء و استخراج آن از یک صحنه را ساده میکند. انسان میتواند هزاران شدت و سایه رنگی را از یکدیگر تمیز دهد در حالیکه فقط تعداد محدودی سطح خاکستری را از هم تشخیص میدهد و همین باعث میشود تصاویر رنگی برای چشم انسان کیفیت بهتری داشته باشد.

۲-شرح تکنیکال

چشم انسان دو نوع حسگر استوانه ای و مخروطی برای دیدن دارد.حسگرهای مخطروطی کیفیت بیشتری دارند و میتوانند رنگ ها را تشخیص دهند.البته این حسگرها برای فعال شدن نیاز به نور دارند(به همین دلیل در تاریکی رنگ ها قابل تشخیص نیستند).

به طور کلی سه رنگ قرمز آبی و سبز را رنگ های اصلی می نامند و دلیل آن سلول های مخروطی می باشد.

۵.۱ فضای رنگی

۵.۱.۱

این فضا بر اساس نحوه درک رنگ توسط بینایی انسان ایجاد شده است. فضای رنگی HSI هر رنگ را با سه جز نشان می دهد در این فضای رنگی مشخصاتی که به طور عمده برای شناسایی یک رنگ از دیگری استفاده می شوند عبارتند از:

- **Hue یا فام**: نشان دهنده ی طول موج رنگ غالب که توسط بیننده دریافت می شود.
- **Saturation یا اشباع**: میزان ترکیب نور سفید با یک فام(میزان خلوص رنگ)، هر چه اشباع بیشتر باشد، خلوص رنگ بیشتر و رنگ سفید اضافه شده کمتر میباشد.
- **Intensity**: شدت رنگ.

برای تبدیل رنگ از فضای RGB به HSI از فرمول های زیر استفاده میکنیم:

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

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

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

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

ترکیب hue و saturation با هم کروماتیسیتی choromaticity می گویند.

چشم انسان در نهایت آن رنگی که دریافت میکند به صورت RGB است اما چیزی که توسط مغز ما درک میشود با ویژگی های کروماتیسیتی و شدت روشنایی بهتر قابل بیان است.

۵.۱.۲

۱. فضای رنگی YIQ:

YIQ یک فضای رنگی مورد استفاده توسط تلویزیون های با سیستم رنگی NTSC می باشد.

در این فضا Y اطلاعات luminance بوده و دو جز I و Q اطلاعات chrominance تصویر را دارا می باشند.این مدل رنگی سعی میکند از نحوه بازخورد چشم انسان نسبت به رنگ استفاده کند. تبدیل از فضای RGB به YIQ به صورت زیر است:

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

برای تبدیل از YIQ به RGB داریم:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.619 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

۲. فضای رنگی XYZ:

این فضای رنگی متناسب با حساسیت اعصاب بینایی به طول موجهای مختلف است و متشکل از سه کمیت X ، Yو Z میباشد.

- کانال Z به صورت تقریبی با طول موجهای کوتاه رنگ مرتبط است.
- کانال X نمایش ترکیبی از طول موجهای بلند و متوسط است.
- کانال Y متناسب با طول موجهای متوسط رنگ میباشد و درخشش یا روشنایی کلی را مشخص میسازد که بیشتر به رنگ سبز وابسته است

تبدیل رنگها از RGB به XYZ :

$$X = 0.607 \cdot R + 0.174 \cdot G + 0.200 \cdot B$$
$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$Z = 0.066 \cdot G + 1.116 \cdot B$$

۳. فضای رنگی HCL:

مشکلی که فضاهای رنگی HSV و HSI دارند،این است که بر اساس saturation متغیر عمل می کنند و اینآن روشی نیست که چشم ما تصاویر را درک میکند.این مشکل به کمک فضای HCL حل می شود.در این فضا کنترل رنگ (hue)، میزان رنگی بودن(chroma) و روشنایی (luminance) توسط ما کنترل می شود.

