CSC566

LAB 7

Name: Amir Hafizi Bin Musa
Student ID: 2024745815
Group: A4CDCS2306A

Image Segmentation using MATLAB (Clustering and Graph-based Ideas)

Tools and Files:

- Using the MATLAB Online from https://matlab.mathworks.com/ as it has the complete version of **Image Processing Toolbox**.
- Sample images used:
  - peppers.png (built-in)
  - dragonFruit.png (own-choice)

Why can image segmentation be viewed as clustering in feature space?

Image segmentation groups pixels into meaningful regions for example objects or backgrounds. Each pixel can be represented as a vector of features like RGB color values or intensity.
Clustering algorithms like k-Means group data points based on similarity, so treating pixels as feature vectors and clustering them effectively partitions the image into regions, thus segmentation becomes a clustering problem.

What are the advantages and limitations of k-Means compared to graph-based segmentation?

Advantages of k-Means:

- Simple, computationally efficient, and easy to implement.
- Works well when regions are distinct in feature space (e.g., clear color differences).

Limitations of k-Means:

- Ignores spatial proximity; may split connected regions with minor feature variations (e.g., a pepper's gradient).
- Requires predefined cluster count (K) and is sensitive to initial centroids.

Advantages of graph-based segmentation:

- Explicitly models spatial relationships (pixel adjacency), preserving object boundaries.
- Robust to feature noise (e.g., retains connected regions even with color gradients).

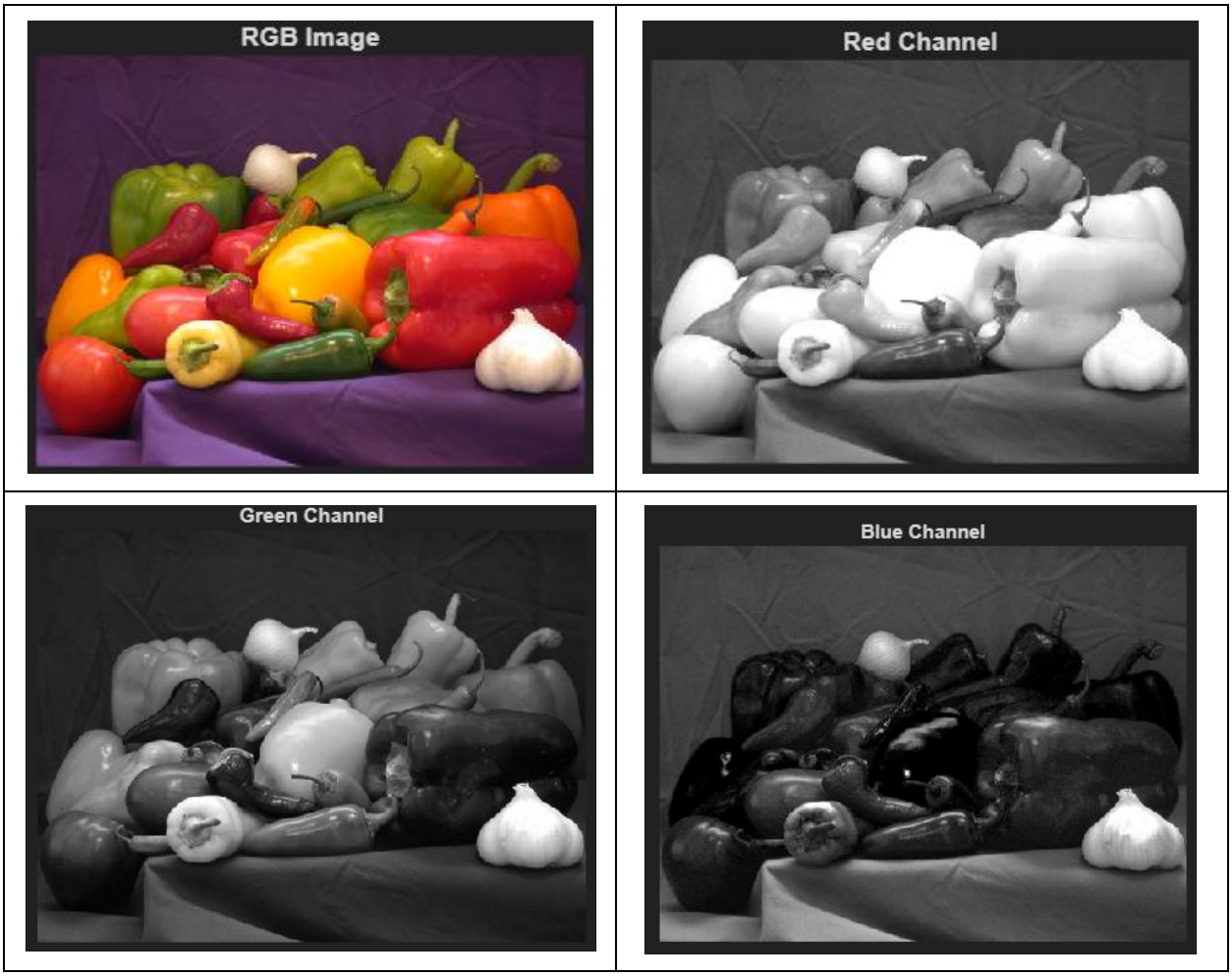
Limitations of graph-based segmentation:

