Resolution-independent superpixels based on convex constrained meshes without small angles

Jeremy Forsythe[1,2], Vitaliy Kurlin[3], Andrew Fitzgibbon[4]

[1]Vienna University of Technology, Favoritenstr. 9-11 / E186, A-1040 Vienna, Austria
[2]Department of Mathematical Sciences, Durham University, Durham DH1 3LE, UK
[3]Computer Science department, University of Liverpool, Liverpool L69 3BX, UK
[4]Microsoft Research, 21 Station Road, Cambridge CB1 2FB, UK

Abstract. The over-segmentation problem for images is studied in the new resolution-independent formulation when a large image is approximated by a small number of convex polygons with straight edges at subpixel precision. These polygonal superpixels are obtained by refining and extending subpixel edge segments to a full mesh of convex polygons without small angles and with approximation guarantees. Another novelty is the objective error difference between an original pixel-based image and the reconstructed image with a best constant color over each superpixel, which does not need human segmentations. The experiments on images from the Berkeley Segmentation Database show that new meshes are smaller and provide better approximations than the state-of-the-art.

1 Introduction: motivations, problem and contributions

1.1 Spatially Continuous Model for Over-segmentation of Images

Digital images are given by pixel values at discrete positions. Since images represent a spatially continuous world, the reconstruction problem should be solved in terms of functions defined over a *continuous image domain*, not over a discretization such as a regular grid. For example, grayscale values across a real image edge rarely drop from 255 (white) to 0 (black), but change gradually over 2-3 pixels, see details in [1, Fig. 1]. Hence a real edge between objects is often not along pixel boundaries and should be considered in the infinite family of line segments with any slope and endpoints having real coordinates. The first algorithm to output subpixel edges with theoretical guarantees is LSD [2].

The *over-segmentation problem* is to split an image into *superpixels* (larger than pixels and usually smaller than real objects) that have a nice shape and low variation of color. *Traditional superpixels* are formed by merging square-based pixels, e.g. by clustering. These superpixels often have irregular shapes with zigzag boundaries and holes inside. The *resolution-independent* approach [1] models a superpixel as a convex polygon with straight edges and vertices at subpixel resolution. Such a polygonal mesh can be rendered at any higher resolution by choosing a best color for each polygon in the *reconstructed image*.

A resulting mesh with constant colors over all polygons can be used to substantially speed-up any higher level processing such as object detection or recognition. Fig. 1 shows that only 231 convex polygons are enough to approximate the original 512×512 image with a small reconstruction error from Definition 1.

1.2 Energy Minimization for Resolution-Independent Superpixels

A real image is modeled as a function I that is defined at any point of a continuous image domain $\Omega \subset \mathbb{R}^2$ and takes values in $\mathbb{R}$ (grayscale) or $\mathbb{R}^3$ (color images). We consider the function $I(\mathbf{x})$ taking the same color value at any point $\mathbf{x} \in \Omega$ within every square pixel B_p considered as a continuous subset of Ω. This function $I(\mathbf{x})$ defines a piecewise constant surface over the image domain Ω.

Fig. 1. Left: 512×512 input. **Middle**: 275 Voronoi superpixels have nRMS≈10.2%. **Right**: 246 superpixels based on a Convex Constrained Mesh have nRMS≈4.48%.

The reconstruction problem is to find a latent image represented by a function $u(\mathbf{x})$ that minimizes the energy $E = \iint\limits_{\Omega} ||I(\mathbf{x}) - u(\mathbf{x})|| d\mathbf{x} + R$, where R is a regularizer that penalizes degenerate solutions or reflects an image prior.

The energy E will be the reconstruction error from Definition 1. Usually $u(\mathbf{x})$ is simpler than $I(\mathbf{x})$ in a certain sense. In our case $u(\mathbf{x})$ will have constant values over geometric polygons (superpixels) that are much larger than original pixels. The regularizer will forbid small angles, because narrow triangles may not cover even one pixel, while large angles (even equal to 180°) cause no difficulties.

So the reconstruction problem is to split a large image into a fixed number of polygons minimizing a difference between the original image function $I(\mathbf{x})$ over many pixels and the reconstructed image $u(\mathbf{x})$ over fewer convex polygons.

1.3 Contribution: Convex Constrained Mesh of Superpixels (CCM)

Here are the stages of the algorithm for resolution-independent superpixels.

1. The Line Segment Detector [2] finds line segments at subpixel resolution.

2. The LSD output is refined to resolve line intersections and small angles.

3. The resulting graph is extended to a triangulation without small angles.

4. Triangles are merged in convex polygons that also have no small angles.

5. The reconstructed image is obtained by finding the best constant color of any convex superpixel after minimizing the approximation error in Definition 1.

The input of the LSD and CCM algorithms above is a grayscale image. The Convex Constrained Mesh (CCM) built at Stage 4 is introduced in Definition 2 and has guarantees in Theorem 5 in terms of the following parameters.

• Min_Angle is the minimum angle between adjacent edges in a final mesh.

• Min_Distance is an approximation tolerance of LSD segments by CCM edges.

The default values are 3 pixels and 30° motivated by a similar angle bound in Shewchuk triangulations used at Stage 3. Here are the main contributions.

• The new concepts of the *reconstruction error* (a new quality measure for resolution-independent superpixels not relying on ground truth segmentations) and a *Convex Constrained Mesh* (CCM) are introduced in Definitions 1–2.

• The *LSD refinement* (Algorithm 3): disorganized line segments are converted into a planar graph well approximating the original LSD with guarantees.

• *Shewchuk's Triangle extension* (Algorithm 4): a triangulation is upgraded to a Convex Constrained Mesh without small angles as guaranteed by Theorem 5.

• The *experiments on BSD* [3] in section 4 show that CCM have smaller sizes and reconstruction errors than other resolution-independent superpixels, also achieving similar benchmark results in comparison with traditional superpixels.

2 Pixel-based and Resolution-Independent Superpixels

A pixel-based image is represented by a lattice L whose nodes are in a 1–1 correspondence with all pixels, while all edges of L represent adjacency relations between pixels. Usually each pixel is connected to its closest 4 or 8 neighbors.

The seminal *Normalized Cuts* algorithm by Shi and Malik [4] finds an optimal partition of L into connected components, which minimizes an energy taking into account all nodes of L. The algorithm by Felzenszwalb and Huttenlocher [5] was faster, but sometimes produced superpixels of irregular sizes and shapes as found by Levinstein at el. [6]. The *Lattice Cut* algorithm by Moore et al. [7] guarantees that the final mesh of superpixels is regular like the original grid of pixels. The best quality in this category is achieved by the Entropy Rate Superpixels (ERS) of Lie et al. [8] minimizing the entropy rate of a random walk on a graph.

The *Simple Linear Iterative Clustering* (SLIC) algorithm by Achanta et al. [9] 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. This gives an average time of 0.2s per BSD500 image.

