安徽大学《数字图像处理（双语）》实验报告（6）

学号：WA2214014 专业：人工智能 姓名及签字：杨跃浙

实验日期：2024 年 12 月 31 日 实验成绩： 教师签字：

【实验名称】： 彩色图像处理

【实验目的】：

1. 熟悉和掌握彩色图像空间
2. 通过 MATLAB 编程实现彩色图像的反色
3. 通过 MATLAB 编程实现两种常见的彩色图像分层
4. 通过 MATLAB 编程实现不同空间的彩色图像直方图均衡

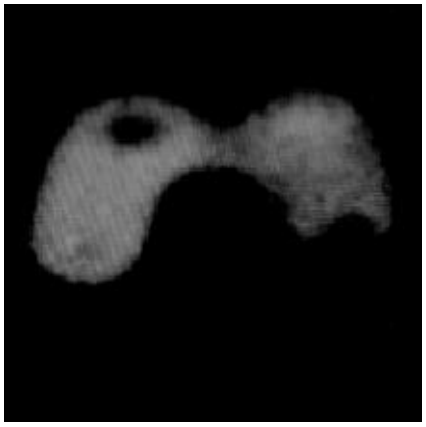
【实验内容】

PROJECT 06-03

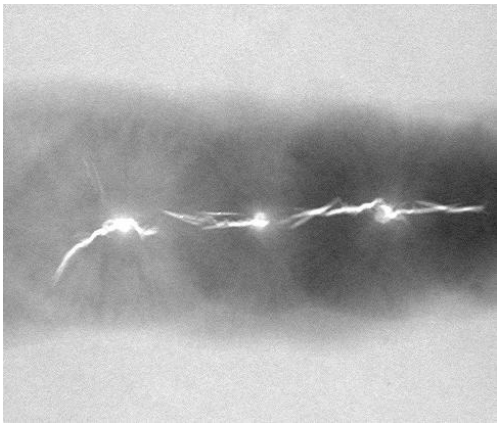
Intensity Slicing

(a) Implement intensity slicing, with the characteristic that you can specify different ranges of gray-level values for the input image and your program will output an RGB image whose pixels have a specified color. You can set the colors in color palette.

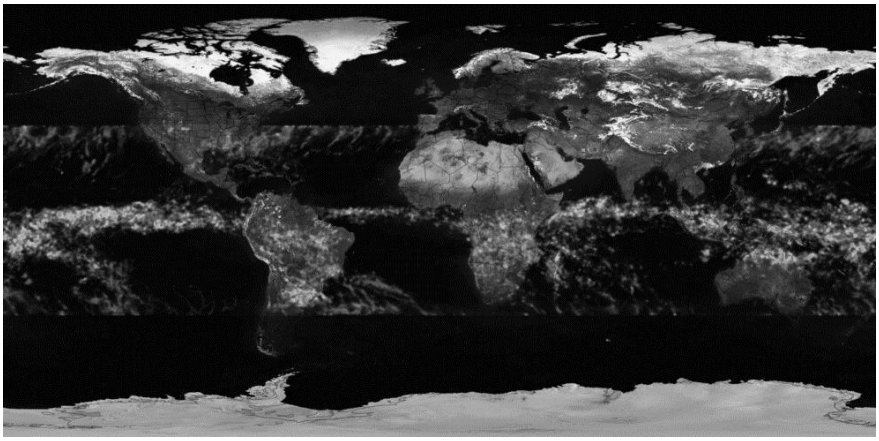
(b) Process the above images with your program with different ranges (8 for (1), 3 for (2), 20 for (3)).



(1) picker_phantom.tif



(2)weld-original.tif



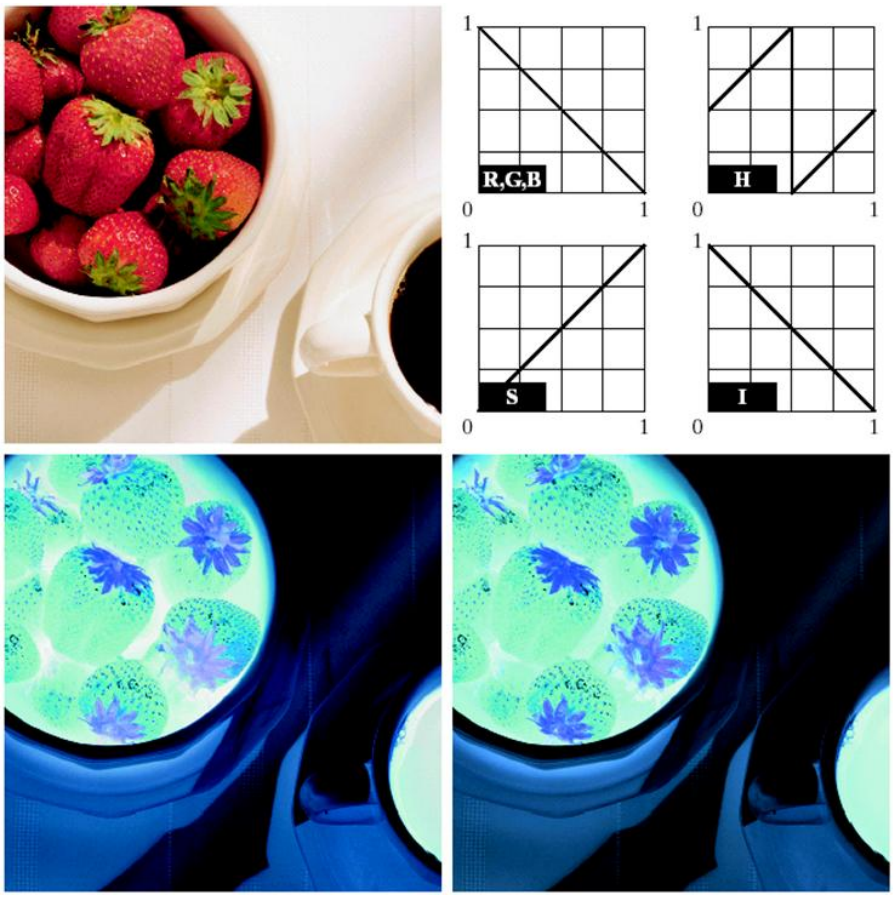
(3) tropical_rain_grayscale.tif

PROJECT 06-04

Color complement

Use the complement function (as show in following figure 6.33(b)) of R,G,B

components and H, S, I components respectively to implement the color image complement of image ‘strawberries’ .



a b
c d

FIGURE 6.33 Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

PROJECT 06-05

Color slicing

Using the following two equation to implement the color slicing on image ‘strawberries’ where n=3 denotes the Red, Green and Blue component. Set W= 0.2549, R=0.1765, a = (0.6863, 0.1608, 0.1922). Pixels outside the cube and sphere were replaced by color (0.5, 0.5, 0.5)

$$s_i = \begin{cases} 0.5 & \text{if } \left[|r_i - a_j| > \frac{W}{2} \right]_{\text{any } 1 \le j \le n} \\ r_i & \text{otherwise} \end{cases}, \qquad i = 1, 2, \ldots, n. \quad (6.5\text{-}7)$$

$$s_i = \begin{cases} 0.5 & \text{if } \sum_{j=1}^{n} (r_j - a_j)^2 > R_0^2 \\ r_i & \text{otherwise} \end{cases}, \qquad i = 1, 2, \ldots, n. \quad (6.5\text{-}8)$$

PROJECT 06-06

Histogram Processing on Color Images

The following image "Fig0637(a)(caster_stand_original)" is a color image of a caster stand containing cruets and shakers.

