KIPPI: KInetic Polygonal Partitioning of Images

Jean-Philippe Bauchet, Florent Lafarge

KIPPI: KInetic Polygonal Partitioning of Images

Jean-Philippe Bauchet Florent Lafarge
Inria – Université Côte d'Azur
Firstname.Lastname@inria.fr

Abstract

Recent works showed that floating polygons can be an interesting alternative to traditional superpixels, especially for analyzing scenes with strong geometric signatures, as man-made environments. Existing algorithms produce homogeneously-sized polygons that fail to capture thin geometric structures and over-partition large uniform areas. We propose a kinetic approach that brings more flexibility on polygon shape and size. The key idea consists in progressively extending pre-detected line-segments until they meet each other. Our experiments demonstrate that output partitions both contain less polygons and better capture geometric structures than those delivered by existing methods. We also show the applicative potential of the method when used as preprocessing in object contouring.

1. Introduction

Algorithms for decomposing images into superpixels, *i.e.* small groups of connected pixels, are now standard tools in computer vision. They are typically used as preprocessing to reduce the algorithmic complexity of subsequent tasks while enforcing the spatial consistency between pixels. Some recent works [2, 9, 14] demonstrated the benefit of handling floating polygons instead of superpixels for scalability, storage and region connectivity reasons. Such resolution-independent representations are particularly appealing for analyzing scenes with strong geometric signatures, as man-made environments. Examples of applications include image rendering [31], urban reconstruction [10] or scene illumination [12].

Existing strategies for generating such polygonal partitions are based either on the vectorization of superpixel boundaries [2], or on the construction of geometric data-structures that conform to pre-detected line-segments [9, 14]. The former easily introduces approximation errors when turning superpixels into polygons and requires a good superpixel connectivity which is difficult to guarantee in practice. The latter, which focuses on positioning polygons on each side of line-segments, is globally more robust and comes with some geometric guarantees but produces less accurate results. In particular, line-segment based methods poorly deal with potential intersections of line-segments that might occur in a spatial neighborhood. They assume that pre-detected line-segments constitute entire or almost entire lineic components of an object. In practice, we observe a line-segment is often a small part of such a component, and needs to be extended to properly capture the underlying structure. We also observe that imposing homogeneously-sized polygons does not allow the capture of meaningful polygons as line-segments are not uniformly distributed on the image domain.

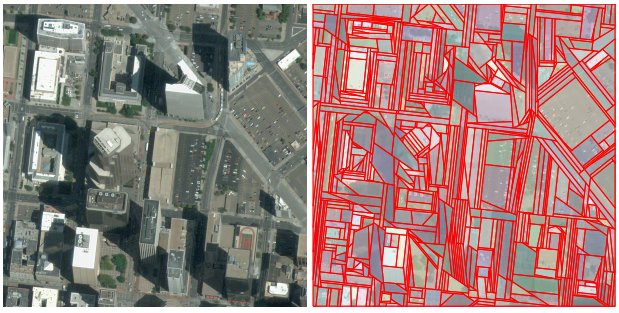


Figure 1. Kinetic partitioning into polygons. Our algorithm decomposes an image (left) into a partition of convex polygons (right). While superpixel-based methods impose homogeneously-sized regions, our polygons are more meaningful, capturing both large components and thin lineic structures that compose, for instance, urban scenes.

Based on these observations, we propose a kinetic approach for partitioning an image into polygons. The key idea consists in progressively extending pre-detected line-segments until they meet each other. We then decide whether line-segments must keep extending based on image gradient considerations and a user-defined number of collisions. This strategy allows us to both recover better junctions in lineic structures and describe objects with polygons more meaningful than superpixel-based polygons. Figure 1 shows an example of such a partition generated from an image exhibiting man-made objects.

The contributions of this work are (i) a computationally efficient kinetic framework able to process big im-

ages in a few seconds, (ii) a shape regularization algorithm from line-segments to preserve parallelism, orthogonality and collinearity relationships, and (iii) an object contouring model that operates on our polygonal partition.

After presenting the related work in Section 2, we detail in Section 3 how line-segments are detected and regularized. The kinetic framework is exposed in Section 4 and evaluated in Section 5. We show the applicative potential of our method on object contouring in Section 6.

2. Related works

Our review of previous work covers the generation of superpixels and polygons, as well as the detection of geometric shapes.

Superpixel decomposition. Region boundary adherence, uniformity of region shapes, computational efficiency, region compactness or simplicity of use constitute the main evaluation criteria to measure the quality of a superpixel decomposition. Algorithms proposed in the literature rarely score high on each of these criteria. Iterative refinement methods as [1, 22, 30] are time and memory efficient and easy to use. More global approaches that exploit energy minimization on graphs as [20, 28] might produce results with better region boundary adherence but are computationally less efficient. By operating at the scale of the pixel, the output of these methods is however resolution-dependent and heavy to store, and also often comes with few guarantees on the region connectivity.

