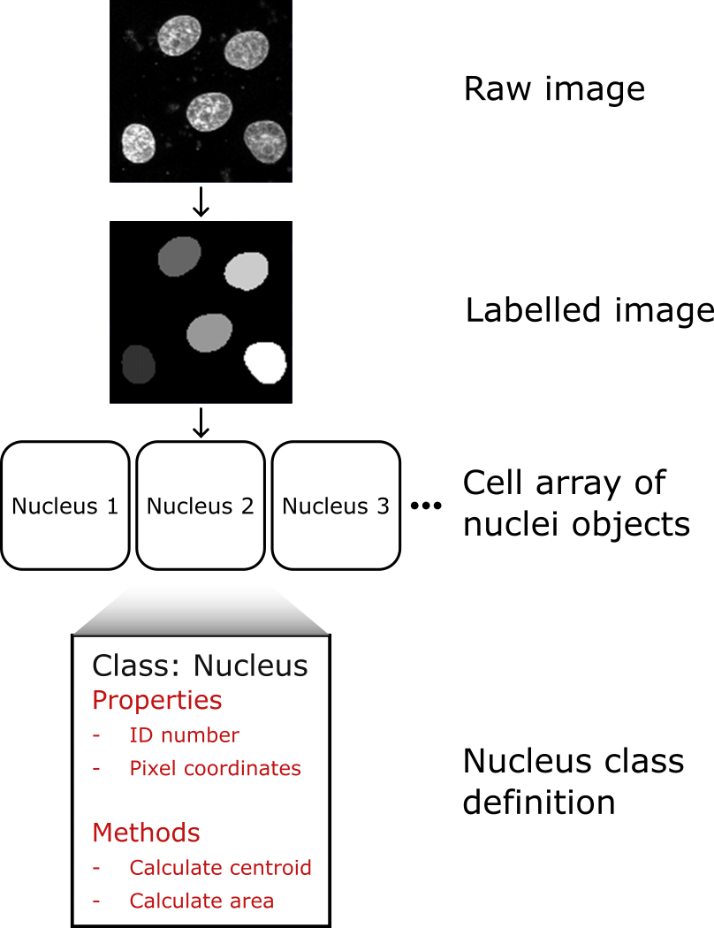
**MATLAB for image processing: Session 3 worksheet**

This session’s worksheet is a continuation of the workflow for segmenting nuclei we worked on in Session 2. At the end of the previous worksheet we had a labelled image, where the all the pixels corresponding to a specific nucleus had the same value (effectively that nucleus’ ID number). This is a functional way to store the nuclei, but any time we want to make measurements for a nucleus, we must first identify which pixels correspond to it. It would be more efficient if we used a data structure that allowed us to keep a numeric array of pixel coordinates corresponding to each nucleus. Any of the mixed data-type structures covered in the slides (structure arrays, cell arrays or tables) can do this, but we can go one better by using an object-oriented programming (OOP) approach. Using an OOP approach allows us to also assign useful functions (e.g. measuring the area or centroid location) to the coordinate stores. In this worksheet, we will build one such OOP model. We will then iterate over each nucleus in the labelled image, create a Nucleus object instance and add the object to a 1D cell array.

1. **Getting the labelled nuclei image**

This first exercise is a repeat of the final main exercise from the Session 2 worksheet. It will take us from a raw image of nuclei to a labelled image. To ensure everyone’s starting with the same materials, the code to do these steps is provided.

Note: For the exercises in this worksheet we’ll use a smaller image, which only contains 5 nuclei. This will make checking results easier.

1. If you haven’t already done so, download the “NucleiImage\_small.tif” image from the Session 3 GitHub repository (<https://github.com/SJCross/MATLAB-course>).
2. Create a new script file and save this to a location accessible to MATLAB.
3. Paste the following code into the new script file

% Clearing the workspace

clear

% Loading the nuclei image

nuc\_im = imread('NucleiImage\_small.tif');

% Applying a 2D median filter

filt\_im = medfilt2(nuc\_im, [5, 5]);

% Calculating and applying threshold

thresh = graythresh(filt\_im)\*255;

log\_im = filt\_im > thresh;

% Filling holes in the binarisation

fill\_im = imfill(log\_im,'holes');

% Creating labelled image of nuclei

label\_im = bwlabel(fill\_im);

% Displaying the labelled image

imshow(label\_im,[]);

1. Run the script from the command window to ensure everything is working correctly. You should see the labelled image as it appears in the figure at the top of this worksheet.
2. **Getting pixel coordinates for a single nucleus**

Here, we will establish a method to create a 2D numeric array containing the coordinates of all pixels corresponding to a specific nucleus. In subsequent exercises, this array will be the coordinate store for the Nucleus class objects. The array should have size Nx2, where N is the number of pixels in that nucleus. The first column of the array should be for the pixel row coordinates and the second for the pixel column coordinates. One way to achieve this is to use the *find* function, which returns the row and column coordinates corresponding to a logical check on an array.

Hint: If you get stuck implementing the *find* function, an example is hidden between the following quotation marks. When pasted into MATLAB the code should appear.

Code: “[rows, cols] = find(label\_im == 2)”

1. For the nucleus number 4 (i.e. pixel values all equal to 4) use the *find* function to get the row and column indices of all pixels. The output from this function should be two Nx1 arrays, where N is the number of pixels in that nucleus.
2. Use array concatenation (covered in Session 2) to combine these 1D arrays to the final Nx2 array.

Note: The row coordinates for nucleus 4 should be approximately 35 and the column coordinates approximately 97.

1. **Defining an object class**

We will now start setting up our nucleus class definition. This definition specifies what properties objects of that class will have as well as how any methods should operate.

1. Create a new script file and save it to a location accessible to MATLAB with the name “nucleus.m”. This will be our class file.

* Create nucleus object class with the following properties. You’ll also need to create a constructor method, which takes these as arguments.
  + Properties:
    - ID (the label assigned to this object)
    - Pixel coordinates
* Add a new method to the class definition, which calculates the centroid of the object (mean row and mean column), displays a message to the command window when it’s run in the format “Object (ID = 7) has centroid (32.5, 46.1)” and returns the centroid as a 2-element numeric array. Test on object 3 (say what the result should be).
* Loop over all object labels (you can refer to the solution to Session2, Exercise 9 for this), creating an object and adding these objects to a cell array
* Loop over all elements of the cell array (i.e. loop over each object) and run the “getCentroid” method. This should print a list of all centroids to the command window (Say what the result should be).