برای تبدیل از RGB به HCL از فرمول های زیر استفاده می شود:

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

$$\alpha = \frac{1}{100} \times \frac{\min(R,G,B)}{\max(R,G,B)} \; ; \; \gamma = 3.$$

Q پارامتری است که شدت luminosity بین یک رنگ کامل اشباع شده و رنگ سفید را مشخص می کند.حال برای سه جز رنگی داریم :

$$H = atan2(G - B, R - G)$$

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

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

۴. فضای رنگی I1I2I3 :

فضای رنگی I1I2I3 به عنوان شکل ناهمبسته
اجزای فضای RGB که حاصل تبدیل Karhunen-Loeve است معرفی گردیده است.
برا تبدیل رنگها از RGB به I1I2I3 از فرمول های زیر استفاده میشود:

$$I1 = 1/3\ (R + G + B)$$
$$I2 = 1/2\ (R - B)$$
$$I3 = 1/4\ (2G - R - B)$$

۵. فضای رنگی YUV:

این فضای رنگی یک بازنمایی جدید از فضای RGB برای انتقال بهینه می باشد.این فضا از luminance(Y) و دو بخش رنگی (U,V) تشکیل شده است. مزیت اصلی مدل YUV در پردازش تصویر جدا کردن اطلاعات رنگی و luminance است.دلیل آن این است که جز luminance یک تصویر بدون اثرگذاری بر جز رنگی میتواند مورد پردازش قرار گیرد.
تبدیل از فضای RGB به YUV به صورت زیر است:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

برای تبدیل از YUV به RGB داریم:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.5806 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

فضای رنگی YUV معمولا برای پردازش ویدیو/تصاویر رنگی و انتشار داده ها در تلویزیون های آنالوگ به کار می رود.

۵.۲
چندی سازی

فرآیند چندی سازی بر روی تصاویر رنگی باعث کاهش تعداد رنگ های متمایز در یک تصویر را کاهش می دهد. هدف این است که کاهش تعداد رنگ ها با کمترین کاهش کیفیت تصویر همراه باشد.
این فرآیند برای نمایش تصاویر با تعداد زیادی رنگ بر روی دستگاه هایی با تعداد رنگ محدود لازم است.

برای چندی سازی از یک quantizer استفاده می کنیم:

اگر تابع f را داشته باشیم، نتیجه اعمال quantizer بر آن Q(f) می باشد.

اگر داشته باشیم:

Decision Levels = $\{t_k , k = 0, \dots, L\}$

Reconstruction Levels = $\{r_k , k = 0, \dots, L-1\}$

$If\ f \in [t_k, t_{k+1})$

$Then\ Q(f) = r_k$

این که این سطح ها به چه صورت باشند،انواع چندی سازی های مختلف را تشکیل می دهد.

همچنین اگر L سطح داشته باشیم، $\lceil \log_2 L \rceil$ بیت برای ذخیره سازی نیاز داریم.

۵.۲.۱

چندی سازی یکنواخت و تعداد سطوح ثابت:

فاصله یکنواختی بین سطح های تصمیم همجوار با سطح های بازسازی همجوار وجود دارد.

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

پارامترهایی که چندی سازی یکنواخت دارد به صورت زیر است:

L = تعداد سطوح

B = $f_{max} - f_{min}$

q = فاصله چندی سازی

Q = $\frac{B}{L}$

تابع چندی سازی یکنواخت به صورت زیر است:

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

در تصاویر رنگی *RGB* برای این که کل تصویر را چندی سازی کنیم هر مولفه را جداگانه را به تنهایی چندی سازی می کنیم.در صورتی که بخواهیم رنگ تصویر عوض نشد باید در هر سه مولفه ، از تعداد سطوح یکسان استفاده می کنیم.

۵.۲.۲

چندی سازی یکنواخت با تعداد سطوح متفاوت:

مانند بخش قبل است با این تفاوت که مولفه آبی ۴ سطح خاکستری(۲ بیت) داشته و مولفه های قرمز و سبز ۸ سطح خاکستری (۳ بیت) دارند.در شرح نتایج به تاثیر چندی سازی با تعداد سطوح متفاوت پرداخته خواهد شد.

۵.۲.۳

کاهش تعداد رنگ:

مسئله ای که با آن در هنگام چاپ، رنگ آمیزی و... مواجه هستیم محدود بودن تعداد رنگ ها است و اگر بخواهیم تصویری را چاپ کنیم باید تعداد رنگ های آن را کاهش دهیم.

