Color

علیرضا لرستانی

اطلاعات گزارش

تاریخ: **05**/10/1399

واژگان کلیدی:
HSI
RGB
Quantization
قطعه بندی
فضای رنگی
Gray Scale
Hue
Saturation
Intensity
Uniform quantization
k-mean clustering
PSNR
MSE
YUV
Lab
YIQ

چکیده

از آنجایی که رنگ یکی از مولفه های اساسی در توصیف تصاویر است، در این بخش با فضاهای رنگی شامل فضای رنگی RGB و HSI آشنا شده و برخی از متد های پردازش تصویر در این فضا ها را با هم بررسی میکنیم.

۱-مقدمه

رنگ یک توصیف گر قدرتمند است که تشخیص یک شی و استخراج آن از صحنه را ساده میکند.

همانطور که میدانید چشم انسان قابلیت ادراک تفاوت بسیار محدود و انگشت شماری در فضای خاکستری یا grayscale را دارد اما این مقدار در فضای رنگی به چندین هزار افزایش میکند. یعنی چشم انسان میتواند به راحتی چندین هزار رنگ را از هم تمییز دهد.

از این رو آشنایی با فضای رنگی در تصاویر و توانایی پردازش بر روی تصاویر رنگی از اهمیت بالایی برخوردار است.

نوشتار حاضر، به بررسی و پیاده سازی (با زبان برنامه نویسی متلب) تعدادی عملیات پردازش تصویر در فضای رنگی میپردازد.

* پست الکترونیک نویسنده مسئول:
alirezalorestani2010@gmail.com

1. دانشجو، دانشکده مهندسی، دانشگاه فردوسی مشهد

۲-شرح تکنیکال

ابتدا به بررسی و تشریح فضاهای رنگی خواهیم پرداخت:

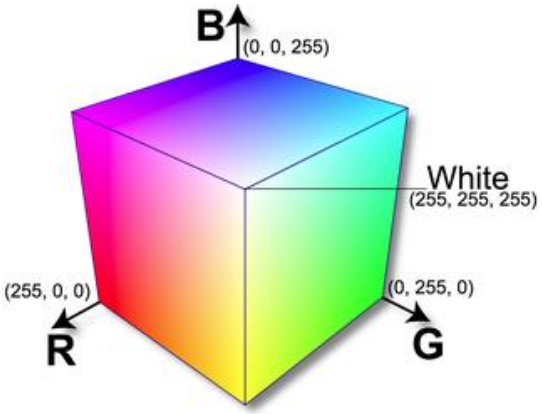
- **RGB**

در فضای رنگی RGB، هر تصویر رنگی از سه تصویر قرمز، آبی و سبز که در واقع همان رنگ های اصلی هستند تشکیل شده است. برای توصیف هر رنگ از ترکیب های مختلف این سه رنگ استفاده میشود.

به طور مثال رنگ سفید را اینگونه توصیف میکنیم:

White in RGB = (255, 255, 255)

در ضمن لازم به ذکر است که در چشم انسان گیرنده های RGB وجود دارند.



مکعب بالا به خوبی فضای رنگی RGB را نشان میدهد. همانطور که مشاهده میکنید با حرکت بر روی محور ها تنها سه رنگ اصلی قرمز، آبی و سبز را خواهیم داشت و همچنین با حرکت بر روی قطر مکعب در گوشه مقابل به راس مختصات مکعب، رنگ سفید را خواهیم داشت.

- **HSI**

همچنین برای توصیف تصاویر رنگی میتوانیم از فضای رنگی HSI استفاده کنیم.

در این فضا برای توصیف تصاویر از سه مولفه hue و saturation و intensity استفاده میکنیم.

پارامتر I در واقع همان میانگین بدون وزن رنگ هاست.

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

پارامتر H نیز نشان دهنده طول موج رنگ غالب است.

