Extracting Geometric Structures in Images with Delaunay Point Processes

Jean-Dominique Favreau, Florent Lafarge, Adrien Bousseau, Alex Auvolat

Extracting Geometric Structures in Images with Delaunay Point Processes

Jean-Dominique Favreau, Florent Lafarge, Adrien Bousseau and Alex Auvolat

Abstract—We introduce Delaunay Point Processes, a framework for the extraction of geometric structures from images. Our approach simultaneously locates and groups geometric primitives (line segments, triangles) to form extended structures (line networks, polygons) for a variety of image analysis tasks. Similarly to traditional point processes, our approach uses Markov Chain Monte Carlo to minimize an energy that balances fidelity to the input image data with geometric priors on the output structures. However, while existing point processes struggle to model structures composed of inter-connected components, we propose to embed the point process into a Delaunay triangulation, which provides high-quality connectivity by construction. We further leverage key properties of the Delaunay triangulation to devise a fast Markov Chain Monte Carlo sampler. We demonstrate the flexibility of our approach on a variety of applications, including line network extraction, object contouring, and mesh-based image compression.

Index Terms—Spatial point process, Delaunay triangulation, Geometric structures, Line network extraction, Object contouring, Image compression

1 Introduction

Many Vision tasks involve the extraction of geometric *structures* from images. Typical examples include the extraction of networks of blood vessels from retinal images, the extraction of building footprints from urban satellite images, or the extraction of 3D surfaces from multiple photographs of a scene. While these structures cover chains or regions of pixels in the input images, they are often converted into sets of geometric *primitives* for subsequent global analysis and efficient storage. For instance, vessel networks can be represented as planar graphs made of line segments, building footprints can be represented as closed polygons, and 3D surfaces can be represented as triangular meshes.

Unfortunately, existing algorithms often decouple the detection of local primitives from the construction of global structures. Continuing on our examples, object contouring methods typically detect line segments along image discontinuities before assembling them to form polygons [1], [2], and multiview stereo reconstruction algorithms extract 3D points by feature matching before interpolating them with a surface mesh [3], [4]. While this two-step approach reduces computational burden, the quality of the resulting structures depends heavily on the local decisions taken during primitive detection.

As an alternative to primitive detection, *generative models* seek to synthesize structures and measure their agreement with image data. In particular, *point processes* have shown their ability to generate configurations of geometric elements that align with image content [5]. However, the synthesis of large-scale structures often requires strong interactions between geometric primitives, which are hard to model with existing formulations. For example, a point process that would generate independent line segments is very unlikely to produce coherent line networks where segments only join at their endpoints. The key idea of our work is to constrain point processes to only produce well-connected geometric structures.

• *J.-D. Favreau, F. Lafarge, A. Bousseau and A. Auvolat are with Inria Sophia Antipolis, France. E-mail: Firstname.Lastname@inria.fr*

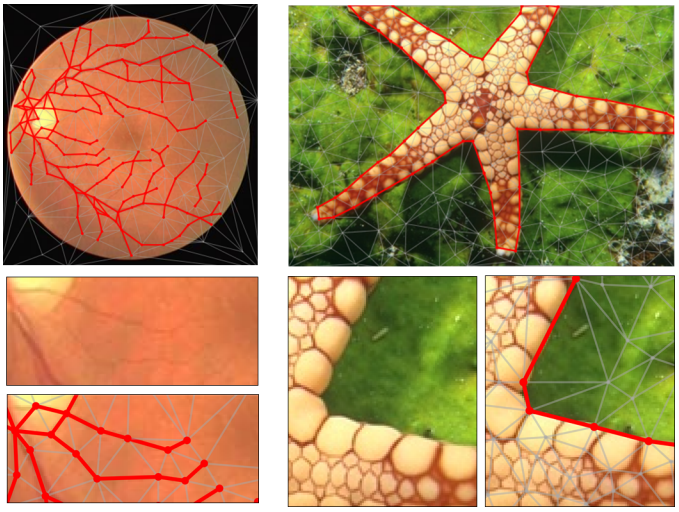


Fig. 1. Example applications of Delaunay Point Processes to extract planar graphs representing blood vessels in retina images (left), and complex polygons representing object silhouettes (right). The point distribution creates a dynamic Delaunay triangulation while edge and facet labels specify the geometric structure (see red edges on close-ups).

This paper focuses on the extraction of 2D structures and leaves the extraction of 3D entities to future work. Our main observation is that any 2D structure composed of non-overlapping lines or polygons can be embedded in a triangulation of the image domain. Given this representation, generating linear or triangular primitives amounts to inserting new vertices in the triangulation, which is a standard operation for existing geometry libraries [6]. Extracting global structures then amounts to grouping subsets of edges or facets of the triangulation. By construction, our representation offers strong geometric guaranties, such as the fact that line segments and polygons always meet vertex-to-vertex or edge-to-edge. We further build on properties of the Delaunay triangulation to propose an efficient sampler for fast stochastic

optimization of the geometric structures we wish to extract. We demonstrate the versatility of this approach to extract 2D structures for a variety of Vision tasks.

In summary, our main contributions are (i) a general framework to extract geometric structures for a variety of Vision problems, (ii) an efficient stochastic optimization to find high-quality structures, and (iii) models for line network extraction, object contouring, and image compression; demonstrating the potential of our approach on real-world tasks.

2 Background on Point Processes

We first provide the necessary background on point processes and their applications before describing our novel approach, which we call *Delaunay Point Processes*. We refer the reader to the book by Descombes [7] for a detailed presentation of the theory of point processes for image analysis.

Definition. Point processes are probabilistic models that describe random configurations of points in a continuous bounded domain K, where the number of points of a configuration and their positions in the domain are random variables [8]. They have been introduced in Vision by Baddeley and Moller to extend traditional Markov Random Fields (MRFs) with an object-based formalism [9]. We denote $\Omega = \bigcup_{n\in\mathbb{N}} \Omega_n$ the configuration space of a point process, where the sub-spaces Ω_n correspond to configurations of exactly n points distributed in K. We denote $\boldsymbol{p} \in \Omega$ a realization of this point process, and $p \in \boldsymbol{p}$ a point of the resulting configuration.

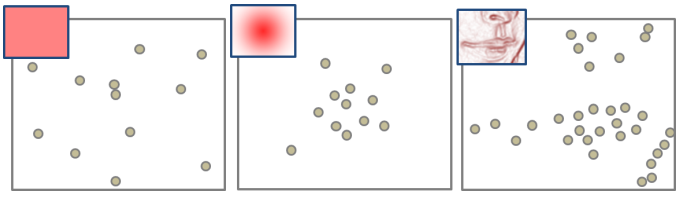


Fig. 2. Point processes distribute points randomly in a bounded domain. While the left example illustrates a uniform distribution, the middle and right examples show point processes guided by a non-uniform density h (top left insets). In particular, the right example uses the image gradient magnitude as a density to distribute points along image contours.

The simplest point process is the homogeneous Poisson point process, for which the number of points follows a discrete Poisson distribution and the position of the points follows a uniform distribution. As illustrated on Figure 2, more complex configurations can be obtained by guiding the point process with a density $h(.)$ defined in Ω. Intuitively, $h(\boldsymbol{p})$ measures the probability of the realization $\boldsymbol{p}$ to occur. By carefully designing the density $h(.)$, practitioners can model processes where the number and position of the points are consistent with input data and where neighboring points obey specific spatial interactions. In the remaining of this paper we often express the density as a Gibbs energy $U(.)$ which we seek to minimize

$$h(\boldsymbol{p}) \propto \exp -U(\boldsymbol{p}). \tag{1}$$

Markovian property. Similarly to Markov Random Fields, the Markovian property of a point process provides a spatial dependency between neighboring points in a configuration. Formally, a point process of density h is *Markovian under the neighborhood relationship* $\sim$ if and only if $\forall \boldsymbol{p} \in \Omega$ such that $h(\boldsymbol{p}) > 0$, and $\forall q \in K$, $h(\boldsymbol{p} \cup \{q\})/h(\boldsymbol{p})$ only depends on q and its neighbors $\{p \in \boldsymbol{p} \,:\, q \sim p\}$. In other words, when adding a point to a configuration, the resulting variation of density only depends on the new point and its neighbors in the configuration. As discussed next, the Markovian property is essential to many efficient optimization algorithms because it guarantees that the variation of energy induced by a local perturbation of a configuration can be computed using a small number of points around that perturbation.

The symmetric relationship $\sim$ is usually defined via a maximal Euclidean distance ϵ between two points of K such that

$$p_i \sim p_j = \{(p_i, p_j) \in \boldsymbol{p}^2 : i > j, ||p_i - p_j||_2 < \epsilon\} \tag{2}$$

Figure 3-left shows a realization of such a point process for $K \subset \mathbb{R}^2$.

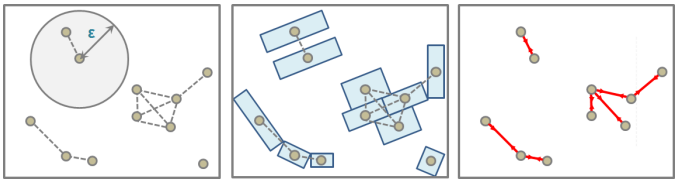


Fig. 3. Markovian point processes. Traditional point processes exploit the Markovian property to define pairs of interacting points, typically a maximal Euclidean distance ϵ between two points (left). Such processes are used for detecting objects in images by associating a simple geometric shape, eg a rectangle [10], to each point (middle), and for extracting line-networks by selecting a subset of pairs of interacting points [11] (right).

