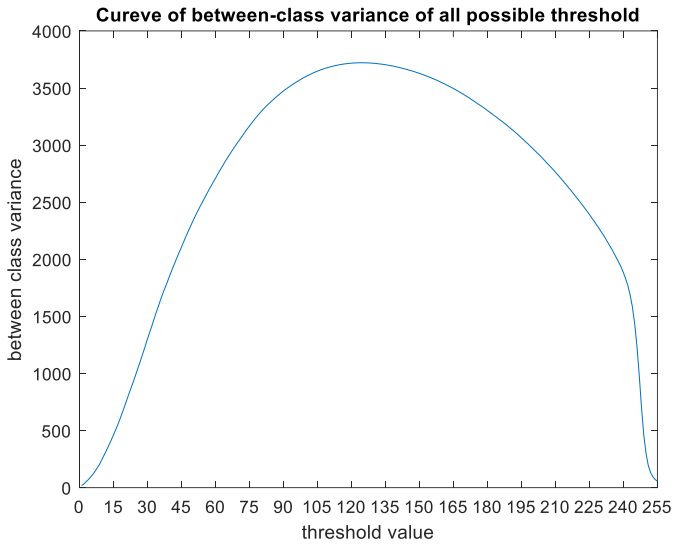
Project 6 solution

1. Plot of the curve of between-class variance depending on all possible threshold values (20%)



➔ Otsu threshold k=124.

2. Plot of the curve of between-class variance depending on all possible threshold values (20%)

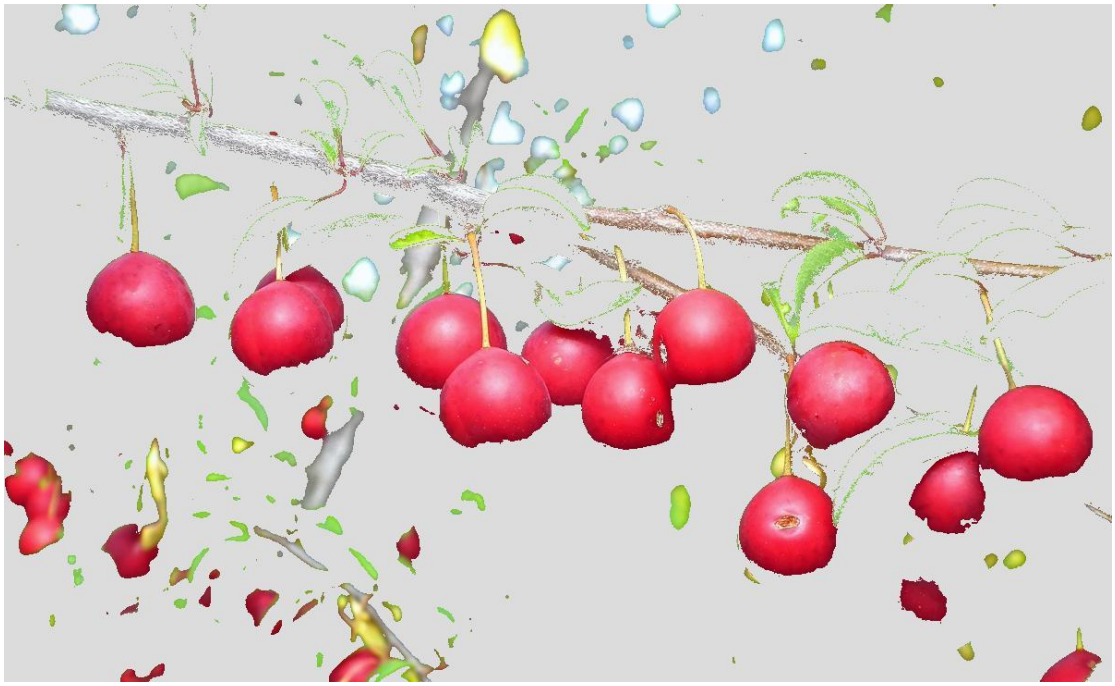


Figure 1. Image of patterns extracted by Otsu's algorithm (threshold k=124)

