Manifold SLIC: A Fast Method to Compute Content-Sensitive Superpixels (Supplementary Material)

Yong-Jin Liu, Cheng-Chi Yu*, Min-Jing Yu*
Tsinghua University, China
{liuyongjin,ycc13,yumj14}@tsinghua.edu.cn

Ying He
Nanyang Technological University, Singapore
YHe@ntu.edu.sg

1. Further Discussion on Splitting and Merging Operators

The manifold SLIC algorithm is largely based on the Lloyd method to compute restricted CVT on the image manifold $\mathcal{M}$. The major difference is the splitting and merging operators (see lines 12 and 15 in Algorithm 2), which are not available in the conventional Lloyd method. It is known that the Lloyd method has only linear convergence rate and produces a local optimal solution. Although global optimization tools, such as simulated annealing, can improve the quality, they are computationally expensive and not practical for high-resolution images. We observe that the splitting and merging operators are low-cost heuristic that can improve *both* runtime performance and quality of the content-sensitive superpixels. Figure S4 demonstrates that those operators lead to superpixels with better distribution, i.e., there are more superpixels in content-rich regions and fewer superpixels in content-sparse regions. In our implementation, we use a fixed iteration number 20. In this supplementary material, we also present the pseudo-codes of the two operators to ease implementation and Figure S1 shows an example.

Function 1 $split(\lambda_i, s_i)$

1: Generate four new seeds $s_{i1} = (u_i - \frac{\lambda_i S}{2}, v_i + \frac{\lambda_i S}{2})$, $s_{i2} = (u_i - \frac{\lambda_i S}{2}, v_i - \frac{\lambda_i S}{2})$, $s_{i3} = (u_i + \frac{\lambda_i S}{2}, v_i - \frac{\lambda_i S}{2})$ and $s_{i4} = (u_i + \frac{\lambda_i S}{2}, v_i + \frac{\lambda_i S}{2})$.
2: Set $\lambda_{i1} = \lambda_{i2} = \lambda_{i3} = \lambda_{i4} = \lambda_i/2$.

2. Additional Results & Comparison

Figure S2 shows several typical results of manifold SLIC. We can clearly see that the superpixels adhere to image boundaries well and they are content sensitive, i.e., superpixels are small in content-dense regions and large in content-sparse regions. The content sensitive feature is due to the fact that regions of high color variance have larger areas on $\mathcal{M}$. Figure S3 shows more results on the BSDS500 benchmark [1].

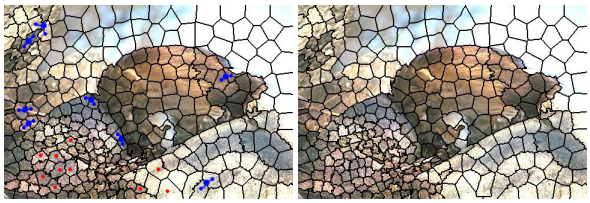

Figure S1. *Left*: Red dots denote the to-be-split superpixels. Blue dots on the edges mean the two neighboring superpixels (with small blue dots in the center) will be merged. *Right*: Superpixels after splitting and merging.

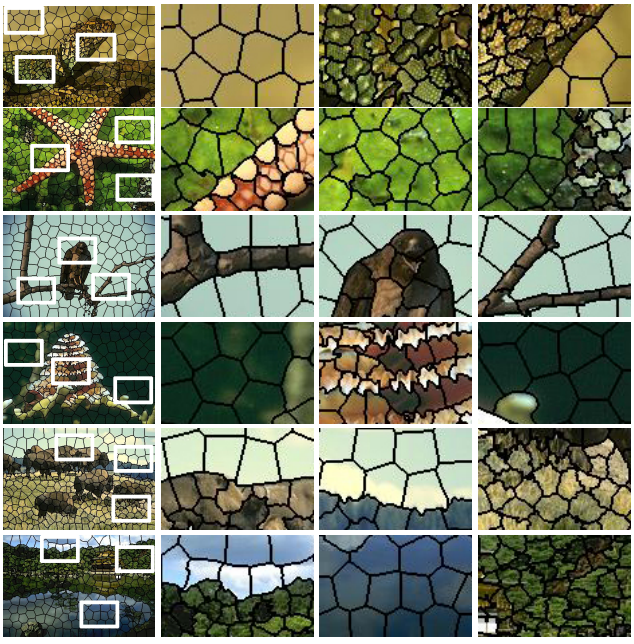

Figure S2. Experimental results. We compute 300 superpixels for each image. Content sensitivity can be verified in the closeup views.

*C. Yu and M. Yu contributed equally to this paper

[1] https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

Function 2 $merge\_voronoi\_cells(I)$

Require: A labeled image I in which every pixel has a label i (means in the superpixel of seed s_i). The image is represented in two data structures: a 2D array $I(u, v)$ and a four-connectivity graph G of pixels.

Ensure: The image with updated labels after merging superpixels.

