SIAM J. SCI. COMPUT.
Vol. 36, No. 2, pp. A792–A827

AN ITERATIVE ALGORITHM FOR COMPUTING MEASURES OF GENERALIZED VORONOI REGIONS[*]

LISA J. LARSSON[†], RUSTUM CHOKSI[†], AND JEAN-CHRISTOPHE NAVE[†]

Abstract. We present and analyze a fast algorithm for directly computing measures of generalized Voronoi regions associated with generators of arbitrary codimension. The algorithm consists of solving one eikonal equation to construct a kernel-based operator whose iteration accumulates mass along the closest generator. In particular, the algorithm does not require the computation of the Voronoi diagram or the gradient of the solution to the eikonal equation. The algorithm is shown to be first order and converge very quickly. By discretizing the distance to the generators on the grid (and not the generators themselves), very accurate geometric information is used even for coarse grids. Several examples are presented, including the computation of the population influence associated with the Los Angeles County highway system. The method is also one of the key ingredients for the fast computation of centroidal Voronoi tessellations (CVTs) of general rigid objects (e.g., rigid curves and surfaces) in higher dimensions. We present a few simple examples of these generalized CVTs.

Key words. iterative algorithm, measures of generalized Voronoi regions, eikonal equation, Markov operator

AMS subject classifications. 65D18, 68U05

DOI. 10.1137/130935598

1. Introduction.

In this article, we present and analyze an algorithm for directly computing measures of generalized Voronoi regions. Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and $\Gamma_1, \ldots, \Gamma_n \subset \Omega$ a fixed collection of subsets of Ω with codimension in $\{1, \ldots, d\}$. We call the sets $\Gamma_1, \ldots, \Gamma_n$ generators. Each generator has a corresponding *generalized Voronoi region*, which is simply the set of points in the domain that are closer to that generator than to any other generator. We assign the points that are equidistant to two or more generators via a tie-breaking rule. More precisely, we define the *generalized Voronoi region*, V_i, as follows:

$$V_i := \left\{ x \subset \Omega \,\middle|\, \operatorname{dist}(x, \Gamma_i) \le \operatorname{dist}(x, \Gamma_j) \text{ for } j \in \{1, \ldots, n\} \setminus \{i\},\, i < j \right\},$$

where

$$\operatorname{dist}(x, \Gamma_i) := \inf_{y \in \Gamma_i} |x - y|.$$

The *generalized Voronoi diagram* associated with the generators is the set of points which are equidistant to two or more generators, i.e.,

$$\cup_{i \neq j} \left(\partial V_i \cap \partial V_j \right).$$

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section September 4, 2013; accepted for publication (in revised form) January 29, 2014; published electronically April 15, 2014.
http://www.siam.org/journals/sisc/36-2/93559.html

[†]Department of Mathematics and Statistics, McGill University, Montreal, QC, Canada (lisa.powers@mail.mcgill.ca, rchoksi@math.mcgill.ca, jcnave@math.mcgill.ca). The work of the first author was supported by a Schulich Graduate Fellowship at McGill University. The work of the second author was supported by NSERC Canada under their Discovery Grant Program. The work of the third author was supported by NSERC Canada under their Discovery Grant Program and Discovery Accelerator Program.

In the special case where each generator is a singleton, the regions correspond to the well-studied *Voronoi regions* which comprise a *Voronoi diagram* or *Voronoi tessellation.*

Voronoi diagrams [26, 36, 15] have been the subject of intense study in computational geometry. For a collection of points, the Voronoi diagram consists of straight line segments. Efficient algorithms for generating the Voronoi diagram are readily available (see [4] for a survey). Thus measures associated with each polygonal region can be computed efficiently.

For the planar case where generators are not points (instead, they are open or closed curves), much work has been done to produce efficient algorithms to compute the generalized Voronoi diagram. Algorithms exist for various types of planar input curves (linear and circular arcs [50], rational curves [13], circles or additively weighted points [24, 25, 14], and NURBS curves [42]; see also [1, 2, 8, 7]). Additionally, there has been work to speed up computation of the generalized Voronoi diagram using the GPU, first by [18]. This algorithm becomes computationally intensive for general curves; more recently, a jump flooding algorithm on the GPU has been proposed to compute the generalized Voronoi diagram of planar curves efficiently [51]. In this algorithm, the curves are discretized on the computational grid, and the generalized Voronoi diagram can be recovered accurately for a large grid. Due to memory constraints, a three-dimensional analogue of this algorithm on the GPU is not yet feasible.

In fact, for generators which are surfaces in three dimensions, efficient algorithms for computing the generalized Voronoi diagram are far less developed. Algorithms exist only for simple geometrical ansatzes. For three-dimensional spheres, the generalized Voronoi diagrams have been calculated in [22, 3, 16]. These generalized Voronoi diagrams in $\mathbb{R}^3$ have applications in the analysis of proteins and other molecular structures [23, 39]. The generalized Voronoi diagrams in $\mathbb{R}^3$ of planes and cylinders have also been considered in [16]. The generalized Voronoi diagram is far more complicated, even for curved generators in the plane (cf. Figure 1).

