AMME4710 Computer Vision and Image Processing
Tutorial Activity Week 6, 2022

THE UNIVERSITY OF SYDNEY

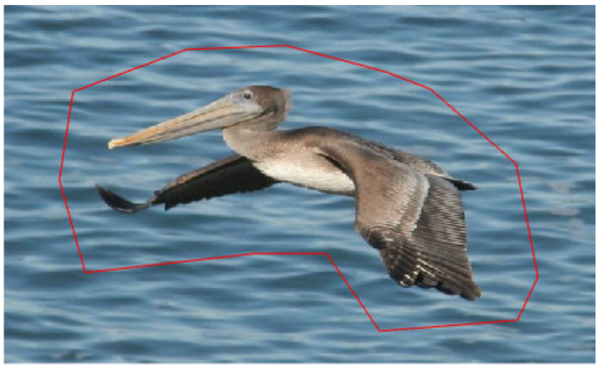
Image Segmentation

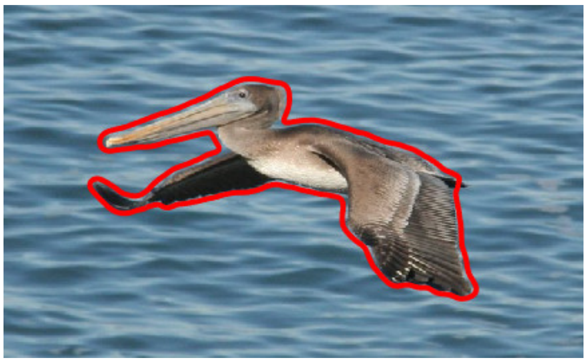
- This tutorial activity is to be completed during the Week 6 tutorial (Friday 9th September 2022)
- When you have completed the activity, you will need to show your results to the tutor who will check your work and check that you have sufficiently completed the task

Objectives

- This tutorial will introduce you to concepts in image segmentation and explore methods for semi-supervised segmentation and unsupervised image segmentation routines based on image clustering

1. Active Contours and Snakes





"Active contours" (or snakes) is a semi-automated image segmentation technique that attempts to refine an initial contour to best fit a target shape in an image. The algorithm attempts to find an optimal position for a closed curve which best fits to the boundary of an object in the image, based on an energy minimisation routine which is a combination of fit to the edge while maintaining the shape of the contour via terms controlling the elasticity and curvature of the contour.

Download the file "example_code_week6.zip" from the black board and open up the files "snakes.m" and "snakes_demo.m". This code implements a variation on the snake algorithm known as a "balloon-force snake". This approach is a modification to the objective function presented in the lecture that also applies a "pressure force" term to the energy function that makes the curve behave like a balloon [1]. In this model, the contour moves out (inflation) or in (deflation) along its normal vectors thus eliminating the need for the initial curve to be close to the solution (correct edges) to converge. The Balloon Snake passes over relatively weak edges and it stops on salient edges. By placing balloon snake inside/outside the object of interest and inflate/deflate contour to its boundary, this new model is useful for objects that are difficult to locate or too complex to trace by a traditional snake algorithm.

Run the demonstration MATLAB script "snakes_demo.m". The demonstration presents the user with an image and an option to click points to form an initial contour around the object. The first image shown is an MRI image of a heart cavity. Click points along the inside of the small dark region in the center of the image. Make sure you click points in a clockwise direction. For the second image (pelican in flight) click points along the outside of the pelican. After you have clicked points, a snakes algorithm is run to refine the initial curve to fit each target shape using the function `snake`.

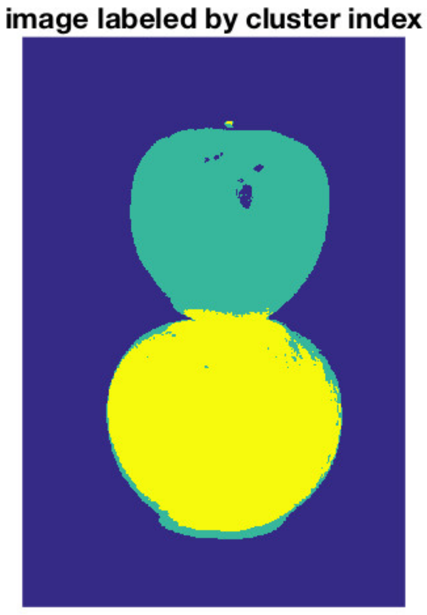
The function `snake` (in "snake.m") takes as an input an image, calculated gradient values, an initial curve (represented by a series of 2D points) and four parameters that control the energy function:

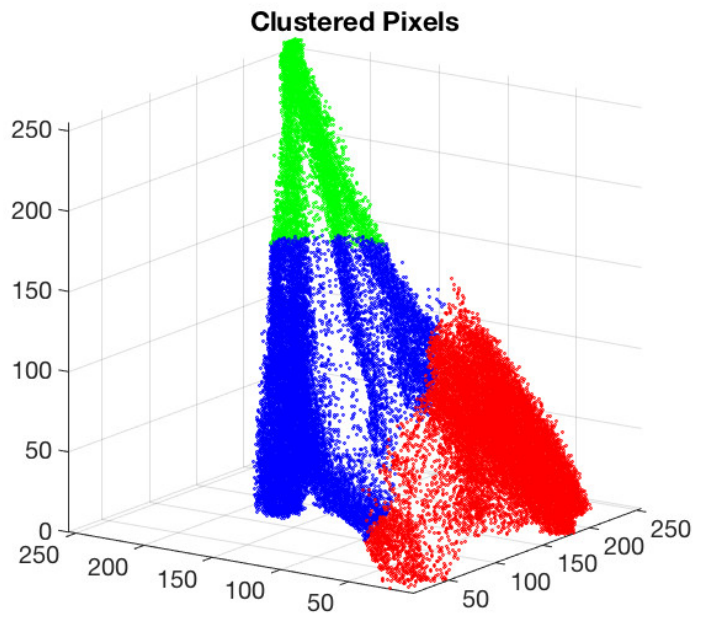
- **`alpha`**: controls the elasticity term of the curve (internal energy)
- **`beta`**: controls the rigidity/curvature term of the curve (internal energy)
- **`kappa`**: controls the external energy term associated with the image gradient
- **`lambda`**: controls the balloon force term: this term is a constant force that makes the contour continue to expand (positive values) or contract (negative values) until the contour is reached.

Tutorial Activity 1: Snakes

- Have a read through the demo program "snakes_demo.m" and get an understanding of how the two examples are performed
- Make a copy of this script and try running the algorithm with different input images and varying the parameters alpha, beta, kappa and lambda. Note that `snake` expects gradients that are computed over a grayscale image: make sure you convert images to grayscale if required.
- You may need to vary the filter parameters for which the gradient is calculated over, and for colour images you may need to apply a colour transformation to accentuate the gradient around the object you are targeting. See the second half of the example to see how a weighted combination of image channels can be used to maximise the difference along a target object contour.

2. Image Segmentation using K-means clustering





K-means is a data clustering algorithm designed to split N data points into K clusters such that the within cluster sum of squares is minimised. K-means can be used to segment the pixels in an image into K distinct classes based on colour, intensity, textural, spatial and/or other properties of the pixels themselves in an automated way.

MATLAB provides an implementation of k-means clustering via the function `kmeans`. `kmeans` takes an N x M matrix of input data points (N points, each with M dimensions) and a scalar K and returns a list of cluster index values for each point, and the K cluster centers. To apply `kmeans` to cluster the pixel values in a colour image, you first need to convert the image data into a list of points. For example, to cluster based on RGB colour alone into K = 4 clusters:

```
points = reshape(im, size(im,1)*size(im,2), 3);
[cluster_idx, cluster_center] = kmeans(points, 4);
```

where `im` is a three-channel colour image (RGB). To transform the resulting list of indices into a segmented image, the list must be reshaped into an image array:

```
pixel_labels = reshape(cluster_idx, size(im,1), size(im,2));
imagesc(pixel_labels)
```

The function `kmeans` provides a variety of optional parameters that can control the type of distance metric used to compute distance between points (defaults to a standard Euclidean distance) and the number of replicate the clustering process to avoid unlucky convergence (you should set the parameter `'Replicates'` to a number greater than 1). To visualise a clustered pointcloud in a (up to) 3D feature space (i.e. clusters based on RGB), you can use the function `scatter3`, which produces a 3D scatter plot and provides the ability to colour the points. For example:

```
[m,n,d] = size(im);
scatter3(points(:,1),points(:,2),points(:,3),ones(n*m),1),c)
```

colours the points using the vector `c`, which can be an n*m by 1 vector specifying the cluster number of each point (i.e. from `cluster_idx`) or an n*m by 3 vector of per-point colours (RGB values between 0 and 1). For example, to colour the points based on cluster id, with an equal spread of hues for each cluster:

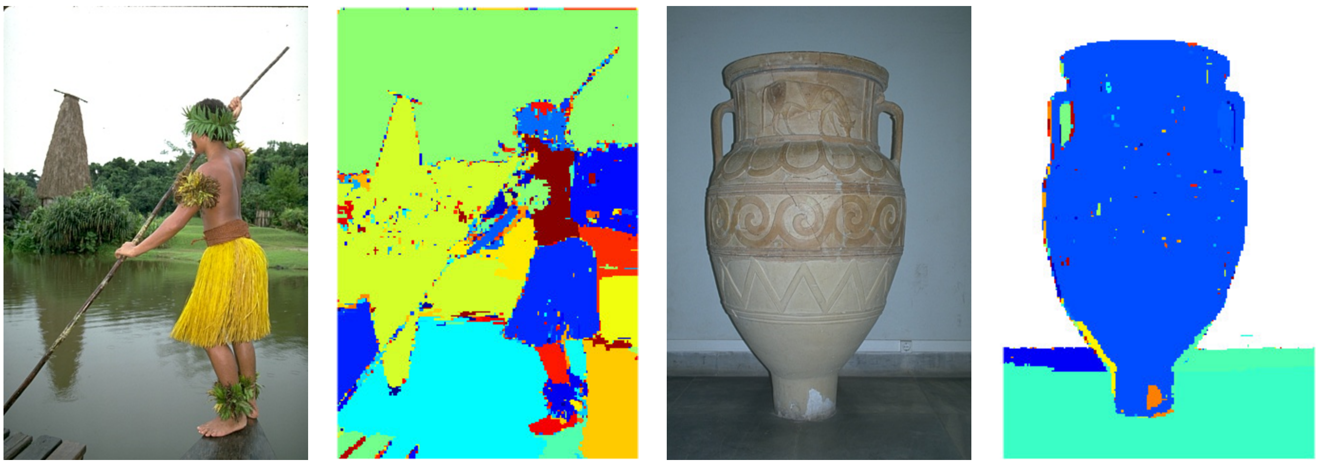
```
c = zeros(n*m,3);
for i = 1:length(cluster_idx)
    c(i,:) = hsv2rgb([cluster_idx(i)/K,1,1]);
end
```

Produces a vector of colours where each point is coloured by its cluster index, where clusters are coloured by equally spreading out values in hue.

Tutorial Activity 2: K-means clustering and Image Segmentation

- Develop MATLAB code to perform image segmentation based on k-means colour clustering for a user-selectable value of K
- Experiment with the performance of this segmentation approach in the example images by varying K. Try varying the input colour space of the image array (i.e. by using the functions rgb2hsv, rgb2lab etc.) prior to clustering and note the way in which data is clustered differently in difference colour spaces
- Extend you approach to include either a texture response layer and/or a spatial layer:
- For adding spatial information into the feature space, add columns to the data matrix in points that represent the x and y locations of each pixel in the data point list, and include these in the clustering.
- For adding a textural layer, implement a simple texture filter (see week 3 tutorial) and add the output of this filter as an additional column in the data matrix.
- Note that the range of values (in feature space) of the spatial and textural dimensions are different to the 0-255 colour space values provided by the RGB dimensions. You will need to scale your spatial/textural values to control the degree to which clusters lock onto colour/spatial/textural properties (i.e. if these dimensions have a range that is greater than 0-255, more emphasis will be put on segmenting these dimensions, and having numbers that have a smaller range than 0-255 will put less emphasis with respect to colour.

3. Mean Shift Image Segmentation



Mean shift is a data clustering technique that overcomes some of the limitations of k-means (does not require user to select K and can fit to non “circular” clusters) that can be applied to image segmentation. Mean shift works by finding modes in the image feature space (regions of locally-maximal point density) and segmenting feature-space points based on where they lie in the basin of attraction to these points. When applied to image segmentation, mean shift is also typically supplemented with a system for maintaining spatial discontinuity in clustered regions that are not joined in the image space [3].

Open up the files “meanshsegm.m” and “meanshsegm_demo.m” from “example_code_week6.zip”, available on the black board. The function `meanshsegm` implements a mean shift image segmentation algorithm that given an image array, produces a labelled image array (i.e. a single channel image with integers representing the segment number of each pixel). The function uses two parameters to control the segmentation: a spatial kernel size `hs` and a feature range kernel size `hr`. `hr` defines the size of a square region in feature space that controls the window for which feature points are included during the mean shift process. `hs` controls the threshold in terms of spatial pixels in the image for which pixels are considered to be of the same class during cluster assignment and during the creation of segments based on joined spatial regions in the image. The implementation of mean shift clustering in this function uses a feature space that is composed of both the colour and spatial pixel locations (a 5D space) during clustering.

The mean shift algorithm has a reasonably high level of computational complexity (for large window sizes, the computation time can scale quadratically with the number of image pixels). The MATLAB implementation provided here is also not optimised, and hence the algorithm can take (potentially) up to a few minutes to complete, depending on window sizes used.

Tutorial Activity 3: Mean Shift Image Segmentation

- Run the mean shift demo script “meanshsegm_demo.m”. Try changing the image loaded in the script (on line 11) and try changing the parameters `hr` and `hs` (on line 21) and observing the performance of the algorithm on different images.

References and Further Reading:

[1] Cohen, L., Cohen, I., 1993. Finite element methods for active contour models and balloons for 2D and 3D images. IEEE Trans. on Pattern Analysis and Machine Intelligence 15(11), 1131–1147, 1993.

[2] J. Liang, T. McInerney, D. Terzopoulos, "United Snakes", Medical Image Analysis, 10, 215–233, 2006.

[3] D. Comaniciu, P. Meer, "Mean shift: a robust approach toward feature space analysis", IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002.

[4] R. Szeliski, "Computer Vision: Algorithms and Applications", Springer, 2010 (Section 5)

[5] MATLAB Image Processing Toolbox Documentation, https://au.mathworks.com/products/image.html, Mathworks, 2022.