- More complex (requires graph construction and optimization).
- May be slower for large images compared to k-Means.

Task 1: **Load and inspect the images**

```
% Load sample image (or replace with your own image path)
I = imread('peppers.png');
figure;
imshow(I);
title('Original Image');

% Convert to double for processing
I_d = im2double(I);

% Extract RGB channels
R = I_d(:,:,1);
G = I_d(:,:,2);
B = I_d(:,:,3);

% Display RGB image and individual channels
figure;
subplot(2,2,1); imshow(I_d); title('RGB Image');
subplot(2,2,2); imshow(R); title('Red Channel');
subplot(2,2,3); imshow(G); title('Green Channel');
subplot(2,2,4); imshow(B); title('Blue Channel');

% Ensure grayscale for channels
colormap(gray);
```

Image Result of Task 1:



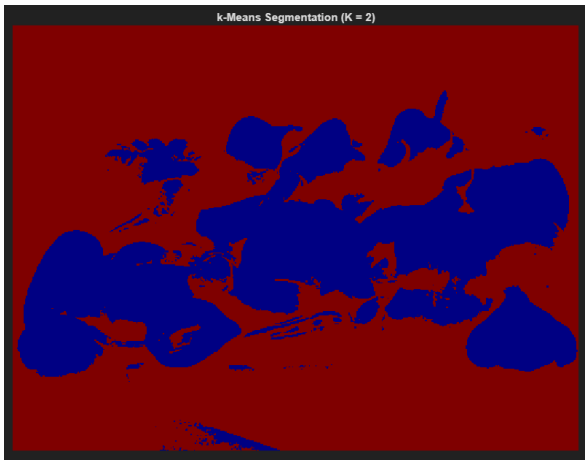
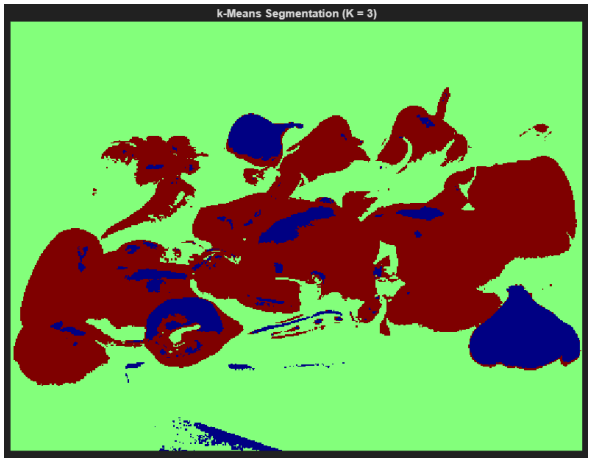
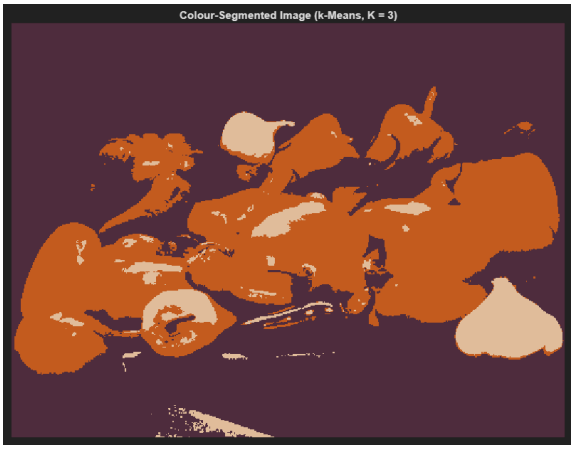
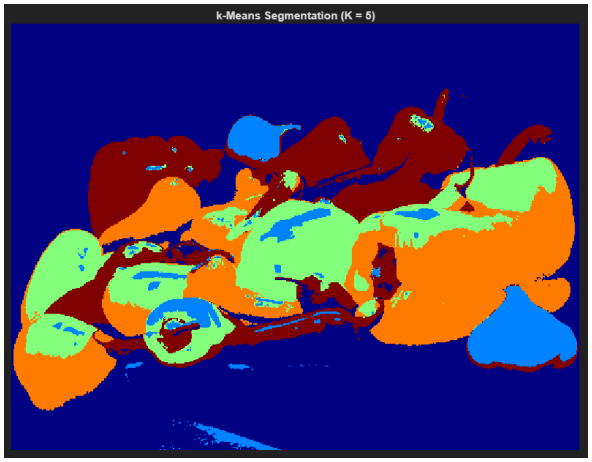
Q1: From visual inspection, what meaningful regions do you expect a good segmentation to produce for this image?