(a) Demonstrate the histogram of its intensity component

(b) Implement the Histogram Equalizing on the intensity component, without altering the hue and saturation. Comparing the result to the original image.

(c) Further increase the image's saturation component and compare the result to (b)

(d) Demonstrate the histogram of its intensity component of the processed image after (c) and compare the distribution to (a).

【实验代码和结果】

Project06-03:

Code:

intensity_slicing.m

```
function slicedRGB = intensity_slicing(grayImage, numSlice,
colorMap)
grayDouble = double(grayImage);

% Define intensity thresholds for slicing
grayMin = min(grayDouble(:));
grayMax = max(grayDouble(:));
thresholds = linspace(grayMin, grayMax, numSlice + 1);

% Initialize RGB output image
[rows, cols] = size(grayImage);
slicedRGB = zeros(rows, cols, 3);

% Map each intensity range to corresponding color
for i = 1:numSlice
mask = (grayDouble >= thresholds(i)) & (grayDouble <
thresholds(i + 1));
for c = 1:3
slicedRGB(:, :, c) = slicedRGB(:, :, c) + mask * colorMap(i,
c);
end
end
end
```

Project1.m:

```
% Define input images and their respective slice numbers
imageFiles = {
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/IMAGES/picker_phantom.tif', 8;
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/IMAGES/weld-original.tif', 3;
```

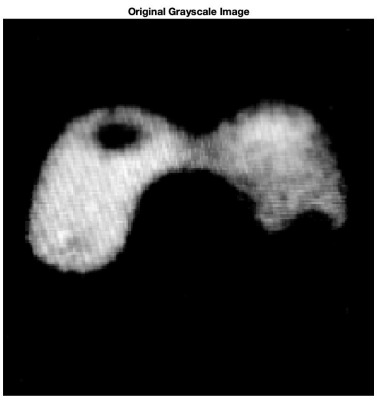
```
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image
Processing/Class6/IMAGES/tropical_rain_grayscale.tif', 20
};

% Define the output folder
outputFolder =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/Figure';
% Process each image
for i = 1:size(imageFiles, 1)
% Load the current image and slice number
imgFile = imageFiles{i, 1};
numSlice = imageFiles{i, 2};
grayImage = imread(imgFile);

% Generate a colormap for the given slice number
colorMap = parula(numSlice);
slicedRGB = intensity_slicing(grayImage, numSlice,
colorMap);

% Display results side-by-side
figure('Name', ['Results for ', imgFile], 'Position', [100,
100, 1200, 500]);

subplot(1, 2, 1);
imshow(grayImage, []);
title(['Original Grayscale Image']);

subplot(1, 2, 2);
imshow(slicedRGB);
title(['Intensity Sliced (', num2str(numSlice), '
slices)']);

% Save the results
[~, imgName, ~] = fileparts(imgFile);
saveFigurePath = fullfile(outputFolder, [imgName,
'_comparison.png']);
saveas(gcf, saveFigurePath);
```

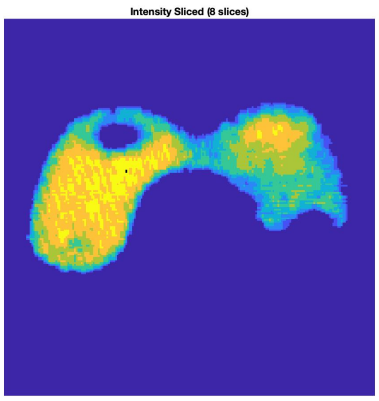
```
saveImagePath = fullfile(outputFolder, [imgName,
'_sliced.png']);
imwrite(slicedRGB, saveImagePath);

% Close the figure
close(gcf);
end
```

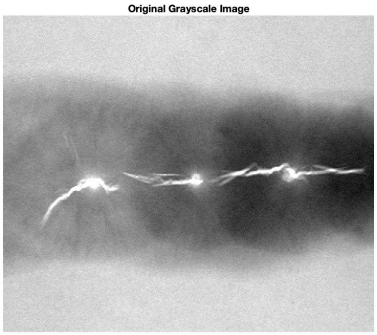
Result:

