Resolution-independent meshes of superpixels

Vitaliy Kurlin and Philip Smith

Department of Computer Science, University of Liverpool

Abstract. The over-segmentation into superpixels is an important pre-processing step to smartly compress the input size and speed up higher level tasks. A superpixel was traditionally considered as a small cluster of square-based pixels that have similar color intensities and are closely located to each other. In this discrete model the boundaries of superpixels often have irregular zigzags consisting of horizontal or vertical edges from a given pixel grid. However digital images represent a continuous world, hence the following continuous model in the resolution-independent formulation can be more suitable for the reconstruction problem.

Instead of uniting squares in a grid, a resolution-independent superpixel is defined as a polygon that has straight edges with any possible slope at subpixel resolution. The harder continuous version of the over-segmentation problem is to split an image into polygons and find a best (say, constant) color of each polygon so that the resulting colored mesh well approximates the given image. Such a mesh of polygons can be rendered at any higher resolution with all edges kept straight.

We propose a fast conversion of any traditional superpixels into polygons and guarantees that their straight edges do not intersect. The meshes based on the superpixels SEEDS (Superpixels Extracted via Energy-Driven Sampling) and SLIC (Simple Linear Iterative Clustering) are compared with past meshes based on the Line Segment Detector. The experiments on the Berkeley Segmentation Database confirm that the new superpixels have more compact shapes than pixel-based superpixels.

1 Introduction

1.1 Over-segmentation for low-level vision

The important problem in low-level vision is to quickly detect key structures such as corners and edges where color intensities substantially change. The over-segmentation problem is to split an image into *superpixels*, which are small patches of square-based pixels having similar colors and positions.

Traditional superpixels often have irregular shapes with zigzag boundaries of only horizontal or vertical short edges. This rigid discretization can be avoided if we allow edges of any length and direction, because continuous objects are much better represented by polygons not restricted to a given pixel grid.

The color intensity in real images always changes gradually over 2-3 pixels without jumps, see [18, Fig. 1]. Hence an edge between different objects can be inside a square pixel, not along its sides. These hurdles disappear if we look for *continuous objects* represented by pixel values discretely sample on a grid.
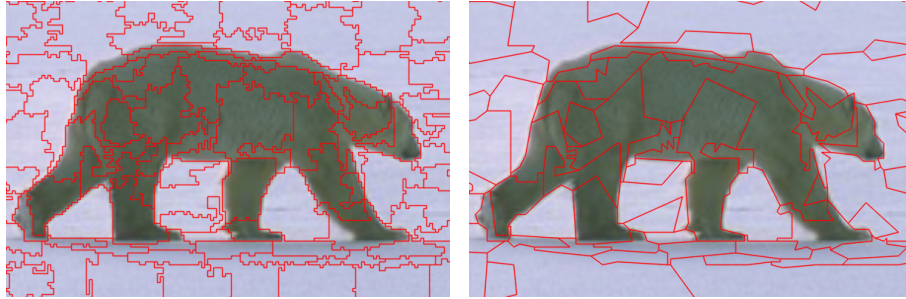
Fig. 1. SLIC superpixels (left) with zigzag boundaries of pixel-based superpixels are converted into a resolution-independent mesh (right) of polygons with straight edges that can be rendered at any higher resolution for better and smoother animations.

1.2   Resolution-independent polygonal superpixels

Digital images represent a continuous world around us, but are restricted to a fixed pixel grid. We consider the over-segmentation problem in the following *resolution-independent* formulation introduced by Viola et al. [18].

We split an image into a fixed number of possibly non-convex polygons so that

• all polygons have straight edges and vertices with any real coordinates (not restricted to a given pixel grid, so independent of an initial image resolution);

• the resulting colored mesh (with a best constant color over each polygon) approximates the original image, e.g. by minimizing an energy in section 4.

Such a polygonal mesh can be rendered at any higher resolution and is called *resolution-independent*. In general, a *mesh* for a rectangular image $I \subset \mathbb{R}^2$ is a graph $G \subset \mathbb{R}^2$ that contains the boundary of I and consists of non-intersecting line segments that split I into possibly non-convex polygons. Fig. 2 and 18 show long thin superpixels for the tripod legs in the famous cameraman image.

1.3   Key contributions to the state-of-the-art for superpixels

• We solve the over-segmentation problem for polygonal *resolution-independent* superpixels that have few straight edges with infinitely many possible slopes.

• The algorithm RIMe in section 3 can convert any pixel-based superpixels into a resolution-independent mesh with quality guarantees in Theorem 3.