A good segmentation for the peppers image should produce regions corresponding to distinct visual elements like the red pepper, green pepper, yellow pepper, and the background. Additionally, subtle regions like highlights (bright spots) or shadows (dark areas) are separated if their colour statistics are distinct enough.

Task 2: **k-Means segmentation in RGB space**

```
% Task 2: k-Means segmentation in RGB space
% Reshape image into N x 3 feature matrix (N = total pixels)
[nRows, nCols, ~] = size(I_d);
pixels = reshape(I_d, nRows*nCols, 3);  % N x 3 matrix

% Experiment with K = 2, 3, 5
K_values = [2, 3, 5];
for K = K_values
    % Run k-Means clustering
    [idx, C] = kmeans(pixels, K, ...
                      'Distance', 'sqeuclidean', ...
                      'Replicates', 3);
% 'Replicates' to improve stability

    % Reshape cluster labels back to image size and visualize
    segmentedLabels = reshape(idx, nRows, nCols);
    figure;
    imagesc(segmentedLabels);
    axis image off;
    title(sprintf('k-Means Segmentation (K = %d)', K));
    colormap(jet);  % Use jet colormap for distinct labels

    % Create color-segmented image (replace pixels with cluster
centers)
    segmentedRGB = reshape(C(idx, :), nRows, nCols, 3);
    figure;
    imshow(segmentedRGB);
    title(sprintf('Colour-Segmented Image (k-Means, K = %d)', K));
end
```

Image Result of Task 2:

k-Means Segmentation(K=2)	Colour-Segmentated Image(k-Means, K=2)
	
k-Means Segmentation(K=3)	Colour-Segmentated Image(k-Means, K=3)
	
k-Means Segmentation(K=5)	Colour-Segmentated Image(k-Means, K=5)
	

Q2: Which value of K gives the most meaningful segmentation for this image? Why?

1. K=2 (Under-Segmentation)

- **Observation:** This setting divides the image into just two clusters. Looking at the image, we can see it essentially creates a **binary mask**: Foreground vs. Background.
- **Why it fails:** It groups all the vegetables (red peppers, yellow peppers, and garlic) into a single "object" blob. It loses all distinction between the different types of vegetables.

2. K=3 (Insufficient Detail)

- **Observation:** From the image, we see three colors. The algorithm likely separates the background, the mid-tones (shadows or darker red peppers) and the highlights.
- **Why it fails:** It still clumps distinct objects together. For example, the distinct color of the garlic (white) and the yellow peppers might be merged with the red peppers or the background depending on lighting. It does not clearly distinguish between the types of vegetables.