برای این کار از الگوریتم kmeans برای خوشه بندی استفاده میکنیم. ورودی این الگوریتم داده ها و پارامتر k که تعداد خوشه ها است، می باشد.این الگوریتم k مرکز خوشه پیدا کرده و هر داده را به خوشه ای که به آن نزدیک تر است دسته بندی میکند.

الگوریتم kmeans :

۱. ابتدا k میانگین که نماینده خوشه ها هستند را به صورت تصادفی مقداردهی میشود.
۲. سپس، این دو مرحله پایین را به تناوب چندین بار اجرا می‌کنیم تا میانگین‌ها به یک ثبات کافی برسند و یا مجموع واریانس‌های خوشه‌ها تغییر چندانی نکنند:
۳. هر داده را به نزدیک ترین خوشه مربوطه اختصاص بده
۴. مرکز جدید هر خوشه را میانگین داده های درون آن قرار بده.
۵. در نهایت هر داده را به خوشه های نهایی اختصاص بده.

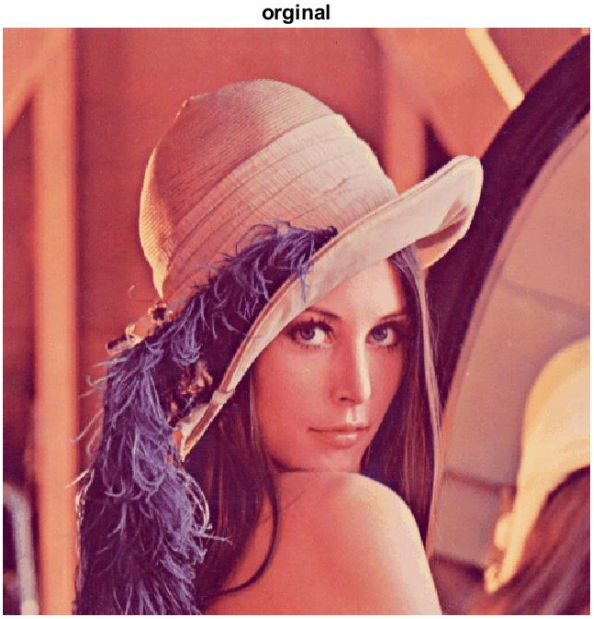
با اجرای kmeans رنگ های نزدیک به هم به یک خوشه اختصاص می یابند.

