Image Segmentation using

SLIC SuperPixels

and DBSCAN Clustering

Peter Kovesi
Image Analysis Group
Centre for Exploration Targeting
The University of Western Australia

April 2013

This segmentation approach makes use of Achanta et al's SLIC superpixels and the DBSCAN clustering algorithm. Application of the SLIC superpixel algorithm forms an over-segmentation of an image. These superpixels are then processed using the DBSCAN algorithm to form clusters of superpixels to generate the final segmentation. The approach is simple and relatively fast.

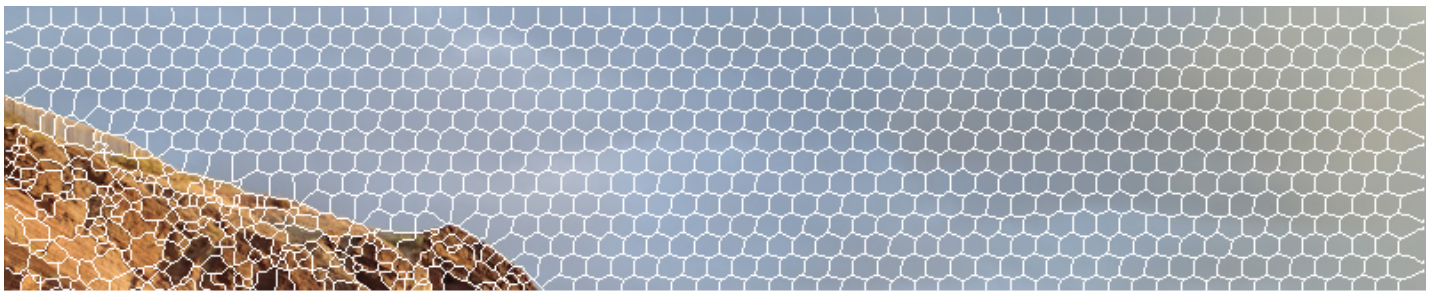
The speed of the DBSCAN clustering process is grearly facilitated by forming an adjacency matrix of the regions produced by the super-pixelization process. This constrains the number of distance measurement tests required

References:

- R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Susstrunk. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods" PAMI. Vol 34 No 11. November 2012. pp 2274-2281.
- Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226-231.

Example Segmentation

Original image. Hallett Cove, South Australia



Superpixels generated by SLIC

The following code segments the image into 3000 superpixels using a weighting factor of 10 relating spatial distances to colour distances, resulting superpixels of area less than 10 pixels are eliminated, and superpixel attributes are computed from the median colour values. Note I prefer to use a relative low weighting of spatial distance to colour distance in forming superpixels. The SLIC algorithm does not enforce continuity of superpixels which means that superpixels can break up unto multiple regions if the pixels within a region are significantly different and the superpixel weighting of spatial distance to colour distance is low. This is not a problem, indeed I consider it an advantage because it means that with a low weighting you reduce the chance of undersegmenting an image and the fidelity of feature boundaries is better preserved. The disjoint superpixels are made distinct by the functions cleanupregions.m or mcleanupregions.m. Note also that the superpixel seed points have been initialised in a hexagonal grid rather than a square one. The idea is that this results in a segmentation that will be nominally 6-connected which hopefully facilitates any subsequent post-processing that seeks to merge superpixels.