3. K=5 (The "Sweet Spot")

- **Observation:** Based on the image, the segmentation becomes semantically clear.
- **Why it works:** The scene appears to naturally contain about 5 dominant color groups:
    1. **Dark Background** (Dark Purple)
    2. **Red Peppers**
    3. **Yellow/Orange Peppers**
    4. **White Garlic**
    5. **Green Stems / Shadows**
- With K=5, the algorithm has enough clusters to assign a unique color to each of these distinct groups. You can clearly see the garlic is separated from the peppers, and the yellow peppers are distinguished from the red ones.

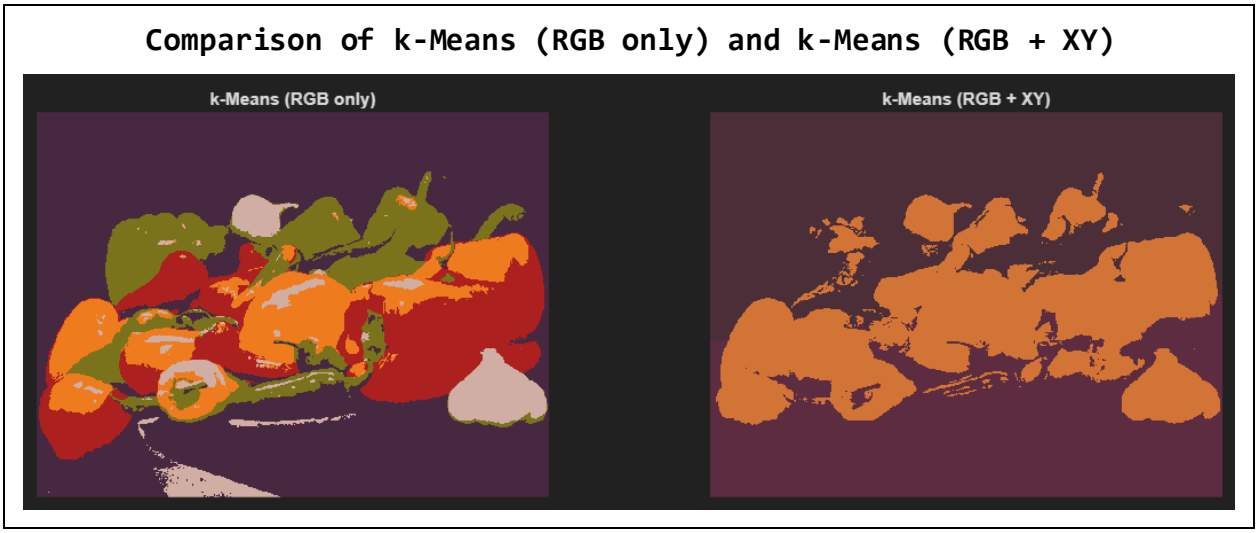
Conclusion

K=5 is the best choice because the complexity of the model (5 clusters) matches the actual **color complexity of the scene**. It avoids under-segmentation (K=2) while likely avoiding the noise that might come from over-segmentation (where a single pepper might get split into multiple unnecessary colors).

Task 3: **Adding spatial information (RGB + XY)**

```
% Task 3: k-Means with spatial (RGB + XY) features
% Generate pixel coordinates (x, y) and normalize
[xGrid, yGrid] = meshgrid(1:nCols, 1:nRows);  % xGrid (columns), yGrid
(rows)
xNorm = xGrid / nCols;  % Normalize x to [0,1]
yNorm = yGrid / nRows;  % Normalize y to [0,1]
features = [pixels, xNorm(:), yNorm(:)];  % N x 5 matrix: [R G B x y]

% Run k-Means on 5D features (use same K as Task 2, e.g., K=3)
K = 3;
[idx_xy, C_xy] = kmeans(features, K, ...
                        'Distance', 'sqeuclidean', ...
                        'Replicates', 3);

% Reshape labels and create segmented image using RGB part of centers
labels_xy = reshape(idx_xy, nRows, nCols);

% Extract RGB from cluster centers (K x 3)
segmentedRGB_xy = reshape(C_xy(:,1:3), K, 3);

% Map labels to centers
segImage_xy = reshape(segmentedRGB_xy(idx_xy, :), nRows, nCols, 3);

% Compare RGB-only vs RGB+XY results
figure;
subplot(1,2,1); imshow(segmentedRGB); title('k-Means (RGB only)');
subplot(1,2,2); imshow(segImage_xy); title('k-Means (RGB + XY)');
```

Image Result of Task 3:



Q3: Compare the two results.