۲-شرح نتایج

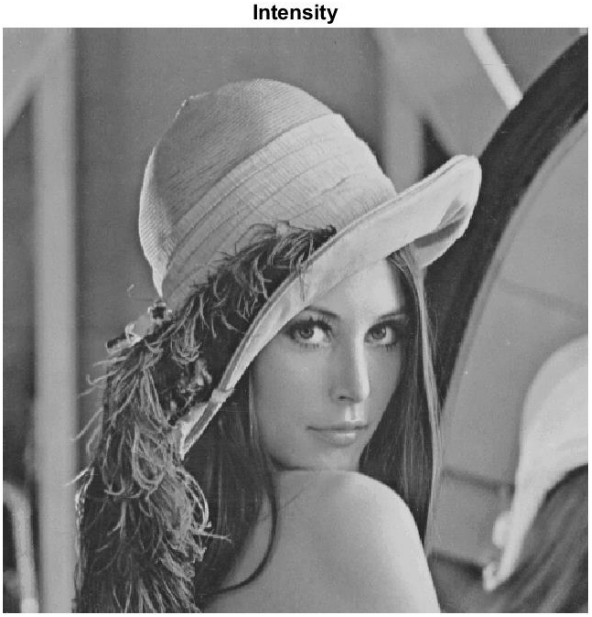
۵.۱.۱

Hue

orginal



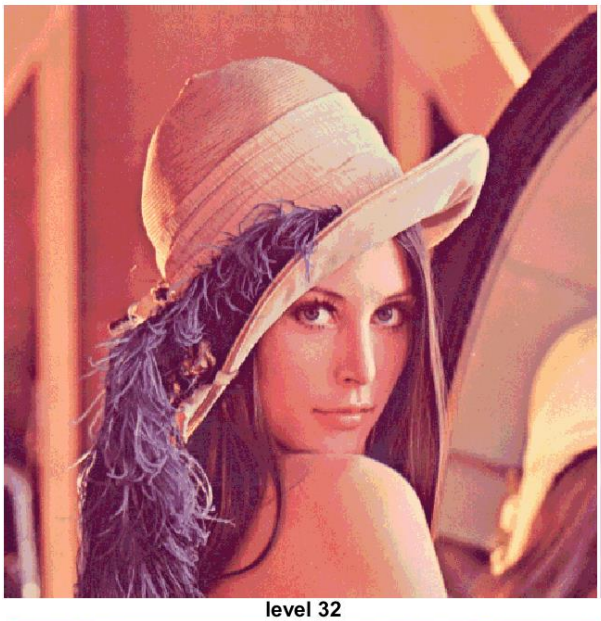
Intensity



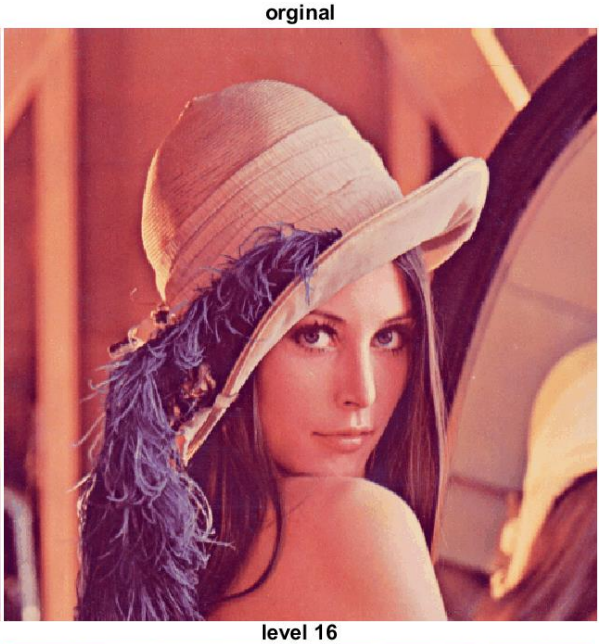
Saturation

۵.۲.۱

level 8

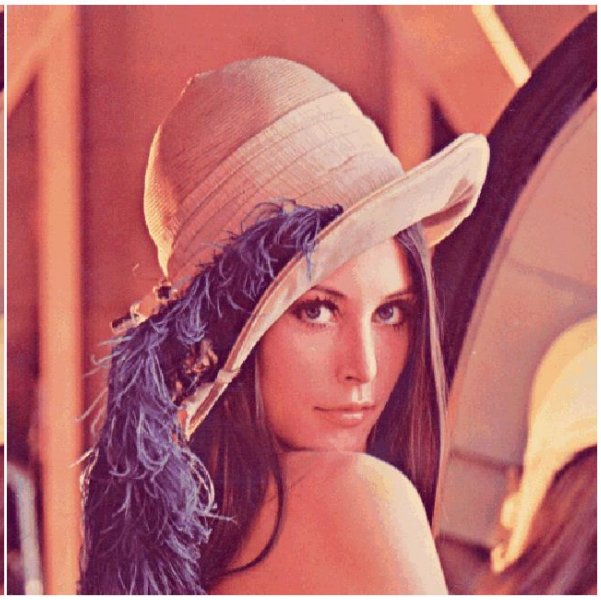


orginal



level 32

level 16



level 64

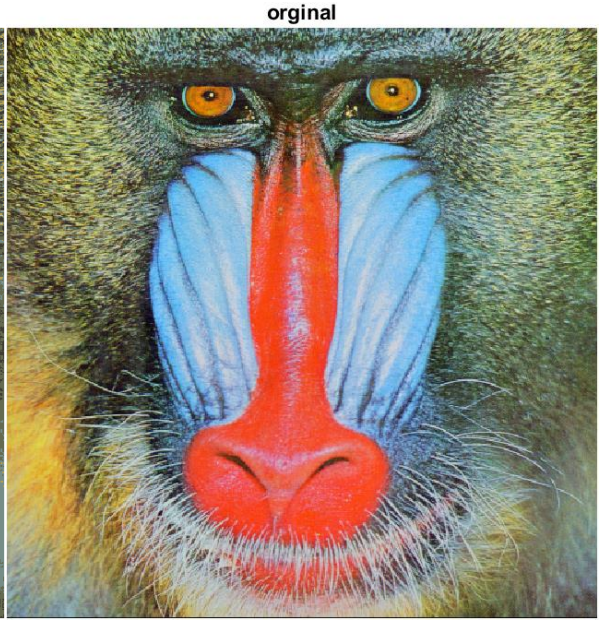
سطح	psnr	mse
64	40.89	8.55
32	36.38	21.25
16	30.68	70.56
8	24.05	280.32

همانطور دیده میشود با کاهش تعداد سطوح mse افزایش یافته و نرخ سیگنال به نویز نیز افزایش می یابد.
دلیل آن هم کم تر بودن data loss زمانی که تعداد سطوح بیشتری داریم، می باشد.

۵.۲.۳

8 colors

orginal



32 colors

16 colors

colors	psnr	mse
32	27.26	85.97
16	27.89	160.05
8	25.16	304.36

هر چه تعداد رنگ ها کمتر باشد،data loss بیشتری خواهیم داشت.

کد ها:

۵.۱.۱

```
img = imread('Images/5/Lena.bmp');
figure
imshow(img);
title('orginal');

R = im2double(img (:,:,1));
G = im2double(img (:,:,2));
B = im2double(img (:,:,3));


tetha = acosd((0.5*((R-G)+(R-B)))./((R-G).^2+(R-B).*(G-B)).^0.5);


%H component
X = B-G;
t = X<=0;
h1 = tetha .*t;
t = X>0;
h2 = (360-tetha) .*t;
H = h1+h2;
H = H./360;

%S component
S = 1 - (3.*min(min(R,G),B)./(R+G+B));

%I component
I = (R+G+B)/3;



figure
imshow(H);
title('Hue');
figure
imshow(S);
title('Saturation');
figure
imshow(I);
title('Intensity');
```

۵.۲.۱

```
img = imread('Images\5\Lena.bmp');


L = 64; %change for diffrent level

R = uniform_quantizer(img(:,:,1),L);