3. Images of patterns extracted by K-means clustering with different threshold values (30%)

Figure 2. Images of patterns extracted by K-means clustering with T =10

Figure 3. Images of patterns extracted by K-means clustering with T =5

Figure 4. Images of patterns extracted by K-means clustering with T =1

4. Source code:

```
clear
close all
%% Otsu algorithm
img = imread('fruit on tree.tif');
figure
imshow(img)

img = double(img);
r = img(:,:,1);
M = size(img,1);
N = size(img,2);

gray_level = [0:1:255].';      % gray level 0~255
count = imhist(uint8(r));
pdf = count/(M*N);
mG = sum(gray_level' .* pdf);  % global average
for k = 0:255                  % threshold 0~255
```

```
    P1 = sum(pdf(1:k+1));
    P2 = 1 - P1;
    m1 = (1/P1) .* sum(gray_level(1:k+1).*pdf(1:k+1));
    m2 = (1/P2) .* sum(gray_level(k+2:end).*pdf(k+2:end));
    sigma2_B(k+1) = P1 * P2 * (m1 - m2) .^2;
end
[max_k index] = max(sigma2_B);
optimal_k = index - 1;

% Plot of the curve of between-class variance depending on all
possible threshold values
figure
plot(sigma2_B)
ylabel('between class variance')
xlabel('threshold value')
title('Cureve of between-class variance of all possible threshold')
xlim([0 255])
set(gca,'XTick',[0:15:255])

final_img_binary = zeros(M,N);
for ii = 1:M
    for jj = 1:N
        if img(ii,jj) > optimal_k
            final_img_binary(ii,jj) = 1;
        else
            final_img_binary(ii,jj) = 0;
        end
    end
end

final_img = zeros(M,N,3);
for ii = 1:M
    for jj = 1:N
        if img(ii,jj) > optimal_k
            final_img(ii,jj,:) = img(ii,jj,:);
        else
            final_img(ii,jj,:) = [220 220 220];
        end
```

```
    end
end

% binary image
figure
imshow(uint8(final_img_binary),[])
title('Otsu algorithm (binary image)')

% Plotted in the same way as the colorslicing
figure
imshow(uint8(final_img),[])
title('Otsu algorithm')


%% K-means clustering
clear

img = imread('fruit on tree.tif');
img = double(img);

[M,N,comp] = size(img);
rgb = reshape(img, M*N, comp);

T = [1 5 10];  % Threshold values
cluster = 2;

rng('default');
rand_C = randperm(M*N, cluster);
C = rgb(rand_C, :);
C_keep = C;

iter_history = [];
norm_history = [];

for nn = 1:length(T)
    C = C_keep;
    iter = 0;
    C_prev = [];
```

```
    while true
        C_prev = C;
        iter = iter + 1;
        dis = sum(rgb.^2, 2)*ones(1, cluster) + (sum(C.^2, 2)*ones(1,
M*N))' - 2*rgb*C';
        [~, label] = min(dis, [], 2) ;
        for i = 1:cluster
            C(i, :) = mean(rgb(label == i , :));
        end

        if (norm(C(1,:)-C_prev(1,:)) + norm(C(2,:)-C_prev(2,:)))  <=
T(nn)
            break;
        end
    end
    iter_history = [iter_history iter];
    norm_history = [norm_history (norm(C(1,:)-C_prev(1,:)) +
norm(C(2,:)-C_prev(2,:)))];

    C(2,:) = [220 220 220];             % Mid-gray tone
    temp = C(label, :);
    temp(label==1, :) = rgb(label==1, :); % Original full-color
    img_seg = reshape(temp, M, N, comp);
    output = uint8(255*mat2gray(img_seg));

    figure
    imshow(uint8(img_seg),[])
    text = ['K-means clustering (T=' num2str(T(nn)) ')'];
    title(text)
end
```