Inference. Reversible Jump Markov Chain Monte Carlo (RJMCMC) [12] is a popular family of algorithms to search for configurations that maximize the density h, or equivalently, that minimize the energy U. A RJMCMC sampler simulates a discrete Markov chain $(X_t)_{t\in\mathbb{N}}$ on the configuration space Ω, converging towards a target density specified by U.

Algorithm 1 provides the pseudo-code of a RJMCMC sampler for point processes. The algorithm starts with a random configuration $\boldsymbol{p}_0$. At each iteration, the current configuration $\boldsymbol{p}$ of the chain is perturbed to a configuration $\boldsymbol{p}'$ according to a proposition density $\mathcal{Q}(\boldsymbol{p} \to .)$, also called a *kernel*. The perturbations are local, which implies that the energy variation between configuration $\boldsymbol{p}$ and $\boldsymbol{p}'$ depends only on a few points thanks to the Markovian property of the point process. The configuration $\boldsymbol{p}'$ is then accepted as the new state of the chain with a probability that depends on the ratio of kernels $\mathcal{Q}(\boldsymbol{p} \to \boldsymbol{p}')$ and $\mathcal{Q}(\boldsymbol{p}' \to \boldsymbol{p})$, the energy variation between $\boldsymbol{p}$ and $\boldsymbol{p}'$, and a relaxation parameter T_t. We next detail the role of kernels and relaxation, followed by a discussion of existing work on object and structure extraction using point processes.

Kernels. For many applications, the kernel $\mathcal{Q}$ is formulated as a mixture of kernels $\mathcal{Q}_m$ associated with probabilities q_m

$$\mathcal{Q}(\boldsymbol{p} \to .) = \sum_m q_m \mathcal{Q}_m(\boldsymbol{p} \to .), \tag{4}$$

where each kernel $\mathcal{Q}_m$ is typically dedicated to a specific type of perturbation. The kernel mixture must satisfy two necessary conditions to guarantee the convergence of the Markov chain.

Algorithm 1 RJMCMC sampler for point processes

1- Initialize $X_0 = \boldsymbol{p}_0$ and T_0 at $t = 0$;
2- At iteration t, with $X_t = \boldsymbol{p}$,

- Choose a kernel $\mathcal{Q}_m$ according to probability q_m
- Perturb $\boldsymbol{p}$ to $\boldsymbol{p}'$ according to $Q_m(\boldsymbol{p} \rightarrow .)$
- Compute the Green rate

$$R = \frac{\mathcal{Q}_m(\boldsymbol{p}' \rightarrow \boldsymbol{p})}{\mathcal{Q}_m(\boldsymbol{p} \rightarrow \boldsymbol{p}')} \exp\left(\frac{U(\boldsymbol{p}) - U(\boldsymbol{p}')}{T_t}\right) \quad (3)$$

- Choose $X_{t+1} = \boldsymbol{p}'$ with probability $\min(1, R)$, and $X_{t+1} = \boldsymbol{p}$ otherwise

First, to make the Markov chain *irreductible*, the kernel mixture must allow any configuration in Ω to be reached from any other configuration with a finite number of perturbations. Second, to make the Markov chain *reversible*, each kernel must be able to propose a perturbation and its reverse with a non-zero probability. This second condition is necessary to compute the kernel ratio in Equation 3, which provides a detailed balance between a perturbation and its reverse [13].

Point processes typically rely on a *birth and death* kernel that adds or removes a point from $\boldsymbol{p}$ [7]. This kernel is parameterized by a birth probability P_b, the death probability P_d being equal to $1 - P_b$. Denoting $|\boldsymbol{p}|$ the number of points in the current configuration, and λ the parameter of the discrete Poisson distribution that governs the number of points in $\boldsymbol{p}$, the kernel ratio for a birth event can be expressed as

$$\frac{\mathcal{Q}_m(\boldsymbol{p}' \rightarrow \boldsymbol{p})}{\mathcal{Q}_m(\boldsymbol{p} \rightarrow \boldsymbol{p}')} = \frac{P_d}{P_b} \frac{\lambda}{|\boldsymbol{p}| + 1}. \quad (5)$$

Similarly, the kernel ratio for a death event is expressed as $\frac{P_b}{P-d} \frac{|\boldsymbol{p}|}{\lambda}$. Intuitively, λ represents the expected number of points in the output configuration. Choosing a birth (respectively a death) when the number of points in the current configuration is higher (resp. lower) than λ will reduce the chance of accepting the proposed perturbation.

Relaxation. The relaxation parameter T_t, also called the *temperature*, controls the acceptation rate of the RJMCMC sampler. A high temperature allows the algorithm to explore very different configurations, including configurations that temporarily increase the energy. In contrast, a low temperature encourages the algorithm to only accept perturbations that decrease the energy. A common practice is to start with a high temperature for initial exploration of the solution space, before decreasing the temperature to converge to a local miminum. Although a logarithmic decrease of T_t is necessary to ensure convergence to the global minimum from any initial configuration, practitioners typically use a faster geometric decrease of the form

$$T_t = T_0.\alpha^t \quad (6)$$

where T_0 is the initial temperature and α controls the speed of decrease and is typically set to a value inferior yet close to 1. Such a geometric decrease gives an approximate solution close to the optimum [14].

From points to objects. Many Vision tasks involve the extraction of extended objects rather than infinitesimal points. *Marked point processes* are a family of point processes that tackle such tasks by associating each point with a parametric object. For example buildings can be represented with rectangles [10], persons with cylinders [15], [16], or textures with sets of parametric shapes [17]. Figure 3-middle illustrates a realization of a marked point process where each point is associated with a rectangle defined by its orientation, width and length [10]. The domain of this marked point process is thus $K \times M$ with $K \subset \mathbb{R}^2$ and $M =]-\frac{\pi}{2}, \frac{\pi}{2}] \times [l_{min}, l_{max}] \times [L_{min}, L_{max}]$. Marked point processes are particularly effective for the extraction of groups of objects from images because they can model rich spatial interactions and because they do not require the number of objects to be known a-priori. More details on marked point processes can be found in [7].

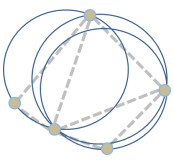
From points to structures. While marked point processes are well adapted to the extraction of groups of disconnected objects, their application to the extraction of connected structures is more challenging because local perturbations may affect the structure globally, breaking the Markovian property necessary for efficient RJMCMC sampling. Prior work attempted to extract geometric structures by designing the point process energy such that it encourages neighboring objects to connect. For example, Lacoste et al. [18] and Sun et al. [19] extract line networks by encouraging line segments to form a graph, while Drot et al. [20] segment images by encouraging triangles to form a tessellation. However, this strategy does not scale well because the probability of sampling objects that connect together decreases quickly with the number of objects. To the best of our knowledge, junction-point processes [11] is the only solution designed to extract structures that are well-connected by construction. As illustrated in Figure 3-right, junction-point processes exploit the $\sim$neighborhood relationship of Equation 2 to define a graph over $\boldsymbol{p}$, where edges link pairs of neighboring points. A subset of edges is then selected to form a planar graph, which represents the output line network. However, a planar graph should not contain crossing edges. Enforcing this topological constraint makes RJMCMC sampling of junction-point processes very slow because a large majority of perturbations yield crossings.

3 Delaunay Point Processes

As discussed in the previous section, existing attempts to extract structures with point processes were strongly limited in their ability to enforce the connectivity of the structure elements. We address this challenge by embedding the point process into a Delaunay triangulation from which we extract structures as groups of edges or triangles. Since the triangulation forms a tesselation of the image plane, our structures are well-connected by construction. While a few studies also combined spatial point processes with Delaunay triangulations [21], [22], they only demonstrated the synthesis of point configurations. In contrast, we augment Delaunay Point Processes with point, edge and facet parameters to extract geometric structures from images.

3.1 Delaunay-Based Neighborhoods

The Delaunay triangulation of a point configuration $\boldsymbol{p}$ subdivides the image domain K into triangles such that no point in $\boldsymbol{p}$ is inside the circumcircle of any triangle. Figure 4-left shows the Delaunay triangulation of the same point configuration as in Figure 3.



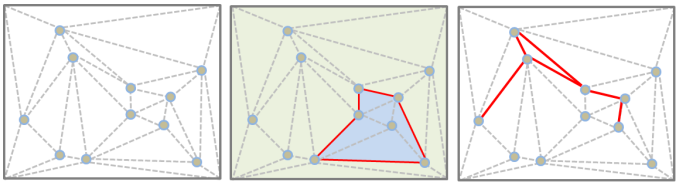


Fig. 4. Delaunay Point Processes. Contrary to traditional point processes, pairs of interacting points are defined more naturally by a Delaunay triangulation instead of an arbitrary distance parameter ϵ (left). Exploiting such a geometric meta-structure allows us to partition the image domain into complex polygons by jointly labeling the triangles (middle) or to extract planar graphs by jointly labeling the edges (right). We add the four corner points of Domain K to $\boldsymbol{p}$ for computing the Delaunay triangulation so that K is entirely partition by triangles.