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

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

و در پایان پارامتر S میزان خلوص رنگ سفید را مشخص میکند:

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

در این رابطه min(R, G, B) در واقع مشخص کننده ی خلوص رنگ سفید است. اگر هر یک از 3 پارامتر R، G و B صفر باشد در نتیجه رنگ سفیدی نداریم و آن رنگ خالص است.

در ضمن قابل ذکر است که مغز انسان تصاویر را طبق این مدل درک میکند.

- **Quantization**

به طور کلی نوعی فرایند فشرده سازی است که در آن ویژگی های نزدیک به هم را یکی در نظر میگیریم.

در این نوشتار موضوع کلام ما color quantization است که در واقع یک فرآیند ساده برای کاهش تعداد رنگ های موجود در تصویر است. این فرآیند بدین صورت کار میکند که پیکسل های مجائر به هم که رنگ های نزدیکی به یکدیگر دارند را یک قطعه در نظر میگیریم.

نکته ی حائز اهمیت در این در این فرانید این است که تصویر نهایی باید تا حد ممکن به تصویر اصلی شبیه باشد.

یکی از تکنیک های ساده و سریع کاهش تعداد رنگ در تصاویر uniform quantization است. این فرایند کاملا مستقل از تصویر ورودی و رنگ های موجود در آن عمل میکند و هر رنگ را به مقدار جدیدی نظیر میکند.
قابل حدس است که این عملیات چندان عالی عمل نخواهد کرد و هیچ گونه تضمینی مبنی بر این که تصویر نهایی حداکثر شباهت به تصویر اصلی را دارا باشد، ندارد.

تکنیک بعدی برای دستیابی به این هدف، استفاده از الگوریتم k-means است.
توضیح کلی این الگوریتم به شرح زیر است:
ابتدا عدد k که مشخص کننده تعداد رنگ های مطلوب ما پس از کاهش است را مشخص میکنیم.
سپس از کل فضای رنگی k رنگ را انتخاب میکنیم.
و در پایان به ازای هر رنگ در هر پیکسل تصویر اصلی نزدیک ترین رنگ در بین k نمونه ی انتخاب شده را انتخاب و جایگزین میکنیم.
این تکنیک بهینه تر از تکنیک قبلی عمل کرده و مارا به نتیجه ی مطلوب (که حداکثر شباهت به نمونه ی اصلی را داراست) نزدیک تر میکند.

در پایان این بخش نیز به توضیح PSNR و MSE میپردازیم:
Peak signal-to-noise ratio یا PSNR یک اصطلاح مهندسی بوده که نسبت بین حداکثر توان یک سیگنال و قدرت تخریب نویز آن را نشان میدهد و معمولا با واحد دسی بل لگاریتمی بیان میشود.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m}\sum_{j=0}^{n}[I(i,j) \quad K(i,j)]^2$$

The PSNR (in dB) is defined as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$
$$= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$
$$= 20 \cdot \log_{10}(MAX_I) \quad 10 \cdot \log_{10}(MSE)$$

مقدار MAX_I در حالتی که هر پیکسل با 8 بیت معرفی میشود، 255 خواهد بود.

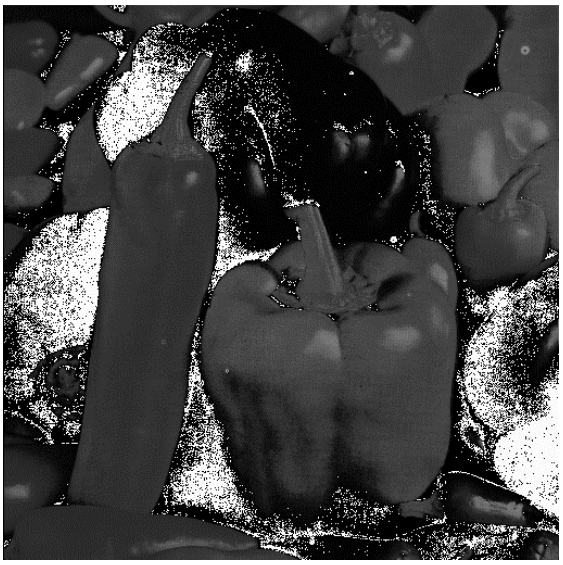
۳-شرح نتایج و نتیجه گیری

5.1.1

ابتدا تصویر اصلی که در فضای RGB است را با هم مشاهده میکنیم:

