ECE 549 Computer Vision: Homework 4

Xianming Liu

NetID: xliu102

1. SLIC Superpixels

a) Distance Function for Measuring the Similarity

The distance function includes two parts: distance in pixels' color space (e.g., Lab used in my implementation) and spatial distance, the Euclidean distance between two pixels' coordinates. Generally, it is defined as:

$$\begin{aligned} d(i,j) &= d_c(i,j) + m * d_s(i,j) \\ &= \|c(i) - c(j)\|^2 + m * \|(x_i, y_i) - (x_j, y_j)\|_2 \end{aligned}$$

where $c(i)$ and $c(j)$ are the Lab* color vectors (3-dimensional), m is the normalized weights defined as:

$$m = compactness/S^2$$

b) Different Weights

The comparisons between different color/spatial weights are shown in Figure 1.

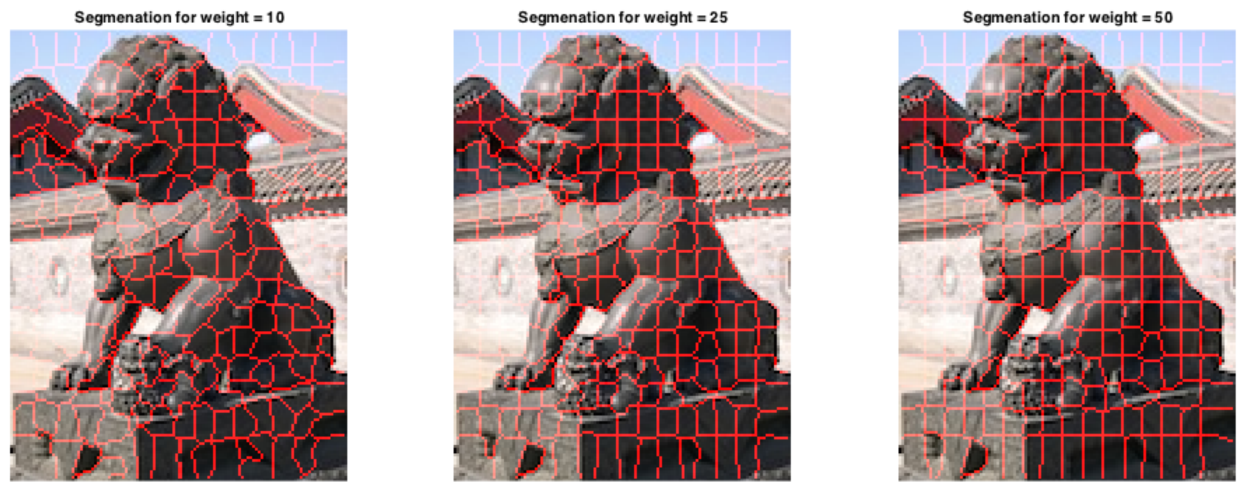
Segmenation for weight = 10
Segmenation for weight = 25
Segmenation for weight = 50

Figure 1: The comparisons of different weights, 10, 25, and 50 respectively. K = 256.

As shown in the figure, the weight determines how compact the segmentation is. With larger spatial weights, the segmentations are more compact and of more regular shapes; while with smaller weight, the segmentations are of arbitrary shapes and are strongly affected by pixel colors.

c) Error Map

The error map is shown in Figure 2.

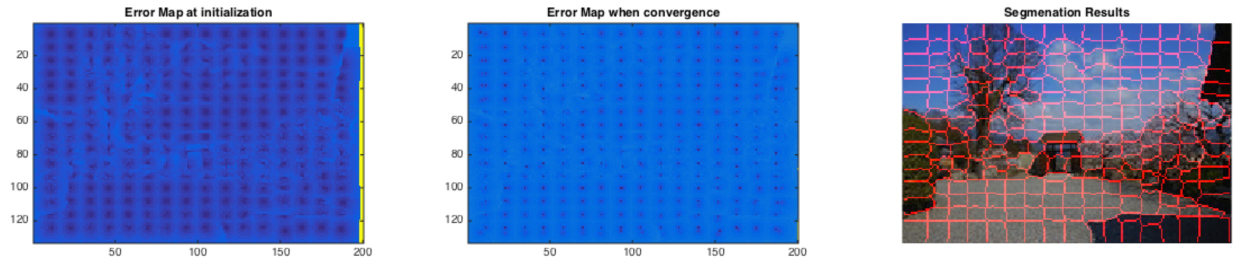

Figure 2: Error Map: left) Initialization Error Map; middle) convergence error map; right) segmentation and cluster centers.

d) Different K

The segmentation with different K is shown in Figure 3.