We denote the set of edges and facets of the Delaunay triangulation of $\boldsymbol{p}$ as $C_2(\boldsymbol{p})$ and $C_3(\boldsymbol{p})$ respectively. This triangulation offers a convenient neighborhood relationship $\sim_D$ for point processes: two points are neighbors if they are connected by an edge in the Delaunay triangulation

$$p_i \sim_D p_j = \{(p_i, p_j) \in \boldsymbol{p}^2 : (p_i, p_j) \in C_2(\boldsymbol{p})\}. \quad (7)$$

We define Delaunay Point Processes as Markovian point processes supported by the Delaunay neighborhood $\sim_D$. Delaunay Point Processes inherit from several interesting properties of the Delaunay triangulation:

- Parameter-free neighborhood. Traditional point processes require a parameter to specify the area of attraction of the neighborhood relationship. Tuning this parameter is often problem-dependent and strongly impacts result quality. Instead, Delaunay edges connect neighboring points without requiring any parameter.
- Geometric guarantees. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation, and thus tends to avoid skinny triangles. This property is especially beneficial to reduce imprecision when measuring radiometric quantities over the pixels covered by a triangle, as is the case in our applications to object contouring (Section 4.2) and image compression (Section 4.3).
- Uniqueness. The Delaunay triangulation of a point set is unique, unless four or more points are inscribed on the same circle, which is very unlikely when the point coordinates are expressed in floating point precision. This property implies that sampling a point set is equivalent to sampling a triangulation.
- Efficient sampling. Perbubations during RJMCMC sampling, *e.g.* removing or adding a point in $\boldsymbol{p}$, only affects the Delaunay triangulation locally. Moreover, such perturbations correspond to the basic operators offered by existing computational geometry libraries to modify Delaunay triangulations [6].
- Flexibility. The Delaunay triangulation is a flexible geometric representation to address numerous Vision problems. As illustrated on Figure 4, we can select edges of the triangulation to represent line-networks, we can group triangles to represent closed contours, or we can assign labels to the triangles to segment the image into parts.

3.2 Marks and Energy Formulation

Similarly to marked point processes, we tackle the extraction of extended geometric structures by associating parameters – or marks – to the elements of the Delaunay point process. However, while traditional marked point processes only associate parameters to points, we also associate parameters to the edges and facets formed by the points. This novel feature of Delaunay point processes is key to extract well-connected line-networks and polygons, which are better expressed via edges and triangles than via punctual primitives.

We denote a geometric structure as $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{m})$ where $\boldsymbol{p}$ defines the geometric configuration of the triangulation, while $\boldsymbol{m}$ represents the set of additional parameters on the triangulation elements. For example, $\boldsymbol{m}$ can identify active edges for line-network extraction, or assign different labels to facets for polygonal object segmentation (Figure 4). Note that $\boldsymbol{m}$ can also take real values such as colors, as demonstrated in our application to image compression (Section 4.3).

Our study of various structure extraction tasks led us to formulate a generic energy U for Delaunay point processes, which we express as the sum of two terms balanced by a parameter $\alpha \in [0, 1]$

$$U(\boldsymbol{x}) = (1-\alpha)U_{fidelity}(\boldsymbol{x}) + \alpha U_{prior}(\boldsymbol{x}). \quad (8)$$

The first term, $U_{fidelity}(\boldsymbol{x})$, measures the agreement of the configuration with image data. For instance, it can measure the alignment of the Delaunay edges with image contours for line network extraction, as detailed in Section 4.1. The second term, $U_{prior}(\boldsymbol{x})$, encodes shape priors on the structures we wish to extract. In the example of line network extraction, U_{prior} can penalize acute angles between successive edges to favor smooth polylines. These two terms can be expressed with local energies on points, edges and facets of the Delaunay triangulation. Note that U not only measures the quality of an output structure $\boldsymbol{x}$, but also accounts for the quality of the underlying triangulation $\boldsymbol{p}$.

3.3 Sampling procedure

We use the RJMCMC algorithm detailed in Algorithm 1 to search for a good approximation of the optimal configuration. In all our applications, the sampling operates on configurations of geometric structures $\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{m})$, which live in a wider space than the point configurations $\boldsymbol{p}$. We now propose three kernels to explore this configuration space: birth or death of a point, relocation of a point, and alteration of a mark. Each of these operators only affect a configuration $\boldsymbol{x}$ locally, which is critical for efficient evaluation of the energy variation at each iteration (Equation 3).

Birth and death kernel adds or removes a point from $\boldsymbol{p}$, as detailed in Section 2. In computational geometry terms, it inserts or removes a vertex from the Delaunay triangulation as illustrated in Figure 5. In practice, we give birth and death the same probability ($P_b = P_d = 0.5$). In case of a death, we select one of the points from $\boldsymbol{p}$ randomly. In case of a birth, we create a new point in the image domain K. While we could draw the position of this point from a uniform distribution, this is often inefficient for Vision applications where the structures of interest lie along image contours. We achieve much faster sampling by following a distribution specified by image gradients. Once a vertex is added (respectively removed), we update the marks of its adjacent edges and facets by uniform sampling from the mark

domain.

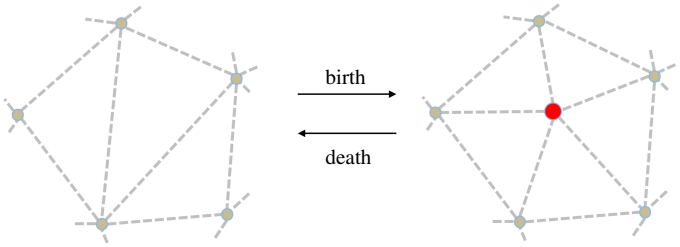


Fig. 5. Birth and Death kernel. A birth inserts a new vertex in the triangulation and recomputes the edge connectivity around it by applying edge flips recursively until the circumcirle condition is valid everywhere. A death removes a vertex and its adjacent edges and reconnect its adjacent vertices so that the circumcirle condition is valid.

Point relocation modifies the position of a random point in $\boldsymbol{p}$. We make this operator efficient by constraining the point to remain in its *safety domain*, which corresponds to the domain in which a vertex can move without producing edge flips in the Delaunay triangulation (Figure 6). We draw the new position of a vertex p from a uniform distribution over the safety domain. As the safety domain for the reverse move is identical, the kernel ratio in the Green rate (Equation 3) is equal to 1. Note that this kernel does not modify the marks of the configuration.

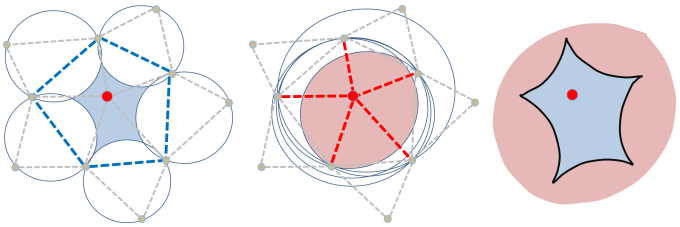


Fig. 6. Safety domain for point relocation. The blue region (left) corresponds to the region where the red vertex can move without entering the circumcirle of another triangle, *i.e.* without flipping the blue edges. The red region (middle) corresponds to the region where the red vertex can move without leaving the circumcirle of any three successive adjacent vertices, *i.e.* without flipping the red edges. The safety domain, drawn with a black border (right), is the intersection of the blue and red regions. In this example, the blue region lies entirely inside the red region, although this is not true in the general case.

Mark alteration changes the value of a mark in $\boldsymbol{m}$. In practice, we randomly select a point, an edge or a facet of the Delaunay triangulation depending on which type of element the marks are associated with. We then draw a new random value for the mark following a uniform distribution. The kernel ratio is thus equal to 1.

In practice, we give equal probability to the three kernels at each iteration of Algorithm 1, *i.e.* $q_m = \frac{1}{3}$. They play different roles during the sampling procedure. Birth and death is the core operator to simulate Delaunay triangulations with varying complexity, and to access any configuration of the solution space. Point relocation and mark alteration allow local adjustments that would take many iterations to obtain using solely birth and death. As illustrated in Figure 7, using only the birth and death kernel gives a fast energy decrease at the beginning of the optimization as

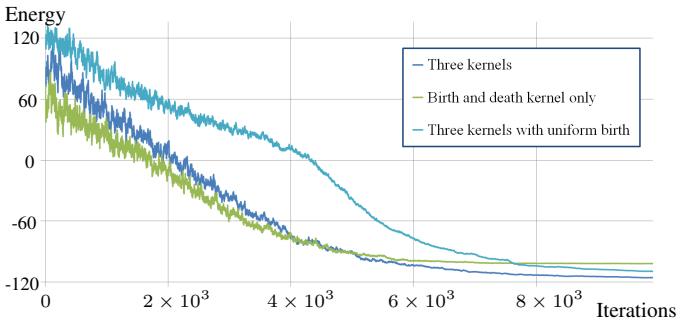


Fig. 7. Energy decrease with different combinations of kernels. A better energy is reached with our combination of three kernels than with just a birth and death kernel or with the three kernels with uniform birth.

the Delaunay vertices quickly align with the main image gradients. However, the energy reaches a high plateau later on, when the low temperature prevents the sampling to propose successive births and deaths that would be necessary to eventually displace a point to another position, or replace a mark by another one. Including the point relocation and mark alteration kernels allows the optimization to decrease the energy further. Figure 7 also shows that the optimization reaches high quality configurations faster when we guide the birth kernel with the image gradients instead of distributing new points uniformly.