G = uniform_quantizer(img(:,:,2),L);
B = uniform_quantizer(img(:,:,3),L);
x = img*0;
x(:,:,1) = R;
x(:,:,2) = G;
x(:,:,3) = B;
mse_error = immse(x,img);
psnr_error = psnr(x,img);
```

```
figure
imshow(img);
title('orginal')
figure
imshow(x);
title("level "+L+"")


function output = uniform_quantizer(gray_img,L)

    %dynamic range
    fmax = max(max(gray_img));
    fmin = min(min(gray_img));
    B = fmax - fmin;

    %quantization interval
    q = B/L;

    [M,N] =size(gray_img);
    output=zeros(M,N,'uint8');
    for r=1:M
        for c=1:N
            f = gray_img(r,c);
            Qf = floor((f-fmin)/q)*q+q/2+fmin;
           output(r,c)=Qf;
        end
    end
end
```

۵.۲.۳

```
img = imread('Images\5\Baboon.bmp');

L = 32;  %change

figure
imshow(img);
title("orginal")
out = kmeans_quantizer(img,L);
figure
imshow(out);
title(""+L+" colors")

mse_error = immse(out,img);
psnr_error = psnr(out,img);
function output = kmeans_quantizer(img,clusters)

    [M,N,dim] = size(img);

    samples =zeros([M*N,dim],'double');
    count = 1;
    for c=1:N
        for r=1:M
            samples(count,:)=reshape(img(r,c,:),[1,3]);
            count = count + 1;
        end
```

```
    end

    [idx,centers] = kmeans(samples,clusters,'MaxIter',1000);
    centers = uint8(centers);
    result = centers(idx,:);
    result = reshape(result,[M,N,dim]);
    output = result;
end
```

پایان