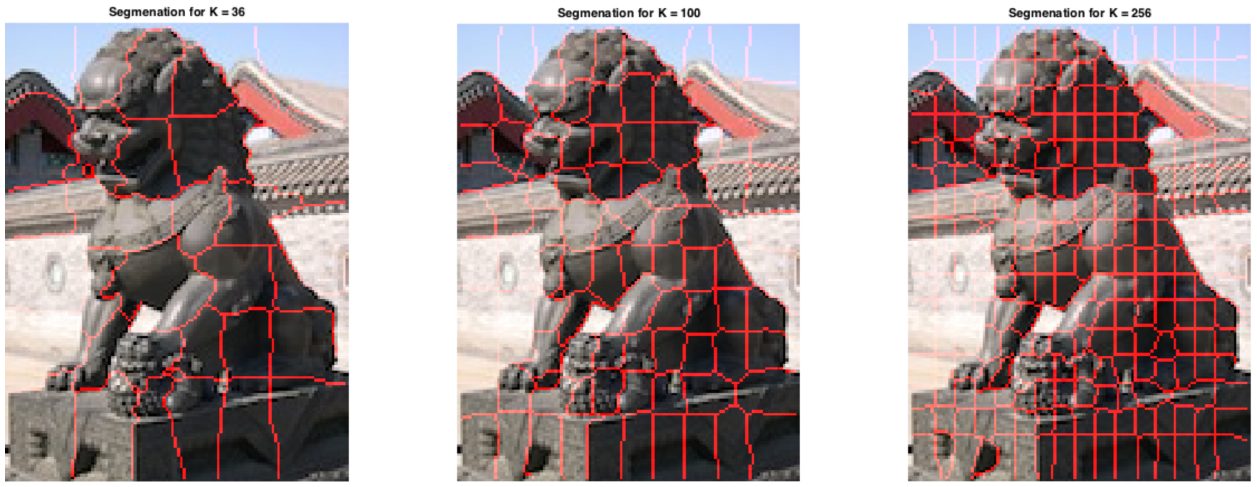

Figure 3: Superpixel segmentation with different K: 36, 100, and 256 respectively.

Runtime for each K (seconds):

