CS663, Assignment 3
Instructor: Prof. Suyash Awate
Mean Shift Segmentation

Yash Sanghvi, Pradyot Prakash
13D070042, 130050008

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 [outputImage] = myMeanShiftSegmentation(inputImage,
    numIterations, spaceSigma, intensitySigma)
        k = 400;

        indices = double(zeros(size(inputImage, 1)^2, 2));

        count = 0;
        for j = 1:size(inputImage, 1)
                for i = 1:size(inputImage, 1)
                        count = count + 1;
                        indices(count, :) = double([i, j]);
                end
        end

```

```
        imageRepresentation = double([indices, reshape(inputImage, [size(inputImage, 1)^2, 3])]);
        newImageRepresentation = double(zeros(size(imageRepresentation)));

        for iter = 1:numIterations
                iter
                f = bsxfun(@(x,y) x./y, imageRepresentation, sqrt(2)*[spaceSigma, spaceSigma, intensitySigma, intensitySigma, intensitySigma]);
                [IDX, D] = knnsearch(f, f, 'K', k);

                for count = 1:size(inputImage, 1)^2
                        nbrs = imageRepresentation(IDX(count, 2:end), :);
                        wts = exp(-((D(count, 2:end)).^2));
                        u = sum(bsxfun(@(x,y) x.*y, nbrs, wts'));
                        sum_wts = sum(wts);
                        u = u/sum_wts;
                        newImageRepresentation(count, :) = double([imageRepresentation(count, 1), imageRepresentation(count, 2), u(1, 3:end)]);
                end
                sum(sum(abs(imageRepresentation - newImageRepresentation)))/(size(imageRepresentation,1))
                imageRepresentation = newImageRepresentation;

        end
        outputImage = uint8(round(reshape(imageRepresentation(:, 3:5), [size(inputImage, 1), size(inputImage, 1), 3])));
end
```

Implementation Details

We considered the algorithm with a 128x128 image due to lack of computing power leading to increased computation time. We took K = 400 nearest neighbors and took their weighted average to determine the mean shift. Note that the weights were determined by a 5 dimensional Gaussian vector in space (X, Y) and color(R, G, B). We updated just the color values keeping the position coordinates intact. This was repeated over N = 20 iterations for the algorithm to converge.

Result Images

Optimum Parameters

hs = 16
hr = 25
Number of Iterations = 20