SEEDS (Superpixels Extracted via Energy-Driven Sampling) by Van den Bergh et al. [10] seems the first superpixel algorithm to use a *coarse-to-fine optimization*. The colors of all pixels within each fixed superpixel are put in bins, usually 5 bins for each color channel. Each superpixel has the associated sum of deviations of all bins from an average bin within the superpixel. This sum is maximal for a superpixel whose pixels have colors in one bin. SEEDS iteratively maximizes the sum of deviations by shrinking or expanding superpixels.

Almost all past superpixels have *no geometric or topological constraints*, only in a soft form of a regularizer [11]. If a final cluster of pixels in SLIC is disconnected or contains holes, post-processing is needed. TopoCut [12] by Chen et al. has a hard topological constraint in a related problem of image segmentation.

The *key limitation of pixel-based superpixels* is the fixed resolution of an original pixel grid. Resolution-independent superpixels are the next step in approximating images by polygons whose vertices have any subpixel precision.

The *only past resolution-independent superpixels* by Duan and Lafarge [13] and new CCM superpixels use constrained edges from the LSD algorithm of Grompone von Gioi et al. [2], which outputs thin rectangles such that the color substantially changes at their long middle lines, see Fig. 3. The parameters are a tolerance τ for angles between gradients and a threshold ε for false alarms.

Voronoi superpixels [13] are obtained by splitting an image into Voronoi faces whose centers are chosen along LSD edges. The natural input would be a set of centers, however the algorithm first runs LSD [2] and then chooses centers on both sides of LSD edges. So the edges were soft constraints without proved guarantees yet. By Theorem 5 all given edges are a hard constraint for CCMs.

A *Shewchuk triangulation* is produced by the state-of-the-art Triangle software [14] that guarantees a lower bound (as large as 28°) for all angles. A Convex Constrained Mesh introduced in Definition 2 extends a Shewchuk triangulation to a mesh of convex polygons that also have no small angles by construction.

3 A Convex Constrained Mesh (CCM) with Guarantees

A superpixel in Definition 1 can be a union of square pixels or any polygon.

Definition 1 *Let an image I have n pixels, each pixel be the 1×1 square B_p and have $Intensity(p)\in[0,255]$. Let I be split in superpixels F_j (polygons or unions of pixels) with $Color(F_j)\in[0,255]$, $j=1,\dots,s$. The* Reconstruction Error *is*

$$RE = \min \sum_{pixels\ p} \Big(Intensity(p) - \sum_{j=1}^{s} Area(B_p\cap F_j)\, Color(F_j) \Big)^2, \qquad (1a)$$

where the minimum is over all $Color(F_j)$, $j=1,\dots,s$. The internal sum in RE is small, because each square B_p non-trivially intersects only few superpixels F_j, so the intersection $Area(B_p\cap F_j)$ is almost always 0 (when B_p is outside F_j)

or 1 (when F_j covers B_p). For a fixed splitting $I = \cup_{j=1}^{s} F_j$, the function RE quadratically depends on $Color(F_j)$, which are found from a linear system.

$$\textit{The normalized Root Mean Square is } nRMS = \sqrt{\frac{RE}{n}} \cdot \frac{100\%}{255}. \tag{1b}$$

The reconstructed image *is the superpixel mesh with all optimal $Color(F_j)$ minimizing $nRMS$. This colored mesh can be rendered at any resolution, see Fig. 2.*

In Definition 1 if a superpixel F_j is a union of square pixels, then $\text{Area}(B_p \cap F_j)$ is always 0 or 1, so the optimal $\text{Color}(F_j)$ is the mean color of all pixels in F_j.

Fig. 2. Left: 589 Voronoi superpixels (mesh and reconstruction) have $nRMS \approx 9.22\%$. **Right**: 416 CCM superpixels (red mesh and reconstruction) have $nRMS \approx 6.32\%$

Another important motivation for the new CCM superpixels is in Fig. 2, where the reconstructed image from Definition 1 in the second picture is considered as the input for any higher level processing. Since boundaries of a Voronoi mesh may not well approximate constrained edges, the reconstructed image may miss long thin structures, such as legs of a camera tripod in Fig. 2.

Definition 2 *Let G be a planar straight line graph with angles at least $\varphi \leq 60°$. A* Convex Constrained Mesh $\text{CCM}(G)$ *is a piecewise linear complex such that*

(2a) $\text{CCM}(G)$ *has convex polygons with angles* $\geq \text{Min\_Angle} = \arcsin\left(\frac{1}{\sqrt{2}} \sin\frac{\varphi}{2}\right)$*;*

(2b) the graph G is covered by the edges of the Convex Constrained Mesh $\text{CCM}(G)$*.*

Any Shewchuk triangulation is an example of a Convex Constrained Mesh. However, Definition 2 allows general meshes of any convex polygons without small angles. We build CCM by converting the LSD output in Algorithm 3 into a planar graph G without self-intersections and then by extending G into a polygonal mesh without small angles. All steps below are needed to satisfy main Theorem 5. Subsection 4.1 confirms that CCMs are smaller than past meshes.

Algorithm 3 *We convert disorganised line segments with self-intersections from the LSD output into a straight line graph as follows, see details in [15].*

(3.1) When a segment almost meets *another segment (within the offset parameter* Min_Distance = 3 *pixels), we extend the first one to a proper intersection .*

(3.2) When two segments almost meet (endpoints within Min_Distance*), we extend both to the intersection to avoid small angles/triangles in Algorithm 4.*

(3.3) When segments meet, we insert their intersection as a vertex in the graph.

Algorithm 4 *We extend a graph G from Algorithm 3, see details in [15].*

(4.1) The Triangle [14] extends the constrained edges of the graph G to a triangulation that has more edges, no angles smaller than Min_Angle $= 30°$.

(4.2) We merge adjacent faces along their common edge e if the resulting face is still convex. If two new angles at the endpoints of e are almost convex, we try to perturb them within Min_Distance *to guarantee convexity and no small angles.*

(4.3) We collapse unconstrained edges if all constrained edges remain fixed.

The steps above guarantee no small angles in CCM. Theorem 5 is proved in [15].

Theorem 5 *Let line segments $S_1, \dots, S_k$ have m intersections. Algorithm 3 builds a* CCM *in time $O((k+m)\log(k+m))$ so that*

(5a) any internal angle in a CCM *face is not smaller than* Min_Angle*;*

(5b) the union $\cup_i S_i$ is covered by the Min_Distance-*offset of the* CCM*'s edges.*

4 Experimental Comparisons and Conclusions

The sizes and reconstruction errors of the CCM and Voronoi superpixels are compared in subsections 4.1 and 4.2. Then two more superpixel algorithms SLIC [9] and SEEDS [10] are also included into BSD benchmarks in subsection 4.3.

4.1 Sizes of CCMs, Shewchuk's Triangulations and Voronoi meshes

The first picture in Fig. 3 is the original LSD output. The second picture shows the graph G obtained by the LSD refinement in Algorithm 3. The refined LSD output has more edges than the original LSD, because we include boundary edges of images and also intersection points, which become vertices of graphs.