4 Applications

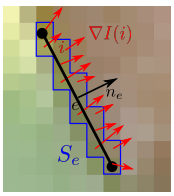
We now demonstrate the versatility of Delaunay point processes on three Vision tasks involving geometric structures: line-network extraction, object contouring, and mesh-based image compression. We provide for each application a brief discussion of related work.

4.1 Line-network extraction

Line networks form important structures in many application domains such as medical imaging (vessel networks), remote sensing (road networks), document analysis (line drawings). While many pixel-based algorithms have been proposed to detect such structures [23] or separate its components, *e.g.* strokes in line drawings [24], pixel chains often need to be vectorized for further analysis of the resulting planar graph. Several methods rely on a two-step procedure to extract planar graphs by first generating an overcomplete graph that is later simplified using optimization [25], [26], [27], [28], [29]. In contrast, our approach samples dynamic planar graphs over the image without resorting to a fixed, overcomplete intermediate representation.

Given the Delaunay triangulation of a point configuration $\boldsymbol{p}$, we model a line-network by associating each edge with a binary activation variable indicating if it belongs to the structure or not. Formally, we define the mark space as $\boldsymbol{m} = (m_e)_{e \in C_2(\boldsymbol{p})}$ with $m_e \in \{0, 1\}$ the activation mark of edge e. We denote $\widetilde{C_2}(\boldsymbol{p})$ the set of active edges in $C_2(\boldsymbol{p})$.

Energy. We design the data fidelity term to encourage active edges to align with strong image gradients. To do so, we define for each active edge $e \in \widetilde{C_2}$ an energy term that measures the strength of the image gradient and its



alignment with edge e. Summing over all active edges gives

$$U_{fidelity}(\boldsymbol{x}) = \sum_{e \in \widetilde{C_2}(\boldsymbol{p})} \frac{1}{|S_e|} \sum_{i \in S_e} \exp(-\mu |\nabla I(i) \cdot \boldsymbol{n}_e|) - \gamma \quad (9)$$

with S_e the pixels covered by e, $\nabla I(i)$ the image gradient at pixel i, and $\boldsymbol{n}_e$ the unit vector orthogonal to edge e (see inset).The parameter μ controls the sensibility to image noise, and γ is an offset to make the unary term negative for strong well-aligned gradients, which encourages their capture by the end structure. We set $\mu = 8$ and $\gamma = 0.5$ in our experiments.

We design the shape prior to penalize isolated edges, short edges, and sharp angles between adjacent edges. The two last criteria prevent the line network to zigzag over image contours. We achieve this behavior with two terms, $w_l(e)$ measuring the length of edges and $w_c(p)$ evaluating the connectivity of active edges at vertices

$$U_{prior}(\boldsymbol{x}) = \sum_{e \in C_2(\boldsymbol{p})} w_l(e) + \beta \sum_{p \in \boldsymbol{p}} w_c(p). \quad (10)$$

Parameter β balances these two terms, we fixed it to 1 in our experiments.

We define the edge length penalty $w_l(e)$ to be close to 1 when the edge is shorter than a threshold, and close to 0 otherwise. While we could use a Heaviside function to model this penalty, such as discontinuous energy would hinder the stochastic optimization. We use instead a smooth sigmoid function of the form $\mathcal{S}_{a,b}(|e|) = (1 + \exp(a(\frac{2|e|}{b} - 1)))^{-1}$ with $|e|$ the edge length and a, b two real values. In our context, b corresponds to the desired minimal length of edges, which we typically fix to 5% of the image diagonal in our experiments. We set the positive constant a to 5 to shape the sigmoid like a smoothed Heaviside function.

The connectivity penalty $w_c(p)$ should penalize both isolated active edges and pairs of active edges forming sharp angles. We achieve this behavior by setting $w_c(p) = 0$ if vertex p has no adjacent active edge, $w_c(p) = 1$ if vertex p has one (*i.e.* isolated) adjacent active edge, and $w_c(p) = \mathcal{S}_{a,b}(\delta(p))$ if vertex p has more than one adjacent active edge, where δ_p is the maximal value of dot products between any pair of active edges around p. We set $a = 5$ and $b = cos(\frac{\pi}{12})$ in our experiments.

Experiments. We applied our model to extract line networks from different application domains, such as vessels in organic images (retina in Figure 1, leaf in Figure 10), pen strokes in line drawings (Figure 8), regular edge patterns in man-made textures (tiles in Figure 10). Our model performs well on this diverse set of images thanks to its generic fidelity term, which only depends on image gradients (Equation 9). The insets in Figure 8 show that our model extracts clean line intersections from rough drawings. Recovering such topological information is a necessary step for many line drawing vectorization algorithms [27], [28].

A major strength of our approach over existing two-steps strategies is its ability to jointly recover the geometric configuration of the Delaunay triangulation and identify its active edges. Figure 9 illustrates the benefits of this joint procedure compared to a two-steps method that first estimates the position of the vertices, and then estimates the activation of the edges. We implemented the first step by sampling points according to an energy that encourages them to be on strong image gradients and to form

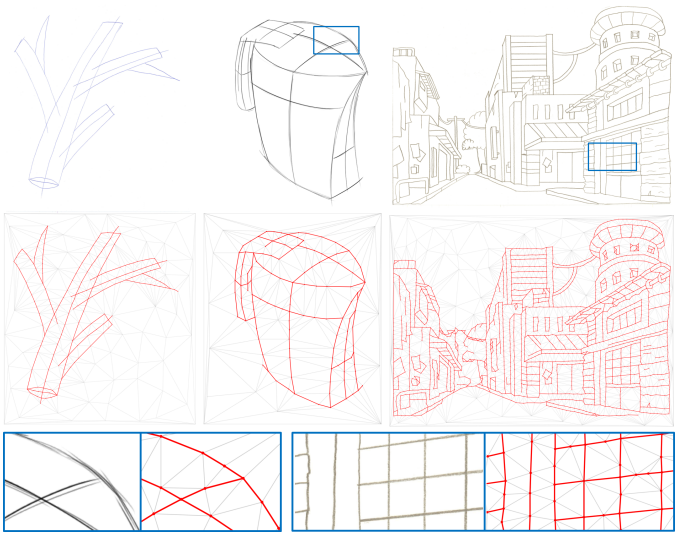


Fig. 8. Vectorization of line-drawings. Our Delaunay point process recovers the center-line of sketchy pen strokes in bitmap line drawings (active edges shown in red). The insets shows that our model produces well-connected structures even in the presence of multiple overlapping strokes and complex regular patterns.

long edges, *i.e.* $U_{fidelity}(\boldsymbol{x}) = \sum_{p \in \boldsymbol{p}} (1 - |\nabla I(p)| - \gamma)$ and $U_{prior}(\boldsymbol{x}) = \sum_{e \in C_2(\boldsymbol{p})} w_l(e)$. Note that we cannot encourage alignment of the edges to image gradients at this point, since we don't know yet which edges will form the end structure. The second step identifies these active edges by minimizing the complete energy (Equation 9 and 10) using only the mark alteration kernel of RJMCMC to keep the triangulation fixed. The comparison shows that the two-steps approach often misses line junctions in the final network because such junctions are not captured by the triangulation during the first step.

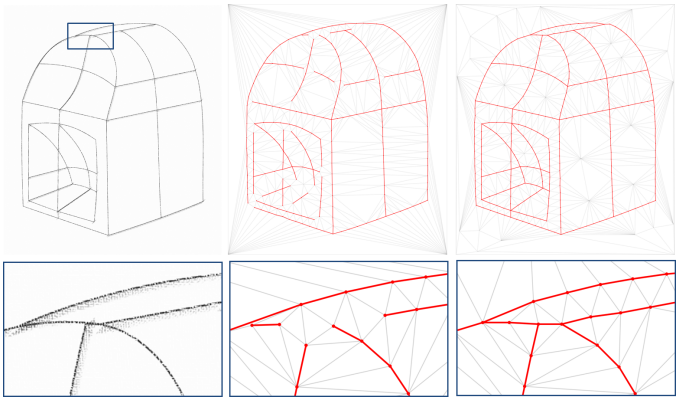


Fig. 9. Comparison with a two-steps optimization for line drawing vectorization. When sampling points independently of the marks (middle), edges of the Delaunay triangulation often miss important line junctions, which cannot be recovered by a subsequent marking step. Our Delaunay point process samples points and marks jointly, which favors the emergence of a well-connected line network (right).