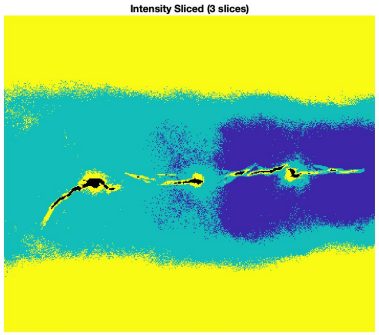
选择不同的 colorMap 函数会有不同的结果：

当选择 parula() 时，结果为：

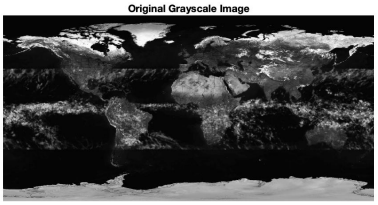




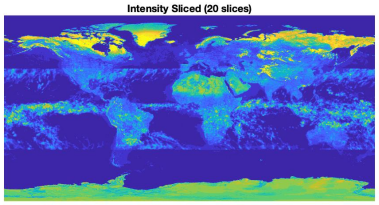


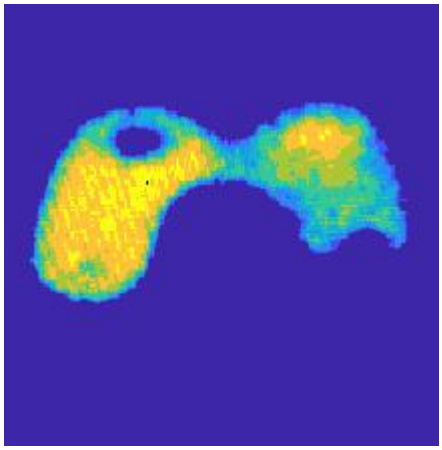


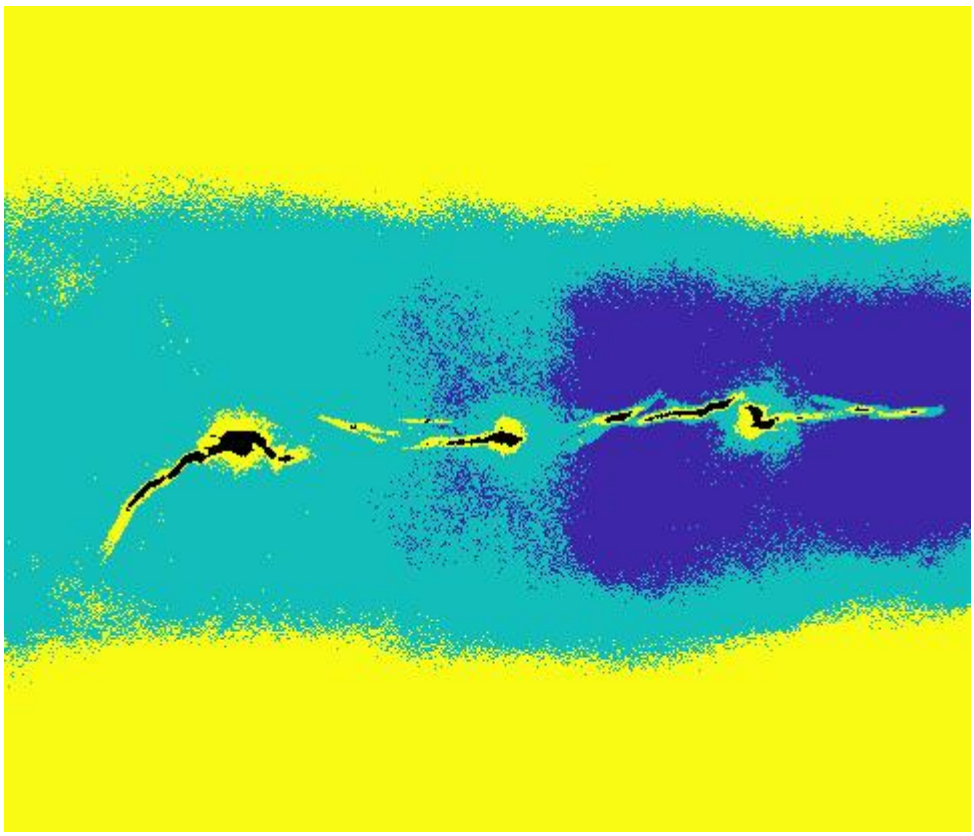
Original Grayscale Image

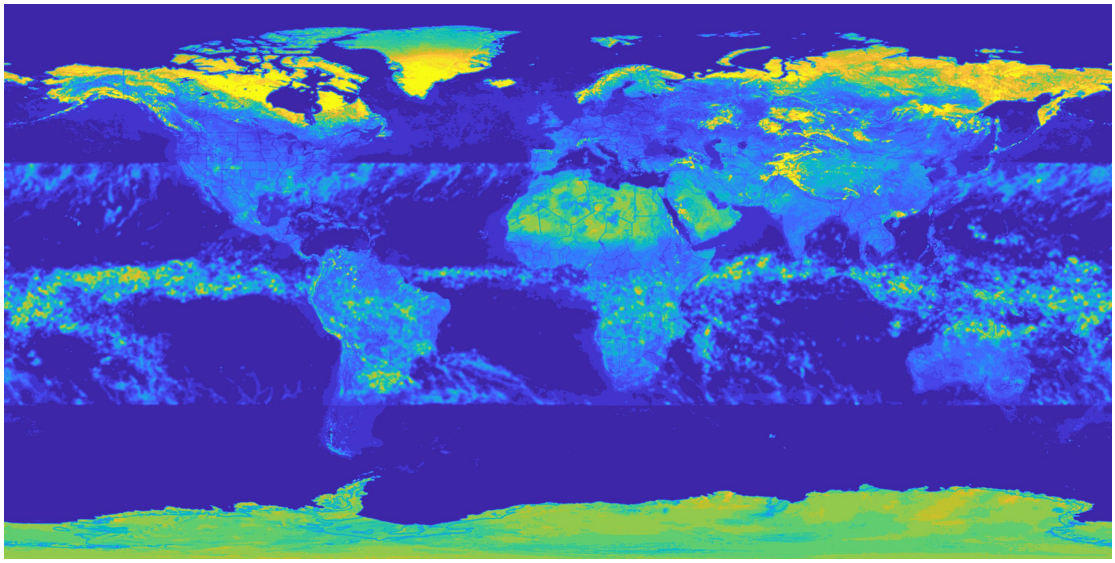


Intensity Sliced (20 slices)