1. k-Means (RGB only) - The "Semantic" Approach

- **What it sees:** This algorithm looks only at color. It doesn't care if a pixel is in the top-left corner or the bottom-right; if two pixels are the same shade of red, they belong to the same group.
- **The Result:** This provides a much better segmentation for this specific image.
    - It clearly distinguishes between the red peppers, yellow peppers, and white garlic because their colors are distinct.
    - It captures the semantic meaning of the objects (e.g., "this is a red vegetable," "this is a white vegetable").

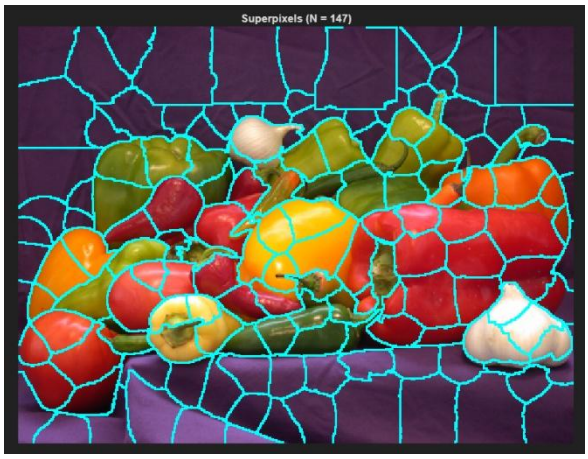
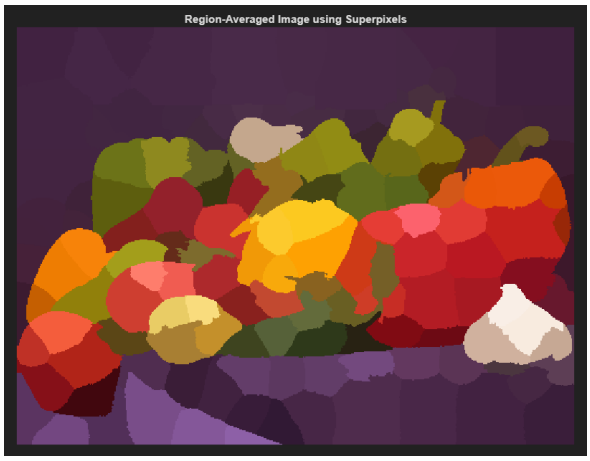
2. **k-Means (RGB + XY) - The "Spatial" Approach**

- **What it sees:** This algorithm treats the **pixel coordinates (X, Y)** as features alongside color (R, G, B). It calculates similarity based on color and physical distance. Pixels that are far apart are considered "different," even if they are the exact same color.
- **The Result:** In this specific case, the result is **worse** (or "under-segmented").
    - **The Super-Blob Effect:** Because the vegetables are all touching each other (spatially close), and the spatial coordinates (X, Y) likely have large numerical values compared to the color values, the algorithm prioritizes grouping nearby pixels together.
    - It merges distinct objects (red vs. yellow peppers) into a single foreground cluster because they are neighbors, sacrificing color accuracy for spatial coherence.

Task 4: **Simple graph-based idea using superpixels**

```
% Task 4: Superpixel segmentation (graph-based approximation)
% Generate superpixels (adjust numSuperpixels based on image
complexity)
numSuperpixels = 150;  % Start with 150; tweak for better results
[L, N] = superpixels(I_d, numSuperpixels);
% L = superpixel labels, N = actual number of superpixels

% Visualize superpixels with boundaries
BW = boundarymask(L);  % Binary mask of superpixel boundaries
figure;
imshow(imoverlay(I_d, BW, 'cyan'));
% Overlay boundaries in cyan

title(sprintf('Superpixels (N = %d)', N));

% Compute mean color per superpixel and create region-averaged image
outputImage = zeros(size(I_d));
for k = 1:N
    mask = (L == k);  % Logical mask for superpixel k
    for ch = 1:3
        channel = I_d(:,:,ch);  % Extract RGB channel
        % Mean color of the channel in superpixel k
        meanVal = mean(channel(mask));
         % Assign mean to region
        outputImage(:,:,ch) = outputImage(:,:,ch) + meanVal .* mask;
    end
end

% Display region-averaged image
figure;
imshow(outputImage);
title('Region-Averaged Image using Superpixels');
```