حال برای این تصویر مقادیر hue، saturation و intensity را محاسبه میکنیم و تصویر را در هر تنها با هر یک از این سه پارامتر به طور جداگانه نمایش میدهیم

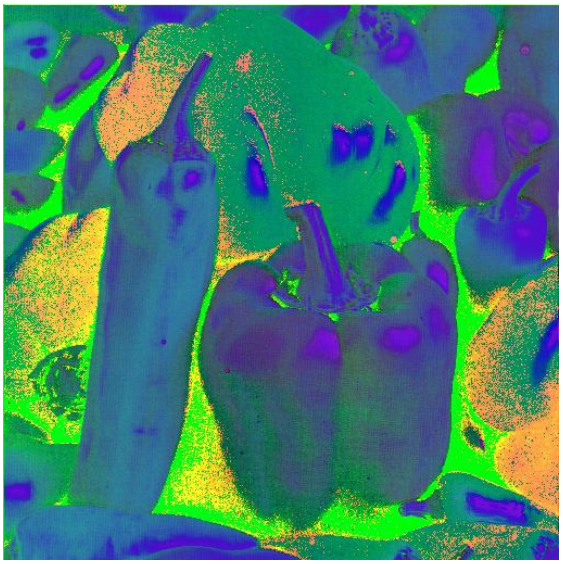
Hue



Saturation

Intensity

حال با ادغام این سه تصویر به تصویر زیر خواهیم رسید:



همانطور که پیش تر نیز توضیح داده شده هر یکه از سه پارامتر H، S و I به ترتیب بیانگر طول موج رنگ غالب، میزان خلوص رنگ سفید و میانگین بدون وزن رنگ ها هستند.

5.1.2

- **فضای رنگ YIQ**

فضای رنگ YIQ در تلویزیون/ ویدئو آمریکا و سایر کشورهایی که از استاندارد کمیته ی ملی سیستم تلویزیون استفاده میکنند، به کار میرود. در این مدل Y روشنایی است، I و Q حاوی اطلاعات رنگ هستند. به طوری که I اطلاعات رنگ آبی – نارنجی و Q اطلاعات رنگ بنفش – سبز را نگهداری میکند.

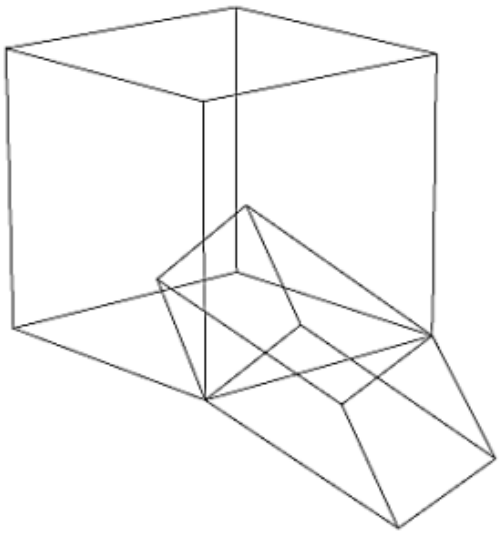
تبدیل بین این فضای رنگ و RGB بسیار ساده است:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

همچنین معکوس این تبدیل به شکل زیر است:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

خطی بودن تبدیلات و در نتیجه سادگی پیاده سازی، آن ها را به گزینه ای مناسب برای پردازش تصویر رنگی بدل کرده است.



مکعب RGB و تبدیل YIQ آن

- **مدل رنگی Lab**

Lab کاملترین فضای رنگی است که توسط کمیته بین المللی نورپردازی تعیین شده و تمام رنگ های قابل مشاهده برای چشم انسان را توصیف میکند.