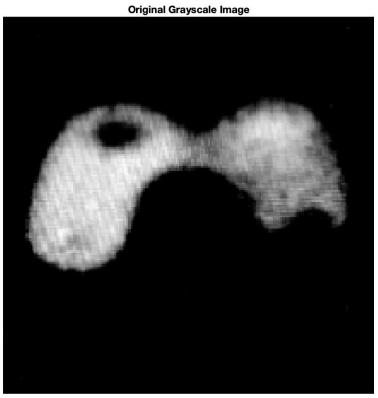


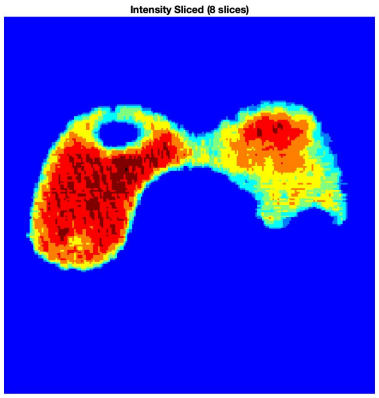


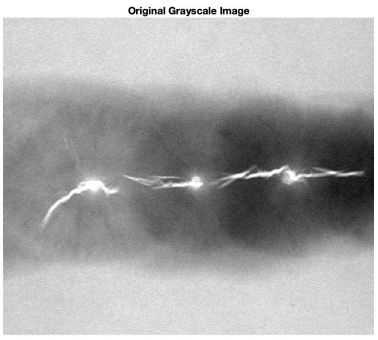


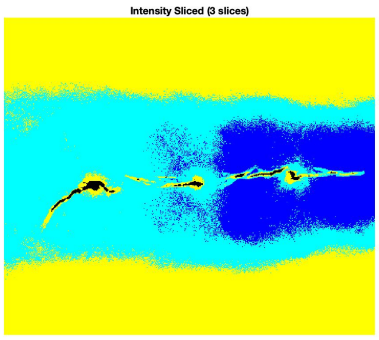


当选择为 jet（）时，结果为：

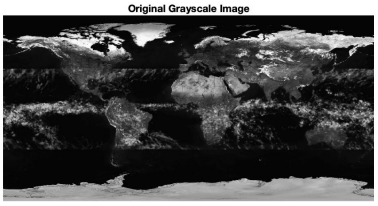




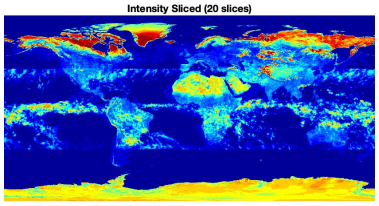


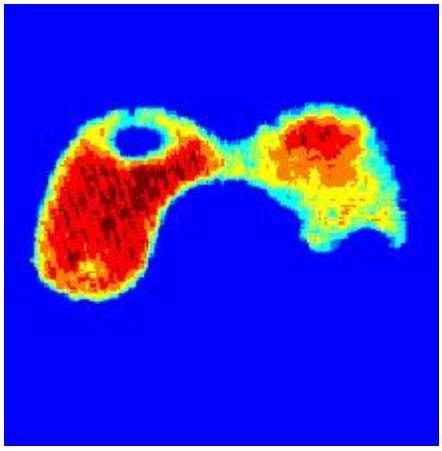


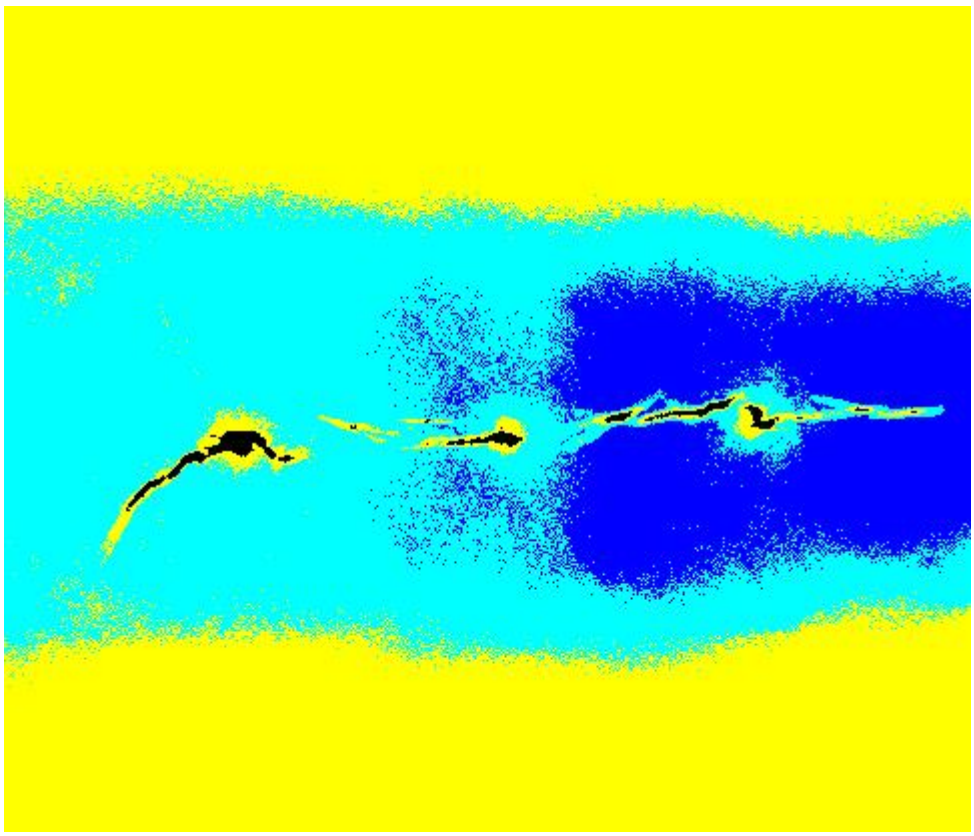
Original Grayscale Image

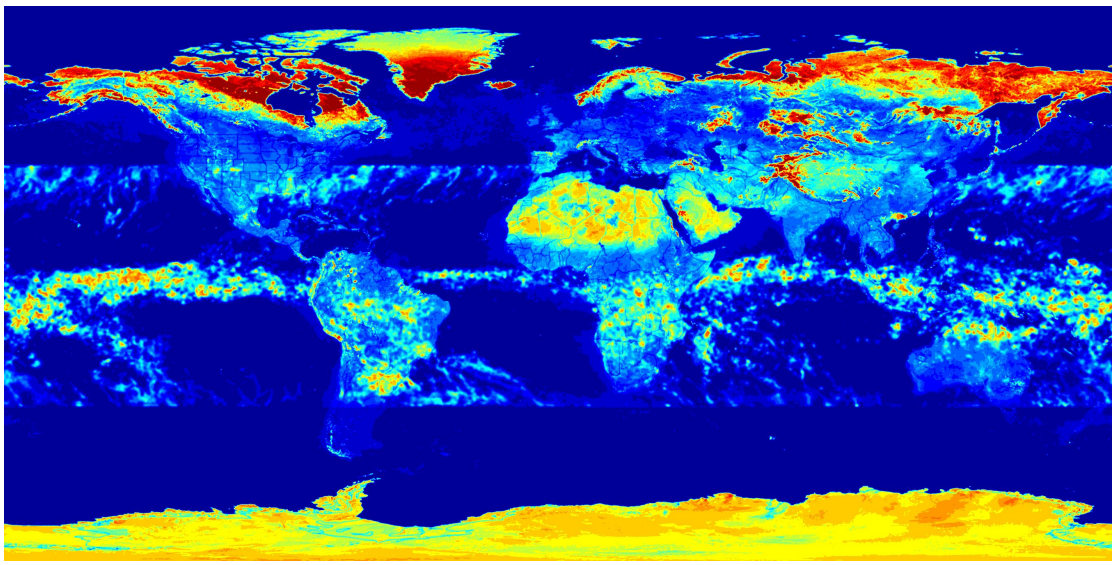


Intensity Sliced (20 slices)









我在实验过程中，选择了其他的 colorMap 函数，不同的 colorMap 会有相差极大的结果，但是仍然没有很理想的结果，书本中可能用了比较复杂的映射函数才能达到比较好的效果。