Initialize the flag $visit = FALSE$ for each pixel.
for each pixel p in I with flag $visit == FALSE$ **do**
  Set l be the label of p.
  In G, depth-first search the nodes of label l to find unvisited neighboring superpixels and set the flag $visit = TRUE$ for all the pixels of label l in I.
  Put all unvisited neighboring seeds into a list $unlist$ and set $merge = FALSE$.
  if $unlist$ is not empty and $Area(\mathcal{V}_{\mathcal{M}}(\Phi(s_l))) < Area(\mathcal{M})/8K$ **then**
    Pop a seed s_k from $unlist$.
    Merge s_l and s_k: in G depth-first search the nodes of label k; set the flag $visit = TRUE$ for all the pixels of label k in I and relabel these pixels to be l; compute the new position of merged seed.
    Set $merge = TRUE$.
  else
    while $unlist$ is not empty and $merge == FALSE$ **do**
      Pop a seed s_k from $unlist$.
      if $Area(\mathcal{V}_{\mathcal{M}}(\Phi(s_i)) \bigcup \mathcal{V}_{\mathcal{M}}(\Phi(s_j))) < \Xi/5$ **then**
        Merge s_l and s_k: in G depth-first search the nodes of label k; set the flag $visit = TRUE$ for all the pixels of label k in I and relabel these pixels to be l; compute the new position of merged seed.
        Set $merge = TRUE$.
      end if
      Remove s_k from $unlist$.
    end while
  end if
end for
if $merge == TRUE$ **then**
  Relabel the pixels so that their IDs are in sequence.
end if

As mentioned in Section 1 in the main paper, manifold SLIC is inspired by the simplicity and high performance of SLIC and the content-aware nature of SSS. We observe that the superpixels produced by manifold SLIC are of similar quality to those by SSS, but manifold SLIC runs 10 times faster than SSS. Computational results also show that manifold SLIC outperforms SLIC in terms of under segmentation error, boundary recall ratio, and achievable segmentation accuracy. Note that both SLIC and manifold SLIC has an $O(N)$ time complexity, which is independent of the number of superpixels K. Figure S4 provides additional results demonstrating that the content-sensitive nature of manifold SLIC.

Figure S3. More experimental results by manifold SLIC.

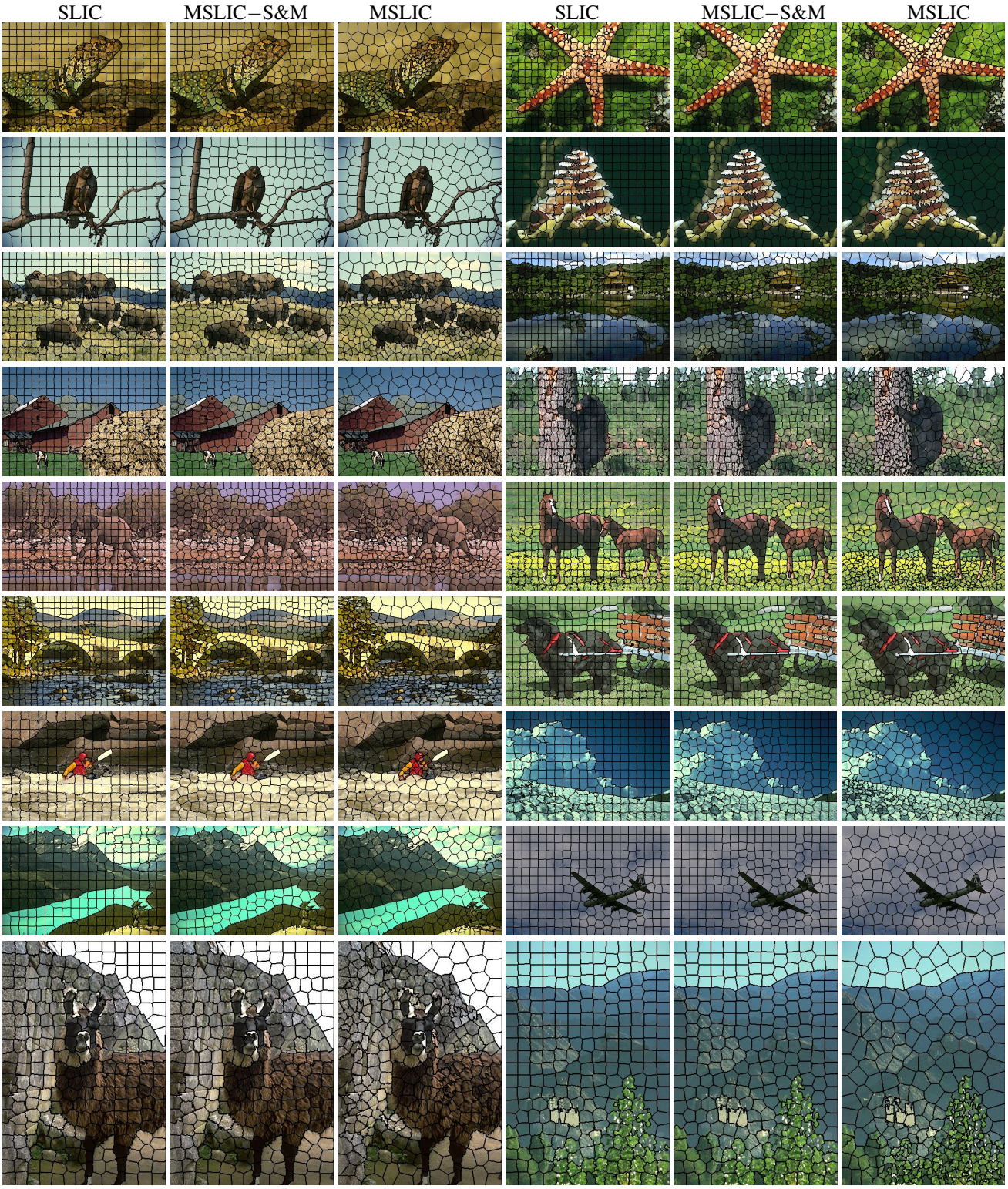

Figure S4. Comparison of SLIC, manifold SLIC with and without the splitting and merging operators, denoted by MSLIC and MSLIC−S&M, respectively. We set the compactness parameter to 40 in SLIC.