سه مختصات L*,a*,b* بیانگر مشخصاتی میباشند که ذیلاً ذکر می گردد.

L* : نشان دهنده شدت روشنائی است. L* = 0 به منزله سیاه و L* = 100 نشان دهنده پراکندگی روشنائی یا نور کامل است.

a* : موقعیت آن بین سبز و قرمز متغیر است، مقادیر منفی a* نشان دهنده رنگ های سبز و مقادیر مثبت آن به منزله رنگهای قرمز هستند.

b* : موقعیت آن بین آبی و زرد متغیر است، مقادیر منفی b* نشان دهنده رنگ های آبی و مقادیر مثبت آن به منزله رنگهای زرد هستند.

مدل رنگ Lab مدل جامعی بوده و مد رنگی CMYK و مد رنگی RGB زیر مجموعه‌ای از این مدل هستند.

علامت ستاره (*) بالای سه نماد مختصاتی یعنی Lو aو bنشان دهنده استفاده از سیستم جدید رنگ در سیستم رنگی CIELAB است. این سیستم نسل جدید همان سیستم قدیمی CIELAB است.

$$\begin{cases} L^* = 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16 & \text{for } \frac{Y}{Y_n} > 0.008856 \\ L^* = 903.3\left(\frac{Y}{Y_n}\right)^{1/3} & \text{for } \frac{Y}{Y_n} \le 0.008856 \end{cases}$$

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right)$$

$$b^* = 200\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)$$

$$\begin{cases} f(u) = u^{1/3} & \text{for } u > 0.008856 \\ f(u) = 7.787u + \frac{16}{116} & \text{for } u \le 0.008856 \end{cases}$$

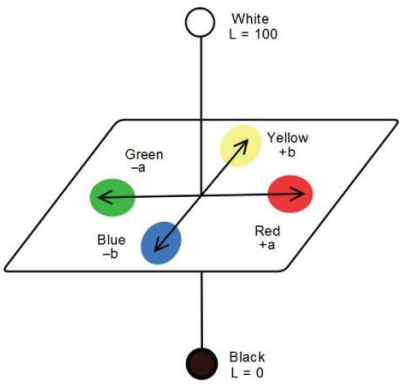
که در آن:

$$[X_n, Y_n, Z_n] = [0.950450,\ 1.000000,\ 1.088754]$$

و

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

در زیر این فضای رنگی سه بعدی به تصویر کشیده شده است:



- **مدل رنگ YUV**

این فضای رنگی زمانی اختراع شد که مهندسان می خواستند تلویزیون رنگی را در یک زیرساخت سیاه و سفید داشته باشند. آنها به یک روش انتقال سیگنال نیاز داشتند که با تلویزیون سیاه و سفید سازگار باشد و در عین حال بتواند رنگ را نیز اضافه کند.

این فضای رنگی رنگ را با Y یا همان luma یا همان روشنایی و دو جز کرومینانس UV توصیف میکند.

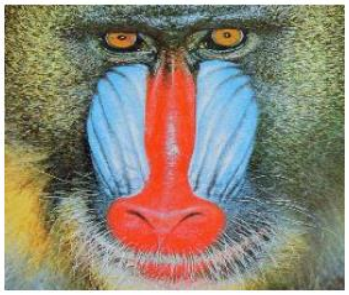
سیگنال های U و V می گویند بدون تغییر در روشنایی ، رنگ یک نقطه خاص را تغییر دهد.

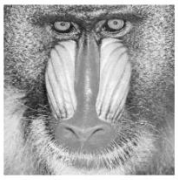
تلویزیون های سیاه و سفید قبلی فقط از اطلاعات روشنایی (Y) استفاده می کردند. اطلاعات رنگی تصویر(U,V) به صورت جداگانه توسّط یک زیر حامل اضافه شد تا گیرنده های سیاه و سفید قدیمی همچنان قادر به دریافت تصاویر باشند.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

در ادامه تصویری در فضای RGB را به سه تصویر جداگانه در فضای RUV تبدیل میکنیم.

