CS663, Assignment 3 **Yash Sanghvi, Pradyot Prakash**
Instructor: Prof. Suyash Awate 13D070042, 130050008
Mean Shift Segmentation

Problem Statement

(40 points) Image Segmentation using mean shift.
Input image: 3/data/baboonColor.png.
Take this 512 512 pixel image, smooth it using Gaussian convolution with standard deviation 1 pixel width, and subsample the smoothed image by a factor of 2 in each spatial dimension to produce a 256 256 image. Use this smaller-sized image for the following experiment. If this image is still too large for your computers memory, then you may resize further.
Implement the algorithm formean-shift image segmentation using both color (RGB) and spatial coordinate (XY) features. Tune parameters suitably to get a segmented image with at least 5 segments and no more than 50 segments. To improve code efficiency, you may use Matlab functions like knnsearch(), bsxfun(), etc. For this image, about 20 iterations should be sufficientfor reaching close to convergence. You may select a random subset of nearest neighbors, in feature space, for the mean-shift updates to reduce running time. Each iteration can run in about 10-20 seconds on a typical personal computer.

1. Write a function myMeanShiftSegmentation.m to implement this.
2. Display the (i) original image along with (ii) the segmented image that shows color-coded pixels (and, thus, segments) using the color component of the converged feature vectors.
3. Report the following parameter values: Gaussian kernel bandwidth for the color feature, Gaussian kernel bandwidth for the spatial feature, number of iterations.

Code

Mean Shift Segmentation

```
function [processedImage, outputImage] = myMeanShiftSegmentation(inputImage)

        W = 2;
        sigma1 = 0.66;
        numIterations = 20;
        spaceSigma = 20;
        intensitySigma = 20;

        sigmas = -2 .* ([spaceSigma, spaceSigma, intensitySigma, intensitySigma, intensitySigma] .^ 2);
        sigmaProds = (2*pi)^2.5 * spaceSigma * spaceSigma * intensitySigma * intensitySigma * intensitySigma;

        k = 1 + 100;
```

```
13
14          h = fspecial('gaussian', [W, W], sigma1);
15
16          for i = 1:3
17                  processedImage(:, :, i) = myShrinkImageByFactorD
                        (imfilter(myLinearContrastStretching(
                        inputImage(:, :, i)), h), 2);
18          end
19
20          numPixels = size(processedImage, 1) * size(
                processedImage, 2);
21          imageRepresentation = zeros(numPixels, 5);
22          newImageRepresentation = zeros(size(imageRepresentation)
                );
23
24          % initialize imageRepresentation
25          count = 0;
26          for x = 1:size(processedImage, 1)
27                  for y = 1:size(processedImage, 2)
28                          count = count + 1;
29                          imageRepresentation(count, :) = [x/255,
                                y/255, processedImage(x, y, 1),
                                processedImage(x, y, 2),
                                processedImage(x, y, 3)];
30                  end
31          end
32
33          for i = 1:numIterations
34                  fprintf('Iter: %d\n', i);
35                  [IDX, D] = knnsearch(imageRepresentation,
                        imageRepresentation, 'K', k, 'IncludeTies', ~
                        true); %handle cell situation later
36
37                  for x = 1:size(processedImage, 1)
38                          for y = 1:size(processedImage, 2)
39                                  rowInRepresentationMatrix = (x
                                        -1) * size(processedImage, 2)
                                         + y;
40
41                                  points = IDX(
                                        rowInRepresentationMatrix, 2:
                                        end); % k-1 closest points
                                        here
42                                  currentPoint =
                                        imageRepresentation(
                                        rowInRepresentationMatrix, :)
                                        ;
43
```

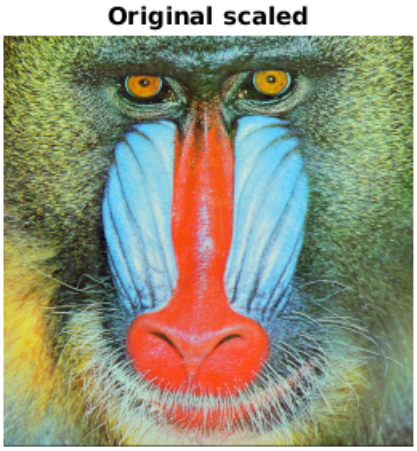
```
44                                  u = zeros(1, 5);
45                                  w = 0;
46
47                                  for t = 1:size(points, 2)
48                                          point =
                                               imageRepresentation(
                                               points(1, t), :);
49                                          w1 = exp(sum(((
                                               currentPoint - point)
                                               .^2) ./ sigmas))/
                                               sigmaProds;
50
51                                          u = u + w1*point;
52                                          w = w + w1;
53                                  end
54
55                                  u = u./w;
56                                                  newImageRepresentation
                                                     (
                                                     rowInRepresentationMatri
                                                     , :) = [
                                                     currentPoint
                                                     (1, 1),
                                                     currentPoint
                                                     (1, 2), u(1,
                                                     3), u(1, 4),
                                                     u(1, 5)];
57
58                                          end
59                                  end
60
61                                  imageRepresentation = newImageRepresentation;
62                          end
63
64                          % row = zeros(numPixels, 1);
65                          % col = zeros(numPixels, 1);
66                          % for i = 1:size(imageRepresentation, 1)
67                          %       row(i, 1) = imageRepresentation(i, 1);
68                          %       col(i, 1) = imageRepresentation(i, 2);
69                          % end
70                          %
71                          % figure;
72                          % plot(row, col, 'r*');
73                          % hold on;
74
75                          outputImage = zeros(size(processedImage));
76                          for x = 1:size(processedImage, 1)
77                                  for y = 1:size(processedImage, 2)
```

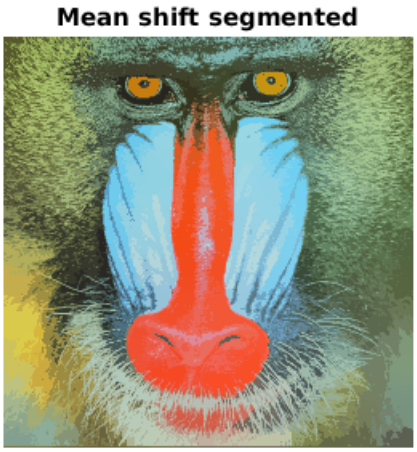
```
78                          rowInRepresentationMatrix = (x-1) * size
                               (processedImage, 2) + y;
79                          for i = 1:3
80                                  outputImage(x,y,i) =
                                       imageRepresentation(
                                       rowInRepresentationMatrix, i
                                       +2);
81                          end
82                  end
83          end
84
85  end
```

Implementation Details

We took a K=200 nearest neighbours and took their weighted average to determine the mean shift. Note that they weights were determined by a 5 dimensional Gaussian vector in space (X,Y) and colour (R,G,B). This was repeated over N iterations for the image to converge.

Result Images





Optimum Parameters

hs = 20
hr = 20 Number of Iterations = 20