Figure 10 and Table 1 provide qualitative and quantitative comparisons of our model to existing line-network extraction methods based on point processes. Using a marked point process with line segments [30] results in many isolated segments as the algorithm struggles to enforce connectivity. Junction-point processes [11] better model connectivity, but have difficulties

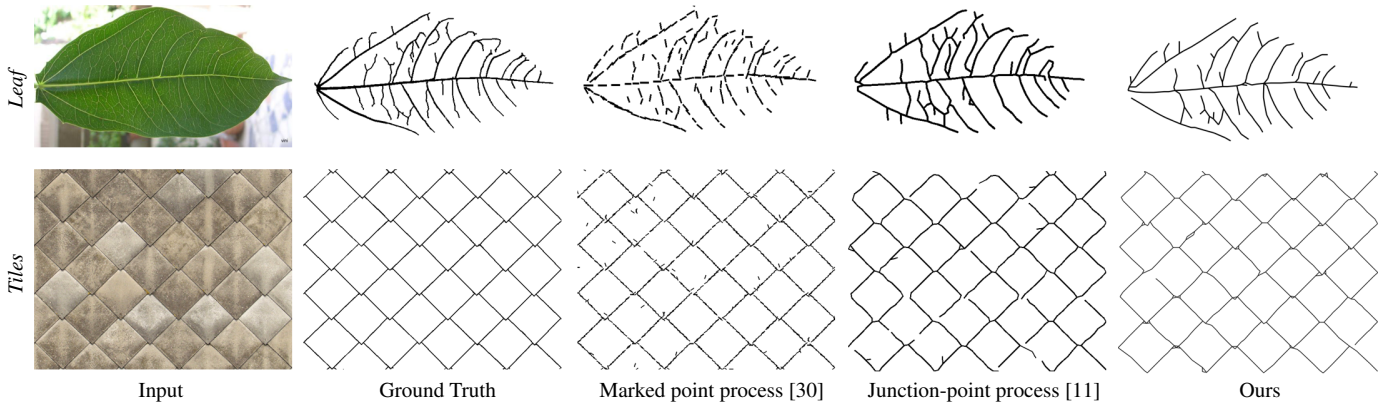


Fig. 10. Visual comparisons with existing point processes. Marked point process [30] produces configurations of mostly disconnected line-segments. Junction-point process [11] better preserves the connectivity of edges but recover badly complex junctions of at least four branches (see junctions in *tiles*). Our Delaunay point process exhibits better connectivity and accuracy for both cyclic (bottom) and acyclic (top) line-networks.

TABLE 1
Quantitative comparisons with existing point processes. Our Delaunay point process outperforms marked point process [30] and junction-point process [11] in terms of precision and F-measure while being faster.

		Precision	F-measure	Time
Leaf	Junction-point process [11]	0.59	0.64	73s
	Marked point process [30]	0.76	0.70	33s
	ours	**0.79**	**0.73**	20s
Tiles	Junction-point process [11]	0.46	0.54	227s
	Marked point process [30]	0.67	0.72	103s
	ours	**0.70**	**0.74**	70s

TABLE 2
Quantitative evaluation on BSDS500. Our method is competitive with different dedicated pipelines. In particular,our accuracy is higher than the one of the Weighted Optimal Transport (OT) pipeline [26] when output complexity becomes very low, *i.e.* when compression is lower than 3%.

		Compression				
		1%	2%	3%	4%	5%
Accuracy	Weighted OT	0.16	0.194	0.22	**0.236**	**0.241**
	Binary OT	0.135	0.161	0.179	0.184	0.187
	Edge Contraction	0.142	0.148	0.164	0.177	0.194
	ours	**0.185**	**0.208**	**0.223**	0.231	0.238

extracting high degree junctions such as the crossings of the tiles. Our method extracts well-connected networks and gives higher precision scores.

We evaluated our algorithm quantitatively on the first 100 images of the Berkeley segmentation dataset BSDS500 [31], which provides ground truth contour lines drawn by five humans on each image. While our basic formulation relies on the image gradient magnitude to locate lines (Eq. 9), we used a more accurate boundary probability map for this experiment [32]. We ran our algorithm on each image several times to produce a range of results with different output compactness. We measure accuracy as the mean Intersection-over-Union scores of our pixelized output against the five provided ground truths. We measure compression as the percentage of vertices of the output planar graph with respect to the average number of pixels of the ground truth boundaries. Tab. 2 shows how accuracy evolves in function of compression.

We compared our algorithm with three approaches capable of producing planar graphs likes ours. For a fair comparison, all approaches use the same boundary probability maps in the experiments, *i.e.* those produced with the method by Isola et al. [32]. The first approach (weighted OT), simplifies pixels chains into planar graphs using an optimal transport formulation where pixels are weighted by their boundary probability [26]. The second approach (binary OT) first applies adaptive thresholding [33] on the boundary probability map to only operate the simplification by optimal transport [26] on pixels with high boundary probability. The last approach (Edge Contraction) binarizes the boundary probability map by adaptive thresholding [33], chains pixels with a high boundary probability using a Delaunay Triangulation, and operates edge contraction with a cost function set as the edge length [6]. Tab. 2 shows our method is competitive with these three approaches. In particular, by discarding weak discontinuities, binary OT and Edge contraction baselines reduce pixel chaining ambiguities on strong discontinuities but produce results with lower accuracies than our method and the weighted OT baseline. The accuracy of the latter drops at high compression rates where vertices start connecting across uniform image regions. For such cases, weighted OT tends to draw edges across uniform regions, while our simplified line-networks remain consistent with the image content since we explicitly account for both the connectivity of the network and its distance to the input image data. Fig. 11 provides a visual comparison of this behavior at low compression rates.

4.2 Object contouring

Object contouring by polygonal shapes provides a compact and structure-aware representation of the object silhouette. Polygonal contours are particularly well suited to represent man-made objects like buildings, cars or furniture that are dominated by straight segments. Existing object polygonalization methods typically start by detecting line-segments, which are then assembled into polygons. This second step can be done by searching for cycles in a graph of line-segments [2], or by connecting line-segments using gap filling [1]. Another strategy for object contouring consists in over-segmenting the image before extracting objects as groups of superpixels [34]. However, obtaining polygonal objects with this strategy either requires a preprocessing step to convert pixels or superpixels into small polygons [35] [36], or a post-processing

Fig. 11. Visual comparisons on line-network extraction at low compression rates. Our method produces a compact line network that captures the dominant boundaries of the image (top). In contrast, the dedicated pipeline based on optimal transport [26] often connects vertices across uniform regions (bottom).

step to vectorize chains of pixels into polygons [37], which often introduces inaccuracy. Recent work considered the use of recurrent neural networks to sequentially predict the vertices of a polygonal object contour [38], but such a black-box algorithm offers little control on the complexity of the outcome. Closer to our work are Polygonal Markov Fields [39], which are stochastic models designed to sample polygons in images. Based on local operators that add or remove vertices to a polygon, these models struggles to explore topological variations and remain very slow to converge on natural images. Our model also shares ideas with the work of Ren et al. [40] who builds on a constrained Delaunay triangulation to fill gaps in object contours. However, they formulate contour completion as an edge labeling task on a fixed triangulation, while our method lets the triangulation evolve dynamically to best capture the object contour.

To achieve polygonal object contouring within our framework, we associate each facet of the Delaunay triangulation with a binary activation variable indicating if it belongs to the object or not. 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. Formally, we define the mark space as $\boldsymbol{m} = (m_f)_{f \in C_3(\boldsymbol{p})}$ with $m_f = \{0, 1\}$ the activation mark of triangle f. We guide the object segmentation with a pixelwise probability map H. The computation of this probability map depends on the application scenario, as detailed in our experiments.

Energy. We express our data fidelity energy as the sum of two terms, one measuring the agreement between the binary mark of each facet and the underlying probability map, the other one encouraging homogeneous colors within each facet to preserve image contours

$$U_{fidelity}(\boldsymbol{x}) = \frac{1}{|I|} \sum_{f \in C_3(\boldsymbol{p})} \sum_{i \in f} (1 - H(i|m_f)) + \beta_1 \, |I_{|f}| \sigma^2_{I_{|f}} \tag{11}$$

where $|I|$ is the number of pixels of image I, $|I_{|f}|$ is the number of pixels inside facet f, $\sigma^2_{I_{|f}} \in [0, 1]$ is the normalized variance of pixel colors inside facet f, and $H(i|m_f)$ is the probability of assigning mark m_f to pixel i. The parameter β_1 balances the two terms, we fixed it to 1 in our experiments.

Our shape prior for object contouring uses the same term as for line network extraction to penalize short edges. In addition, we define a smoothness term based on Potts model to favor compact polygons. Summing the two terms gives

$$U_{prior}(\boldsymbol{x}) = \sum_{e \in C_2(\boldsymbol{p})} w_l(e) + \beta_2 \, w_s(e) \tag{12}$$

where the edge length penalty $w_l(e)$ is defined as in equation 10, and $w_s(e) = |e|$ if the two facets adjacent to edge e have different marks, and $w_s(e) = 0$ otherwise. The parameter β_2 balances the two terms, we fixed it to $\frac{1}{|I|}$ in our experiments.

Experiments. We tested our contouring model on the Berkeley segmentation dataset [31] as well as on images with regular man-made structures, such as facades and urban aerial photographs. 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 $H(i|m_f)$ of a pixel i to belong to class m_f as its normalized RGB distance to the closest color in the set of scribbled pixels belonging to that class

$$H(i|m_f) = \frac{\min\limits_{j \in S_{m_f}} \|I(i) - \widehat{I}(j)\|_2^2}{\min\limits_{j \in S_0} \|I(i) - \widehat{I}(j)\|_2^2 + \min\limits_{j \in S_1} \|I(i) - \widehat{I}(j)\|_2^2} \tag{13}$$

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, but this is beyond the scope of this paper.