We use $\phi = 30°$ for the LSD refinement, which leads to Min_Angle $\approx 10.5°$ in Shewchuk's Triangle [14]. We compare Shewchuk triangulations on the original LSD output and CCM on the refined LSD output in Fig. 3, where the 3rd picture shows a zoomed-in green box with many tiny triangles. The final picture in Fig. 3 contains only few faces after merge operations in Algorithm 4. The ratio of Shewchuk triangles to the number of faces in CCMs across BSD is 7.6.

The first step for Voronoi superpixels [13] is to post-process the LSD output when close and near parallel lines are removed, because the target application was satellite images of urban scenes with many straight edges of buildings. Then long thin structures such as legs of a camera tripod in Fig. 3 are represented only by one edge and may not be recognized in any further processing.

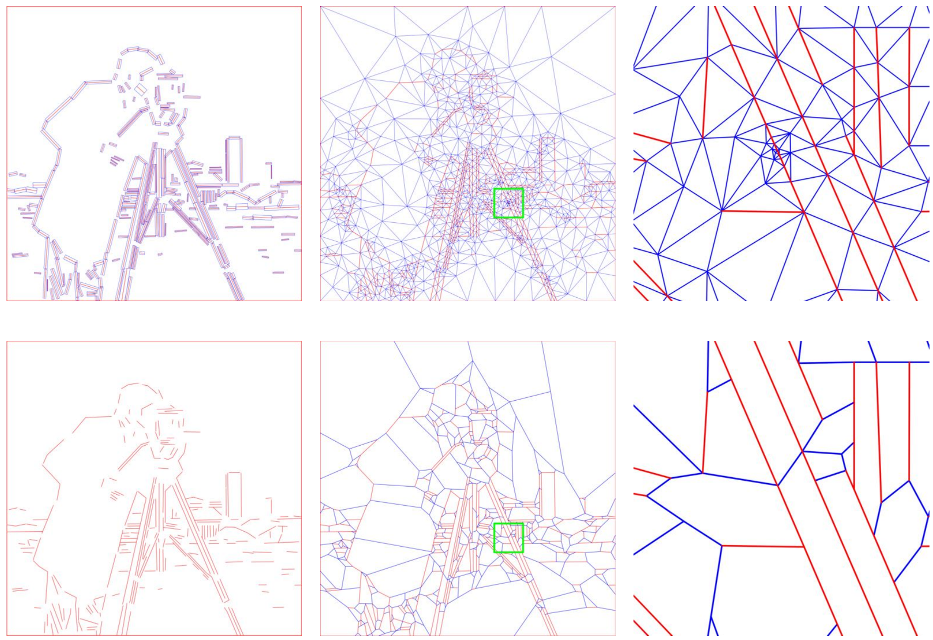

Fig. 3. Top left: 259 LSD red middle segments in blue rectangles before the refinement in Algorithm 3. **Bottom left**: the refined LSD output (a graph G) with 294 edges. **Top middle**: Shewchuk triangulation $T(G)$ with 2260 triangles. **Bottom middle**: the Convex Constrained Mesh CCM(G) with 416 faces. **Top right**: zoomed in green box with tiny triangles. **Bottom right**: zoomed in green box, all tiny triangles are merged.

That is why the LSD refinement in section 3 follows another approach and offers guarantees leading to Theorem 5. Table 1 displays the average ratios of face numbers over BSD images. Even when the parameter Eps_Radius of Voronoi superpixels is increased to 12, these ratios converge to a factor of about 3.25.

4.2 Approximation Quality of the CCM and Past Superpixels

Since the aim of superpixels is to approximate a large image by a reconstructed image based on a smaller superpixel mesh, the important quality is the standard statistical error $nRMS$ over all pixels, which is introduced in Definition 1.

Table 1. Ratios of the face numbers for CCM and Voronoi meshes on the same LSD edges, averaged across BSD images [3]. The parameter Eps_Radius is in pixels.

Eps_Radius of a superpixel	4	5	6	7	8	9	10	11	12
Mean $\frac{\text{Voronoi superpixels [13]}}{\text{number of faces in CCM}}$	8.91	6.21	4.86	4.03	3.96	3.43	3.27	3.27	3.26

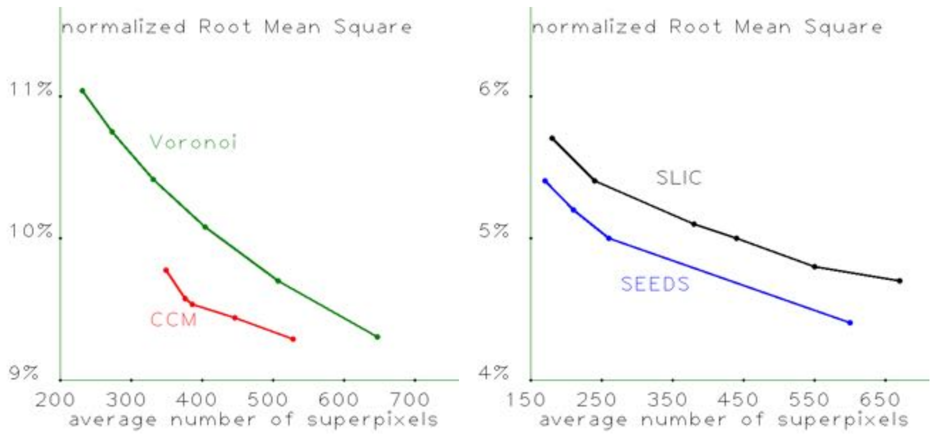

Fig. 4. The normalized Root Mean Squares in percents for Voronoi and CCM superpixels (on the left), SLIC and SEEDS (on the right) averaged over BSD500 images.

Fig. 5. Left: 791 Voronoi superpixels (mesh and reconstruction) with $nRMS$ ≈8.45%.
Right: 791 CCM superpixels (red mesh and reconstruction) with $nRMS$ ≈7.22%.

Fig. 4 shows that the reconstructed images of CCM superpixels better approximate original images than Voronoi superpixels. Some convex polygons of CCMs are much larger than Voronoi superpixels, simply because the corresponding regions in images indeed have almost the same intensity, e.g. the sky. Hence taking the best constant color over each superpixel is reasonable.

Voronoi superpixels have similar sizes, because extra centers are added to empty regions using other non-LSD edges. Despite CCMs being obtained from only LSD edges without using colors, the reconstructions have smaller errors in comparison with Voronoi meshes containing more superpixels in Fig. 5.

Fig. 4 confirms smaller approximation errors of CCM superpixels across all BSD500 images, where we used the same LSD parameters for CCM and Voronoi superpixels. For all superpixels, we computed optimal colors minimizing the reconstruction error and measured $nRMS$ in percents, see Definition 1.

Each BSD experiment outputs 500 pairs (number of faces, nRMS). We average each coordinate of these pairs and output a single dot per experiment. The first red dot at $(377.1, 9.626\%)$ in Fig. 4 means that CCMs have 377 faces and an approximation error of 9.6% on average. For a fixed image, the LSD algorithm outputs roughly the same number of edges for all reasonable parameters τ, ε.