Polygon decomposition. A natural way to decompose an image into polygons is to vectorize the chains of boundary pixels of superpixels. Achanta *et al.* [2] position vertices when at least three superpixels meet, and vectorize the chain of pixels in between these vertices using a Douglas-Peucker based-algorithm [8]. It produces an image partitioning into non-convex polygons that approximate the initial superpixels. The alternative way consists in fitting geometric data-structures on the image. Duan *et al.* [9] decompose images into uniformly-sized convex polygons by building a Voronoi tessellation that conforms to pre-detected line-segments. Gevers *et al.* [16] and Forsythe *et al.* [14] build a Delaunay triangulation, before regrouping triangles to form polygons. The former operates by iteratively splitting triangles with heterogeneous radiometry whereas the latter uses a constraint Delaunay triangulation that conforms to pre-detected line-segments. Although exploiting line-segments to guide the polygonal partitioning is computationally efficient, existing methods [9, 14] fail to properly recover the junctions of lineic structures, leading in best situations to the generation of many thin polygons around a junction.

Shape detection and regularization. Fitting geometric shapes in images is an efficient way to synthesize a huge number of pixels into a few parametric functions. Line-segments constitute the most common geometric shapes for analyzing images, especially when the observed structures are lineic [18]. If the Hough detector has been widely used in the literature for decades, more recent algorithms improved the quality of line-segment detection while guaranteeing fast running times [11], and even false detection control [29]. However line-segments returned by such algorithms do not preserve geometric regularities of observed structures, as for instance line alignments in a regular layout of windows on a facade image. Global regularization is often a valuable processing to both (i) correct imprecisions and (ii) reduce output complexity by removing redundant shapes. Existing regularization methods typically operate from 3D shapes either by iterative refinements [24] or by energy minimization [25].

3. Shape detection and regularization

We now expose the first step of our algorithm that consists in detecting and regularizing line-segments.

Detection of Line-segments. We use the Line-Segment Detector (LSD) [29] to extract line-segments from images. Based on a region-growing approach operated on image gradients, this algorithm has several interesting properties including a linear algorithmic complexity in the number of pixels in the image and a mostly parameterless control scheme with respect to other existing algorithms.

Global regularization. We optionally operate a global regularization of line-segments in order to (i) correct imprecisions and (ii) reduce the occurrence of skinny cells in the subsequent image partitioning detailed in Section 4. This process is mainly designed for images with man-made objects without strong perspective effects. We propose two quadratic formulations performed sequentially for computational efficiency that first re-orient and then re-align LSD line-segments with respect to the three principal geometric regularities used for characterizing shapes of man-made objects, *i.e.* parallelism, orthogonality and collinearity.

By denoting by $x_i \in [-\theta_{max}, \theta_{max}]$ the quantity to be added to the initial orientation of the line-segment i with respect to its center, we formulate the line-segment re-orientation problem by minimizing the energy

$$U(\mathbf{x}) = (1-\lambda)D(\mathbf{x}) + \lambda V(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = (x_1, .., x_n)$ is a configuration of perturbations operated on the n line-segments, $D(\mathbf{x})$ and $V(\mathbf{x})$ represent a data term and pairwise potential respectively, and $\lambda \in [0, 1]$ is a parameter weighting these two terms, typically 0.8 in our experiments.

Data term $D(\mathbf{x})$ discourages strong angle deviations with respect to their initial orientation. It is expressed by

$$D(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i}{\theta_{max}}\right)^2 \quad (2)$$

Pairwise potential $V(\mathbf{x})$ encourages pairs of spatially-close line-segments which are nearly-parallel or nearly-orthogonal to be exactly parallel or orthogonal:

$$V(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n}\sum_{j>i}\mu_{ij}}\sum_{i=1}^{n}\sum_{j>i}\mu_{ij}\frac{|\theta_{ij} - x_i + x_j|}{4\theta_{max}} \quad (3)$$

where θ_{ij} measures how far the relative angle α_{ij} between line-segments i and j is from a straight or right angle. Formally, $\theta_{ij} = \alpha_{ij} \pmod{\pi}$ if $\alpha_{ij} \in [-\frac{\pi}{4}, \frac{\pi}{4}[\cup[\frac{3\pi}{4}, \frac{5\pi}{4}[$ and $\theta_{ij} = \alpha_{ij} - \frac{\pi}{2} \pmod{\pi}$ otherwise.

The dummy variable μ_{ij} returns 1 if line-segments i and j are (i) spatially close and (ii) $|\theta_{ij}| < 2\theta_{max}$, and 0 otherwise. We consider that two line-segments are spatially close if, after building a Delaunay triangulation of points regularly sampled on all the line-segments, at least one Delaunay edge connects their respective sampled points. In practice, sampled points are distant by 10 pixels. Note that time for building a Delaunay triangulation is rather negligible with respect to other operations. Such a neighborhood strongly reduces the number of irrelevant interactions with respect to a standard Euclidean distance by imposing a direct visibility in between line-segments.