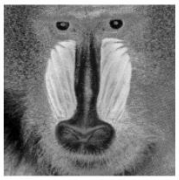
تصویر اصلی





Y

U



V

5.2.1

ابتدا مقادیر محاسبه شده برای دو پارامتر psnr و mse را در جدول زیر مشاهده میکنیم:

	MSE	PSNR
L=8	2828.7584	13.6148
L=16	1557.9114	16.2054
L=32	938.9541	18.4044
L=64	647.6914	20.0171

سپس تصویر اصلی را با هم مشاهده میکنیم:

سپس تصویر پس از عمل quantize با مقدار L = 8 را مشاهده میکنیم:

با مقایسه ی مقادیر به دست آمده برای mse و psnr مشاهده میکنیم که مقدار mse از همه بیشتر و مقدار psnr از همه کم تر است.

میتوان نتیجه گرفت که این تصویر بیشترین اختلاف را با تصویر اصلی دارا بوده و تعداد رنگ هایش به شدت محدود شده به طوری که با تصویر اصلی تفاوت فاحشی پیدا کرده است.

سپس تصویر پس از عمل quantize با مقدار $L = 16$ را مشاهده میکنیم:

در مقایسه با $L = 8$ شرایط بهتری دارد اما همچنان مقدار mse آن بسیاز بزرگ و مقدار psnr آن کوچک است که باز هم خبر از اختلاف نه چندان مطلوب با تصویر اصلی میدهد.

سپس تصویر پس از عمل quantize با مقدار $L = 32$ را مشاهده میکنیم:

همان طور که از تصویر پیداست اصالت آن به مراتب از دو تصویر قبلی بیشتر شده است. مقادیر داخل جدول نیز اثباتی بر این مدعا هستند.

و در نهایت تصویر پس از عمل quantize با مقدار $L = 64$ را مشاهده میکنیم:

با این که مقادیر mse و psnr همچنان چندان مطلوب نیستند اما به مراتب بهتر از 3 تصویر قبلی هستند و همانطور که مشاهده میشود تصویر حاصل نیز بیشترین میزان شباهت به تصویر اصلی را داراست.

5.2.2

ابتدا تصویر اصلی را مشاهده میکنیم:

سپس تصویر quatize شده را مشاهده میکنیم:

همانطور که مشاهده میشود از آنجایی که تعداد بیت های رنگ آبی را کم تر از دو رنگ دیگر در نظر گرفتیم، تصویر به سبز و قرمز متمایل شده است.

همجنین کاهش رنگ ها سبب کاهش رنگ سفید در تصویر نیز گشته است.

5.2.3

در این بخش عمل کاهش را با الگوریتم k-means انجام میدهیم. همانطور که پیش تر گفتیم این الگوریتم عملکرد بهتری از uniform quantization دارد.

ابتدا تصویر اصلی را مشاهده میکنیم:

سپس تصویر حاصل پس از quantizaton با k = 8 را مشاهده میکنیم:

سپس تصویر حاصل پس از quantizaton با k = 16 را مشاهده میکنیم:

سپس تصویر حاصل پس از quantizaton با k = 32 را مشاهده میکنیم:

همانطور که مشاهده میشود الگویتم k-means عملکرد مطلوبی دارد و هرچه عدد k بیشتر میشود نتیجه ی حاصل رضایت بخش تر نیز میشود.

4-پیوست

- 5.1.1

```
img = original_img;
img = im2double(img);
r = img(:,:,1);
g = img(:,:,2);
b = img(:,:,3);
Hue = acos((0.5*((r-g)+(r-b)))./((sqrt((r-g).^2+(r-b).*(g-b)))+eps));
Hue(b>g) = 2*pi - Hue(b>g);
Hue = Hue/(2*pi);

Saturation = 1 - 3.*(min(min(r,g),b))./(r+g+b+eps);

Intensity = (r+g+b)/3;

hsi_image = cat(3,Hue,Saturation,Intensity);
```