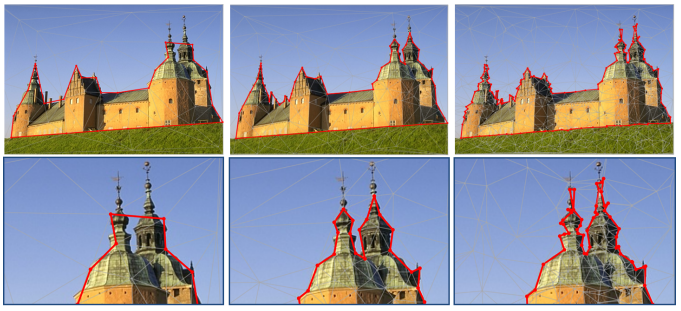


Fig. 12. Trade-off between fidelity and simplicity. Parameter α offers control over the complexity of the output polygon. A low α value gives more weight to the fidelity term of the energy, resulting in more complex polygons that tightly fit to the object silhouettes.

Figure 12 illustrates the trade-off between fidelity and simplicity for different values of parameter α, keeping the Poisson parameter fixed. Although we cannot control the exact number of edges in the output polygons, tuning α has a direct impact on polygon complexity. Figure 13 shows the results of this model on a variety of images with organic and man-made shapes. Our method extracts low-complexity polygons that accurately capture the object silhouettes, despite the simplicity of our color model H. Our method performs best on man-made objects composed

Fig. 13. Object contouring on a few example images. Our Delaunay point process samples polygons that capture the silhouettes of foreground objects. The user first draws a few scribbles that roughly characterize the objects of interest (blue lines) and the image background (red lines). Output polygons preserve details, such as the flower petals and the vase handles, while having low complexity.

of piecewise-linear contours, such as the facade elements in the street-level picture and the roofs in the aerial picture.

Figure 14 provides a visual comparison to several two-steps strategies, for an increasing number of user scribbles. We first compare to a pixel-based segmentation algorithm (GrabCut [41]), which requires many scribbles to capture fine details accurately. In contrast, our approach achieves better segmentations with fewer scribbles by working at the scale of Delaunay triangles. Converting the GrabCut pixel segmentation to a polygon as a post-process [37] reduces accuracy further. We also compare to running GraphCut segmentation [42] on a graph of polygonal superpixels [35], using $H(i|m_f)$ for the unary term and a Potts model for the pairwise term. However, we only obtained satisfactory results when using small superpixels, which result in very complex output polygons. In contrast, our method achieves both high accuracy and low polygon complexity, even when very few scribbles are provided.

We tested our algorithm on the first 100 images of the HKU-IS dataset proposed by Li and Yu [43] for evaluating salient object detection methods. To avoid providing foreground and background scribbles for each image, we replaced the color model used in our basic formulation (Eq. 13) by the saliency map returned by the automatic algorithm of Li and Yu [43]. We measure accuracy as the Intersection-over-Union score of our pixelized output foreground region against ground truth. We measure compression as the percentage of vertices of the output polygons with respect to the number of pixels of the ground truth region boundary. Tab. 3 shows how accuracy evolves in function of compression.

TABLE 3
Quantitative evaluation on HKU-IS dataset. Our method outclasses two-step pipelines in which object extraction and geometry simplification are performed independently.

		Compression				
		5%	6%	7%	8%	9%
Accuracy	Contour vectorization	0.73	0.77	0.81	0.83	0.84
	Voronoi	0.71	0.73	0.75	0.76	0.77
	Voronoi with regularization	0.7	0.71	0.72	0.73	0.74
	ours	**0.76**	**0.8**	**0.84**	**0.86**	**0.87**

We compared our method with several two-step pipelines that produce polygonal contours. These pipelines exploit the same saliency maps in the experiments, *i.e.* those produced by the method of Li and Yu [43]. The first pipeline (contour vectorization) consists in i) extracting pixel-based regions by thresholding the saliency map and ii) simplifying the chains of pixels of the region contours into polygons using the Douglas-Pecker algorithm

Fig. 14. Visual comparisons with two-steps object contouring methods given different sets of input scribbles. The GrabCut pixel-based segmentation [41] requires many input scribbles to correctly capture horse silhouettes (see the bottom parts of the legs). Converting the output pixel chains to polygons using Douglas-Peucker algorithm [37] accentuates their defects, whereas pre-segmenting the input image into polygonal superpixels [35] only give satisfactory results when small superpixels are used, *i.e.* for high output complexity. In contrast, our Delaunay point process produces low complexity polygons that accurately capture the horses, even when only two scribbles are provided.

[37]. The second pipeline (Voronoi) exploits the inverse strategy: the image is first simplified into a Voronoi diagram using a polygonal decomposition method [35] before extracting polygon-based regions by thresholding the saliency map averaged over each cell of the diagram. The last pipeline (Voronoi with regularization) is a variant in which the thresholding operation has been replaced by a graph-cut approach that explicitly favors simple output polygons. The three pipelines perform object extraction and geometry simplification independently. Voronoi with regularization improves the visual quality of output polygons with respect to thresholded Voronoi, but produces slightly less accurate polygons. The contour vectorizaton pipeline produces more accurate results than Voronoi when the saliency map is of high quality. The accuracy values obtained at different compression rates are higher for our method because it is able to merge regions to achieve the best accuracy/compactness trade-off.

4.3 Image compression

Our third application consists in representing an image as a colored triangular mesh, as illustrated in Figure 15. While this geometric representation is not as flexible as wavelet-based compression schemes [44], we show that it achieves competitive compression rates on images dominated by smooth color variations (Figure 17). Our approach is inspired by prior work on Delaunay-based image compression [45], [46], [47], image vectorization [48], and image stylization [49]. However, existing methods employ heuristic or greedy strategies to define the location of the Delaunay vertices. In contrast, Delaunay point processes allow us to jointly optimize the position and color of the vertices to best balance image reproduction with image compression.

We represent a color image as a Delaunay triangulation, where each vertex is associated to a color and each triangle interpolates the colors of its vertices bilinearly. Formally, we define the mark space as $\boldsymbol{m} = (c_p)_{p\in\boldsymbol{p}}$ where c_p is a RGB color. Since we assume that the Delaunay triangulation is uniquely defined by its vertices, we can store the image compactly as a list of colored points.

Fig. 15. Image compression by Delaunay point process. A Delaunay triangulation with only 2.8K colored points (right) is sufficient to approximate a 262Kpixels image (left) with a structural similarity (SSIM) greater than 0.97. Each triangle is colored by bilinear interpolation of its three vertices. Here we display the Delaunay edges in grey for visualization.

Energy. We design the point process energy to offer a trade-off between fidelity to the input image and simplicity of the output mesh. The fidelity term measures the per-pixel error between input and output,

$$U_{fidelity}(\boldsymbol{x}) = \frac{1}{|S|}\sum_{i\in S} ||I(i) - I_{\boldsymbol{x}}(i)||_2^2, \qquad (14)$$