Time cost when [K=36] = 1.899039e-01

Time cost when [K=100] = 3.936170e-01

Time cost when [K=256] = 8.787849e-01

e) Benchmark Evaluation

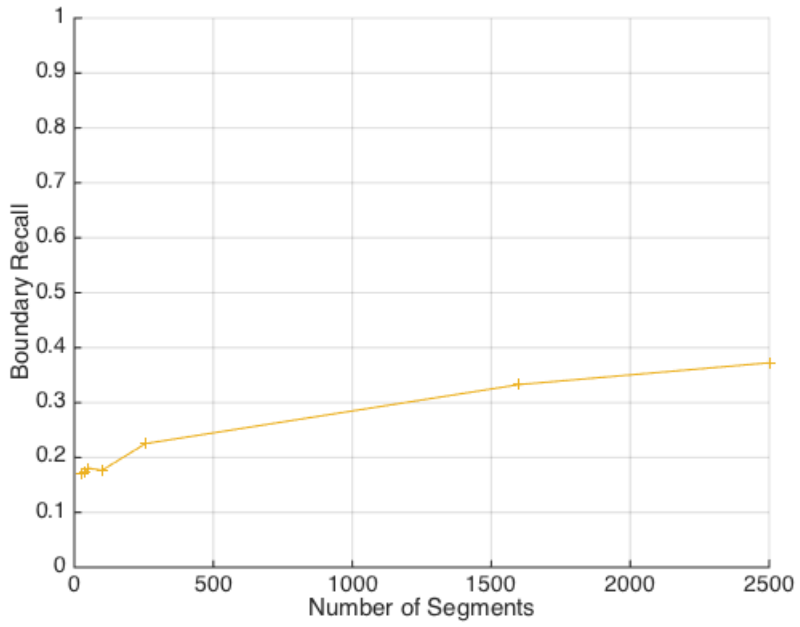

Figure 4: Evaluations - Boundary Recall

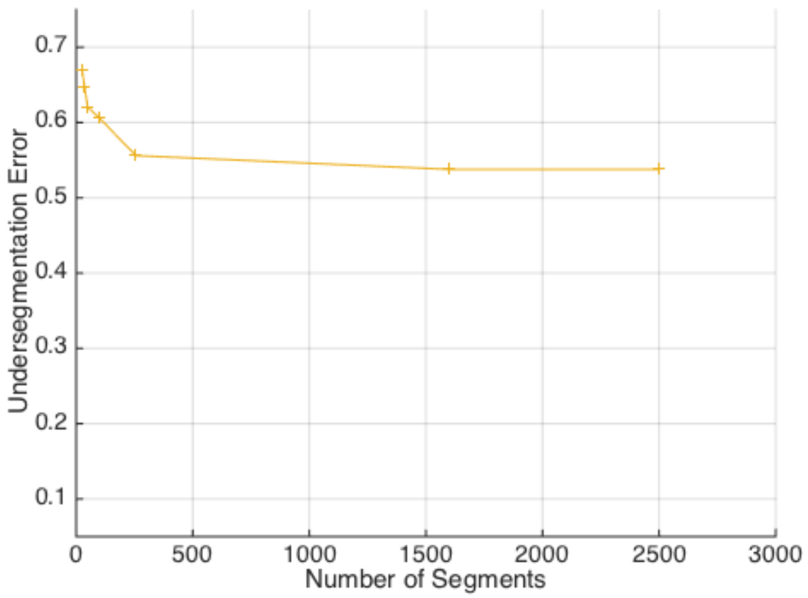

Figure 5: Evaluation: Undersegmentation Error

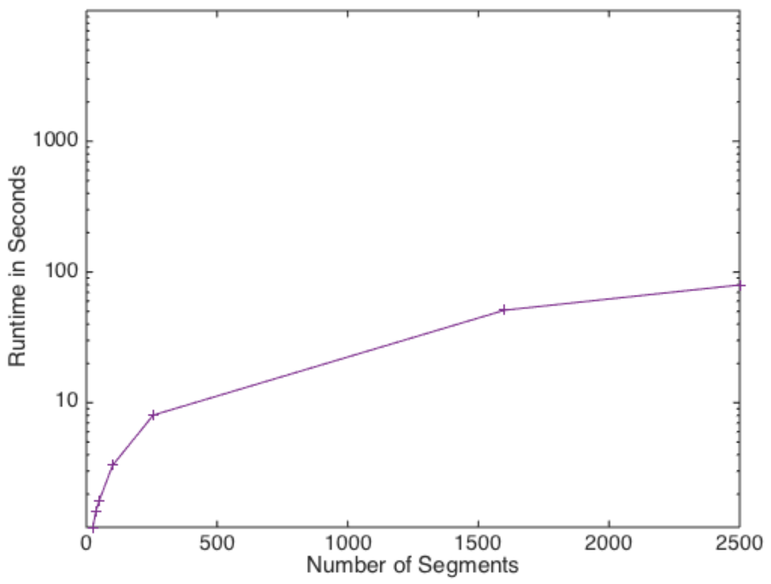

Figure 6: Average Runtime

Table 1: Evaluation numbers for different K

K	25	36	49	100	256	1600	2500
AvgRecall	0.17	0.1723	0.1807	0.1768	0.2253	0.3327	0.3724
AvgUnder	0.6699	0.6478	0.6206	0.6058	0.5561	0.5361	0.5378
AvgRuntime	1.1151	1.4641	1.7737	3.3117	8.0380	51.0373	79.6368

f) Extra Credit: Using Lab Color Space

In my implementation, I implemented the algorithm using Lab* color space instead of RGB color.

