# Programming assignment 3 – Mean-shift

## 1 Basic Implementation

The Mean Shift algorithm clusters a $d$-dimensional data set by associating each point to a peak of the data set's probability density function. For each point, Mean Shift computes its associated peak by first defining a spherical window at the data point of radius $r$ and computing the mean of the points that lie within the window. Note that in the mean shift paper a kernel which assigns distance decaying weights to all other points is used, while we simply use a spherical window which gives all points within a distance $r$ uniform weights and 0 to all other points. The algorithm then shifts the window to the mean and repeats until convergence ($\epsilon = .01$ works well). Within each iteration, the window will shift to a more densely populated portion of the data set until a peak is reached. You will implement this process as the function:

```
function peak = findpeak(data,idx,r)
```

where data is the $d$-dimensional data set ($d \times n$ matrix), idx is the column index of the data point for which we wish to compute its associated density peak and $r$ is the search window radius. The algorithm's dependence on $r$ will become apparent from the experiments performed below. Implement the mean shift function, which calls `findpeak` for each point and then assigns a label to each point according to its peak. This function should have the following prototype:

```
function [labels, peaks] = meanshift(data, r)
```

where `labels` are the peak labels (a vector of length $n$ with each entry $\in [1, K]$ where $K$ is the number of distinct peaks) and `peaks` is a $d \times K$ matrix storing the density peaks found using meanshift as its columns. Note that Mean Shift requires that peaks are compared after each call to `findpeak` and for similar peaks to be merged. For our implementation of Mean Shift, we will consider two peaks to be the same if the distance between them is $\leq \frac{r}{2}$. Also, if the peak of a data point is found to already exist in peaks then for simplicity its computed peak is discarded and it is given the label of the associated peak in `peaks`.

Debug your algorithm using `two_clusters.mat` with $r = 2$ (this should give two clusters). Plot your result using the `plot3dclusters` function. The result of running `meanshift` on `two_clusters.mat` is depicted in Figure 1. If you are having doubts about your program working correctly, you can generate more synthetic data using the function `generatedata`.

## 2 Optimizations

Unfortunately, the Mean Shift algorithm you just implemented is too slow to be realized for image segmentation. We will therefore incorporate several speedups into our implementation.

### 2.1 MATLAB vectorization (avoid loops!)

First of all, you will definitely want to exploit MATLAB's ability to handle matrix operations efficiently. For example, if you want to find all data points with label equal to 1 and compute their mean, write the following:

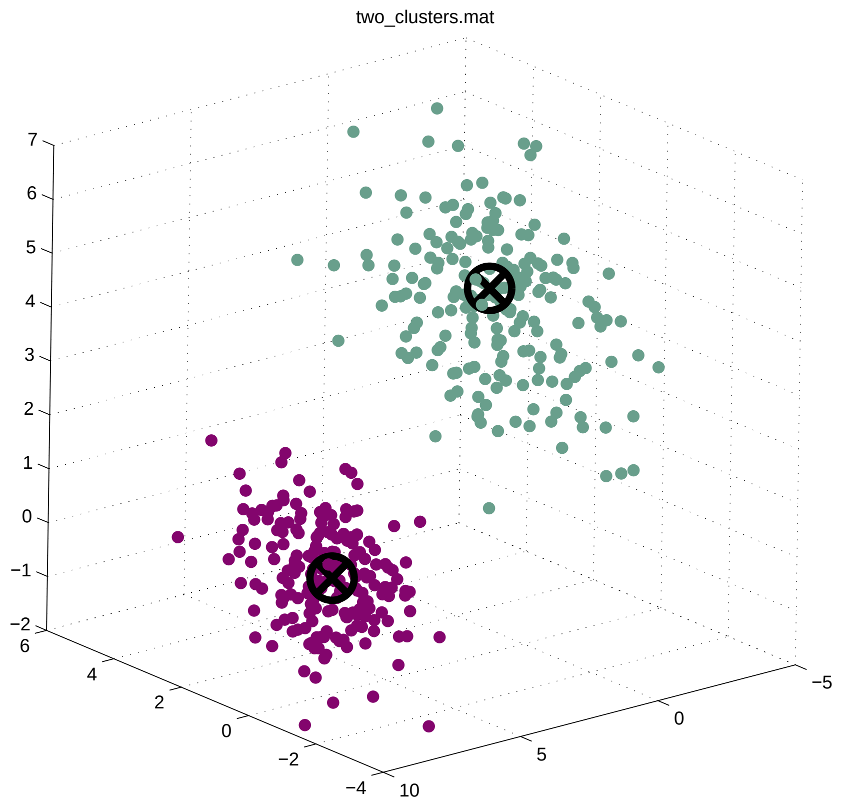

Figure 1: Mean Shift Clustering Result

```
currdata = data(:,labels == 1);
currmean = mean(currdata,2);
```