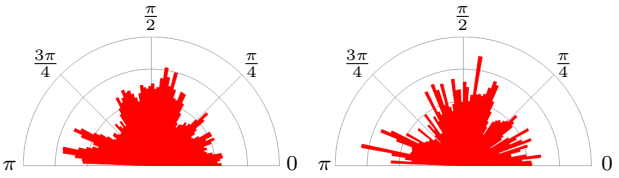


Figure 2. Line-segment re-orientation on a 9 Mpixels satellite image of Seoul city. The quite-uniform orientation histogram of initial line-segments (left) makes a few dominant orientations appear after the regularization (right).

Assuming there are m non-zero μ_{ij}, we introduce a new set of variables $\mathbf{y} = (y_1, .., y_m)$ so that our formulation can be turned to a quadratic optimization problem with $(n+m)$ variables and $2(n+m)$ linear constraints:

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & (1-\lambda)\sum_{i=1}^{n}\left(\frac{x_i}{\theta_{max}}\right)^2 + \lambda\sum_{k=1}^{m} y_k \\
\text{subject to} \quad & x_i \leq \theta_{max}, \quad i = 1, \ldots, n \\
& -x_i \leq \theta_{max}, \quad i = 1, \ldots, n \\
& y_k \leq \frac{1}{4\theta_{max}}(\theta_{ij} - x_i + x_j), \quad k = 1, \ldots, m \\
& -y_k \leq \frac{1}{4\theta_{max}}(\theta_{ij} - x_i + x_j), \quad k = 1, \ldots, m
\end{aligned} \quad (4)$$

This minimization problem is solved using a standard optimization library [15].

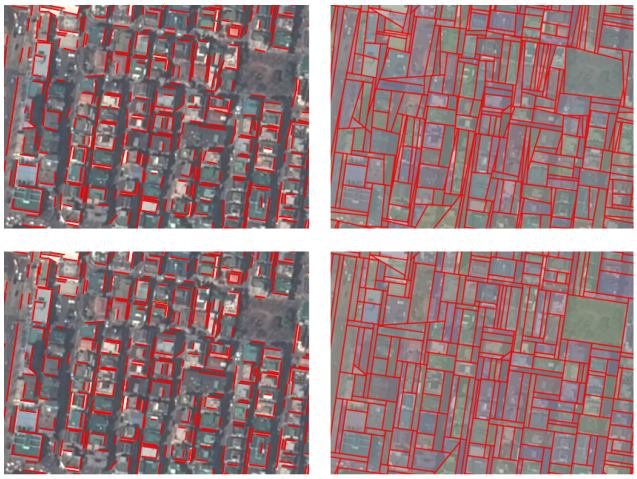


Figure 3. Global regularization of line-segments. Floating line-segments detecting by LSD (top left) yields a complex polygonal partition with many meaningless polygons (top right). By regularizing them (bottom left), we both simplify the partition with typically around 20% less polygons, and improve the polygon alignments with typical building layouts.

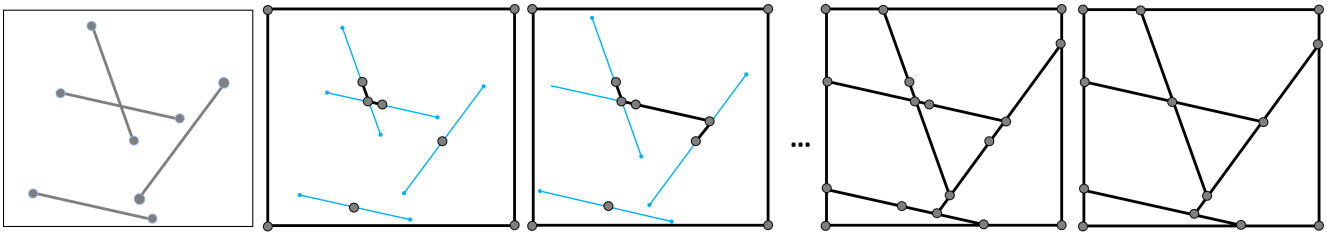
We then use an analogous formulation to re-align line-segments. By now denoting by $x_i \in [-d_{max}, d_{max}]$ the translation to be operated on the line-segment i along its orthogonal vector, we minimize the energy $U(\mathbf{x})$ of Equation 1 with $D(\mathbf{x}) = \sum_{i=1}^{n}(x_i/d_{max})^2$ and $V(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j>i}\mu'_{ij}(|d_{ij} - x_i + x_j|/4d_{max})$. Here, d_{ij} corresponds to the distance between the support lines of parallel line-segments i and j, whereas μ'_{ij} returns 1 if (i) $\mu_{ij} = 1$, (ii) line-segments i and j are parallel, and (iii) $d_{ij} < 2d_{max}$, and 0 otherwise. In our experiments, we typically fixed θ_{max} and d_{max} to 5° and 1 pixel. Figures 2 and 3 show the impact of regularization on urban scenes.

4. Kinetic partitioning

We now present our partitioning algorithm, after a brief introduction to Kinetic Data Structures.

4.1. Background

A kinetic data-structure consists in a set of geometric *primitives*, whose coordinates are continuous functions of time. The purpose of kinetic frameworks [4, 17] is to maintain the validity of a set of statements that apply to such a data-structure. These statements, called *certificates*, are built upon *predicates*, which are functions of the geometric primitives that return a discrete set of values. Most often, predicates evaluate the sign of an algebraic expression binding two primitives or more, and therefore convey an idea of interaction between them.



