CSC566 – Image Processing

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

Duration: 2 hours

1. Learning outcomes

By the end of this lab, you should be able to:

1. Treat image segmentation as a clustering problem in feature space.
2. Implement colour image segmentation using k-Means in MATLAB.
3. Use spatial information in the feature vector and observe its effect.
4. Use MATLAB's superpixel function as a simple graph-based segmentation and compare the results.

2. Tools and files

- MATLAB (any recent version with Image Processing Toolbox).
- Sample images:
    - peppers.png (built-in)
    - Any natural image of your choice (download or use your own photo, e.g. fruits, scenery, buildings).

3. Pre-lab recap (from videos)

Before you start coding, write short answers (in your notebook) to:

1. Why can image segmentation be viewed as clustering in feature space?
2. What are the advantages and limitations of k-Means compared to graph-based segmentation?

You do not need to submit this, but it will help you later in the discussion.

4. Task 1 – Load and inspect the image (15 min)

1. Start MATLAB and load the sample image:

   ```
   I = imread('peppers.png');   % or your own image

   figure; imshow(I); title('Original Image');
   ```

2. Convert to double precision for processing:

   ```
   I_d = im2double(I);
   ```

3. Display the separate colour channels to recall RGB representation:

   ```
   R = I_d(:,:,1);

   G = I_d(:,:,2);

   B = I_d(:,:,3);

   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');
   ```

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

5. Task 2 – k-Means segmentation in RGB space (35 min)

You will now treat each pixel as a 3D vector [R G B] and cluster these vectors using k-Means.

1. Reshape the image into a feature matrix:

   ```
   [nRows, nCols, ~] = size(I_d);

   pixels = reshape(I_d, nRows*nCols, 3);  % N x 3 matrix
   ```

2. Choose a number of clusters, e.g. K = 3 and run k-Means:

   ```
   K = 3;

   [idx, C] = kmeans(pixels, K, ...

                  'Distance', 'sqeuclidean', ...

                  'Replicates', 3);
   ```

3. Reshape cluster labels back to image size and visualise:

   ```
   segmentedLabels = reshape(idx, nRows, nCols);

   figure;

   imagesc(segmentedLabels);

   axis image off;

   title(['k-Means Segmentation (K = ' num2str(K) ')']);

   colormap('jet');
   ```

4. Create a colour-segmented image (each pixel replaced by its cluster centre):

   ```
   segmentedRGB = reshape(C(idx, :), nRows, nCols, 3);

   figure;

   imshow(segmentedRGB);

   title('Colour-Segmented Image (k-Means in RGB space)');
   ```

Q2: Experiment with K = 2, 3, 5.

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

6. Task 3 – Adding spatial information (RGB + XY) (25 min)

The video on "Segmentation as Clustering" mentioned that spatial proximity can help.

1. Build a feature vector that includes **pixel coordinates** (x, y):

```
[xGrid, yGrid] = meshgrid(1:nCols, 1:nRows);

% Normalise coordinates to [0,1] to keep scale comparable

xNorm = xGrid / nCols;

yNorm = yGrid / nRows;

features = [pixels, xNorm(:), yNorm(:)];   % N x 5 matrix:
[R G B x y]
```

2. Run k-Means on this 5D feature space:

```
K = 3;

[idx_xy, C_xy] = kmeans(features, K, ...

                        'Distance', 'sqeuclidean', ...

                        'Replicates', 3);

labels_xy = reshape(idx_xy, nRows, nCols);

segmentedRGB_xy = reshape(C_xy(:,1:3), K, 3);

% Recreate segmented image using RGB part of centres:

segImage_xy = reshape(segmentedRGB_xy(idx_xy, :), nRows,
nCols, 3);

figure;

subplot(1,2,1); imshow(segmentedRGB);      title('k-Means
(RGB only)');

subplot(1,2,2); imshow(segImage_xy);       title('k-Means
(RGB + XY)');
```

Q3: Compare the two results.

Does adding (x, y) coordinates reduce scattered regions and produce more spatially coherent segments? Explain with reference to your output.

7. Task 4 – Simple graph-based idea using superpixels (25 min)

You will now approximate graph-based segmentation using MATLAB's superpixels function. This method internally builds a graph over pixels.

1. Generate superpixels:

```
numSuperpixels = 150;   % you may adjust this

[L, N] = superpixels(I_d, numSuperpixels);

figure;

BW = boundarymask(L);

imshow(imoverlay(I_d, BW, 'cyan'));

title(['Superpixels (N = ' num2str(N) ')']);
```

2. Compute the mean colour of each superpixel and create a region-averaged image:

```
outputImage = zeros(size(I_d));

for k = 1:N

    mask = (L == k);

    for ch = 1:3

        channel = I_d(:,:,ch);

        meanVal = mean(channel(mask));

        temp = outputImage(:,:,ch);

        temp(mask) = meanVal;

        outputImage(:,:,ch) = temp;

    end

end

figure;

imshow(outputImage);

title('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.

8. Task 5 – Apply to your own image (15 min)

Repeat Task 2 and Task 4 using your own chosen image (e.g. scenery, building, etc.).

Take note of:

- A case where k-Means merges two objects that should be separate.
- A case where superpixels follow the boundary more accurately.

9. Short lab report/discussion points

For your short write-up (or for in-class discussion), prepare:

1. One figure showing **original image**, **k-Means (best K)**, and **superpixels** side by side.
2. A short explanation (5–8 lines) comparing:
   - k-Means in RGB vs k-Means in RGB+XY
   - k-Means vs superpixel-based segmentation
3. One suggestion on how these segmentation methods could be used in a real application (e.g. medical imaging, surveillance, object detection).