Project06-04：

Code:

hsi2rgb.m:

```
function rgbImage = hsi2rgb(hsiImage)
% Convert an HSI image to RGB color space.
H = hsiImage(:, :, 1) * 360; % Convert to degrees
S = hsiImage(:, :, 2);
I = hsiImage(:, :, 3);

% Initialize
[rows, cols] = size(H);
R = zeros(rows, cols);
G = zeros(rows, cols);
B = zeros(rows, cols);

% H in [0, 120)
idx = (H >= 0 & H < 120);
```

```
B(idx) = I(idx) .* (1 - S(idx));
R(idx) = I(idx) .* (1 + S(idx) .* cosd(H(idx)) ./ cosd(60 -
H(idx)));
G(idx) = 3 * I(idx) - (R(idx) + B(idx));

% H in [120, 240)
idx = (H >= 120 & H < 240);
H(idx) = H(idx) - 120;
R(idx) = I(idx) .* (1 - S(idx));
G(idx) = I(idx) .* (1 + S(idx) .* cosd(H(idx)) ./ cosd(60 -
H(idx)));
B(idx) = 3 * I(idx) - (R(idx) + G(idx));

% H in [240, 360)
idx = (H >= 240 & H <= 360);
H(idx) = H(idx) - 240;
G(idx) = I(idx) .* (1 - S(idx));
B(idx) = I(idx) .* (1 + S(idx) .* cosd(H(idx)) ./ cosd(60 -
H(idx)));
R(idx) = 3 * I(idx) - (G(idx) + B(idx));

% Combine
rgbImage = cat(3, R, G, B);
rgbImage = im2uint8(rgbImage); % Scale back to [0, 255]
end
```

rgb2hsi.m:

```
function hsiImage = rgb2hsi(rgbImage)
% Convert an RGB image to HSI color space.
rgbImage = im2double(rgbImage);
R = rgbImage(:, :, 1);
G = rgbImage(:, :, 2);
B = rgbImage(:, :, 3);

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

% Calculate Saturation
minRGB = min(cat(3, R, G, B), [], 3);
```

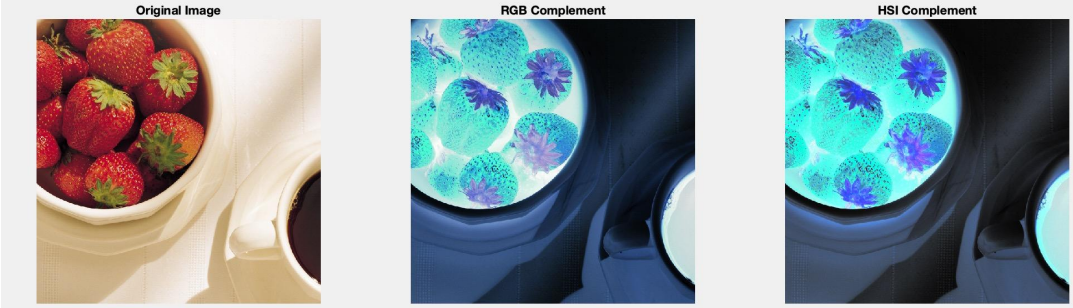
```
S = 1 - (minRGB ./ I);
S(I == 0) = 0; % Handle division by zero

% Calculate Hue
num = 0.5 * ((R - G) + (R - B));
den = sqrt((R - G).^2 + (R - B) .* (G - B));
H = acosd(num ./ (den + eps));
H(B > G) = 360 - H(B > G);
H = H / 360; % Normalize to [0, 1]

% Combine into HSI image
hsiImage = cat(3, H, S, I);
end
```

Project2.m:

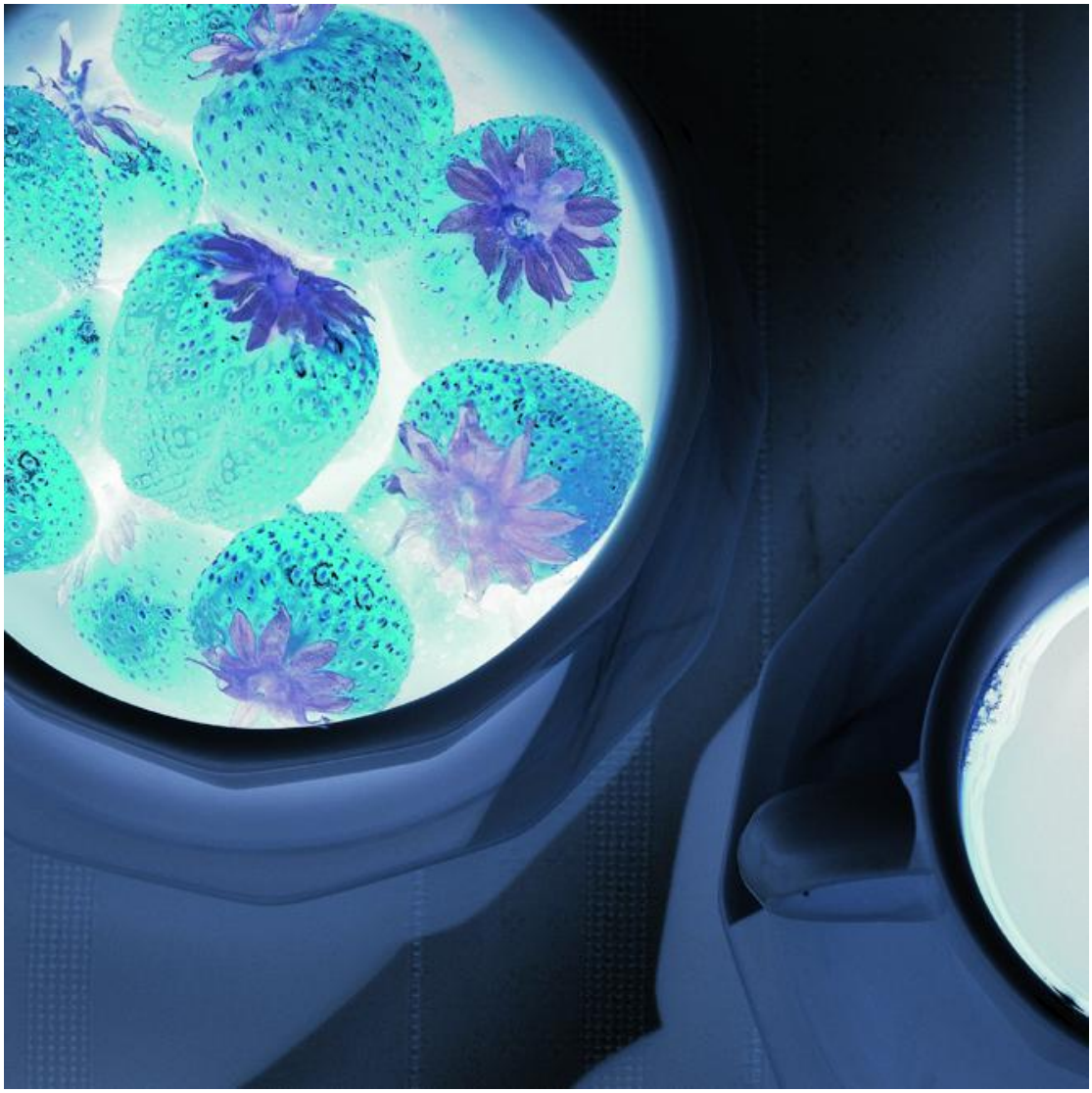
```
inputImageFile =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/IMAGES/Fig_strawberries.tif';
outputFolder =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/Figure';

% Read the input image
rgbImage = imread(inputImageFile);

% RGB Complement
rgbComplement = 255 - rgbImage;

% Convert RGB to HSI
hsiImage = rgb2hsi(rgbImage);

% HSI Complement
% Complement Hue by shifting 180 degrees (0.5 in normalized
range)
hsiComplement = hsiImage;
hsiComplement(:, :, 1) = mod(hsiComplement(:, :, 1) + 0.5,
1); % Hue
hsiComplement(:, :, 2) = hsiComplement(:, :, 2); % Saturation
```

```
hsiComplement(:, :, 3) = 1 - hsiComplement(:, :, 3); %
Intensity

% Convert back to RGB
hsiComplementRGB = hsi2rgb(hsiComplement);

% Display Results
figure('Name', 'Color Complement Results', 'Position', [100,
100, 1200, 800]);

% Original Image
subplot(1, 3, 1);
imshow(rgbImage);
title('Original Image');

% RGB Complement
subplot(1, 3, 2);
imshow(rgbComplement);
title('RGB Complement');

% HSI Complement
subplot(1, 3, 3);
imshow(hsiComplementRGB);
title('HSI Complement');

% Save Results
imwrite(rgbComplement, fullfile(outputFolder,
'strawberries_rgb_complement.jpg'));
imwrite(hsiComplementRGB, fullfile(outputFolder,
'strawberries_hsi_complement.jpg'));
```