(a) Detected line-segments (b) Initialization (c) First event (d) Last event (e) Finalization

Figure 4. Illustration of the kinetic partitioning mechanism. Line-segments (a) are converted into an initial planar graph (b). As extremities of primitives extend (blue dots), they meet each other, which enriches the planar graph with new nodes and edges (c,d). After the last collision, the planar graph is simplified by removing unnecessary nodes (e).

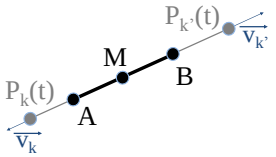
As primitives move, *events* may occur when certificates become invalid. Kinetic frameworks show a strong algorithmic interest to dynamically order the times of occurrences of the events within a *priority queue*. When an event actually happens on top of the priority queue, the geometric objects responsible for a certificate failure and the priority queue itself are updated, so that the kinetic data structure remains valid at any time of the simulation.

Examples of kinetic data structures include dynamic Delaunay triangulations of a set of moving vertices [3], or polyhedral surface reconstruction from point clouds [6].

4.2. Algorithm

We propose a kinetic framework in which the line-segments are progressively lengthening in the image domain. The underlying data-structure is a dynamic planar graph $G_t = (V_t, E_t)$ that partitions the image domain, with V_t and E_t the set of vertices and edges respectively at time t. When line-segments intersect, the complexity of the graph evolves with typically the insertion of new vertices and edges so that it remains planar. We define below the primitives, certificates and update operations of our kinetic formulation.

Primitives. Because the two extremities of a line-segment should be able to expand independently, our primitives correspond to half line-segments. Formally, a detected line-segment between points A and B generates two primitives $s_k(t) = [MP_k(t)]$ and $s_{k'}(t) = [MP_{k'}(t)]$ where fixed point M is the mid-point of A and B, and moving points $P_k(t)$ and $P_{k'}(t)$ evolve with time such that



$$P_k(t) = A + \overrightarrow{v_k} \times t \quad (5)$$

$$P_{k'}(t) = B + \overrightarrow{v_{k'}} \times t \quad (6)$$

where $\overrightarrow{v_k}$ (respectively $\overrightarrow{v_{k'}}$) is the speed vector of primitive $s_k(t)$ (resp. $s_{k'}(t)$) of direction $\overrightarrow{MA}$ (resp. $\overrightarrow{MB}$) and intensity v_k (resp. $v_{k'}$). In our experiments, v_k is set to 1.

Certificates. For each primitive s_i, we define the certificate function $C_i(t)$ as

$$C_i(t) = \prod_{\substack{j=1 \\ j \neq i}}^{N} Pr_{i,j}(t) \quad (7)$$

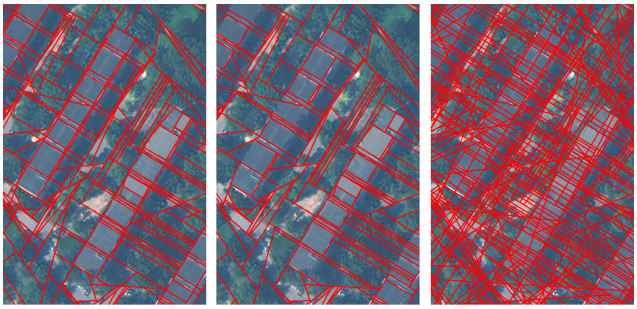
where N is the number of primitives of the kinetic system, and $Pr_{i,j}(t)$ the predicate function that returns 0 when primitive s_i enters in collision with primitive s_j, *i.e.* when the distance from point to line-segment $d(P_i(t), s_j(t)) = 0$, and 1 otherwise. Primitive s_i is called the *source primitive*, and s_j, the *target primitive*. We also call *collision point*, the point located at the intersection of two primitives.

Initialization. We construct the planar graph at $t = 0$ by inserting as vertices (i) the mid-point of each segment, (ii) the four corner points of the image domain, and (iii) points located at the intersection of two line-segments, if any. We set edges between the four successive corner points as well as in between possible intersection points and the mid-points of their corresponding line-segments, as illustrated in Figure 4-(b). We also create the priority queue by computing and sorting all the times for which certificates $C_i(t) = 0$ for $i = 1..N$. Instead of considering all possible pairs of line-segments at once, we compute several priority queues in successive time intervals $[kT, (k+1)T[$ to reduce the algorithmic complexity. In practice, when k is incremented, a new priority queue is built from events occurring within this temporal range. By defining the bounding box of a primitive as the smallest image-aligned square that contain the primitive at time $(k+1)T$, and by assuming primitives extend at constant speed, we easily find these events as the pairs of primitives whose bounding boxes overlap. T is fixed to 50 in our experiments, which is a good compromise between running time and memory consumption.

Updating operations. The planarity property of our graph is broken when an event happens, *i.e.* when one of the N certificates become null. We repair it by first inserting the collision point in the graph. When three primitives

or more are concurrent, we do not insert this point if it already exists. We then update the edge set of the graph by (i) inserting a new edge between the collision point and the last collision point of the source primitive, and (ii) splitting the edge supporting the target primitive with respect to the collision point, as illustrated in Figure 4-(c).