• The experiments in section 4 confirm that the resolution-independent meshes based on SEEDS and SLIC superpixels, achieve better results on objective measures, perform similarly to SEEDS and SLIC on the BSD benchmarks.

• RIMe beats all other resolution-independent superpixels on the objective reconstruction error and benchmarks of the Berkeley Segmentation Database [2].
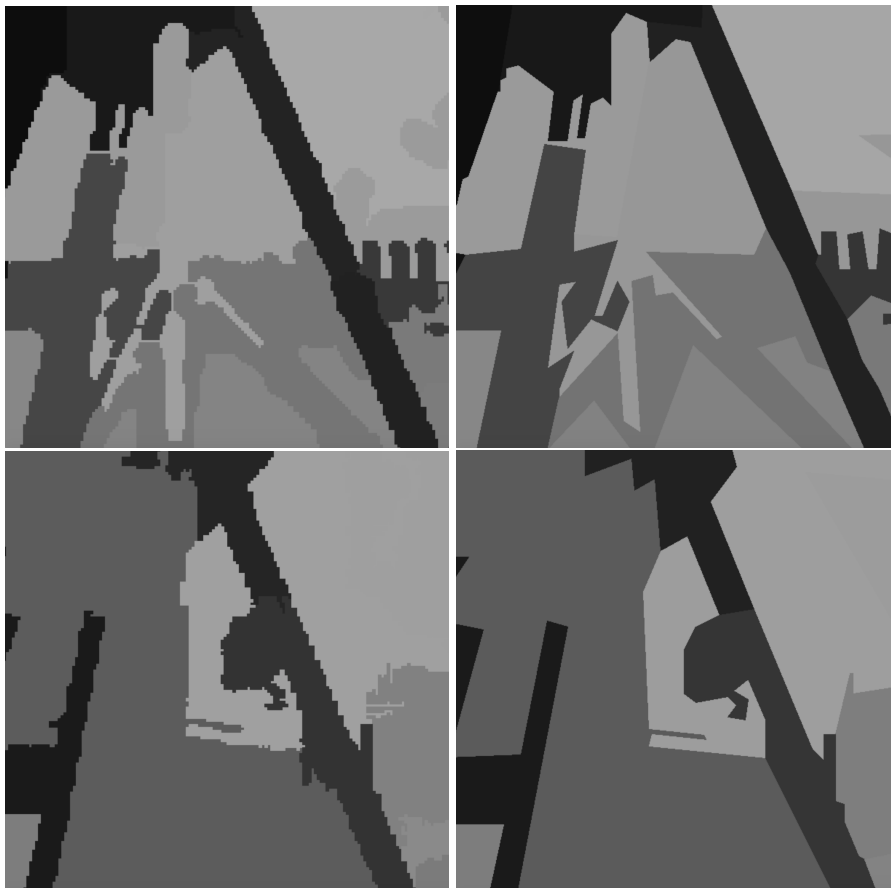
Fig. 2. The pixel-based superpixels on the left (SEEDS and SLIC) have irregular zigzag boundaries, which are straightened by the algorithm RIMe on the right.

2   Review of the past work on superpixels

We review the widely used algorithms for pixel-based and resolution-independent superpixels for the harder reconstruction problem of continuous real-life objects.

2.1   Pixel-based superpixel algorithms

The first successful algorithms were based on the graph of the 4-connected pixel grid [16, 5, 11]. The *Lattice Cut* algorithm by Moore et al. [15] guarantees that the final mesh of superpixels is regular as the original grid of pixels. The best quality in this category is achieved by Entropy Rate Superpixels (ERS) of Lie et al. [13] minimizing the entropy rate of a random walk on a graph. Based on *Compact*

Superpixels by Veksler and Boykov [17], the fastest algorithm is by Zhang et al. [20] processing an average image from BSD500 in 0.5 sec. Our experiments will use the algorithms SEEDS, SLIC from OpenCV, VLFeat libraries.

The *Simple Linear Iterative Clustering* (SLIC) algorithm by Achanta et al. [1] forms superpixels by k-means clustering in a 5-dimensional space using 3 colors and 2 coordinates per pixel. Because the search is restricted to a neighborhood of a given size, the complexity is $O(kmn)$, where n and m are the numbers of pixels and iterations. The later *Linear Spectral Clustering* (LSC) by Li et al. [12] is based on a weighted k-means clustering in a 10-dimensional space. The SMURF algorithm by Luengo et al [14] obtains superpixels in a parallelized way within larger super-regions and alternates split/merge steps at several levels.