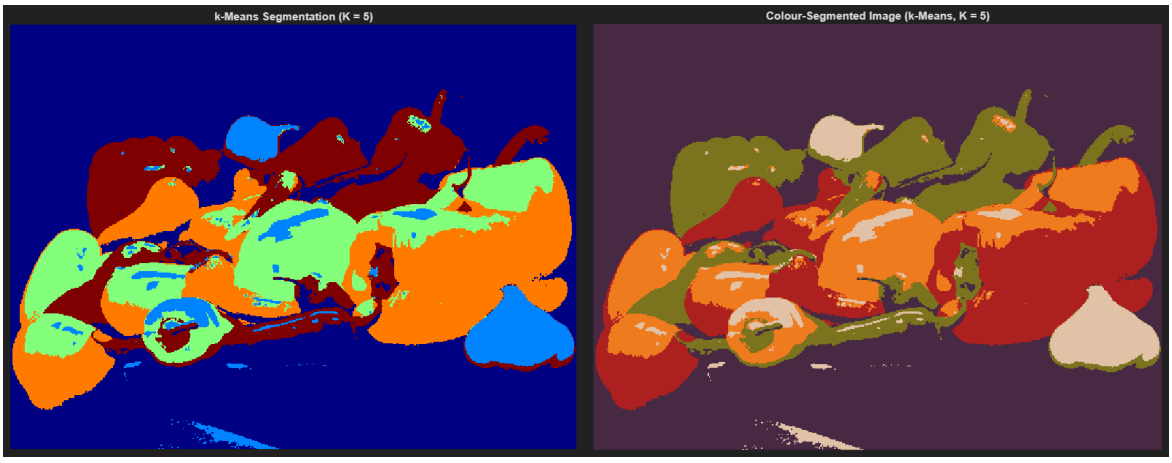
Image Result of Task 4:

Superpixels (N = 147)	Region-Averaged Image using Superpixels
	

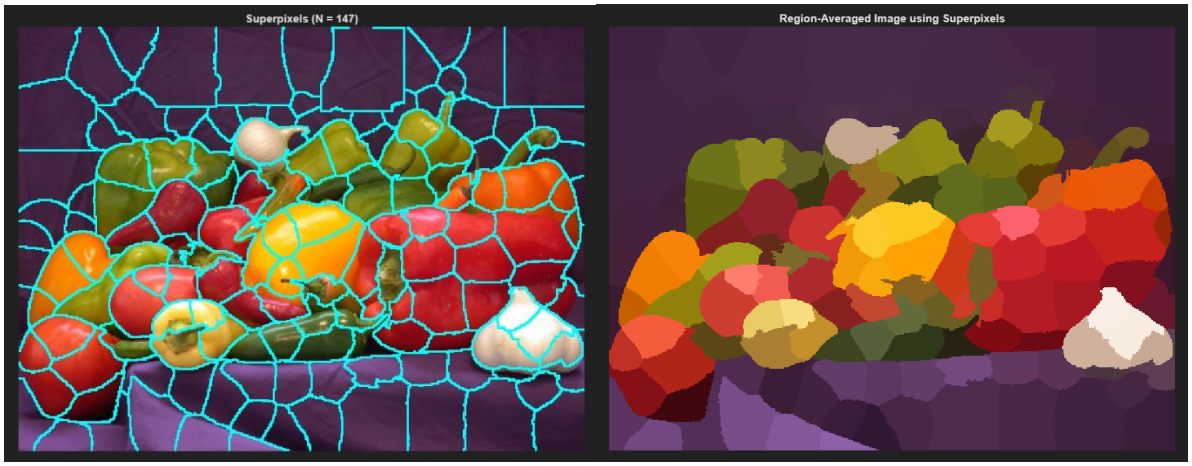
Q4: Compare the superpixel result with your best k-Means segmentation. Which one preserves object boundaries better? Give one example from your image.

The Example: The Touching Red Peppers

The clearest evidence is visible in the group of red peppers on the right side of the image.