So smaller CCMs seem impossible, because all LSD edges are hard constraints, while all faces should be convex. To get larger CCMs, we stop merging faces in Algorithm 4 after getting a certain number of convex faces. The five experiments on Voronoi superpixels with Eps_Radius $= 7, 8, 9, 10, 11$ produced 5 dots along a decreasing curve. Fig. 4 implies that Voronoi meshes require more superpixels (507.3 on average) to achieve the similar $nRMS = 9.696\%$.

4.3 Standard Benchmarks for CCM and Past Superpixels

The benchmarks BR and CUE are designed for pixel-based superpixels and use human segmentations from BSD [3], see details in [15]. We discretize CCM and Voronoi superpixels by drawing lines in OpenCV to detect boundary pixels. We put all pixels into one superpixel if their centers are in the same polygon.

It is unfair to compare discretized resolution-independent superpixels and pixel-based superpixels on benchmarks designed for the latter superpixels. CCM achieves smaller undersegmentation errors than SEEDS/SLIC and most importantly beats Voronoi superpixels on the objective $nRMS$ as well as on BR.

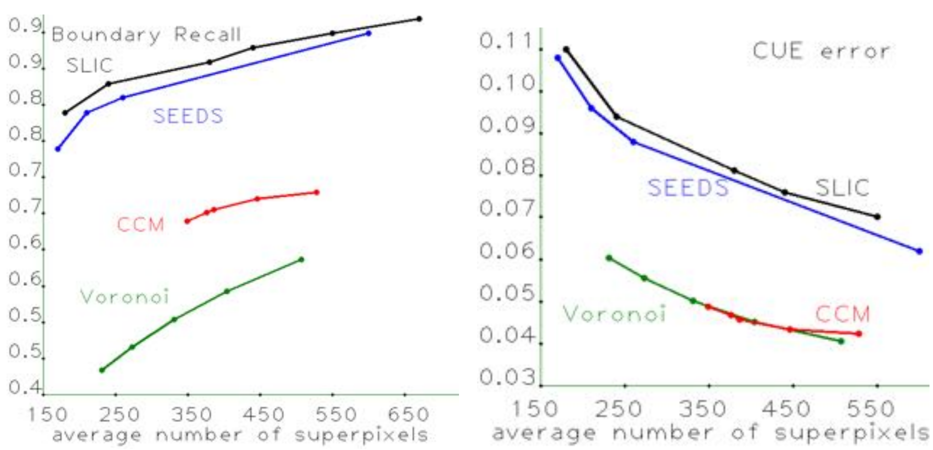

Fig. 6. Left: Boundary Recall (BR). **Right**: Corrected Undersegmentation Error.

Pixel-based superpixels SLIC and SEEDS achieve better results on $nRMS$ and Boundary Recall (BR) in Fig. 6, because their superpixels can have irregular boundaries (of only horizontal and vertical edges). However, humans are more likely to sketch straight edges than boundaries consisting of short zigzags.

So irregular pixel-based superpixels are often split by straight ground truth boundaries. Resolution-independent superpixels are convex polygons with straight edges and are expected to have smaller undersegmentation errors in Fig. 6.

Since only a Windows demo is available for Voronoi superpixels [13], we couldn't directly compare the running times of resolution-independent superpixels. We worked on a different platform and confirm that the running time for the CCM on a laptop with 8G RAM is about 0.15s across BSD500 images.

The key contribution is the new concept of a Convex Constrained Mesh (CCM), which extends any constrained line segments to a mesh of convex polygons without small angles. The paper focused on the quality of CCM superpixels, which seem ideal for detecting long thin structures in urban scenes, see Fig. 2.

• Theorem 5 guarantees the approximation quality and no small angles in CCMs, which also have smaller sizes on the same input in comparison with [14], [13].

• The CCM outperforms the only past algorithm [13] for resolution-independent superpixels on BR (Boundary Recall) and the new error $nRMS$ in Fig. 4, and even outperforms pixel-based superpixels on the CUE benchmark in Fig. 6.

The first author was supported by the project FWF P24600-N23 at TU Wien.

References

1. Viola, F., Fitzgibbon, A., Cipolla, R.: A unifying resolution-independent formulation for early vision. In: Proceedings of CVPR. (2012) 494–501
2. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: a line segment detector. Image Processing On Line **2** (2012) 35–55
3. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Transactions PAMI **33** (2011) 898–916
4. Shi, J., Malik, J.: Normalized cuts and image segmentation. Transactions PAMI **22** (2000) 888–905
5. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int J Computer Vision **59** (2004) 167–181
6. Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Siddiqi, K.: Turbopixels: fast superpixels using geometric flows. Transactions PAMI **31** (2009) 2290–2297
7. Moore, A., Prince, S., Warrell, J.: Lattice cut – constructing superpixels using layer constraints. In: Proceedings of CVPR. (2010) 2117–2124
8. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: Proceedings of CVPR. (2011) 2097 – 2104
9. 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)
10. Van de Bergh, M., Boix, X., Roig, G., Van Gool, L.: Seeds: superpixels extracted via energy-driven sampling. Int J Computer Vision **111** (2015) 298–314
11. Veksler, O., Boykov, Y., Mehrani, P.: Superpixels and supervoxels in an energy optimization framework. In: Proceedings of ECCV. (2010) 211–224
12. Chen, C., Freedman, D., Lampert, C.: Enforcing topological constraints in random field image segmentation. In: Proceedings of CVPR. (2011) 2089–2096
13. Duan, L., Lafarge, F.: Image partitioning into convex polygons. In: Proceedings of CVPR (Computer Vision and Pattern Recognition). (2015) 3119–3127
14. Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Computational Geometry: Theory and Applications **22** (2002) 21–74
15. Forsythe, J., Kurlin, V., Fitzgibbon, A.: Resolution-independent superpixels based on convex constrained meshes (full version) (2016) http://kurlin.org.

A Planar Graphs and Voronoi Meshes

To avoid any confusion, we continue numbering all definitions, figures and pages as in the paper. Definition 6 introduces the convenient concept of a *PL complex* that formally covers all essential cases including

- our input (a set of points and constrained line segments);
- any PSLG (a planar straight line graph) as defined in [1];
- Steiner Delaunay [2] and Shewchuk triangulations [1];
- our final output (a Convex Constrained Mesh CCM).

Definition 6 *[2, Def 2.8] A* piecewise linear (PL) complex C *is a finite set of vertices, edges and polygons (faces) such that*

- *if C contains an edge e, then C contains both endpoints of e;*
- *the boundary of any face is a union of edges from C;*
- *edges from C can meet only at their common endpoint;*
- *faces from C can share only common edges and vertices.*

The domain $|C| \subset \mathbb{R}^2$ *of the PL complex C is the area covered by all vertices, edges and faces of C. If C has no faces, we call C a* graph *or a* Planar Straight Line Graph (PSLG) *[1].*

A PL complex C can consist of disconnected line segments. Definition 7 extends C using extra vertices (called *Steiner* points) to a full triangulation T. All edges of C (possibly subdivided in T) will be called *constrained*, all other edges of T are *unconstrained*.