In addition to graph updates, we also decide whether the source primitive should keep propagating. We stop the propagation of the source primitive if it has entered into collision more than a user-defined number of times K, or else if its potential prolongation aligns well with high gradients in the input image. This second condition allows us to not stop the primitive when an obvious image discontinuity along its supporting line exists. Note that our kinetic data-structure is a motorcycle graph [13] when $K = 1$ and the gradient-based condition is deactivated. Figure 5 shows the impact of the stopping conditions on the output partition.



(a) (b) (c)

Figure 5. Stopping conditions. Setting K to 1 is sufficient to capture the different parts of buildings (a). Deactivating the gradient-based condition leads to omit a few structural components (b) whereas fixing K to a too high value, here 20, gives a too complex partition in which polygons are not meaningful anymore (c).

Finally we update the priority queue by removing the processed event from it, and also, in case the propagation is stopped, all the events created from the certificate function of the source primitive.

Finalization. Once the priority queue is empty, we simplify the planar graph by removing the unnecessary vertices, *i.e.* vertices adjacent to two colinear edges which are thus merged, as illustrated in Figure 4-(e). Optionally, we also remove skinny polygons when the width of their oriented bounding rectangles is lower than 2 pixels. Such polygons, that can hardly be exploited by subsequent tasks, are merged to the biggest adjacent polygon under the condition the new polygon is convex.

The global mechanism of our algorithm is summarized in Algorithm 1. In all our experiments, we set the maximal number of collisions K to 1, except for Figure 5. Note that the returned polygons are convex by construction. As concavities inside polygons only appear when two non-colinear primitives intersect at exactly the same time during the propagation phase, we simply force one of the two primitives to keep propagating. Convexity is an interesting property that makes some geometric computations simpler as polygon intersection, point sampling or Constructive Solid Geometry operations. This is for instance useful in 3D reconstruction [5]. To have non-convex polygons, a possible postprocessing could be to group adjacent convex polygons following a color metric.

Algorithm 1 Pseudo-code of the Kinetic partitioning

```
Initialize the planar graph G
Initialize the priority queue Q
while Q ≠ ∅ do
    Pop the source and target primitives from Q
    Update G
    Test the stopping condition of the source primitive
    Update Q
end while
Finalize the planar graph
```

5. Experiments

We tested our algorithm from large-scale satellite images as well as from the Berkeley dataset [21]. We deactivated the line-segment regularization for Berkeley images which is mainly composed of organic shapes. Our main parameter is the LSD scale which allows us to control the sensitivity to image noise, and thus the amount of input line-segments. Despite our algorithm does not offer an exact control on the output number of polygons, this parameter directly impacts on it, as illustrated in Figure 6.

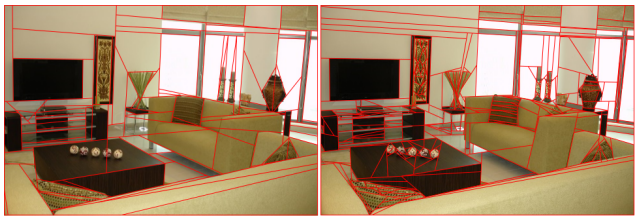


Figure 6. Partition complexity. We can produce partitions with varying numbers of polygons by tuning the sensitivity of the line-segment detector. Left partition with 113 polygons is sufficient for capturing the indoor structure and the main furnitures. Right partition (365 polygons) also captures smaller details as some patterns of the background painting.

We compared our algorithm with state-of-the-art superpixel methods SNIC [2] and ERS [20] and polygonal partitioning methods VORONOI [9] and SNICPOLY [2]. Because these methods are designed to produce homogeneously-sized regions, our output partitions are visually different, combining both large polygons on homogeneous image areas and thin polygons on lineic structures as illustrated in Figure 7. Among the tested methods, only