SEEDS (Superpixels Extracted via Energy-Driven Sampling) by Van den Bergh et al. [3] seems the first superpixel algorithm to use a *coarse-to-fine optimization* that progressively refines superpixels. At the initial coarse level, each superpixel consists of large rectangular blocks of pixels. At the next level, all blocks are subdivided into four rectangles and any boundary block can move to an adjacent superpixel, an so on until all blocks become pixels. SEEDS puts the colors of all pixels within each fixed superpixel are put in bins (five for each color channel) and iteratively maximizes the sum of deviations of all bins from an average bin within every superpixel. The color deviation is maximal for a superpixel whose pixels have colors in one bin.

The similar Coarse-to-Fine (CtF) algorithm by Yao et al. [19] minimizes the discrete Reconstruction Error from section 4. The recent improvement of this Coarse-to-Fine approach [9] allows the user to make shapes of superpixels more round by giving more weight to an isoperimetric quotient.

2.2   Edge detection at subpixel resolution

Both past algorithms for resolution-independent superpixels (Voronoi and CCM: Convex Constrained Meshes) are based on the Line Segment Detector algorithm (LSDA), which outputs line segments at subpixel resolution [8]. The parameters are a tolerance τ for angles between gradients and a threshold ϵ for false alarms. For the values $\tau = 22.5°$, $\epsilon = 1$ and the random model of a uniformly distributed gradient field, the LSDA outputs on average at most one false positive.

The LSDA edges have endpoints and gradients with any real coordinates, but the use has no control over a number of line segments in the output. The recent persistence-based line segment detector [10] guarantees edges without intersections and small angles.

2.3   Resolution-independent polygonal superpixels

The *Voronoi superpixels* by Duan and Lafarge [4] split an image into polygons not restricted to a pixel grid. For points $p_1, \ldots, p_k$ (called *centers*), the *Voronoi cell* $V(p_i)$ is the polygonal neighborhood of the center p_i consisting of all points closer (in the Euclidean distance) to p_i than to other centers. These centers are

chosen on both sides of each LSDA edge so that all LSDA edges are covered by the boundaries of Voronoi cells, though no guarantees were proved. The main advantage of Voronoi superpixels is their almost "round" shape, see section 4.

The CCM superpixels (Convex Constrained Meshes) by Forsythe et al. [6] directly include LSDA edges as hard constraints, which improves the Boundary Recall, see section 4. After post-processing LSDA edges to get a straight line graph without self-intersections, this graph is converted into a full mesh of convex polygons that are guaranteed to have no angles smaller than $20°$. The boundaries of CCM superpixels are always in a small offset of LSDA edges.

Both Voronoi and CCM superpixels crucially depend on the quality of LSDA, which may not output a desired number of strongest edges. Pixel-based superpixels are better optimised for the Boundary Recall. The goal of the paper is to transfer this advantage to new resolution-independent superpixels.

3   RIMe: a resolution-independent mesh of polygons

3.1   RIMe algorithm: pipeline, input, parameters and output

The algorithm RIMe converts any pixel-based superpixels into a mesh of polygons whose straight edges approximate boundaries with guarantees in Theorem 3. The *input* is the matrix $s(p)$ of superpixel indices for every pixel $p = (i, j)$.
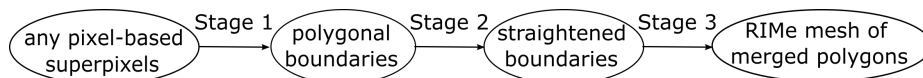

Fig. 3. Pipeline of the RIMe algorithm through 3 stages in subsection 3.2.

RIMe essentially uses the OpenMesh library and saves the final mesh to a .off file. One can find the most optimal color for every polygon and output the colored mesh as a small representation for further processing, see section 4.

3.2   Conversion algorithm RIMe step-by-step

Stage 1: Convert any traditional superpixels into a proper pixel-based mesh by extracting polygonal boundaries of any given pixel-based superpixels. All past algorithms output a matrix $s(p)$ of superpixel labels (integer indices) over all pixels p. Unfortunately, the union of pixels having the same label s is often disconnected (as a subset in the 4-connected or 8-connected grid). Even if connected, one superpixel can be surrounded by another superpixel (in this case the surrounding superpixel can be better split into two smaller superpixels). We convert any output into a mesh whose polygons are connected unions of pixels.