Definition 7 *[2, Def 2.15] A* Steiner constrained Delaunay triangulation *of a complex C is a PL complex* $\mathrm{SDT}(C)$ *such that*

(7a) $\mathrm{SDT}(C)$ *has only triangular faces and covers $|C| \subset \mathbb{R}^2$,*

(7b) if all edges of C are opaque, for any triangle T, the open circumdisk of T has no vertices of C visible from the interior of T.

Condition (7a) means that all faces of C are subdivided into triangles. An edge of C can be subdivided into shorter edges and the domain of $\mathrm{SDT}(C)$ may not be convex. Condition (7b) guarantees that $\mathrm{SDT}(C)$ contains only nice triangles, e.g. a quadrilateral in Fig. 7 should be split by a shorter diagonal rather than a longer one.

If a PL complex C is a finite set of points, then $\mathrm{SDT}(C)$ is a classical Delaunay triangulation, which is dual to the Voronoi mesh below. Let d be the Euclidean distance. For a set $S \subset \mathbb{R}^2$, let $d(p, S) = \inf\{\, d(p, q) \,:\, q \in C\,\}$ be the distance from a point p to S.

Definition 8 *For a set of points $C = \{p_1, \ldots, p_n\}$, the* Voronoi face $V(p_i) = \{q \in \mathbb{R}^2 \,:\, d(q, p_i) \le d(q, C - p_i)\}$ *is the closed neighborhood of p_i consisting of*

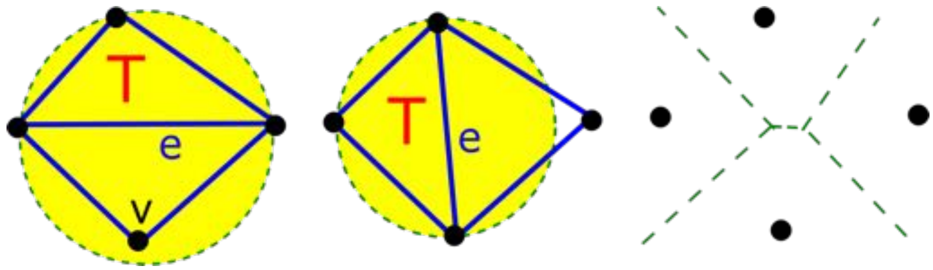

Fig. 7. Left: the yellow open circumdisk of the triangle T contains a vertex v. **Middle**: the circumdisk of T contains no vertices, both triangles belong to $\mathrm{SDT}(C)$. **Right**: the top and bottom Voronoi faces are adjacent, so their centers are connected by e.

points $q \in \mathbb{R}^2$ that are non-strictly closer to the site p_i *than to other points of $C - p_i$. The splitting $V(p_1) \cup \cdots \cup V(p_n)$ is the* Voronoi mesh, *see Fig. 7.*

Then a Delaunay triangulation $\mathrm{DT}(C)$ *on the vertex set C has*

- *an edge between p_i, p_j if the faces $V(p_i) \cap V(p_j) \neq \emptyset$,*
- *a triangle on p_i, p_j, p_k if $V(p_i), V(p_j), V(p_k)$ share a point.*

B Refinement of the Line Segment Detector (LSD)

The input for the LSD algorithm [3] is a grayscale image. The output is an unordered set of thin rectangles in the plane. We consider only the long middle lines of these rectangles as the red constrained edges and also add the boundary edges of the whole image, for example, see the top left picture in Fig. 3.

The resulting red segments may intersect each other, go outside the boundary of the image or form small angles. Sections B.1–sub:split-segments describe how to refine the LSD output and get a graph G without angles smaller than a given bound φ.

We define the *strength* of a line segment S as $+\infty$ if S is on the boundary of the image, else the *strength* is the length of S. We apply each of the refinements to line segments ordered by their strength. So the order of refinements is not random and depends only on line segments that were detected by LSD.

In each of subsections B.1–B.3 all the listed steps below are performed in one go for every pair of line segments from the original LSD output.

B.1 Extending Segments to Line-Segment Intersections

If an endpoint v of one segment is very close to another segment, then a Shewchuk triangulation will have many tiny triangles at the vertex v to avoid small angles. The steps below resolve this singular case by inserting a proper intersection.

Step (B.1a) For any straight segment S_1, we take the infinite line $L(S_1)$ through S_1, see Fig. 8. We find all segments S_2 intersecting the two rays $L(S_1) - S_1$.

Step (B.1b) Among all intersection points of S_2 and $L(S_1) - S_1$, we choose a point p closest to an endpoint of S_1. This choice of p guarantees that if we extend S_1 to p, then no new intersections with other segments are created. The steps below work similarly for the intersection closest to another endpoint of S_1.

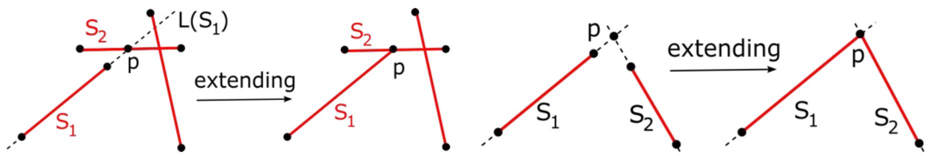

Fig. 8. Left: extend S_1 to $p = L(S_1) \cap S_2$. **Right**: extend S_1, S_2 to $p = L(S_1) \cap L(S_2)$.

Step (B.1c) If $d(p, S_1) >$ Min_Distance, we stop considering p. Otherwise we find the acute angle θ between $L(S_1)$ and S_2. If $\theta \geq \varphi$, we extend S_1 to the new vertex p, which splits S_2 into two new segments, see the left hand side picture of Fig. 8. If any of these new segments within S_2 is shorter than Min_Distance, we remove this segment together with its endpoint different from p.

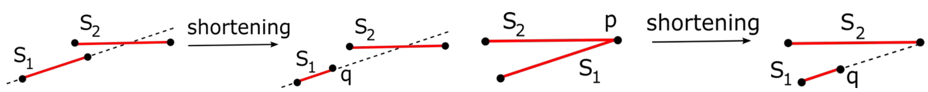

Fig. 9. Left: shorten a segment S_1 to the new endpoint q with $d(q, S_2) =$ Min_Distance. **Right**: shorten a segment S_1 away from S_2 to avoid a small angle $\theta < \varphi$ between S_1, S_2.

Step (B.1d) If $\theta < \varphi$, we find a point $q \in S_1$ with $d(q, S_2) =$ Min_Distance. If q exists, we shorten S_1 to the new endpoint q, see Fig. 9. Otherwise we remove the whole segment S_1, because S_1 is covered by the Min_Distance-offset of S_2.