You will also be computing Euclidean distances from some `x` to all of your data very often, and it is crucial that you avoid writing `for` loops and computing distances one by one. We are providing you with a MATLAB function `ml_distSqr` that given two matrices of points (where points are the columns in the matrices) as input, will return a matrix of squared Euclidean distances between each of the elements in the input.

### 2.2 Basin and Trajectory of Attraction

The first non-MATLAB specific speedup will be to associate each data point that is at a distance $\leq r$ from the peak with the cluster denoted by that peak. This speedup is known as basin of attraction and is based on the intuition that points that are within one window size distance from the peak will with high probability converge to that peak.

The second speedup is based on a similar principle, where points that are within a distance of $\frac{r}{c}$ of the search path are associated with the converged peak, where $c$ is some constant value. Incorporate the above speedups into your Mean Shift implementation by modifying your implementation from before. The resulting modified function should have the following prototypes:

```
function [labels, peaks] = meanshift_opt(data,r,c)
function [peak, cpts] = findpeak_opt(data,idx,r,c)
```

where `cpts` is a $n$ dimensional boolean vector storing a 1 for each point that is within a distance of $\frac{r}{c}$ from the trajectory or within $r$ from the final peak, 0 otherwise. Your implementation of `meanshift_opt` should produce the same result on `two_clusters.mat` with $r = 2$ and $c = 4$ as `meanshift` (the two clusters will start to mix if you let $c = .5$). You might want to generate some data using `generatedata` and try to cluster it. You will have the ground truth labels and can see how well your algorithm is doing. An example of such a clustering can be seen in Figure 2.

## 3 Image Segmentation

In this section you will build upon your optimized Mean Shift implementation to perform image segmentation. To do so, implement the function

```
function [segmIm,peakIm] = segment_meanshift(I,r,c)
```

In this function, $I$ is a color input image, $r$ is the radius used for our window, and $c$ is the trajectory of attraction parameter. `segIm` is an image created by reshaping the `labels` into the size of the input image. `peakIm` is an image the same dimensions as $I$, but with each pixel colored by the RGB value of that pixel's associated peak. This function is constructed by reshaping the image into RGB vectors, converting them to Luv color-space representation, and then clustering the resulting color data using the optimized Mean Shift implemented earlier. Mean Shift clusters using the Euclidean distance metrics, and since Euclidean distance in RGB space does not correlate well to the perceived change in color we will use Luv color space. In `segment_meanshift`, convert to Luv color space by using the provided MATLAB function `rgb2luv`. Then convert the resulting cluster centers back to RGB using the function `luv2rgb`.

### 3.1 Results

First, you will need to cluster the 3D point clouds found in the files `testdata1.mat`, `testdata2.mat`, and `testdata3.mat`. You will need to somehow choose a good value for $r$ and you should use $c = 4$. Produce

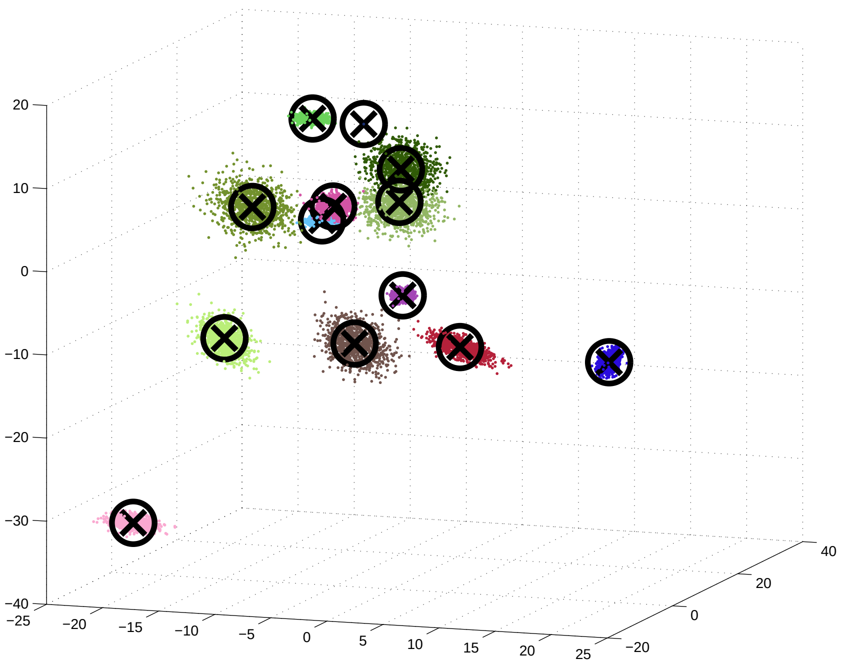

Figure 2: Mean Shift Segmentation of Point Cloud generated from generatedata function.