Step 1.1. Starting from an initial position at a corner of an image, we follow the boundary of a current polygon within the pixel grid and check at every pixel corner whether we should turn left/right or go forward as shown in Fig 4.
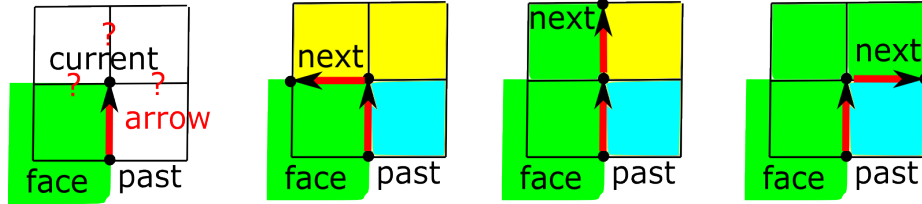

Fig. 4. Choosing the next point at a corner (colors indicate different superpixels).

Step 1.2. When we meet a new superpixel, we save an initial arrow (a directed edge between pixels) to make sure that we later go around the new superpixel.

Step 1.3. After we have returned to the initial position, a closed boundary of a superpixel was traced and we check if there are any unexplored superpixels.

Step 1.4. Check that all found superpixels have the expected areas, otherwise we find more connected components by looking at unexplored boundary pixels from the matrix of superpixel labels, so the number of superpixels can increase.

Step 1.5. If a found polygon still does not have the expected area, it must surround another polygon. We add a straight edge (say, D_1) between two closest vertices on the boundaries of the superpixels to split the surrounding superpixel into two. This edge D_1 cannot intersect another edge (say, D_2), which would contradict Lemma 1 for the quadrangle with the intersecting diagonals D_1, D_2.

Lemma 1 *In any convex quadrilateral with all sides longer than the shortest diagonal, the longest diagonal is longer than any side.*

Proof. Triangle inequality implies that the sum of the two diagonals is greater than the sum of either pair of opposite sides. Assuming that the shortest diagonal is shorter than any side, the longest diagonal should be longer than any side. $\square$

Stage 2: Straighten all boundaries of a mesh from Stage 1. The key advantage of RIMe is the possibility to *run Stage 2 in parallel* for different pairs of polygons.

Step 2.1: Find the edge chain (a sequence of successive non-boundary edges) between any two adjacent polygons.

Step 2.2: Given a chain between vertices A, B, find the maximum Euclidean distance d_C from an intermediate vertex C to the straight line AB, see Fig. 5.

Step 2.3: We replace the whole chain by the segment AB if

$$(1) \qquad \text{the max distance } d_C < 1 \text{ or } d_C < \frac{\text{Color\_Offset}}{\text{Color\_difference}},$$
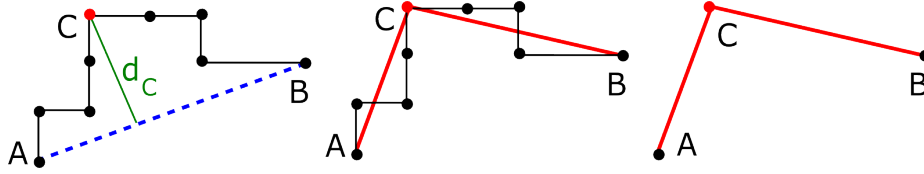
Fig. 5. Straightening polygonal chains of edges recursively in Step 2.3 when d_C is small.

where Color difference is the absolute difference between *average intensities* (estimated from pixel-based superpixels in Stage 1) of the polygons F_1, F_2 on both sides of the chain.

Step 2.4: For $d_C \geq 1$, we check that the new potential edge AB does not intersect any existing edges of the polygons F_1, F_2 sharing the chain. We check if AB has angles more than Min Angle all incident edges in the current mesh.

Step 2.5: If any condition in Steps 2.3-2.4 fails, we recursively straighten the subchains AC, CB as above.

Stage 3: Merge adjacent polygons whose average intensities differ by less than Max Color Dif. Many images have uniform backgrounds and superpixels consisting of pixels with the same intensity. Hence merging the resulting polygonal superpixels keeps important edges from a given image.

3.3 Theoretical guarantees (proved in appendices)

We assume that pixel-based superpixels are given with their average intensities, otherwise these averages can be quickly computed for Step 2.3. Apart from these averages, the algorithm RIMe accesses only a smaller number m of *boundary pixels* that have at least one neighboring pixel from a different superpixel.