B.2 Extending Segments to Line-Line Intersections

If two segments don't intersect as in subsection B.1, but have very close endpoints, we again avoid tiny triangles later inserting a proper intersection.

Similarly to Step (B.1), we find intersection points within Min_Distance. Now we take the intersection of the infinite lines $L(S_1)$ and $L(S_2)$ outside S_1, S_2.

Step (B.2a) For a segment S_1, we consider the infinite line $L(S_1)$ through S_1, and the lines $L(S_2)$ through all the other segments. Then we find all segments S_2 such that either of the rays $L(S_1) - S_1$ meets one of the rays $L(S_2) - S_2$.

Step (B.2b) Among all intersections of $L(S_1) - S_1$, $L(S_2) - S_2$, we choose a point p closest to an endpoint of S_1, do Step (B.2c) for both endpoints of S_1.

Step (B.2c) If $d(p, S_1) <$ Min_Distance and $d(p, S_2) <$ Min_Distance, we find the acute angle θ between $L(S_1)$ and $L(S_2)$. If the angle $\theta \geq \varphi$, we extend S_1

and S_2 to the new vertex p, see Fig. 8. If the angle $\theta < \varphi$, we move the endpoint of the weaker segment to a point q such that $d(q, S_2) = $ Min_Distance, see Fig. 9.

B.3 Splitting Line Segments at Intersection Points

Many segments in the LSD output may actually intersect, so the steps below insert this intersection point to get a planar graph without self-intersections.

Step (B.3a) For each pair of segments S_1, S_2, we check if S_1, S_2 intersect at a point p that is internal to both S_1, S_2. If a new segment is shorter than Min_Distance, we remove it together with its endpoint different from p, see Fig. 10.

Fig. 10. Removing new segments and collapsing segments shorter than Min_Distance.

Step (B.3b) Let θ be the smallest angle between the remaining segments (also denoted by S_1, S_2) with the common endpoint p. If $\theta \geq \varphi$, we stop considering the point p. The steps below similarly work for the symmetric angle θ at p.

Step (B.3c) If $\theta < \varphi$, we shorten the weaker segment S_1 to make the distance $d(S_1, S_2) = $ Min_Distance as in (B.1d), see the right hand side picture of Fig. 9.

Step (B.3d) We collapse any isolated edge shorter than Min_Distance to its mid-point and remove all non-isolated edges shorter than Min_Distance, see Fig. 10.

B.4 Approximation guarantees for the LSD refinement

We further justify all the steps in subsections B.1–B.3 by the following result.

Lemma 9 *Let line segments $S_1, \ldots, S_k$ have m intersections. The LSD refinement described in Appendix B outputs a planar straight line graph G with $O(k+m)$ edges in time $O((k+m)\log k)$ such that*

(9a) any angle in the graph G between adjacent edges is not smaller than φ;

(9b) the union $\cup_i S_i$ of segments is covered by the Min_Distance-*offset of G.*

Proof. Due to subsection B.3, all final segments meet only at their endpoints. A line segment may intersect any other segment only once. Any new intersection may generate 2 extra segments, so G has at most $O(k+m)$ edges.

There are $m = O(k^2)$ intersections of k segments in the worst case. In practice, any segment S detected by LSD meets at most two segments, only one near each endpoint of S, so $m = O(k)$. The output-sensitive swipe line algorithm [4, chapter 2] finds all intersections between segments in time $O((k+m)\log k)$ and can be extended to line-segment intersections in Step (B.1a).

Steps (B.1d), (B.3c) guarantee that all angles are not smaller than φ. A segment S_1 can become longer by at most Min_Distance in Step (B.1c) and shorter in Steps (B.1d), (B.2c), (B.3c), (B.3d). The removed part of S_1 is in the Min_Distance-offset of a stronger segment S_2, which can't be shortened later. □

C A Convex Constrained Mesh Without Small Angles

C.1 Fast Shewchuk Triangulations Without Small Angles

Any planar straight line graph $G \subset \mathbb{R}^2$ can be the input for Shewchuk's Triangle code [1]. The output is a Steiner constrained Delaunay Triangulation $T(G)$ without small angles.

Theorem 10 *[1, Theorem 12] For a planar straight line graph G having n vertices and no angles smaller than $\varphi \le 60°$, in time $O(n\log n)$ one can build a triangulation $T(G)$ without angles smaller than* $\text{Min\_Angle} = \arcsin\left(\frac{1}{\sqrt{2}}\sin\frac{\varphi}{2}\right)$.

If $\varphi = 60°$, then $\text{Min\_Angle} = \arcsin\left(\frac{1}{\sqrt{2}}\sin\frac{\varphi}{2}\right) \approx 20.7°$. If a graph G has angles smaller than Min_Angle, they are preserved in a Shewchuk triangulation. So the LSD refinement in section B is needed to prove main Theorem **??** later.

The existing edges of G are *constrained* and drawn in red. The newly added edges of $T(G)$ are *unconstrained* and drawn in blue. We use OpenMesh [5] to store $T(G)$ and then merge triangles into convex faces as described below.

C.2 Simple and Advanced Merge Operations to Get Convex Faces

Here we process unconstrained edges of the mesh in the decreasing order of length. The steps below are motivated by the aim to simplify the polygonal mesh and get a smaller number of superpixels keeping them convex.

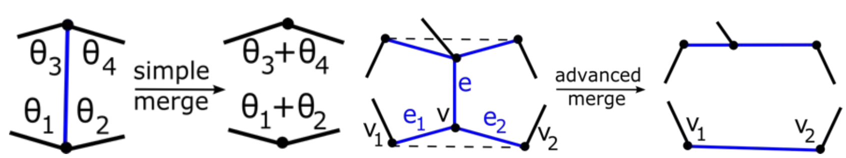

Fig. 11. Left: a simple merge by removing an unconstrained edge e between two faces if the new larger face is convex. **Right**: an advanced merge by removing an unconstrained edge e between two faces if each of its endpoints can be resolved by Step (C.2b) or (C.2c).

Step (C.2a) For each unconstrained edge e, we find 4 angles $\theta_1, \theta_2, \theta_3, \theta_4$ along the edge e at its endpoints, see Fig. 11. The four edges different from e in Fig. 11 are black, because they can have any type (constrained or unconstrained).

Step (C.2b) If $\theta_1 + \theta_2 \leq 180°$ and $\theta_3 + \theta_4 \leq 180°$, the convexity is preserved at both endpoints of e, so we remove e and the new larger face is convex.

If one of the angles $\theta_1 + \theta_2, \theta_3 + \theta_4$ in Step (C.2b) is greater than 180°, the simple merge operation along the edge e leads to a non-convex face. Then we try to make the *triangular cut* in Step (C.2c) without disturbing constrained edges.

Step (C.2c) Assume that $\theta_1 + \theta_2 > 180°$ at the vertex v in Fig. 11, and both edges e_1, e_2 are unconstrained. Then we try replacing $e_1 \cup e_2$ by the new unconstrained edge connecting v_1, v_2 in the last picture of Fig. 11. If no angle becomes smaller than Min_Angle or larger than 180°, then this *triangular cut* is successful.