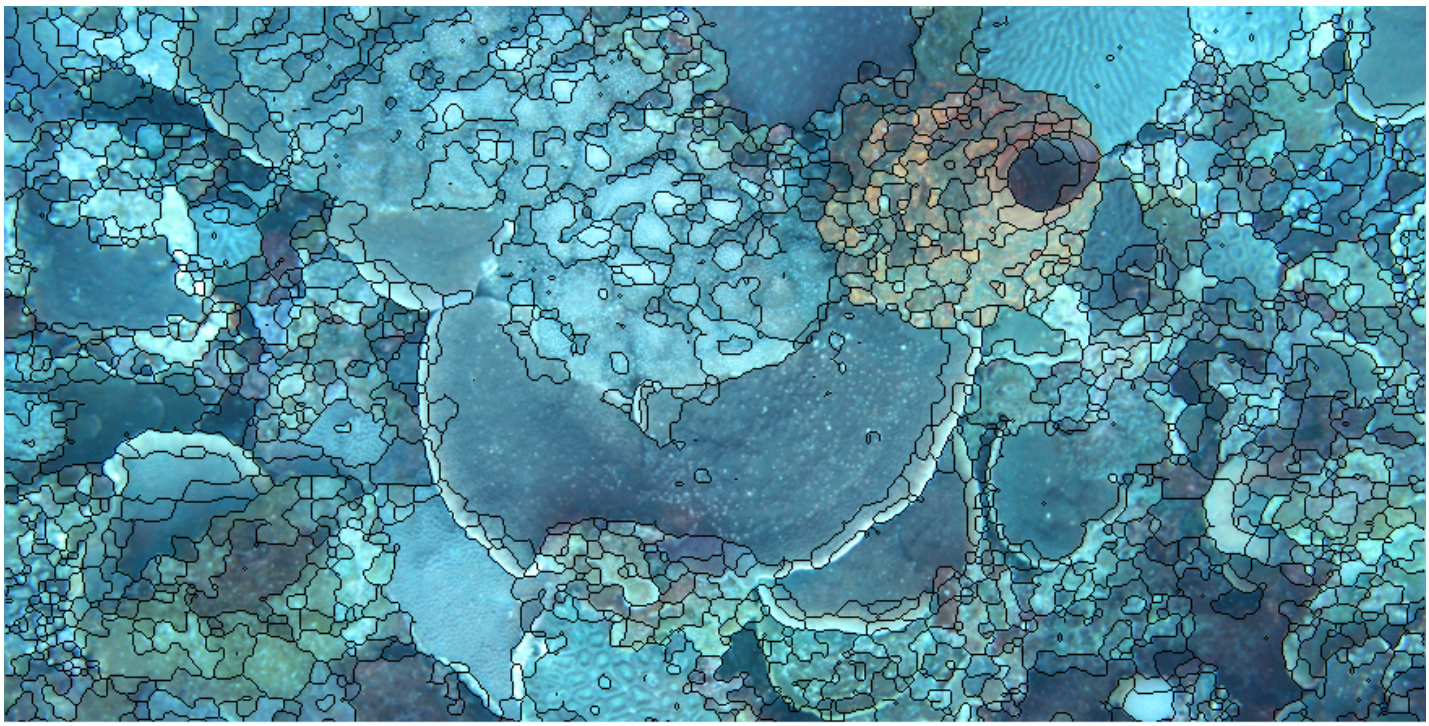
```
>> [l, Am, C] = slic(im, 3000, 10, 1, 'median');
>> show(drawregionboundaries(l, im, [255 255 255]))
```

Correction to code above 14/12/2014: In the call to slic.m the 4th argument, the structuring element radius, should be a value of around 1 to 1.5, not 10 as this web page originally suggested. A value of 10 causes most of the superpixels to be merged to produce just 3 superpixels. Apologies for the confusion and frustration. Thanks to Bin Chen for pointing this out.

Clustering of superpixels using DBSCAN

Here I have chosen a L*a*b* colour difference threshold of 5. Small changes in this value can have a significant effect.

```
>> lc = spdbscan(l, C, Am, 5);
>> show(drawregionboundaries(lc, im, [255 255 255]))
```



An example of this approach applied to a coral image. As with the Hallett Cove image the result is somewhat oversegmented (and undersegmented in some areas too). Segmentation based just on colour is not sufficient, some texture information is needed too.

MATLAB code:

- slic.m Implementation of Achanta, Shaji, Smith, Lucchi, Fua and Susstrunk's SLIC Superpixels.
- spdbscan.m Implements DBSCAN clustering of superpixels.
- cleanupregions.m Cleans up small regions in a segmentation. Used by slic.m
- mcleanupregions.m Morphological version of cleanupregions.m The output is not quite as nice but the execution is much faster.
- finddisconnected.m Finds groupings of disconnected labeled regions. Used by mcleanupregions.m to reduce execution time.
- makeregionsdistinct.m Ensures labeled regions are distinct.
- renumberregions.m Ensures all regions in labeled image have a unique label and that the label numbering forms a contiguous sequence.
- regionadjacency.m Computes adjacency matrix for an image of labeled segmented regions.
- drawregionboundaries.m Draw boundaries of labeled regions in an image.
- maskimage.m used by drawregionboundaries.m
- rgb2lab.m Convenience function for converting an image from RGB to L*a*b* colour space.
- circularstruct.m Generates a circular structuring element, used by mcleanupregions.m
- dbscan.m Basic implementation of DBSCAN

- testdbscan.m Function to test/demonstrate dbscan.m

It is possible that I have missed some function dependencies above. If so rummage through the rest of my functions at MATLAB and Octave Functions for Computer Vision and Image Processing

---