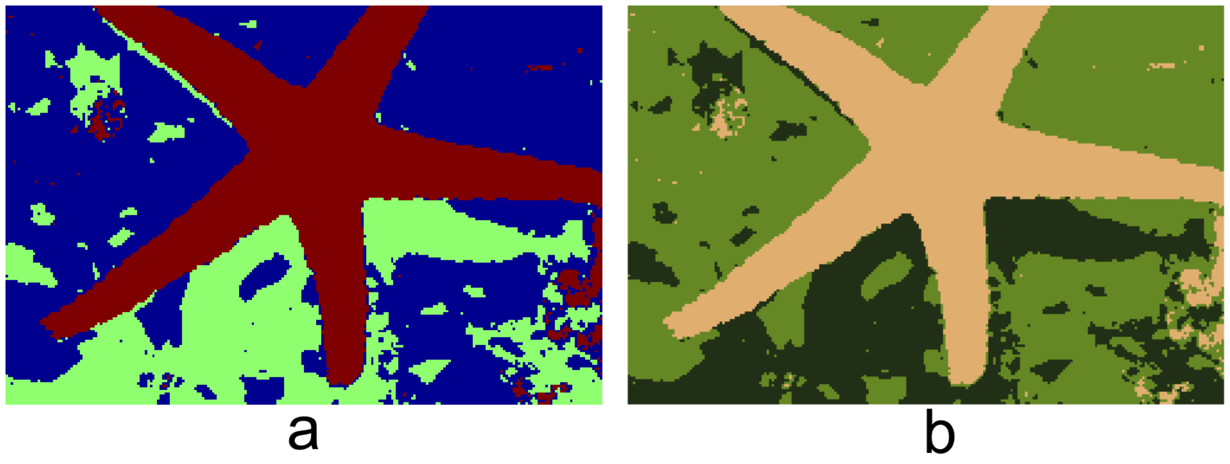

Figure 3: Mean Shift Clustering Result on Starfish Image. a) label image; b) peak image. This result is generated with $r = 20$ and $c = 1$.

three result files `results1.mat`, `results2.mat`, and `results3.mat` such that each file loads the variables `labels` and `peaks` (the output of `meanshift_opt`).

You will also have to submit some results on real images. Note that Mean Shift is still computationally expensive, even with these optimizations. Furthermore, the computational time depends on the parameters $r$ and $c$. For example, with $r = 20, c = 1$, the starfish image in Figure 3 took 3 seconds to to segment; and it took 60 seconds to segment if $r = 10, c = 2$. In general, Mean Shift will take longer if $r$ is small and $c$ is big. In this case, Mean Shift will generate lot of segments which might not be what we want. You will need to play around with different values of $r$ and $c$. With relatively big $r$ and small $c$, your algorithm should not take more than a few minutes to run. If it does, you are doing something wrong, or you need to stop using punchcard-based computers.

Segment the four `.png` images (hat,man,star,sheep) using your image segmentation algorithm. Experiment with different values of $r$ and $c$ so that you are able to segment these four images. Consider $r \in \{5, 10, 20\}$ and $c \in \{1, 2, 4\}$. What effect do $c$ and $r$ have on the run time of your algorithm? How about the visual quality of the peaks image? (You might want to down sample these images when debugging)

## References

The theory behind mean shift clustering is described in the following paper:

```
‘‘Mean shift: A robust approach toward feature space analysis’’ by D.
Comaniciu and
P. Meer, IEEE Trans. Pattern Analysis and Machine Intelligence 24, 2002, 603-619.
```

Note that there are some errors in the equations and you should look at the `meanshift_errata.pdf` document. You will not need to understand all of the mathematical details of this paper as specifics of the algorithm that you will be implementing are outlined below. Of interest is section 4 of the paper, which discusses how a suitable feature space can be defined for image segmentation purposes.