2. EM Algorithm: Dealing with Bad Annotations

2.1 Derivation of EM Algorithm:

EM algorithm is an machine learning algorithm to find maximum likelihood, by iteratively performing E (Expectation) step and M (Maximize) step. It is used to estimate the parameters of statistical models.

For this particular problem, the target function is maximize a posterior probability (MAP) of annotations given all parameters μ_i, σ(mean and standard deviation) and Good / Bad user assignments m_j and the prior to be a good annotator β. The likelihood is defined as the logarithm of posterior probability:

$$\begin{aligned}
\mathcal{L}(X, M, \theta) &= \log\left(\prod_i \prod_{j\in\mathcal{A}(i)} p(x_{i,j}, m_j; \theta)\right) \\
&= \sum_i \sum_{j\in\mathcal{A}(i)} \log p(x_{i,j}, m_j; \theta) \\
&= \sum_i \sum_{j\in\mathcal{A}(i)} \log p(x_{i,j}; m_j, \theta) p(m_j; \theta)
\end{aligned}$$

where θ is the parameter set ($\sigma, \{\mu_i\}, \beta$), $\mathcal{A}_i$ is the set of annotators who labeled image i. Assignment probability $p(m_j = 1; \theta) = \beta$ and $p(m_j = 0; \theta) = 1 - \beta$.

E-Step: taking the conditional expectation of log-likelihood, given all data samples $\{x_{i,j}\}$ and parameters in previous iteration θ^{t-1}:

$$\begin{aligned}
\mathbf{E}[\mathcal{L}|\mathcal{X}, \theta^{t-1}] &= \sum_i \sum_{j\in\mathcal{A}(i)} \mathbf{E}[\log(p(x_{i,j}; m_j, \theta)p(m_j; \theta))|\mathcal{X}, \theta^{t-1}] \\
&= \sum_i \sum_{j\in\mathcal{A}(i)} p(m_j = 1|x_{i,j}, \theta^{t-1}) \log\left(p(x_{i,j}; m_j = 1, \theta)\beta\right) \\
&+ p(m_j = 0; x_{i,j}, \theta^{t-1}) \log\left(p(x_{i,j}; m_j = 0, \theta)(1-\beta)\right) \\
&= \sum_i \sum_{j\in\mathcal{A}(i)} \alpha_{i,j} \log\left(\frac{\beta}{\sqrt{2\pi}\sigma} \exp -\frac{(x_{i,j} - \mu_i)^2}{2\sigma^2}\right) \\
&+ (1 - \alpha_{i,j}) \log(\frac{1-\beta}{10})
\end{aligned}$$

where

$$\begin{aligned}
\alpha_{i,j} &= p(m_j = 1; x_{i,j}, \theta^{t-1}) \\
&= \frac{p(x_{i,j}; m_j = 1, \theta^{t-1})p(m_j = 1; \theta^{t-1})}{p(x_{i,j}; \theta^{t-1})} \\
&= \frac{p(x_{i,j}; m_j = 1, \theta^{t-1})p(m_j = 1; \theta^{t-1})}{\sum_{m_j\in\{0,1\}} p(x_{i,j}; m_j, \theta^{t-1})p(m_j; \theta^{t-1})} \\
&= \frac{\frac{1}{\sqrt{2\pi}\sigma(t-1)} \exp(-\frac{(x_{i,j} - \mu(t-1)_i)^2}{2\sigma(t-1)^2})\beta(t-1)}{\frac{1-\beta(t-1)}{10} + \frac{1}{\sqrt{2\pi}\sigma(t-1)} \exp(-\frac{(x_{i,j} - \mu(t-1)_i)^2}{2\sigma(t-1)^2})\beta(t-1)}
\end{aligned}$$

M-Step:

Maximizing conditional expectation, using partial derivatives.

$$\frac{\partial\mathcal{L}}{\partial\mu_i} = k \cdot \sum_{j\in\mathcal{A}_i} \alpha_{i,j}(x_{i,j} - \mu_i) = 0$$

where *k* is a constant.