Proposition 2 *The algorithm* RIMe *in section 3 has the* linear running time *in the number m of boundary pixels that are not strictly inside one superpixel.*

Let d be the Euclidean distance between points in $\mathbb{R}^2$. The *r-offset* (dilation with a disk of radius r) of any $S \subset \mathbb{R}^2$ is its thickened neighborhood $\{p \in \mathbb{R}^2 \: : \: d(p, S) \leq r\}$, where $d(p, S) = \min\{d(p, q) \: : \: q \in S\}$ is the distance from p to S.

Theorem 3 *The inequality $d_C < 1$ in (1) implies that*

(3a) for any pixel-based superpixels, the edges of a resolution-independent mesh can meet only at endpoints.

Moreover, the conversion algorithm RIMe *guarantees that*

(3b) all angles between incident edges in a final resolution-independent mesh are at least Min Angle,

(3c) any chain of edges between original superpixels with Color difference *is replaced by a polygonal line within the* $\dfrac{\text{Color\_Offset}}{\text{Color\_difference}}$ *-offset of the original chain.*
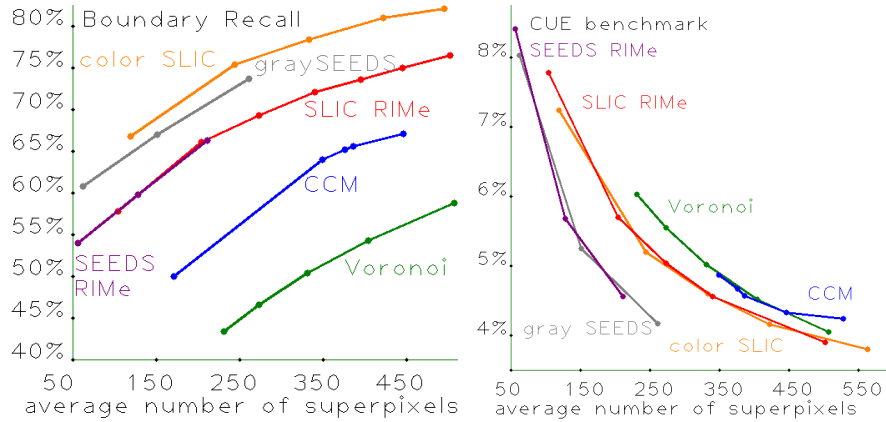
Fig. 6. Boundary Recall ($\uparrow$ better). Corrected Undersegmentation Error ($\downarrow$ better).

The value Color_Offset $= 30$ means that any chain between polygons whose average intensities differ by 10 is straightened within a 3-pixel neighborhood.

4   Experimental comparison of six algorithms on BSD

The *Berkeley Segmentation Database* BSD [2] has 500 images widely used for evaluating segmentation algorithms due to (sometimes imperfect) human-sketched ground truth boundaries. For an image I, let $I = \cup G_j$ be a segmentation into ground truth regions and $I = \cup_{i=1}^{k} S_i$ be an oversegmentation into superpixels produced by an algorithm. Each quality measure below compares the superpixels $S_1, \ldots, S_k$ with the best suitable ground truth for every image from the BSD.

Let $G(I) = \cup G_j$ be the union of ground truth boundary pixels and $B(I)$ be the set of boundary pixels produced by a superpixel algorithm. For a distance ε in pixels, the *Boundary Recall* $BR(\varepsilon)$ is the ratio of ground truth boundary pixels $p \in G(I)$ within 2 pixels from the superpixel boundary $B(I)$.

