Project 2

In this project, the goal is to segment rover from terrain. As usual grad student will have to segment the rover shadow as well. You will compare 5 methods for segmenting the images that you used in Project 1.

1) Felzenszwalb's algorithm on its own with parameters set so that to obtain the final segmentation into rover and background
2) Felzenszwalb's algorithm with parameters set so that an over-segmentation is obtained. The Region adjacency graph (RAG) is built on the segments and hierarchical merging is used on the RAG to obtain the final segmentation into rover and background.
3) SLIC superpixels (over)segmentation. The Region adjacency graph (RAG) is built on the segments and hierarchical merging is used on the RAG to obtain the final segmentation into rover and background.
4) Try a clustering algorithm on top of the SLIC oversegmentation
5) Try a clustering algorithm on top of the Felzenswalb's oversegmentation

The grad students will also try ncuts on the RAGs constructed on the superpixel segmentation and the Felzenszwalb's over-segmentation.

Note: In order to obtain a good superpixel segmentation or Felzenszwalb oversegmentation you should try different configurations of parameters so that to minimize the occurrence of segments that include the boundary between rover and background. In other words, we would like to have superpixels or segments whose boundary coincide with the boundary between rover and background.

Felzenszwalb's efficient graph-based segmentation

The fast image segmentation algorithm proposed in [1] is popular in the computer vision community. The algorithm has a single scale parameter that influences the segment size. The actual size and number of segments can vary greatly, depending on local contrast.

Note that the graph that the code use is an 8-neighbor connectivity graph where the Euclidean distance in color space between the neighboring nodes.

[1] Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Python implementation: Use the function `segmentation.felzenszwalb` in the scikit-image package.

SLIC - K-Means based image segmentation

This algorithm simply performs K-means in the 5d space of color information and image location. As the clustering method is simpler, it is very efficient. It is essential for this algorithm to work in Lab color space to obtain good results. The `compactness` parameter trades off color-similarity and proximity (it is the m in the slides and paper [2]).

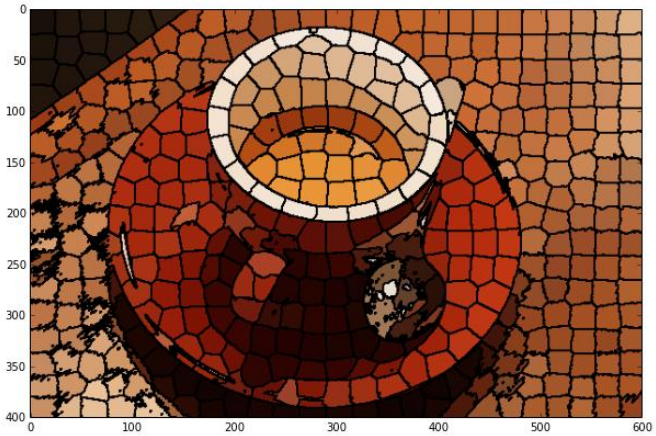
[2] Radhakrishna Achanta,, Appu Shaji, Kevin Smith,, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, SLIC Superpixels Compared to State-of-the-Art Superpixel Methods IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 34, NO. 11, NOVEMBER 2012

Python implementation: Use the function `segmentation.slic` in the scikit-image package.

Grad students: You should understand and try the SLIC0 method for SLIC in which the compactness parameter is adaptively modified. Compare with the best result obtained by a constant value of compactness. You can find reference to the SLIC0 method at https://ivrl.epfl.ch/research-2/research-current/research-superpixels/#SLICO

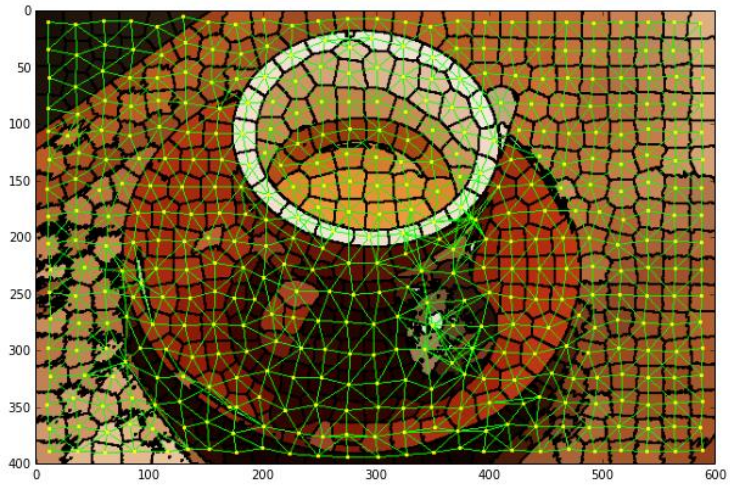
Region Adjacency Graphs

We start from an oversegmentation (Felzenszwalb or SLIC, slic in the picture below).



Region Adjacency Graph (RAG), as the name suggests, represent adjacency of regions with a graph. Each region (segment) in the image is a node in a graph. There is an edge between every pair of adjacent regions (regions whose pixels are adjacent). The weight of between every two nodes can be defined in a variety of ways. For this example, we will use the difference of

average color between two regions as their edge weight. The more similar the regions, the lesser the weight between them.



Python implementation: `graph.rag_mean_color` in the scikit-Image package.

Hierarchical merging of the RAG

You are going to use a function that greedily merges the most similar pair of nodes until no edges lower than *thresh* remain. The similarity is still the difference between the average weights of two segments.

Python Implementation: in you python code first import the functions below from the provided file `plot_rag_merge.py` with the command

```
>>from plot_rag_merge import _weight_mean_color, merge_mean_color
```

Then use `graph.merge_hierarchical` from the scikit-image package with these functions as parameters.

Grad students: NCuts on the RAG

In addition to hierarchical merging you will perform as an alternative Ncuts on the RAG formed by the segments obtained by the Felzenswalb and SLIC algorithms. The algorithm is described in the paper [3].

Python implementation: use the function graph.cut_normalized in the scikit-image package.

[3] Shi, J.; Malik, J., “Normalized cuts and image segmentation”, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 8, pp. 888-905, August 2000.