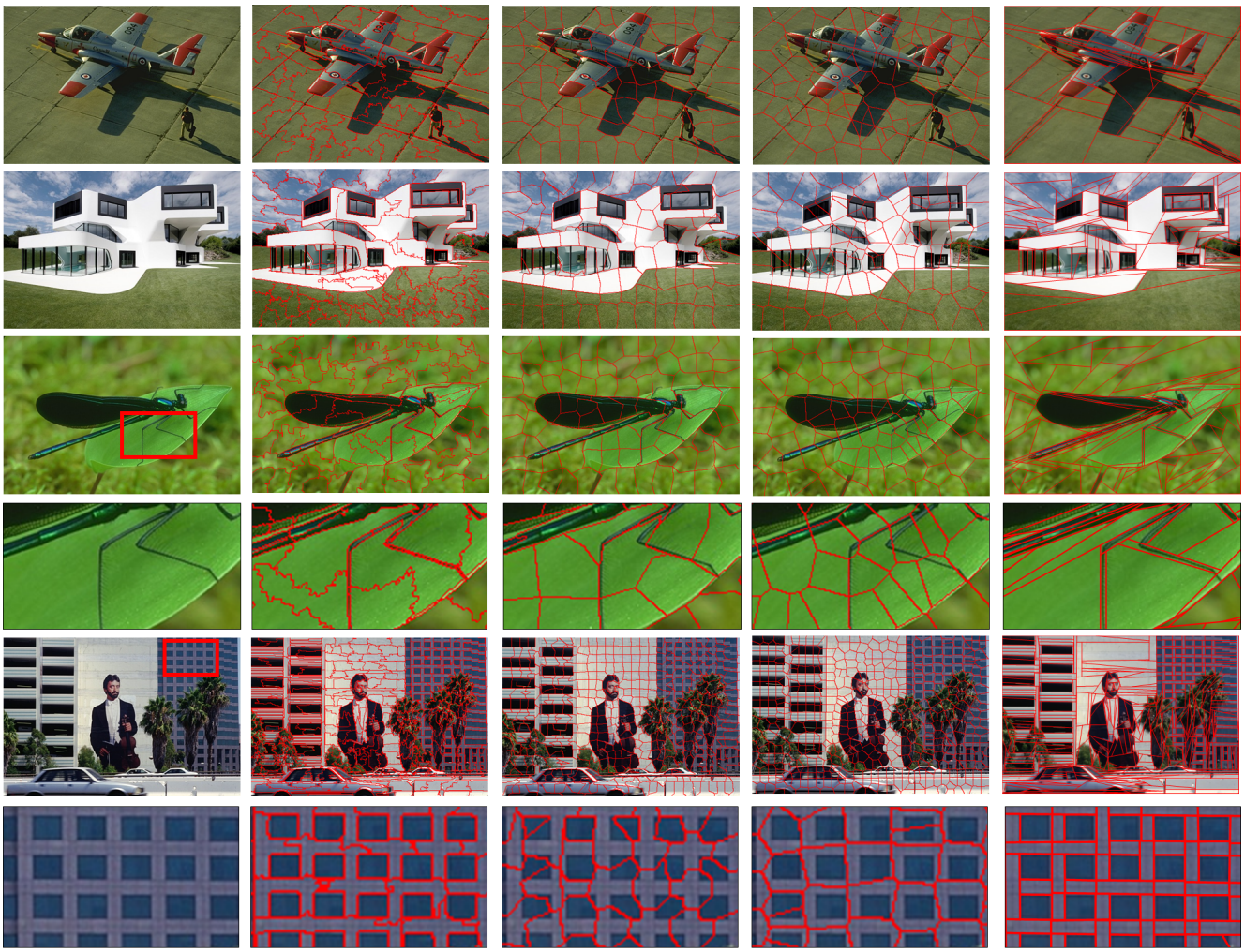


Figure 7. Visual comparisons with superpixel and polygonal partitioning methods. Contrary to existing methods designed to deliver homogeneously-sized regions, our partitions combine large polygons capturing homogeneous areas, as the shadow under the airplane, and thin polygons describing lineic structures as the legs of the dragonfly. For an identical number of output regions, our algorithm produces more meaningful polygons, as those capturing the windows of the facade image.

ERS offers enough flexibility on region shapes to capture lineic structures like us. However, converting ERS superpixels into polygons is a delicate task because of boundary irregularities and region connectivity ambiguities. Our algorithm performs best on man-made scenes in which objects or object parts can be well captured by polygons.

We evaluated our algorithm on the Berkeley300 dataset [21] using standard quality criteria of superpixel methods, in particular the boundary recall as defined in [23] as well as the boundary precision. The former indicates the ratio of Ground Truth contours correctly recovered by the output region boundaries, whereas the latter measures the ratio of output region boundaries that correctly recovers the Ground Truth contours. We measured the boundary precision on the entire image, contrary to some works [2] that compute it on an ϵ-domain around the Ground Truth contours. For measuring the quality criteria from polygonal partitions, the edges of floating polygons have been discretized into pixel boundaries. We measured these criteria for partitions returning between 50 and $1,000$ regions. Figure 8 shows our algorithm outperforms polygonal partitioning methods VORONOI and SNICPOLY on boundary recall by quite a big margin as their scores at a given number of polygons remain lower than ours with twice less polygons. Our algorithm performs best for a number of regions between 400 and 800, with a boundary recall even higher than superpixel method SNIC. Because our partitions contain large-sized polygons, our algorithm even outperforms superpixel methods on the precision to recall curve when recall is higher than 0.85. To get homogeneously-sized polygons, we can apply a Poisson-disk sampling as postprocessing, similarly to [9]. Its effects on the boundary recall are shown through

the curve KIPPI-HOMO: the recall decreases but remains higher than SNICPOLY and VORONOI. When deactivating Poisson disk sampling on VORONOI, the boundary recall improves by a few hundredths but remains lower than SNICPOLY as shown with the curve VORONOI-HETERO.