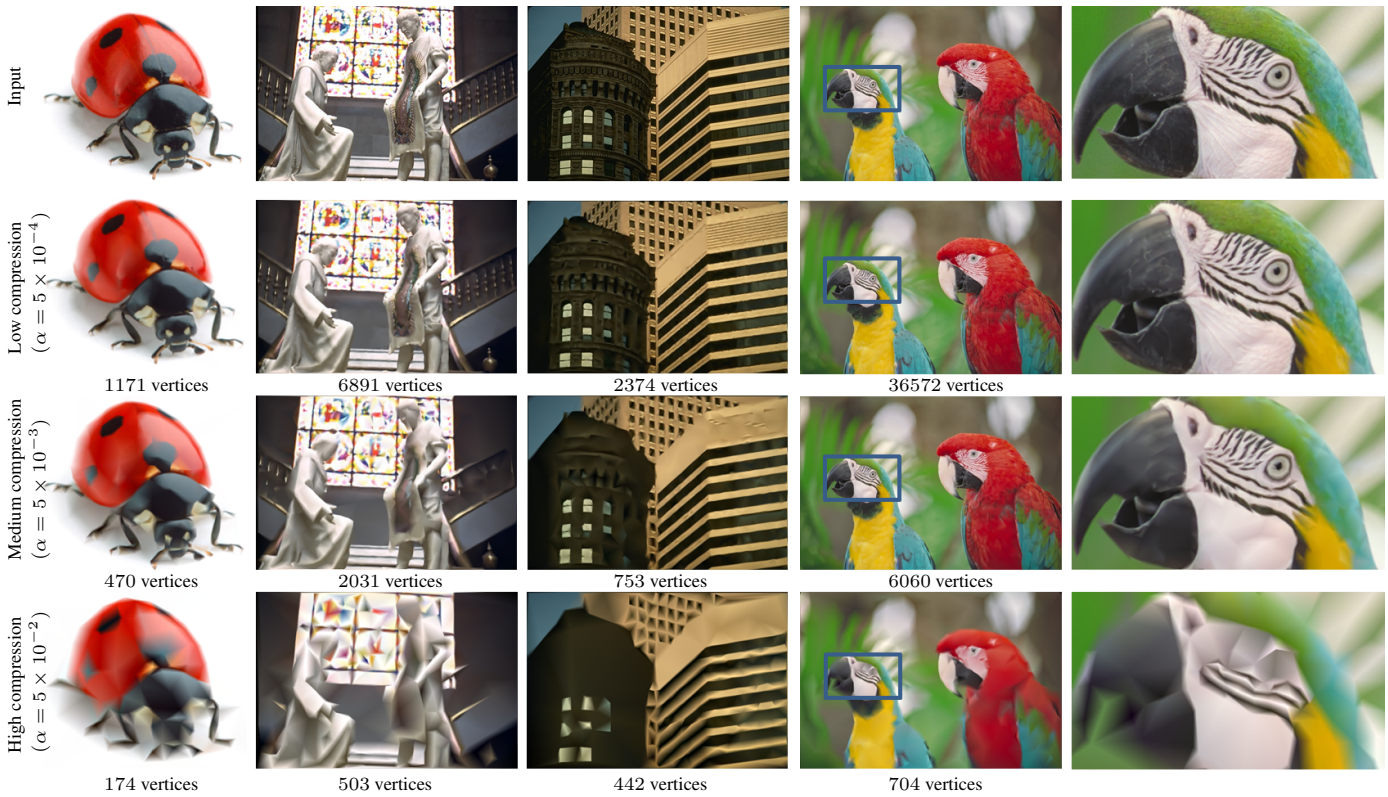


Fig. 16. Parameter α offers a trade-off between visual quality and compression rate. A high α value preserves high frequency details and sharp discontinuities, but gives a low compression rate. Highly compressed results (bottom) expose the underlying triangulation, especially on fine details and regular patterns such as the building facades.

where S is the set of pixels of input image I and $I_{\boldsymbol{x}}$ is the image reconstructed from configuration $\boldsymbol{x}$ using bilinear color interpolation over each triangle. $U_{fidelity}$ can be seen as a sum of unary data terms on each facet. The shape prior penalizes the number of points in the configuration,

$$U_{prior}(\boldsymbol{x}) = \frac{|\boldsymbol{p}|}{\lambda}, \tag{15}$$

where $|\boldsymbol{p}|$ is the number of points in configuration $\boldsymbol{x}$ and λ is the Poisson parameter of the point process.

Experiments. We implemented the algorithm with the *birth and death* and *point relocation* kernels described in Section 3.3. However, we found that we can avoid using the *mark alteration* kernel by computing the optimal color of a mark every time the corresponding point is created or relocated. We compute this color by minimizing Equation 14 over the pixel domain covered by the facets adjacent to the point.

Figure 16 illustrates the effect of parameter α, which weights $U_{fidelity}$ and U_{prior} according to Equation 8. A low α preserves well the input image but generates complex configurations, with typically more points than the Poisson parameter. Increasing α yields simpler configurations where fine details are removed.

We compare the performance of our model to state-of-the-art image compression algorithms in Figure 17. We perform this comparison on images with varying levels of realism and noise (a clipart, a studio photograph, and a real-world photograph). Since our model represents an image as a piecewise-linear function, it performs best on cliparts that are often composed of linear color gradients. Our approach is also competitive on studio photographs that contain large, uniform highlights and soft shadows with little image noise. However, our approach tends to smooth-out the high-frequency grain of real-world photographs, achieving a low SSIM score on such images. Note that, similarly to other vector graphics representations, our colored meshes can be rasterized at any resolution. This model does not outperform the best image compression algorithms on real-world images, but it illustrates the diversity of applications for which Delaunay Point Processes can be used.

5 Discussion

We have introduced Delaunay Point Processes for the extraction of 2D geometric structures composed of line segments or polygons. By building on point processes, our approach simultaneously detects geometric primitives and group them into structures, which is more robust than performing these two tasks in sequence. By building on the Delaunay triangulation, our approach produces well-connected structures and allows more efficient stochastic optimization than point processes based on an Euclidean distance neighborhood. Our three applications demonstrate the flexibility of this framework. We now detail the performance of our method and give some design guidelines for practitioners who would like to apply it to other structure extraction tasks.

Performance. Because the RJMCMC sampler is memoryless, Delaunay point processes are very memory efficient with a constant memory allocation during sampling. Running times range from a few seconds, *e.g.* for the extraction of horse silhouettes on Figure 14, to a few minutes, *e.g.* for the

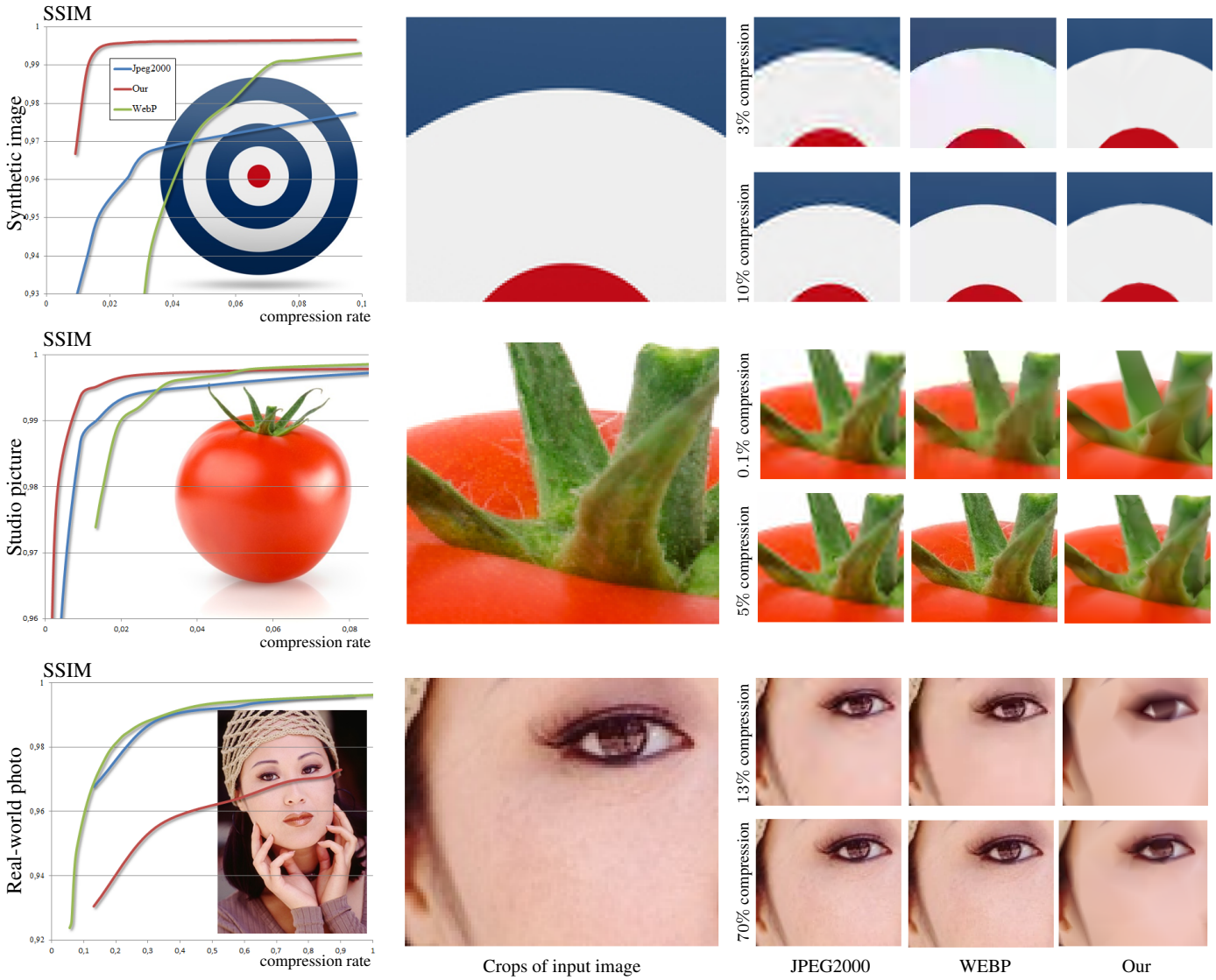


Fig. 17. Comparisons with state-of-the-art image compression algorithms. Our method competes well with image compression standards on synthetic images (top) and, to a lesser extent, on studio photographs (middle). Our use of bilinear color interpolation inside triangles is well suited to the smooth color variations of synthetic images. However, this interpolation tends to remove the fine grain of real world photographs (bottom). As a result, our approach achieves low SSIM scores on the portrait, even though its tendency to smooth out noise may be appreciated in some applications.

compression of the parrots on Figure 16. Timings depend on the input image size and, to a lesser extent, on the complexity of the energy formulation. In particular, the line-network extraction model requires more iterations to converge than our two other models. The design of kernels tailor-made for manipulating Delaunay triangulations allows us to reach attractive timings compared to traditional point processes, especially with the use of an efficient computational geometry library [6]. Note that recent work proposed faster optimization strategies for point processes, such as parallel Monte Carlo samplers [30] and binary labeling of object proposals [50], although these approaches make restrictive assumptions on the energy. Sampling Delaunay point processes in parallel would require an efficient GPU implementation of Delaunay data-structures, which is currently not available in standard geometry libraries.