Van den Bergh et al. [3] suggested the *Corrected Undersegmentation Error* $CUE = \dfrac{1}{k} \sum_i |S_i - G_{max}(S_i)|$, where $G_{max}(S_i)$ is the ground truth region having the largest overlap with S_i. The *Achievable Segmentation Accuracy* is $ASA = \dfrac{1}{k} \sum_i \max_j |S_i \cap G_j|$. If a superpixel S_i is covered by a ground truth region G_j, then $|S_i \cap G_j| = |S_i|$ is the maximum value. Otherwise $\max_j |S_i \cap G_j|$ is the maximum area of S_i covered by the most overlapping region G_j. If we use superpixels for the higher level task of semantic segmentation, then ASA is the upper bound on the number of pixels that are wrongly assigned to final semantic regions. All values of BR, CUE, ASA are in [0,1] and can be measured in percents.
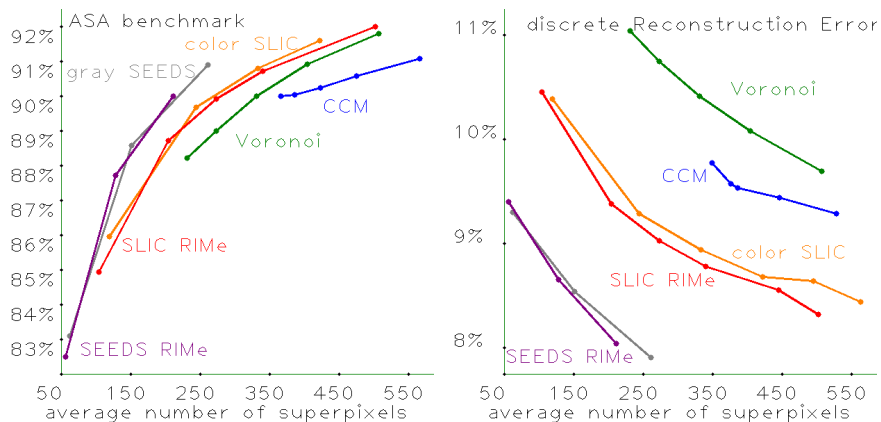
Fig. 7. Achievable Segm. Accuracy (↑ better). Discrete Reconstruction Error (↓ better).

The BSD benchmarks above involve some parameters, *e.g.* the 2-pixel offset for the Boundary Recall, which can be hard to justify. That is why several objective cost functions were proposed in an energy minimization framework. The main energy term for CtF superpixels [19] depends only on pixel intensities as follows and can be called the *discrete Reconstruction Error*:

$$dRE = \sum_{\text{pixels } p} \left|\left|\text{Intensity}_p - \text{average intensity of } S(p)\right|\right|^2, \text{ where}$$

$S(p)$ is the pixel-based superpixel containing a pixel p. For colored images, the intensity can be considered as a vector of 3 colors with any (say, Euclidean) norm. So *dRE objectively measures* how well the colored mesh (with average intensities over all superpixels) approximates the original image over all pixels.

Compactness measures for superpixels were used as regularizers by different authors [17, 19]. We propose the simplest version based on the isoperimetric quotient $Q(S) = \dfrac{4\pi \text{ area}(S)}{\text{perimeter}^2(S)}$, which has the maximum value 1 for a round disk S. The *Compactness* is the average $Comp = \sum\limits_{\text{superpixels S}} \dfrac{Q(S)}{\#\text{superpixels}}$.

Here are the parameters of the RIMe superpixels for benchmarking below.

• Color_Offset = 30 is used for straightening in Step 2.3 and controls approximation guarantees in Theorem 3.

• Max_Color_Dif = 2 is the maximum (grayscale) intensity difference for merging adjacent polygons in Stage 3 (larger values will lead to larger superpixels).

• Min_Angle = 30° (can be 0) is the minimum angle between adjacent edges (only to avoid narrow triangles).
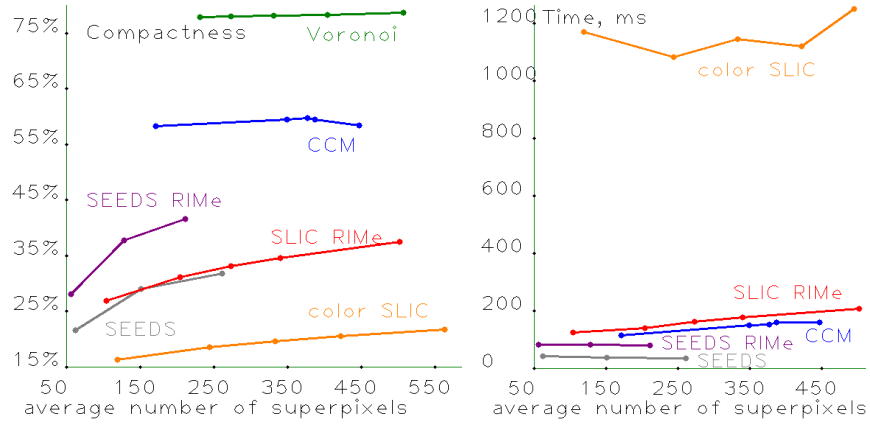
Fig. 8. Compactness (↑ better). Time on 8G RAM 2.6 GHz Intel Core i5 (↓ better).

Fig. 6, 7, 8 show 6 benchmarks for 6 superpixel algorithms. Each dot on the curves corresponds to a single run on 500 images and has the coordinates (average number of superpixels, average benchmark value over BSD).