Step (C.2d) If all angles are in [Min_Angle, 180°] after removing the edge e in Fig. 11, we finish Step (C.2c), else we reverse Step (C.2c) to avoid small angles.

After Step (C.2d) we check if any new edges can be removed by simple merge operations, which further simplifies the mesh. Finally, at any degree 2 vertex with the 180° angle, we replace two adjacent edges by one longer straight edge.

C.3 Collapsing unconstrained edges for a further simplification

We process unconstrained edges of the mesh in the increasing order of length. Both endpoints of any constrained edge are called *fixed* vertices. We will perturb only *non-fixed* vertices whose all incident edges are non-constrained.

Step (C.3a) If an endpoint v of an unconstrained edge e is not fixed, we try to collapse the edge e from the endpoint v towards the opposite endpoint w.

Step (C.3b) If any of the faces around the vertex w is non-convex or has an angle smaller than Min_Angle, we cancel this collapse and consider the opposite edge directed from w to v, or the next edge from the ordered list above.

If e belongs to a triangle, this triangle also collapses, which reduces the number of faces without affecting constrained edges. The average decrease of the number of faces due to collapses above is 10% across BSD500 images.

C.4 Guarantees for a Convex Constrained Mesh CCM

The following result again justifies all the steps in subsections C.2–C.3.

Lemma 11 *For any planar straight line graph G with n vertices and without angles smaller than $\varphi \leq 60°$, in time $O(n \log n)$ one builds $CCM(G)$ such that*

(11a) CCM(G) *has no angles* $\theta < \text{Min\_Angle} = \arcsin\left(\frac{1}{\sqrt{2}} \sin \frac{\varphi}{2}\right)$;

(11b) the graph G is fully covered by the edges of CCM(G).

Proof. Theorem 10 guarantees no small angles in $T(G)$ built in time $O(n \log n)$. All steps in section C check that CCM(G) has no angles $\theta <$ Min_Angle. All edges of G are kept by the merge operations, so the edges of CCM cover G. $\square$

Proof of Theorem 3. Lemma 9 in time $O((k+m)\log k)$ builds a planar straight line graph G with $O(m)$ vertices and angles $\geq \varphi = 2\arcsin(\sqrt{2}\sin \text{Min\_Angle})$. Lemma 11 in time $O(n\log n)$ for $n = O(k+m)$ builds CCM(G) without angles smaller than $\arcsin\left(\frac{1}{\sqrt{2}}\sin\frac{\varphi}{2}\right) = \text{Min\_Angle}$. Now conditions (9b) and (11b) imply (3b). $\square$

D Benchmarks based on Ground Truth Segmentations

The *Berkeley Segmentation Database* BSD500 [6] has 500 images widely used for evaluating segmentation algorithms due to 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 BSD500.

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 $p \in G(I)$ that are within 2 pixels from the superpixel boundary $B(I)$.

$$\text{The Undersegmentation Error } UE = \frac{1}{n}\sum_j \sum_{S_i \cap G_j \neq \emptyset} |S_i - G_j|$$

was often used in the past, where $|S_i - G_j|$ is the number of pixels that are in S_i, but not in G_j. However a superpixel is fully penalized when $S_i \cap G_j$ is 1 pixel, which required ad hoc thresholds, e.g. the 5% threshold $|S_i - G_j| \geq 0.05|S_i|$ by Achanta et al. [7], or ignoring boundary pixels of S_i by Liu et al. [8].

Van den Bergh et al. [9] suggested the more accurate measure, namely

$$\text{the Corrected Undersegmentation Error } CUE = \frac{1}{n}\sum_i |S_i - G_{max}(S_i)|,$$

where $G_{max}(S_i)$ is the ground truth region having the largest overlap with S_i. Neubert and Protzel [10] introduced the Undersegmentation Symmetric Error

$$USE = \frac{1}{n}\sum_j \sum_{S_i \cap G_j \neq \emptyset} \min\{in(S_i), out(S_i)\}, \text{ where}$$

$in(S_i)$ is the area of S_i inside G_j, $out(S_i)$ is the area of S_i outside G_j.

$$\text{The Achievable Segmentation Accuracy } ASA = \frac{1}{n}\sum_i \max_j |S_i \cap G_j|.$$

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 problem of a semantic segmentation, then ASA is the upper bound on the number of pixels that are wrongly assigned to final semantic regions.

Fig. 12, 13, 14 show more details of buulding CCMs.

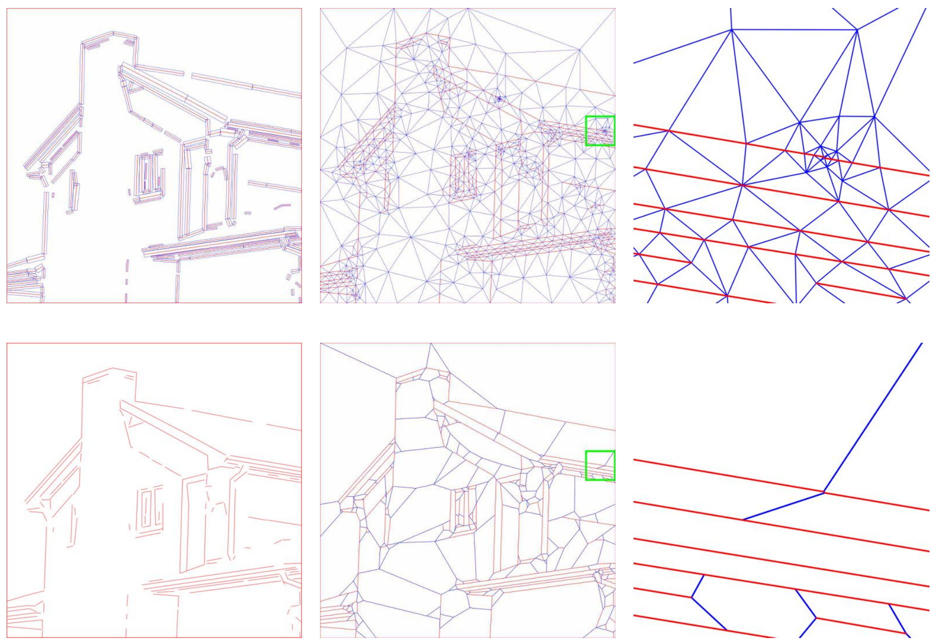

Fig. 12. Top left: the initial LSD output with 154 blue thin rectangles and red middle line segments before the refinement in section 3. **Bottom left**: the refined LSD output (a graph G) with 184 edges. **Top middle**: Shewchuk's triangulation SDT(G) with 1645 triangles on the red graph G in the bottom left picture. **Top right**: zoomed in green box with small triangles. **Bottom middle**: the Convex Constrained Mesh CCM(G) with 275 faces. **Bottom right**: zoomed in green box, all small triangles are merged.