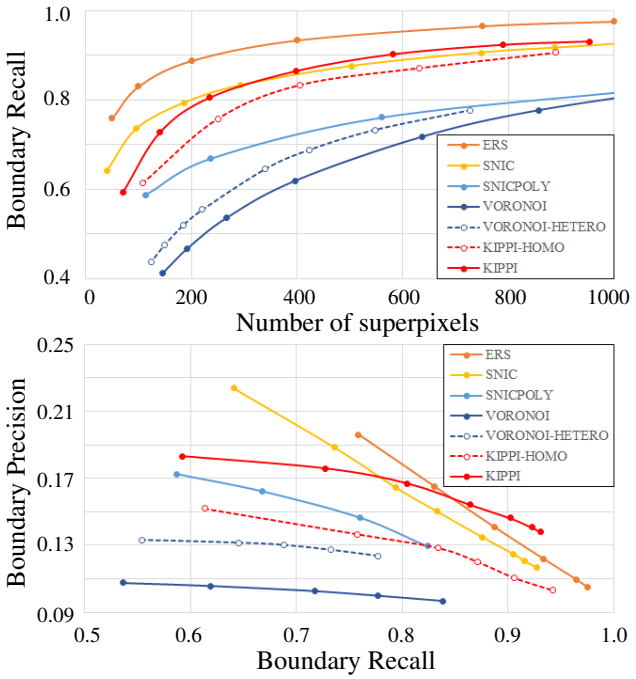


Figure 8. Quantitative evaluation. Our algorithm outperforms polygonal partitiong methods VORONOI and SNICPOLY on boundary recall while approaching the scores of the best superpixel methods. Because we allow polygons to be large for capturing big homogeneous areas, our algorithm offers a better compromise between boundary precision and boundary recall than other methods when recall is high, *i.e.* higher than 0.85.

By reasoning at the scale of geometric shapes instead of pixels, and by exploiting an efficient framework based on Computational Geometry, our algorithm is computationally efficient and scalable. As shown in Table 1, a few minutes are necessary to process a massive satellite image of several hundred millions pixels on a single standard computer. In terms of storage, polygons and their connectivity can be saved in a very compact way with a planar graph.

Our algorithm has a few shortcomings. First, it does not offer to the user an exact control on the number of output polygons. Also, regularization of line-segments is not effective on organic images: it reduces the complexity of the partition at the expense of accuracy. Finally, missing line-segments on small structural parts can lead to under-segmentation situations that are currently not handled by our algorithm. One solution would be to split polygons with heterogeneous radiometry within the kinetic data-structure.

	Facade 154Kpix	Aerial 2.46Mpix	Satellite 106Mpix
# Line-segments	847	3178	171.1K
# Output polygons	530	2488	124.5K
Line-segment detection	52.4 ms	0.59 s	70.7 s
Regularization	72.8 ms	0.35 s	654.5 s
Kinetic partitioning	51.2 ms	0.23 s	45.1 s
Total time	0.195 s	1.41 s	795.6 s

Table 1. Performances on three different image sizes (Facade from Figure 7 -bottom, Aerial from Figure 10-right, and Satellite whose a cropped part is illustrated on Figure 1) in terms of running time.

6. Application to object contouring

Object contouring by polygonal shapes provides a compact and structure-aware representation of object silhouettes, in particular in man-made environments [7, 27]. To achieve polygonal object contouring from our partition, we associate each polygon with a binary activation variable indicating if it belongs to the objects of interest or not, similarly to [19] with superpixels. The output polygonal contours correspond to the set of edges separating active polygons from inactive ones, which ensures that the contours are closed by construction. The problem is formulated as a standard energy minimization with a data term measuring the agreement between the binary variable of each polygon and an underlying probability map H, and a smoothness term based on Potts model to favor compact contours.

#edges: 130 #edges: 308 #edges: 476

Figure 9. Trade-off between fidelity and simplicity. Polygonal partitions with low complexity give compact polygonal contours that roughly approximate the object silhouette (left). More refined partitions allow us to better capture shape details (right).

For each input image, we compute the probability map H from a few user-provided scribbles, which roughly characterize the radiometric distribution of the foreground objects of interest and the image background. We express the probability $H(i|l)$ of a pixel i to belong to class $l = \{0, 1\}$ as its normalized RGB distance to the closest color in the set of scribbled pixels belonging to that class

$$H(i|l) = \frac{\min_{j \in S_l} \|I(i) - \widehat{I}(j)\|_2^2}{\min_{j \in S_0} \|I(i) - \widehat{I}(j)\|_2^2 + \min_{j \in S_1} \|I(i) - \widehat{I}(j)\|_2^2} \quad (8)$$

Figure 10. Object contouring. Using our partition as input, we are able to capture details in the image missed by other algorithms while producing polygons with lower complexity. Note in particular how thin structures as the legs of the dragonfly or the propeller of the airplane are recovered. Our method performs best on man-made objects composed of piecewise-linear contours, as roofs (right).

where S_0 (respectively S_1) is the set of pixels scribbled as foreground (resp. background), and $\widehat{I}$ is the input image convolved by a 11×11 mean filter to remove noise. Note that more advanced methods could be used to predict foreground and background pixels. This would surely lead to better results, but this is beyond the scope of this paper.