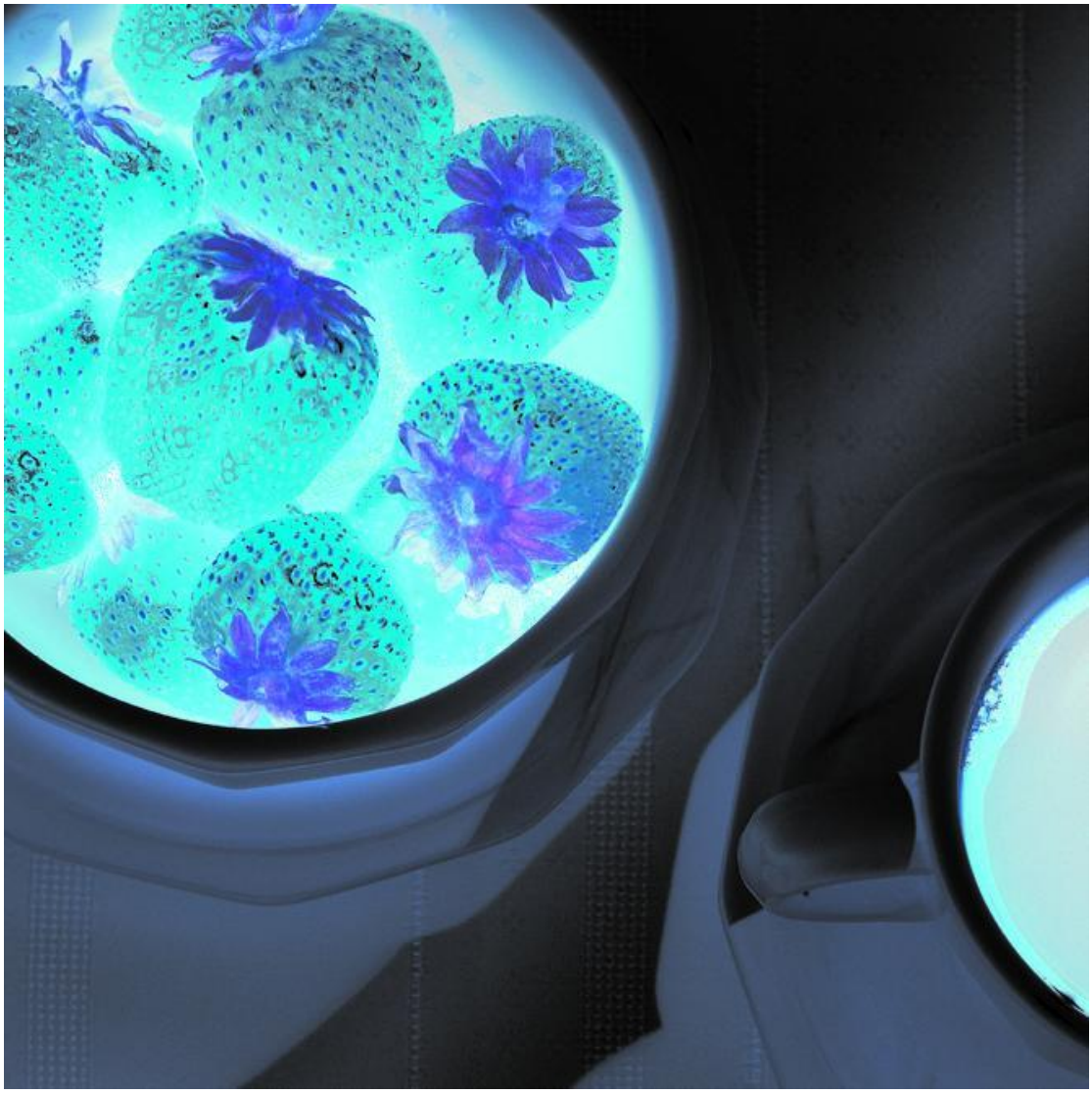
Result:



RGB:



HSI:



Project06-05:

Code:

Project3.m:

```
I =
imread('/Users/youngbean/Documents/Github/Misc-Projects/D
igital Image
Processing/Class6/IMAGES/Fig_strawberries.tif');
I = im2double(I);

% 定义输出文件夹
outputFolder =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/Figure';
```

```
% 2. 定义参数
% 目标中心颜色 a
a = [0.6863, 0.1608, 0.1922]; % (R,G,B)
% 立方体半边长 W/2
W = 0.2549; % 这里指的是整个立方体的边长
% 球体半径
R0 = 0.1765;
% 替换色
outColor = [0.5, 0.5, 0.5];

% 拆出 R/G/B 通道
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);

% 3. 立方体切片（公式 6.5-7）
% 若对任何通道 i 有 |rᵢ − aᵢ| > W/2，则该像素“在立方体外”
maskCube = (abs(R - a(1)) > W/2) | ...
(abs(G - a(2)) > W/2) | ...
(abs(B - a(3)) > W/2);

% 将立方体外的像素替换成 outColor(0.5,0.5,0.5)
I_cube = I; % 复制原图
for c = 1:3
channelC = I_cube(:,:,c);
channelC(maskCube) = outColor(c);
I_cube(:,:,c) = channelC;
end

% 4. 球体切片（公式 6.5-8）
% 若 (R − a_R)^2 + (G − a_G)^2 + (B − a_B)^2 > R0^2，则像素“在球体外”
distSq = (R - a(1)).^2 + (G - a(2)).^2 + (B - a(3)).^2;
maskSphere = distSq > R0^2;

% 将球体外的像素替换成 outColor
I_sphere = I;
for c = 1:3
```

```
channelC = I_sphere(:,:,c);
channelC(maskSphere) = outColor(c);
I_sphere(:,:,c) = channelC;
end

% 5. 显示并保存结果
figure('Name','Color Slicing
Demo','Position',[100,100,1500,400]);
subplot(1,3,1); imshow(I); title('Original');
subplot(1,3,2); imshow(I_cube); title('Cube-based
Slicing');
subplot(1,3,3); imshow(I_sphere); title('Sphere-based
Slicing');

outCube = fullfile(outputFolder, 'strawberries_cube.png');
outSphere = fullfile(outputFolder,
'strawberries_sphere.png');

% 保存图像
imwrite(I_cube, outCube);
imwrite(I_sphere, outSphere);

disp('Color slicing demo finished. Results saved to:');
disp(outCube);
disp(outSphere);
```

Result:

cube:

sphere:

Project06-06:

Code:

Project4.m:

```
% 设置输入、输出路径
imgFile =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image
Processing/Class6/IMAGES/Fig0637(a)(caster_stand_original)
.tif';
outputFolder =
'/Users/youngbean/Documents/Github/Misc-Projects/Digital
Image Processing/Class6/Figure';
```

```
% 读入图像并转 double [0,1]
I_rgb = imread(imgFile);
I_rgb = im2double(I_rgb);

% 转到 HSV，获取 H/S/V
I_hsv = rgb2hsv(I_rgb);
H = I_hsv(:,:,1);
S = I_hsv(:,:,2);
V = I_hsv(:,:,3);

% (a) 显示 原图 V 分量直方图
fig_a = figure('Name','(a) Original Intensity (Value)
Histogram','Position',[50,200,1200,400]);
subplot(1,2,1);
imshow(I_rgb);
title('Original RGB Image');

subplot(1,2,2);
imhist(V);
title('Histogram of Original Intensity (V)');

% 保存 (a)
saveas(fig_a, fullfile(outputFolder,
'Fig_a_Original.png'));
close(fig_a);

% (b) 对 V 分量做直方图均衡化，不改 H 和 S
V_eq = histeq(V); % 对亮度分量做直方图均衡化
I_hsv_eq = cat(3, H, S, V_eq);
I_rgb_eq = hsv2rgb(I_hsv_eq);

% 显示均衡化结果及直方图
fig_b = figure('Name','(b) Histogram Equalization on
V','Position',[100,200,1200,400]);
subplot(1,2,1);
imshow(I_rgb_eq);
title('Histogram Equalized (V only)');
```

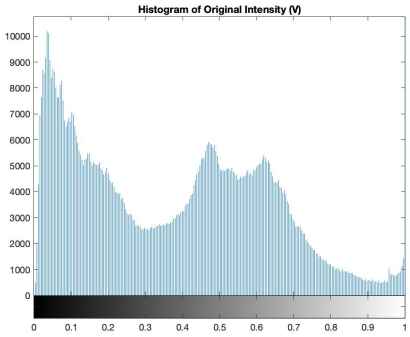
```
subplot(1,2,2);
imhist(V_eq);
title('Histogram of Equalized V');

saveas(fig_b, fullfile(outputFolder,
'Fig_b_EqualizedV.png'));
close(fig_b);

% 对比（b）的结果与原图
fig_b_cmp = figure('Name','(b) Compare Original and
Equalized','Position',[150,250,1200,400]);
subplot(1,2,1);
imshow(I_rgb);
title('Original RGB');

subplot(1,2,2);
imshow(I_rgb_eq);
title('Equalized (V)');

saveas(fig_b_cmp, fullfile(outputFolder,
'Fig_b_Compare.png'));
close(fig_b_cmp);

% （c）在（b）的基础上，“增加饱和度 S”
% 适度提高（比如 +30%），并截断到最大值 1
S_boost = 1.3 * S;
S_boost(S_boost > 1) = 1;

% 保持 V_eq 不变，组合新的 HSV
I_hsv_boosted = cat(3, H, S_boost, V_eq);
I_rgb_boosted = hsv2rgb(I_hsv_boosted);

% 显示结果
fig_c = figure('Name','(c) Increased
Saturation','Position',[200,300,1200,400]);
subplot(1,2,1);
imshow(I_rgb_boosted);
title('Equalized(V) + Boosted(S)');
```

```
subplot(1,2,2);
imhist(I_hsv_boosted(:,:,3));
title('Histogram of V after Saturation Boost (same as eq)');

saveas(fig_c, fullfile(outputFolder,
'Fig_c_BoostedS.png'));
close(fig_c);

% 与 (b) 只做 V_eq 的结果进行对比
fig_c_cmp = figure('Name','(c) Compare with
(b)','Position',[250,350,1200,400]);
subplot(1,2,1);
imshow(I_rgb_eq);
title('From (b) : V_{eq}');

subplot(1,2,2);
imshow(I_rgb_boosted);
title('From (c) : V_{eq} + S_{boost}');

saveas(fig_c_cmp, fullfile(outputFolder,
'Fig_c_Compare.png'));
close(fig_c_cmp);

% (d) 显示 (c) 最终图像亮度分量直方图，并与 (a) 对比
V_after_c = I_hsv_boosted(:,:,3); % 仍然是 V_eq

fig_d = figure('Name','(d) Compare final V histogram vs.
original','Position',[300,400,1200,400]);
subplot(1,2,1);
imhist(V_after_c);
title('Histogram of final image (V)');

subplot(1,2,2);
imhist(V);
title('Histogram of original image (V)');

saveas(fig_d, fullfile(outputFolder,
'Fig_d_FinalHistogram.png'));
close(fig_d);
```

Result:

(a):