In the k-Means(K=5) Result: The algorithm looks only at color. Since all the red peppers are roughly the same shade of red, it merges them into a single, flat red blob. The boundary between the touching peppers is completely erased, losing the information that there are multiple distinct objects there.



In the Superpixel Result: Look at the cyan boundary lines. Even though the peppers are the same color, the algorithm detects the subtle edges and shadows where one pepper ends and the next begins. The cyan lines clearly trace the individual shapes, keeping the touching peppers separated rather than fusing them together.

Superpixels are designed to over-segment an image by adhering to local image edges (gradients). This means it naturally snap to the physical outlines of objects.

- **Boundary Adherence:** The algorithm respects the local spatial relationships, creating smooth boundaries that follow the actual curves of the vegetables.
- **Separation:** It can distinguish between two distinct objects of the same color if there is even a slight edge or shadow between them.

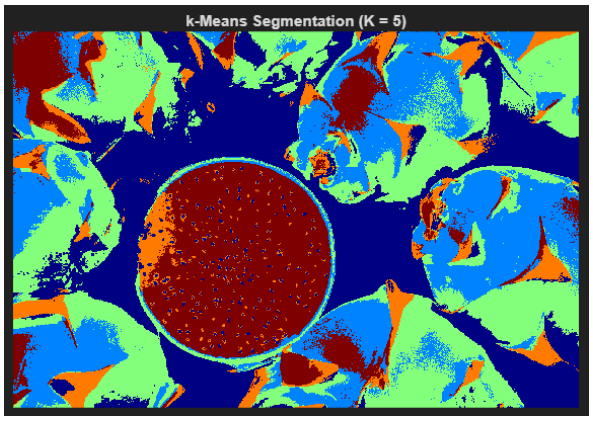
Task 5: Apply to own image (dragonFruit.png)

```
% Task 5: k-Means and superpixels on your own image
% Load your image (ensure path is correct)
own_image = im2double(imread('dragonFruit.png'));
[nRows_own, nCols_own, ~] = size(own_image);
pixels_own = reshape(own_image, nRows_own*nCols_own, 3);

% Task 2 repetition (k-Means on RGB)
K_own = 5;
[idx_own, C_own] = kmeans(pixels_own, K_own, ...
                          'Distance', 'sqeuclidean', ...
                          'Replicates', 3);
segmentedLabels_own = reshape(idx_own, nRows_own, nCols_own);
segmentedRGB_own = reshape(C_own(idx_own, :), nRows_own, nCols_own, 3);
figure;
subplot(1,2,1); imshow(own_image); title('Original (Own Image)');
subplot(1,2,2); imshow(segmentedRGB_own); title(sprintf('k-Means (K = %d)',
K_own));

% Task 4 repetition (superpixels on own image)
numSuperpixels_own = 1018;  % Adjust based on image size/complexity
[L_own, N_own] = superpixels(own_image, numSuperpixels_own);
% Visualize superpixels (optional, but recommended)
BW_own = boundarymask(L_own);
figure;
imshow(imoverlay(own_image, BW_own, 'cyan'));
title(sprintf('Superpixels (N = %d)', N_own));

% Compute region-averaged image for superpixels (same as Task 4)
outputImage_own = zeros(size(own_image));
for k = 1:N_own
    mask = (L_own == k);
    for ch = 1:3
        channel = own_image(:,:,ch);
        meanVal = mean(channel(mask));
        outputImage_own(:,:,ch) = outputImage_own(:,:,ch) + meanVal .* mask;
    end
end

% Compare k-Means and superpixels (adjust K_own and numSuperpixels_own)
figure;
subplot(1,3,1); imshow(own_image); title('Original');
subplot(1,3,2); imshow(segmentedRGB_own); title(sprintf('k-Means (K=%d)',
K_own));
subplot(1,3,3); imshow(outputImage_own); title('Superpixels');
```

Image Result of Task 5:

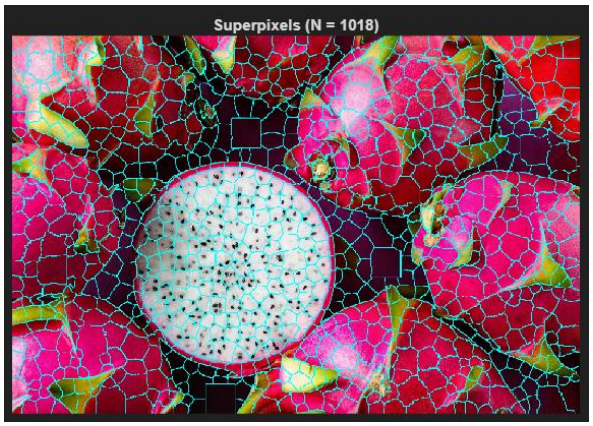
Original

k-Means Segmentation(K=5)