Design guidelines. We have identified several guidelines to follow to develop efficient models for Delaunay Point Processes.

- *Small mark space.* The mark space should only contain a few discrete values to allow efficient exploration by random mark alterations. While our model for image compression relied on the much larger space of 24bits colors, we resorted to a closed-form optimization to estimate the mark value instead of random sampling.
- *Simple energy.* Simple energy formulations improve convergence stability. In particular, we recommend using (i) no more than three energy terms to avoid unstable parameter tuning, and (ii) as-continuous-as-possible energy functions so that the Monte Carlo sampler better guides the current configuration into interesting energy valleys.
- *Application-specific kernels.* Although the birth and death kernel is theoretically sufficient to explore the entire configuration space, application-specific kernels often greatly speed-up the optimization. In particular, data-driven ker-

nels can concentrate perturbations in the interesting areas of the input data, *e.g.* on the high image gradients when extracting line-networks.

- *Reasonably-sized images.* Because the Monte Carlo sampler operates sequentially, timings are strongly impacted by the input image resolution. We obtained competitive performances by running Delaunay point processes on images with a few million pixels. The parallelization of Delaunay point processes constitutes an important research challenge to scale to high-resolution images.

Perspectives. Besides the optimization challenges, an interesting direction for future research would be to develop efficient strategies for marking Delaunay point processes with parametric functions. This would allow the extraction of more complex geometric structures, such as networks of Bezier curves for line drawing vectorization, or non-linear color gradients for image compression. We also would like to investigate the extension of Delaunay point processes to 3D, opening the door to many vision problems that involve the extraction of surfaces and volumes.

ACKNOWLEDGMENTS

This work was supported in part by the ERC starting grant D^3 (ERC-2016-STG 714221) and by research and software donations from Adobe. Special thanks to Dengfeng Chai and Liuyun Duan for the fruitful discussions.

REFERENCES

[1] Z. Zhang, S. Fidler, J. Waggoner, Y. Cao, S. Dickinson, J. Siskind, and S. Wang, "Superedge grouping for object localization by combining appearance and shape informations," in *CVPR*, 2012.

[2] X. Sun, M. Christoudias, and P. Fua, "Free-shape polygonal object localization," in *ECCV*, 2014.

[3] P. Labatut, J.-P. Pons, and R. Keriven, "Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cut," in *ICCV*, 2007.

[4] C. Mostegel, R. Prettenthaler, F. Fraundorfer, and H. Bischof, "Scalable surface reconstruction from point clouds with extreme scale and density diversity," in *CVPR*, 2017.

[5] M. N. M. V. Lieshout, "Stochastic annealing for nearest-neighbour point processes with application to object recognition," *Advances in Applied Probability*, vol. 26, no. 2, 1994.

[6] The CGAL Project, *CGAL User and Reference Manual*, 4.11 ed. CGAL Editorial Board, 2017. [Online]. Available: http://doc.cgal.org/4.11/Manual/packages.html

[7] X. Descombes, *Stochastic geometry for image analysis*. Wiley-ISTE, 2011.

[8] D. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*. Springer, 1988.

[9] A. J. Baddeley and M. V. Lieshout, "Stochastic geometry models in high-level vision," *Journal of Applied Statistics*, vol. 20, no. 5-6, 1993.

[10] M. Ortner, X. Descombes, and J. Zerubia, "Building outline extraction from digital elevation models using marked point processes," *IJCV*, vol. 72, no. 2, 2007.

[11] D. Chai, W. Forstner, and F. Lafarge, "Recovering line-networks in images by junction-point processes," in *CVPR*, 2013.

[12] P. J. Green, "Reversible jump markov chain monte carlo computation and bayesian model determination," *Biometrika*, vol. 82, no. 4, 1995.

[13] P. H. Peskun, "Optimum monte carlo sampling using markov chains," *Biometrika*, vol. 60, 1973.

[14] P. Salamon, P. Sibani, and R. Frost, *Facts, Conjectures, and Improvements for Simulated Annealing*. SIAM Monographs on Mathematical Modeling and Computation, 2002.

[15] W. Ge and R. Collins, "Marked point processes for crowd counting," in *CVPR*, 2009.

[16] A. Utasi and C. Benedek, "A 3-D marked point process model for multi-view people detection," in *CVPR*, 2011.

[17] F. Lafarge, G. Gimel'farb, and X. Descombes, "Geometric feature extraction by a multi-marked point process," *PAMI*, vol. 32, no. 9, 2010.

[18] C. Lacoste, X. Descombes, and J. Zerubia, "Point processes for unsupervised line network extraction in remote sensing," *PAMI*, vol. 27, no. 10, 2005.

[19] K. Sun, N. Sang, and T. Zhang, "Marked point process for vascular tree extraction on angiogram," in *EMMCVPR*, 2007.

[20] S. Drot, X. Descombes, H. Le Men, and J. Zerubia, "Object point processes for image segmentation," in *ICPR*, 2002.

[21] A. Baddeley and J. Moller, "Nearest-neighbour markov point processes and random sets," *International Statistical Review*, vol. 57, no. 2, 1989.

[22] E. Bertin, J.-M. Billiot, and R. Drouilhet, "Spatial delaunay gibbs point processes," *Communications in Statistics. Stochastic Models*, vol. 15, no. 2, 1999.

[23] J. Wegner, J. Montoya-Zegarra, and K. Schindler, "A higher-order crf model for road network extraction," in *CVPR*, 2013.

[24] B. Kim, O. Wang, A. C. Öztireli, and M. Gross, "Semantic segmentation for line drawing vectorization using neural networks," *Computer Graphics Forum*, vol. 37, no. 2, 2018.

[25] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua, "Reconstructing loopy curvilinear structures using integer programming," in *CVPR*, 2013.

[26] F. de Goes, D. Cohen-Steiner, P. Alliez, and M. Desbrun, "An optimal transport approach to robust reconstruction and simplification of 2d shapes," *Computer Graphics Forum*, vol. 30, no. 5, 2011.

[27] G. Noris, A. Hornung, R. W. Sumner, M. Simmons, and M. Gross, "Topology-driven vectorization of clean line drawings," *ACM Trans. on Graphics*, vol. 32, no. 1, 2013.

[28] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity vs. Simplicity: a Global Approach to Line Drawing Vectorization," *ACM Trans. on Graphics (Proc. of Siggraph)*, vol. 35, no. 4, 2016.

[29] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun, "Enhancing road maps by parsing aerial images around the world," in *ICCV*, 2015.

[30] Y. Verdie and F. Lafarge, "Detecting parametric objects in large scenes by Monte Carlo sampling," *IJCV*, vol. 106, no. 1, 2014.

[31] 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.

[32] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *ECCV*, 2014.

[33] P. sung Liao, T. sheng Chen, and P. choo Chung, "A fast algorithm for multilevel thresholding," *Journal of Information Science and Engineering*, vol. 17, 2001.

[34] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Optimal contour closure by superpixel grouping," in *ECCV*, 2010.

[35] L. Duan and F. Lafarge, "Image partitioning into convex polygons," in *CVPR*, 2015.

[36] R. Achanta and S. Susstrunk, "Superpixels and polygons using simple non-iterative clustering," in *CVPR*, 2017.

[37] S. T. Wu and M. R. G. Marquez, "A non-self-intersection Douglas-Peucker algorithm," in *IEEE Symposium on Computer Graphics and Image Processing*, 2003.

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

[39] R. Kluszczyński, M. N. M. van Lieshout, and T. Schreiber, "Image segmentation by polygonal markov fields," *Annals of the Institute of Statistical Mathematics*, vol. 59, no. 3, 2007.

[40] X. Ren, C. Fowlkes, and J. Malik, "Scale-invariant contour completion using conditional random fields," in *ICCV*, 2005.

[41] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut -interactive foreground extraction using iterated graph cuts," *ACM Trans. on Graphics (Proc. of Siggraph)*, vol. 23, no. 3, 2004.

[42] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *PAMI*, vol. 26, no. 9, 2004.

[43] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *CVPR*, 2016.

[44] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. Academic Press, 2008.

[45] L. Demaret and A. Iske, "Adaptive image approximation by linear splines over locally optimal delaunay triangulations," *Signal Processing Letters*, vol. 13, no. 5, 2006.

[46] L. Demaret, N. Dyn, and A. Iske, "Image compression by linear splines over adaptive triangulations," *Signal Processing*, vol. 86, no. 7, 2006.

[47] S. Bougleux, G. Peyre, and L. Cohen, "Image compression with anisotropic geodesic triangulations," in *ICCV*, 2009.

[48] Z. Liao, H. Hoppe, D. Forsyth, and Y. Yu, "A subdivision-based representation for vector image editing," *Trans. on Visualization and Computer Graphics*, vol. 18, no. 11, 2012.

[49] M. Gai and G. Wang, “Artistic low poly rendering for images,” *The Visual Computer*, vol. 32, no. 4, 2016.
[50] T. Pham, S. Rezatofighi, I. Reid, and T.-J. Chin, “Efficient point process inference for large-scale object detection,” in *CVPR*, 2016.