The SLIC algorithm crucially uses 3 color values (converted to a Lab space) and we marked its curve as color SLIC. We ran SEEDS on grayscale version of BSD images and marked the curve by gray SEEDS, because both Voronoi and CCM algorithms accept only a grayscale input needed for LSDA edges. The corresponding outputs of RIMe are marked by SLIC RIMe, SEEDS RIMe.

The two remaining curves are for Voronoi [4] and CCM meshes [7]. Fig. 9 shows that RIMe conversions lead to visually better reconstructions than the only other resolution-independent superpixels on Voronoi and CCM meshes.

5   Conclusions, applications and further problems

Starting from any pixel-based superpixels, the RIMe conversion produces polygonal superpixels with almost the same BSD benchmarks BR, CUE, ASA and the objective error dRE in Figs. 6, 7. In comparison with any pixel-based superpixels, polygonal superpixels have much fewer edges (no long zigzags in boundaries) with slopes of any potential direction (not only horizontal or vertical).

The RIMe superpixels have twice better compactness (more "round") shapes than their original pixel-based superpixels such as SEEDS and SLIC in Fig. 8. The RIMe conversions of SLIC, SEEDS outperform other polygonal resolution-independent superpixels (Voronoi and CCM) on BR, CUE, ASA, dRE.

When a resolution of the cameraman image is decreased as in the last figure of supplementary materials, pixel-based superpixels include more and more visible zigzags, while corresponding RIMe superpixels keep nice straight boundaries.

The key advantage of polygonal resolution-independent superpixels is the possibility to render a polygonal mesh at any higher resolution. This up-scaling can convert low resolution photos into high-resolution images. In Computer Graphics, polygonal meshes with few edges can be easily manipulated to improve animations converted from traditional videos by cheap cameras.

The next step is to optimize positions of branched vertices, for example by minimizing an energy containing the exact reconstruction error and compactness.

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. T-PAMI **34** (2012)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Transactions PAMI **33**, 898–916 (2011)
3. Van de Bergh, M., Boix, X., Roig, G., Van Gool, L.: Seeds: superpixels extracted via energy-driven sampling. Int J Computer Vision **111**, 298–314 (2015)
4. Duan, L., Lafarge, F.: Image partitioning into convex polygons. In: Proceedings of CVPR (Computer Vision and Pattern Recognition). pp. 3119–3127 (2015)
5. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int J Computer Vision **59**, 167–181 (2004)
6. Forsythe, J., Kurlin, V.: Convex constrained meshes for superpixel segmentations of images. J Electronic Imaging **26(6)**(061609) (2017)
7. Forsythe, J., Kurlin, V., Fitzgibbon, A.: Resolution-independent superpixels based on convex constrained meshes. In: Proceedings of ISVC (2016)
8. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: a line segment detector. Image Processing On Line **2**, 35–55 (2012)
9. Kurlin, V., Harvey, D.: Superpixels optimized by color and shape. In: Proc. EMM-CVPR 2017, Lecture Notes in Computer Science. vol. 10746, pp. 297–311 (2018)
10. Kurlin, V., Muszynski, G.: A persistence-based approach to automatic detection of line segments in images. In: Proceedings of CTIC. pp. 137–150 (2019)
11. Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Siddiqi, K.: Turbopixels: fast superpixels using geometric flows. Transactions PAMI **31**, 2290–2297 (2009)
12. Li, Z., Chen, J.: Superpixel segmentation using linear spectral clustering. In: Proceedings of CVPR. pp. 1356–1363 (2015)
13. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: Proceedings of CVPR. pp. 2097 – 2104 (2011)
14. Luengo, I., Basham, M., French, A.: Smurfs: Superpixels from multi-scale refinement of super-regions. In: Proceedings of BMVC (2016)
15. Moore, A., Prince, S., Warrell, J.: Lattice cut – constructing superpixels using layer constraints. In: Proceedings of CVPR. pp. 2117–2124 (2010)
16. Shi, J., Malik, J.: Normalized cuts and image segmentation. Transactions PAMI **22**, 888–905 (2000)
17. Veksler, O., Boykov, Y., Mehrani, P.: Superpixels and supervoxels in an energy optimization framework. In: Proceedings of ECCV. pp. 211–224 (2010)
18. Viola, F., Fitzgibbon, A., Cipolla, R.: A unifying resolution-independent formulation for early vision. In: Proceedings of CVPR. pp. 494–501 (2012)
19. Yao, J., Boben, M., Fidler, S., Urtasun, R.: Real-time coarse-to-fine topologically preserving segmentation. In: Proceedings of CVPR. pp. 216–225 (2015)
20. Zhang, Y., Hartley, R., Mashford, J., Burn, S.: Superpixels via pseudo-boolean optimization. In: Proceedings of ICCV. pp. 211–224 (2011)