Original RGB Image

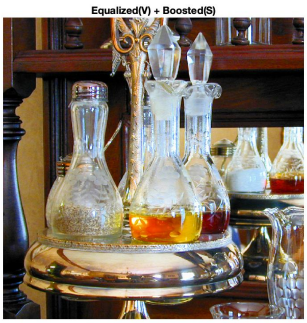

Histogram of Original Intensity (V)
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

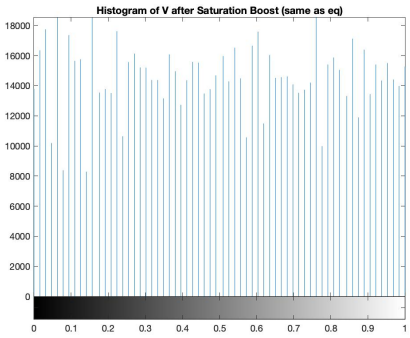
(b):

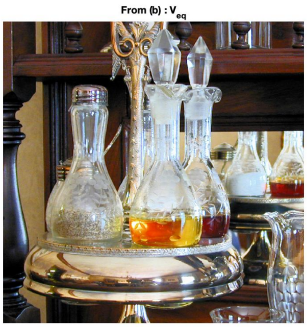

Original RGB

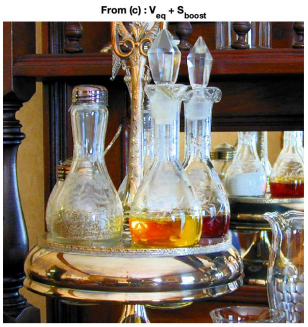

Equalized (V)

(c):

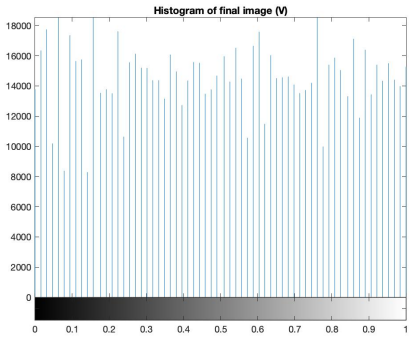

Equalized(V) + Boosted(S)

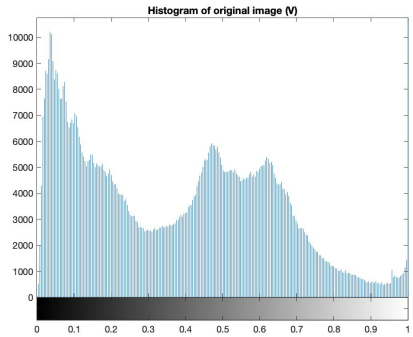

Histogram of V after Saturation Boost (same as eq)
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


From (b) : V_eq


From (c) : V_eq + S_boost

(d):





【小结或讨论】

我在本次实验中先后完成了多个项目，涵盖了灰度级切片，彩色图像反色，彩色图像分层以及彩色图像的直方图均衡等内容，每个项目都给了我对于数字图像处理更深层次的理解。

在 Project06 – 03 中，我通过编写 intensity _ slicing. m 和相应的主脚本，实现了对灰度图像进行强度切片（Intensity Slicing）。在程序中，我先设定若干个阈值区间 $\{[T_0, T_1), [T_1, T_2), \cdots\}$，其中

$$T_k = \text{linspace}\ (\min(I), \max(I), N + 1)$$

再将每个区间映射到特定的颜色。这样，对于灰度范围内的不同区间，就能用不同的 RGB 色块来表现，从而直观地突出图像的强度分层。实测结果表明，无论选择 parula 还是 jet 之类的色带，都可以让图像在分段时展现出不同的伪彩形式。

之后，在 Project06 – 04 中，我使用了 " 颜色反转（Color Complement） " 的思路来处理彩色图像 " strawberries "，并分别在 RGB 和 HSI 空间实现了反色效果。对于 RGB 反色，我在像素级应用了

$$R' = 255 - R,\ G' = 255 - G,\ B' = 255 - B$$

从而得到对立的红，绿，蓝值。而在 HSI 空间里，我采用了将色调 H 加 0.5 （即 $180°$）并对亮度 I 做 $I' = 1 - I$ 的方式来达到反转目的，从而可以更加灵活地控制哪些分量进行变换。结果显示，RGB 反色和 HSI 反色侧重不同：RGB 反色更直接，HSI 反色则能够呈现更贴近人眼视觉感知的差异。

在 Project06 – 05 中，我通过（6．5 – 7）和（6．5 – 8）这两条公式完成了＂颜色切片（Color Slicing）＂，同样以＂strawberries＂图为例。具体来说，我先定义一个目标颜色 $a = (a_R, a_G, a_B)$ 及立方体宽度 W 和球半径 R_0 。当像素 $p = (r, g, b)$ 与中心 a 的差值满足

$$\max(|r - a_R|, |g - a_G|, |b - a_B|) > \frac{W}{2}$$

就认为它位于立方体区域之外，用 $(0.5,0.5,0.5)$ 灰色替换；或者通过

$$(r - a_R)^2 + (g - a_G)^2 + (b - a_B)^2 > R_0^2$$

来判断是否在球面之外，也用灰色替换。这样一来，就能高亮显示我所希望的颜色范围，而排除其他与目标色相差较大的区域。在实际操作中，我明显感受到，立方体切片更适合捕捉单通道有较大波动的像素，而球体切片则更能保留与目标颜色＂整体距离＂接近的像素。

最后，在 Project06 – 06 中，我尝试了对彩色图像进行直方图均衡化，并仅对亮度 (Value) 分量做处理。具体来说，我先把图像从 RGB 转到 HSV 空间，记为 (H, S, V) ，然后在 V 通道上应用

$$V_{eq} = \text{histeq}\ (V)$$

得到均衡化后的亮度。这样可以在增强图像对比度的同时，避免对原图的色调（Hue）和饱和度 （Saturation）产生影响。接着，我又对饱和度 S 进行了适当的提升，如 $S_{boost} = \min(1.3 \times S, 1)$ ，使图像色彩更鲜艳，视觉冲击力更强。比较前后图像的亮度直方图，我看到均衡化后分布更加均匀，也进一步证实了直方图均衡化在增强图像细节方面的作用。

通过这几个项目的综合练习，我对于彩色图像空间（RGB，HSI，HSV）的特性有了更直观的认识，并且熟悉了许多基本操作，包括＂反色＂这种对通道做线性或非线性变换的方法，＂强度切片＂与＂颜色切片＂的区间映射思想，以及对彩色图像只调整亮度分量或其他分量时所带来的不同效果。整个实验过程让我掌握了 MATLAB 在图像处理领域的快速实现能力，也让我学会了更多底层图像处理的思路与技巧。