Seylom AYIVI **HW4**
CS 543 – Computer Vision

1. Mean-Shift segmentation

I used a Gaussian kernel for both spatial and range domains. I experimented with different values of the bandwidth for both domains. Results are summarized below.

The kernel used for the RGB color domain (3D) has the form $\frac{1}{(2*pi*\sigma^2)^{\frac{3}{2}}} e^{-\frac{||x-x_i||^2}{2*\sigma^2}}$

Similarly, the kernel used for the 2D space also has the form $\frac{1}{(2*pi*\sigma^2)} e^{-\frac{\left||x-x_i|\right|^2}{2*\sigma^2}}$

We choose two different values of σ to control the bandwidth of the algorithm.

Computation of the mean shift vector:

- We start by evaluating the 5D domain for point selection. We use two criteria to find neighboring pixels in both color and spatial domain. The Euclidian distance is the measure used to filter out points that fall outside a specified radius 2*hr for RGB/Luv color range and 2*hs for space.
- We multiply RGB/Luv component of the of the pixels in the "window" with the Gaussian kernel for the color range and multiply the x, y component with the space Gaussian kernel.
- We compute the pixels weight by multiplying together both kernels.

I used code online to generate a Luv image from an RGB image and vice versa

$$MeanShift = \frac{\sum_1^n x_i * e^{-\frac{\left||x-x_i|\right|^2}{{h_r}^2}} * e^{-\frac{||x-x_i||^2}{{h_s}^2}}}{\sum_1^n e^{-\frac{\left||x-x_i|\right|^2}{{h_r}^2}} * e^{-\frac{||x-x_i||^2}{{h_s}^2}}} - x$$

In Matlab we obtain the denominator by summing the product of both kernels along the 5 dimensions.

Below are the results obtained before the merging process.

Figure 1 hr = 12 hs = 4

Figure 2 hr=12 hs=4

Results after mode merging using the Luv color space:

Figure 3 hr = 12 hs = 4

Figure 4 hr=12 hs=4

About the runtime of the mean-shift algorithm.

I spent some time to improve the runtime of my. It seems like the costly part of the processing is computing the neighborhood of the pixel. The stopping criterion for the convergence uses the absolute value of the difference between successive computations of the mean.

Mode merging:

When a point reaches its mean during the mean-shift algorithm we assign it to the closest cluster if there is one within a distance hr for color and hs for space. Otherwise we consider the new mean to be a new cluster center.

After we have run mean shift on all point and have obtained a list of the potential cluster center, we perform a second mode merging by removing clusters with less than a predefined number of pixels.

To summarize

First image: Lion.jpg

5d tolerance set to 0.01 ➔ about 90 seconds runtime with hs=4 and hr=12
Number of segments found: 1153

Second image: House2.jpg

5d tolerance set to 0.01 ➔ about 45 seconds runtime with hs = 4 and hr =12
Number of segments found: 543

2. **EM Algorithm**.
    - Please find attached my EM algorithm derivation in file ***seylom_em_derivation.pdf***
    - Please find in em_algorithm_2.m the implementation of my algorithm.
      I was unable to make it converge but would be happy to know what might have been the problem with my approach.
      My first implementation of the algorithm came out with the list of following bad annotators:
      1-2-3-5-7-11-13-14-17-19-23 and 25 but after getting many different result based on the initialization process I decided to rewrite the algorithm to make use of the logsumexp I had not used until then.

3. **Graph Cuts**

- First we process each of the three channels RGB separately by gathering pixel intensities for each channel.
- We use MATLAB's **gmdistribution.fit** function to fit a Gaussian mixture with 1 component for each channel for the foreground (applied to the pixels in the polygon) and used the same approach for pixels outside of the polygon (background).

- Energy expression

Unary term: $u(x) = -\log(\frac{p(c(x);\theta_{foreground}}{p(c(x);\theta_{background}})$

Where both $p(c(x);\theta_{foreground}\ and\ p(c(x);\theta_{background}$ where determined by our Gaussian mixtures derived from ***gmdistribution.fit*** model.

- We pass the model as an argument to MATLAB's ***pdf*** function to retrieve the posterior probability of foreground and background assignment given the parameters of the distribution

The pairwise term (defining our edge potentials) can be written as follows:

$$f(x,y) = k_1 + k_2 * \mathrm{e}^{(-\frac{"Cost(x)-Cost(y)"^2}{2*\sigma^2})}$$

The energy expression is $E(x) = \sum_p \mathrm{D}_p(f(x_p)) - \sum_{\{p,q\}\in N} V\,(f(x_p), f(x_q))$

- Intensity maps

Figure 3- Background map with K= 1 component

Figure 5 Foreground map, K=1 component

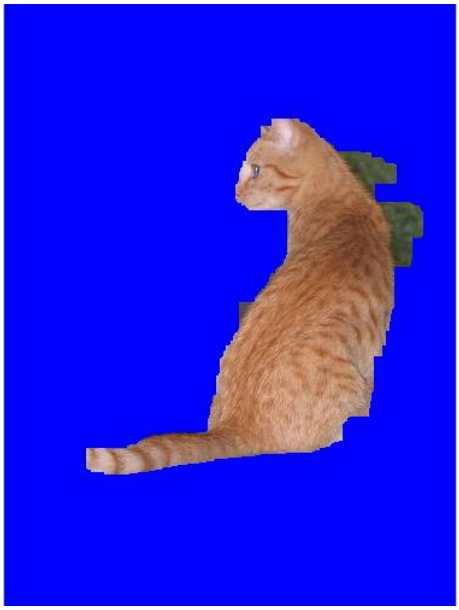
Images where generated using a simple comparison between foreground and background probability from channels with the highest probability. So if the probability of a red pixel being foreground is higher than for the other two channels, we will use that probability and compare similarly with the highest probability for the background channels. We then set the pixel value to 1 if foreground is higher, 0 otherwise.

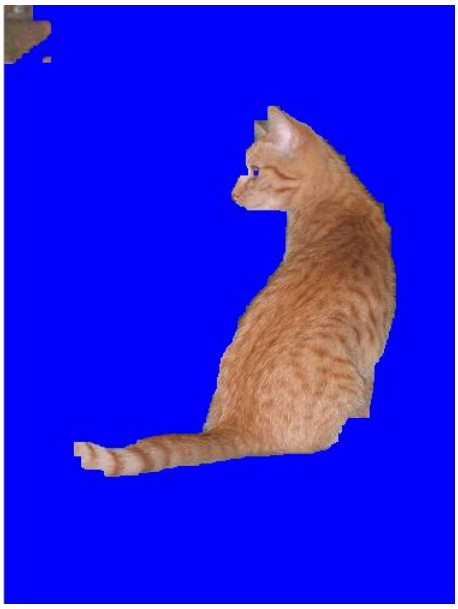
The image above is obtained by using each channel's vote separately and encoding the pixel with a value of 0 or 1 based on the probability for that pixel to be a foreground pixel (or not) for that specific channel. Interestingly, the white area in the image represents the region where all three channels agree on the vote (foreground/background). In order to use this result I used

the assumed independence for each channel in to compute the pixel foreground and background probability by multiplying the probability of all three channels.

- Final Segmentation using graph cut

Code source is in ***graphcut_segmentation_cat.m***





The first result was obtained while using as previously explained a Gaussian mixture for each channel in the original image. I then proceeded to convert to the Lab color-space since it provided better results. I was not able to get as good result with more than 1 component per channel.

The second image was obtained by using the previously describe observation (consensus on the foreground or background label) Results are very close, but I believe the second one does a better job at identifying the true foreground than does the previous one.