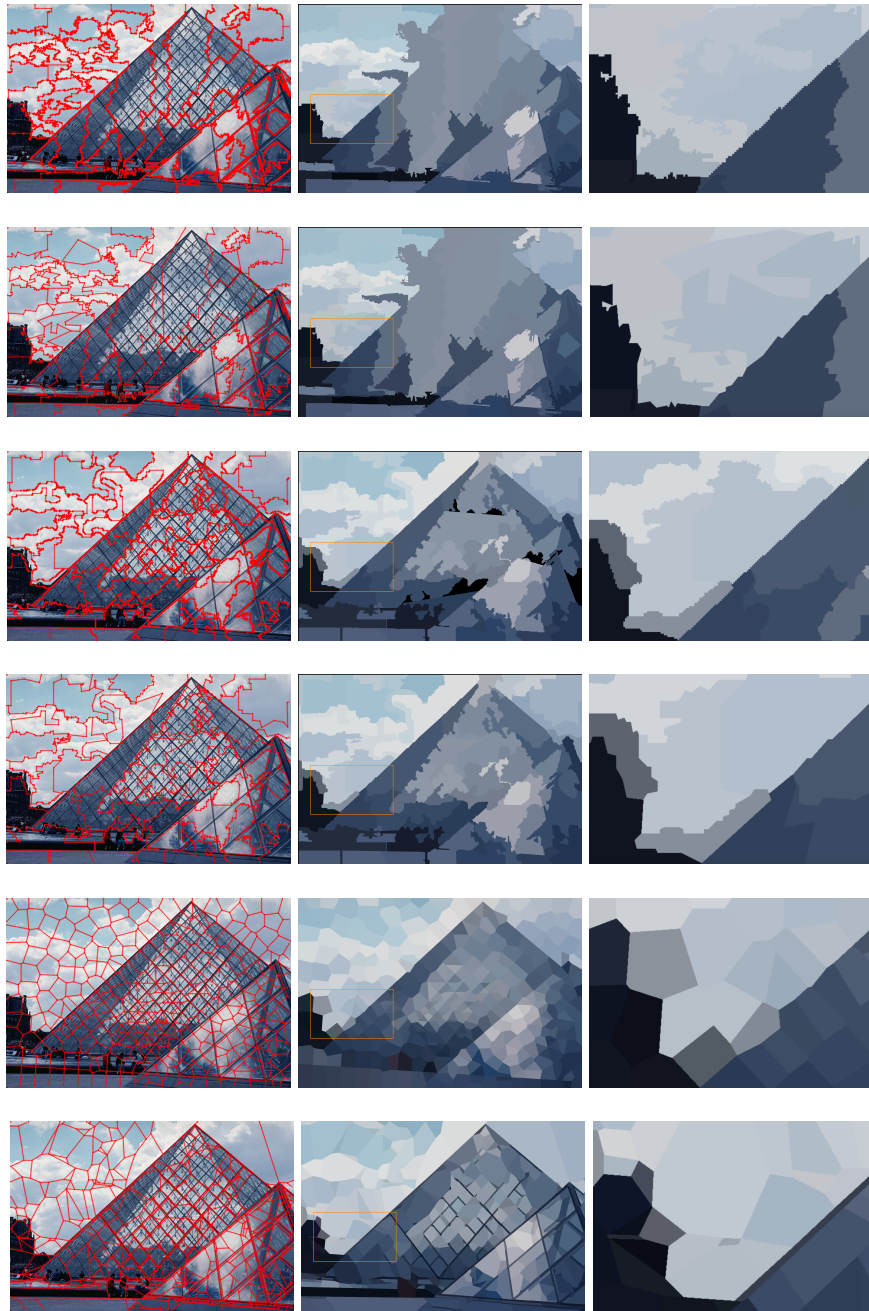**Fig. 9. Row 1:** pixel-based SEEDS. **Row 2:** better SEEDS with RIMe. **Row 3:** SLIC. **Row 4:** better SLIC with RIMe. **Row 5:** Voronoi superpixels. **Row 6:** CCM superpixels. **Column 1:** superpixel meshes. **Column 2:** reconstructions with average colors. **Column 3:** zoomed-in parts of reconstructions by small orange rectangles.