Solve:

$$\mu_i = \frac{\sum_{j \in \mathcal{A}_i} \alpha_{i,j} x_{i,j}}{\sum_{j \in \mathcal{A}_i} \alpha_{i,j}}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma} = \sum_i \sum_{j \in \mathcal{A}_i} \left(-\frac{\alpha_{i,j}}{\sigma} + \frac{\alpha_{i,j}(x_{i,j} - \mu_i)^2}{\sigma^3} \right) = 0$$

with $\sigma \neq 0$, solve σ:

$$\sigma = \sqrt{\frac{\sum_i \sum_j (\alpha_{i,j}(x_{i,j} - \mu_i)^2)}{\sum_i \sum_j \alpha_{i,j}}}$$

Finally,

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_i \sum_{j \in \mathcal{A}_i} \left(\frac{\alpha_{i,j}}{\beta} - \frac{1 - \alpha_{i,j}}{1 - \beta} \right) = 0$$

Solve:

$$\beta_j = \frac{\sum_{i \in S_j} \alpha_{i,j}}{N_j}$$

where N_j is the total number of annotations given by user j, and S_j is the set of images labeled by user j.

2.2 Application to Data

Please refer to the Matlab code for implementation.

(a) "Bad" annotators:

As shown in Figure 7, it is the value of β for each annotator. Use 0.1 as threshold, annotator $[5, 6, 8, 12, 16, 19, 21, 22]$ are bad annotators

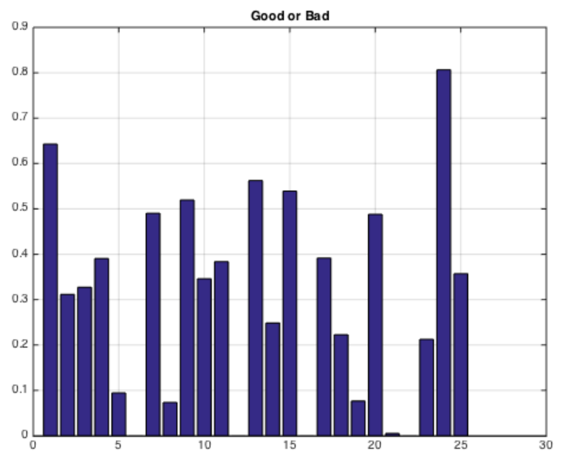

Figure 7: “Bad” annotators

(b) Estimated value of $\sigma = 0.1991$.

(c) Plot estimated mean values of image scores:

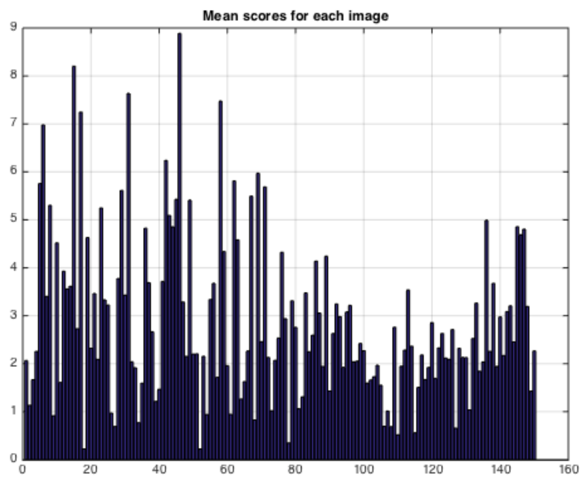

Figure 8: Mean ratings of images

3. Graph-cut Segmentation

a) Explain your foreground and background likelihood function

To estimate the foreground and background likelihood function, two Gaussian Mixture Models are used to estimate the distribution of foreground pixel colors and background respectively. And the likelihood is estimated as the logarithm of probability density functions (pdf). In Matlab implementation, the GMM is estimated using fitgmdist function, and the pdf is estimated using pdf function.

b) Unary and Pairwise term functions

The unary function is defined as:

$$unary_{potential}(x) = -\log\left(\frac{p(c(x);\theta_{foreground})}{p(c(x);\theta_{background}}\right)$$

where $p(c(x);\theta_{foreground})$ is the probability density of pixel x being the foreground, similarly $p(c(x);\theta_{background})$ is the probability being background.