Despite the simplicity of our color model H, Figure 10 shows our method achieves good results with both organic and man-made shapes. Output polygons capture well the object silhouettes while having a low complexity. In particular, it outperforms results returned by Grabcut [26] followed by a Douglas-Peucker vectorization of the border pixels [8]. Replacing our partitions by VORONOI [9] reduces accuracy. In particular, VORONOI partitions cannot handle thin structures and tend to produce complex polygonal contours zigzagging around the true silhouettes.

7. Conclusion

We proposed a kinetic approach to partition images into floating polygons. Whereas existing methods impose homogeneously-sized polygons in the style of superpixels, our line-segment extension mechanism offers more flexibility on polygon shapes. This allows us to better recover geometric patterns contained in man-made and organic images, and capture thin structures without over-partitioning large homogeneous areas. By reasoning at the scale of geometric shapes instead of pixels within a computational geometry framework, our algorithm is scalable and computationally efficient. We demonstrated the strong applicative potential of our algorithm when used as preprocessing in object contouring. In particular, a simple polygon selection model allows us to accurately capture the silhouette of complex objects by a polygon with fews edges.

A natural extension of this work would be to enrich the geometric shapes by more complex parametric functions than line-segments. Quadrics and Bezier curves would bring more versatility to output partitions. We also would like to investigate on the exploitation of such a kinetic framework in 3D where geometric shapes are planes. This would allow the partition of 3D data by polyhedra in a very efficient way.

Acknowledgments

This work was supported by Luxcarta. The authors thank Radhakrishna Achanta and Jean-Dominique Favreau for provinding materials for experimental comparisons.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art

superpixel methods. *PAMI*, 34(11), 2012. 2

[2] R. Achanta and S. Susstrunk. Superpixels and polygons using simple non-iterative clustering. In *CVPR*, 2017. 1, 2, 5, 6

[3] P. K. Agarwal, J. Gao, L. Guibas, H. Kaplan, V. Koltun, N. Rubin, and M. Sharir. Kinetic stable delaunay graphs. In *Symposium on Computational Geometry*, 2010. 4

[4] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(1), 1999. 3

[5] A. Bodis-Szomoru, H. Riemenschneider, and L. Van Gool. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *CVPR*, 2014. 5

[6] M. Brédif, D. Boldo, M. Pierrot-Deseilligny, and H. Maître. 3d building model fitting using a new kinetic framework. *arXiv:0805.0648*, 2008. 4

[7] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017. 7

[8] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitizedline or its caricature. *Cartographica: The International Journal for Geographic Information*, 10(2), 1973. 2, 8

[9] L. Duan and F. Lafarge. Image partitioning into convex polygons. In *CVPR*, 2015. 1, 2, 5, 6, 8

[10] L. Duan and F. Lafarge. Towards large-scale city reconstruction from satellites. In *ECCV*, 2016. 1

[11] M. Dubska, A. Herout, and J. Havel. Pclines line detection using parallel coordinates. In *CVPR*, 2011. 2

[12] S. Duchene, C. Aliaga, T. Pouli, and P. Perez. Mixed illumination analysis in single image for interactive color grading. In *Symposium on Non-Photorealistic Animation and Rendering*, 2017. 1

[13] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry*, 22(4), 1999. 5

[14] J. Forsythe, V. Kurlin, and A. Fitzgibbon. Resolution-independent superpixels based on convex constrained meshes without small angles. In *International Symposium on Visual Computing*, 2016. 1, 2

[15] E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Trans. on Mathematical Software*, 29(1), 2003. 3

[16] T. Gevers and A. W. M. Smeulders. Combining region splitting and edge detection through guided delaunay image subdivision. In *CVPR*, 1997. 2

[17] L. Guibas. Kinetic data structures. In *Handbook of Data Structures and Applications*. Citeseer, 2004. 3

[18] D. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009. 2

[19] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *ECCV*, 2010. 7

[20] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, 2011. 2, 5, 6

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 5, 6

[22] R. Nagar and S. Raman. Symmslic: Symmetry aware superpixel segmentation. In *ICCV*, 2017. 2

[23] P. Neubert and P. Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, 2012. 6

[24] S. Oesau, F. Lafarge, and P. Alliez. Planar Shape Detection and Regularization in Tandem. *Computer Graphics Forum*, 35(1), 2016. 2

[25] T.-T. Pham, T.-J. Chin, K. Schindler, and D. Suter. Interacting geometric priors for robust multimodel fitting. *Trans. on Image Processing*, 23(10), 2014. 2

[26] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Trans. on Graphics*, volume 23, 2004. 8

[27] X. Sun, M. Christoudias, and P. Fua. Free-shape polygonal object localization. In *ECCV*, 2014. 7

[28] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *ECCV*, 2010. 2

[29] R. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *PAMI*, 32(4), 2010. 2

[30] J. Wang and X. Wang. Vcells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations. *PAMI*, 34(6), 2012. 2

[31] K. Yang, Z. Sun, C. Ma, and W. Yang. Paint with stitches: A random-needle embroidery rendering method. In *Proc. of Computer Graphics International*, 2016. 1