- 5.2.1

```
mg = original_img;

 = img(:,:,1);
 = img(:,:,2);
 = img(:,:,3);

8 = quantizingImage(r,3);
8 = quantizingImage(g,3);
8 = quantizingImage(b,3);
uantized8=cat(3,r8,g8,b8);

16 = quantizingImage(r,4);
16 = quantizingImage(g,4);
16 = quantizingImage(b,4);
uantized16=cat(3,r16,g16,b16);

32 = quantizingImage(r,5);
32 = quantizingImage(g,5);
32 = quantizingImage(b,5);
uantized32=cat(3,r32,g32,b32);

64 = quantizingImage(r,6);
64 = quantizingImage(g,6);
64 = quantizingImage(b,6);
uantized64=cat(3,r64,g64,b64);
```

- 5.2.2

```
img = original_img;

r = img(:,:,1);
g = img(:,:,2);
b = img(:,:,3);

r3 = quantizingImage(r,3);
g3 = quantizingImage(g,3);
b2 = quantizingImage(b,2);
quantized_img = cat(3,r3,g3,b2);
```

- 5.2.1 and 5.2.2 (uniform quantization)

```
function quantizedImage = quantizingImage(img,level)
    image = double(img);
    img_size = size(image);
    n = level;
    w = img_size(2);
    h = img_size(1);
    im = image/255;
    new_image = zeros(h,w);
    for i=1:h
        for j=1:w
            th=1/n;
            for k = 1:n
                if(im(i,j)>th)
                    new_image(i,j) = th+1/n;
                    th = th+1/n;
                end
            end
        end
    end
    new_image = uint8(255*new_image);
    quantizedImage = new_image;
end
```

5.2.3 •

```
MAX_IT = 10;
ep = 10;
iteration = 0;
errorsum = 11;
[M,N, c] = size (x);
labels = zeros (M,N);
centroids = uint8 (zeros (1,K, 3));
centroids (1,:,1) = uint8 (linspace (0,256,K));
centroids (1,:,2) = uint8(linspace(0,256,K));
centroids (1,:,3) = uint8 (linspace(0,256,K));

y = zeros (M,N, c);
x = double (x);

sums = zeros (1,K,3);
counts = zeros (1,K);

while (iteration <= MAX_IT && errorsum > ep)
    errorsum = 0;
    for m = 1:M
        for n = 1:N
            smallest_dist = 500;
            closestk = 1;
            for k = 1:K
                dist = sqrt((x(m, n, 1) - double(centroids(1,k,1)))^2 + (x(m,n,2) - double(centroids(1,k,2)))^2 + (x(m, n, 3) - double(centroids(1, k,3)))^2);
                if dist < smallest_dist
                    smallest_dist = dist;
                    closestk = k;
                end %end if
            end %centroid loop
```

```
            errorsum = errorsum + smallest_dist;
            labels(m, n) = closestk;

            counts(closestk) = counts(closestk) + 1;
            sums(1, closestk, 1) = sums(1, closestk, 1) + x(m,n,1);
            sums(1, closestk, 2) = sums(1, closestk, 2) + x(m, n, 2);
            sums(1,closestk, 3) = sums(1, closestk,3) + x(m, n,3);

            y(m, n, 1) = centroids(1, closestk, 1);
            y(m, n, 2) = centroids(1, closestk, 2);
            y(m, n, 3) = centroids(1, closestk, 3);
        end %column
    end %row
    for k = 1:K
        if counts(k) > 0
            centroids(1,k,1) = sums(1,k,1) / counts(k);
            centroids(1,k,2) = sums(1,k,2) / counts(k);
            centroids(1,k, 3) = sums(1,3,3) / counts(k);
        else
            centroids(1, k, 1) = centroids(1,k,1) / 10;
            centroids(1,k,2) = centroids(1,k,2) / 2;
            centroids(1,k,3) = centroids(1, k,3) / 3;
        end
    end
    iteration = iteration + 1;
    errorsum = errorsum / (M*N);
end
```