As for the pairwise potentials, it is defined as a fix term k_1 for classifying adjacent pixels into different classes, and edge response defined by image gradient with weight k_2. The whole pairwise potential is defined as:

$$potential_{edge}(x,y) = k_1 + k_2 \exp\{\frac{-\|c(x)-c(y)\|^2}{2\sigma^2}\}$$

In implementation, since we only consider the pixels x and y nearby, the color difference term could be calculated using image gradient $\Delta I(x,y)$.

c) *Foreground and Background Map*

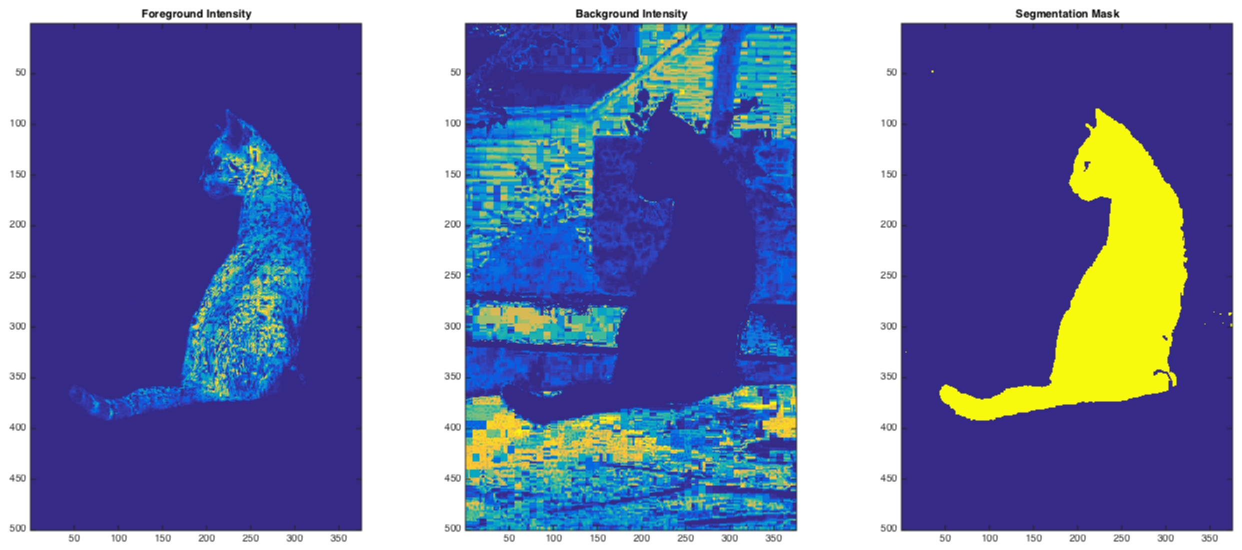

Figure 9: Foreground and Background Map: left) foreground map; middle) background map; right) segmentation mask

d) *Final Segmentation*

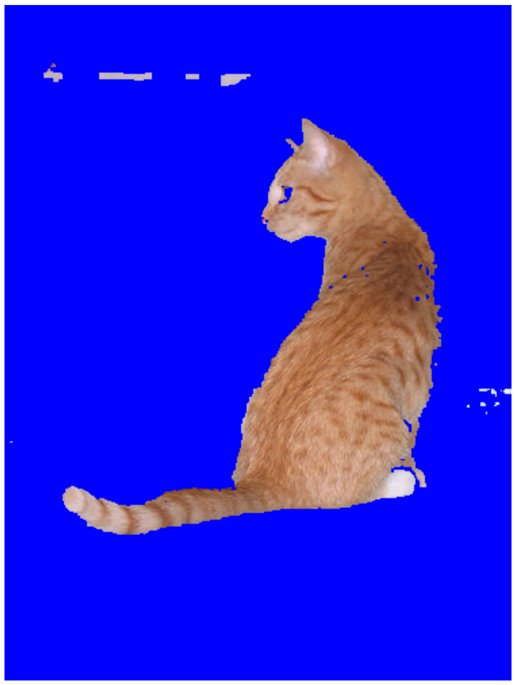

Figure 10: Final Segmentation using Graph-Cut