E Augmentation Problems for Planar Graphs

Building a constrained mesh can be considered as a new augmentation problem for a planar graph [11]. Specifically, detected edges in an image are augmented to a polygonal mesh under the new constraint that small angles are forbidden.

A typical augmentation problem is to extend a set of disjoint straight segments in $\mathbb{R}^2$ to a planar graph satisfying some properties as in [12]. One approach

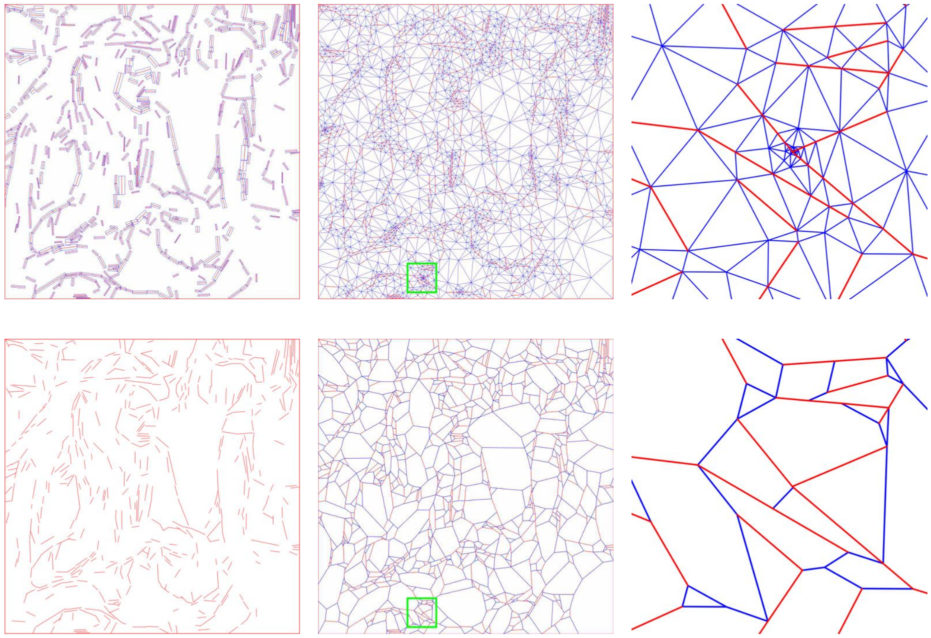

Fig. 13. Top left: the initial LSD output with 524 blue thin rectangles and red middle line segments before the refinement in section 3. **Top middle**: Shewchuk's triangulation SDT(G) with 3906 triangles on the red graph G in the bottom left picture. **Top right**: zoomed in green box with small triangles. **Bottom left**: the refined LSD output (a graph G) with 556 edges. **Bottom middle**: the Convex Constrained Mesh CCM(G) with 791 faces. **Bottom right**: zoomed in green box, all small triangles are merged.

is to extend n disjoint straight segments (in any order) to infinite lines until they first meet another segment or line, which produces at most $n+1$ convex polygons [11]. In practice, endpoints of segments are not in general position. Moreover, near parallel segments lead to small angles and very narrow faces. Hence our augmentation problem becomes harder than previously studied cases [11]. That is why we use Shewchuk triangulations and keep guarantees on small angles.

References

1. Shewchuk, J.R.: Delaunay refinement algorithms for triangular mesh generation. Computational Geometry: Theory and Applications **22** (2002) 21–74
2. Cheng, S.W., Dey, T.K., Shewchuk, J.R.: Delaunay Mesh Generation. Chapman and Hall/CRC Press (2012)
3. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: a line segment detector. Image Processing On Line **2** (2012) 35–55
4. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry : Algorithms and Applications. Springer (2010)

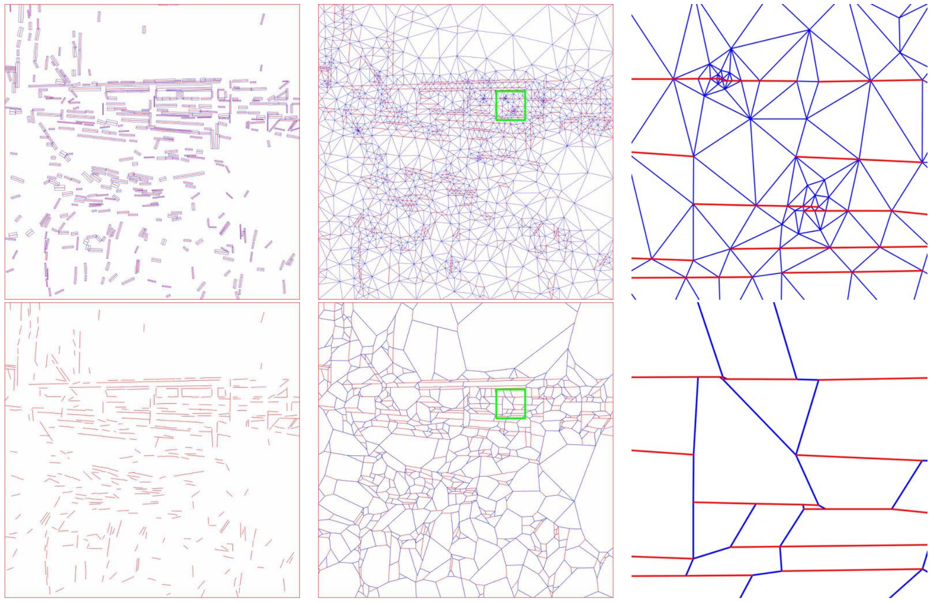

Fig. 14. Top left: the initial LSD output with 394 blue thin rectangles and red middle line segments before the refinement in section 3. **Top middle**: Shewchuk's triangulation SDT(G) with 3083 triangles on the red graph G in the bottom left picture. **Top right**: zoomed in green box with small triangles. **Bottom left**: the refined LSD output (a graph G) with 416 edges. **Bottom middle**: the Convex Constrained Mesh CCM(G) with 626 faces. **Bottom right**: zoomed in green box, all small triangles are merged.

5. Botsch, M., Steinberg, S., Bischoff, S., Kobbelt, L.: Openmesh - a generic and efficient polygon mesh data structure (2002)
6. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Transactions PAMI **33** (2011) 898–916
7. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. Transactions PAMI **34** (2012) 2274–2282
8. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: Proceedings of CVPR. (2011) 2097 – 2104
9. Van de Bergh, M., Boix, X., Roig, G., Van Gool, L.: Seeds: superpixels extracted via energy-driven sampling. Int J Computer Vision **111** (2015) 298–314
10. Neubert, P., Protzel, P.: Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms. In: Proceedings of ICPR. (2014) 996–1001
11. Hurtado, F., Tóth, C.D.: Plane geometric graph augmentation: a generic perspective. In: Thirty Essays on Geometric Graph Theory. (2012) 327–354
12. Al-Jubeh, M., Hoffmann, M., Ishaque, M., Souvaine, D.L., Tóth, C.: Convex partitions with 2-edge connected dual graphs. J. Combinatorial Optimization **22** (2010) 409–425