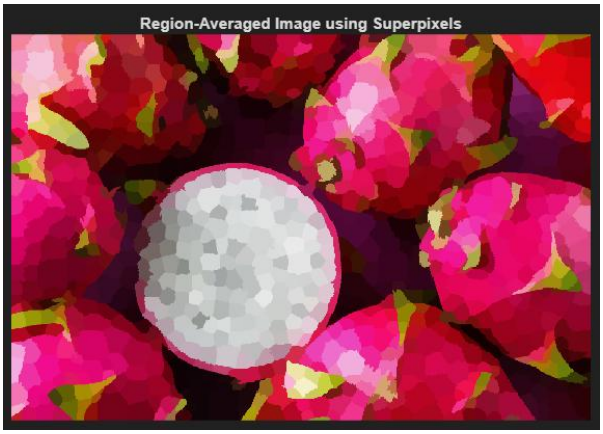


Colour-Segmentated Image(k-Means, K=5)

Superpixels (N = 1018)



Region-Averaged Image using Superpixels



Conclusion:

1. k-Means (Color-Based) is a "Global" Grouper

- **Strength:** It is excellent for color quantization. It reduces an image to its dominant color palette (e.g., simplifying the dragonfruit to just pink, white, and black).
- **Weakness:** It has no concept of "shape." As seen in the pepper image, it merges distinct objects into single blobs if they share the same color. It also creates "noise" (speckles) in textured areas like the dragonfruit flesh.

2. Superpixels are "Local" Boundary Preservers

- **Strength:** This is the superior method for preserving structure. As shown in the dragonfruit image (bottom row), Superpixels adhere to the natural edges of the fruit's scales and the circular slice.
- **Advantage:** Unlike k-Means, Superpixels do not merge touching objects. They create a "stained glass" effect that simplifies the image data while respecting the physical boundaries of the objects.

3. The "Spatial" Warning (RGB + XY)

- Adding spatial coordinates to k-Means without proper scaling (normalization) destroys the segmentation. It proves that feature scaling is critical in machine learning. Otherwise, large numbers (pixel positions) drown out small numbers (color values).

If needing to separate objects like individual peppers or fruit scales, use Superpixels. If simply want to analyze the dominant colors of a scene, use k-Means.

Short lab report/discussion points:

1. Figure of Original, k-Means(K=5) and Superpixels

2. Comparison of Methods

- **k-Means (RGB) vs. k-Means (RGB+XY):**
  k-Means in RGB segments effectively by global color, successfully separating distinct vegetables (e.g., yellow vs. red peppers) based on pixel intensity alone. However, adding unscaled spatial coordinates (RGB+XY) causes the algorithm to prioritize physical proximity over color features. This "spatial dominance" leads to under-segmentation, where distinct objects are merged into large, meaningless blobs simply because they are neighbors.

- **k-Means vs. Superpixels:**
  While k-Means is excellent for reducing an image's color palette, it ignores local structure, often merging touching objects of the same color (like the group of red peppers) into a single shape. Superpixels are superior for defining object shapes; they rely on local image gradients to adhere to natural edges, effectively preserving the boundaries between adjacent items even if they share similar colors.

3. Real-World Application: Medical Imaging

Application: Tumor and Lesion Segmentation in MRI/CT Scans.

How it works?

Medical images often contain complex tissues with subtle boundaries. For example:

- **Superpixels** are used as a preprocessing step to partition the scan into small, homogeneous regions that strictly adhere to tissue boundaries (e.g., the edge of a tumor).
- This drastically reduces the complexity of the data (from millions of pixels to hundreds of regions).
- **k-Means** (or more advanced classifiers) can then be applied to these superpixel regions to classify them as "healthy" or "anomalous" based on their intensity or texture, aiding doctors in rapid diagnosis and surgical planning.