In the following sections, we propose a method to compute integrals over generalized Voronoi regions. As a motivation, we outline an algorithm for explicitly calculating the Voronoi diagram for any generators in any dimension and describe in detail the computational complexities involved. Due to the preventative complexity of this calculation, the integration method we propose does not require the generalized Voronoi regions to be explicitly computed. This allows our method to work efficiently for *general* curves and surfaces.

DEFINITION 1.1. *Given a function $\mu \in L^1(\Omega)$, we define the* measure w_i *associated with the ith generator as follows:*

$$w_i := \int_{V_i} \mu(x)\, dx. \tag{1.1}$$

Our goal is as follows:

> *Given* Ω, generators Γ_i, and an integrable function μ, *compute* w_i without finding the generalized Voronoi regions V_i explicitly.

This paper describes a numerical integration algorithm to achieve this goal. Given the ubiquitous nature of the problem, there are many potential applications where the fast computation of such measures is important. For example, one particular application comes from geographic and urban planning wherein μ denotes a population density: Given a collection of hospitals (points) or highways/subway lines (curves), these measures give the fraction of the population which lives closest to the respective hospital or highway/subway line (cf. section 5.2).

While the method proposed here should be considered as a stand-alone method, a very important and natural application is the *centroidal Voronoi tessellation (CVT) of rigid bodies*. See [11, 12] for a survey of the literature on CVT for point generators. Indeed, let us use the CVT of rigid bodies to motivate why this goal is so important. Calculating a CVT for general rigid bodies in two and three dimensions has largely remained open (see [33] for a method to compute CVTs for nonrigid line segments and graphs, and [17] for an extension of Lloyd's method [32] to shapes in $\mathbb{R}^2$).

What exactly is involved in finding a CVT of a collection of rigid bodies? To this end, suppose we are given curves or surfaces $\Gamma_i \subset \Omega$ which, by rotation and/or translation, we wish to *optimally distribute* throughout Ω in the sense that they are "best centered" within their Voronoi region. If the generators are points, then we wish to find a placement of the points which has the property that the points are also the centroids of their associated Voronoi regions. For general generators, we seek to find a configuration of the generators which minimizes the CVT *energy* (cf. [11]). To define this energy, let us assume the location of the ith generator Γ_i is entirely determined by a location vector and an angle vector. For example, for line segments, the location vector is the midpoint, whereas for spheres, it is the center. Let $\mathbf{X}$ denote the vector of location coordinates and $\boldsymbol{\alpha}$ the vector of angles. To obtain a CVT of the rigid shapes, $\{\Gamma_i\}_{i=1}^n$, one must find the locations $\mathbf{X}$ and angles $\boldsymbol{\alpha}$ that minimize the following energy:

$$(1.2) \qquad \text{CVT energy:} \qquad F(\mathbf{X}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \int_{V_i(\mathbf{X}, \boldsymbol{\alpha})} \operatorname{dist}^2(y, \Gamma_i)\, dy.$$

A key observation is that in order to minimize the CVT energy, for example, via gradient descent or quasi-Newton iteration, one does not need to *find* the Voronoi regions V_i, but rather one needs to *evaluate* integrals of certain functions over the V_i, i.e., evaluate suitable measures of V_i.

In this paper we present and analyze an algorithm for computing measures of generalized Voronoi regions which bypasses the explicit calculation of the Voronoi diagram. Rather, it relies on the iteration of a Markov kernel operator until the input density is accumulated in a neighborhood of the generators. This transforms the problem completely: instead of integrating a density over a priori unknown regions, we evolve the input density to be able to integrate over known regions. Our algorithm

- solves an eikonal equation (cf. (3.2)) once;
- uses the eikonal solution to construct a Markov kernel operator (cf. (3.1) and (3.7)) whose application to an input density moves mass towards the closest generator;
- iterates the operator until all mass from the input density is accumulated in a fixed neighborhood of the generators;
- computes w_i by integrating around this neighborhood.

Recently Saye and Sethian [40, 41] treated generalized Voronoi regions for tracking multiple interacting and evolving regions in complex multiphase physical systems. We note that while their goals and results are different, they use similar tools to tackle the interface-tracking problem.

We remark that the eikonal solution ϕ gives a distance function associated to the closest generator. Given that the spirit of our algorithm is to move mass towards the closest generator, one can naturally see the analogue with a gradient flow associated

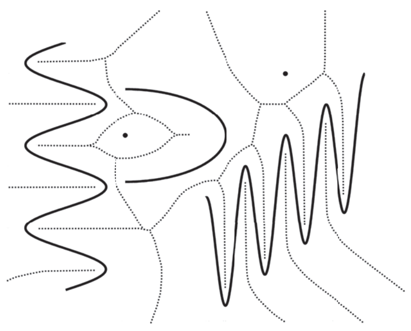


FIG. 1. *The generalized Voronoi diagram and the points in the domain that are equidistant to the same generator (in dashed lines). Along these dashed lines, the gradient of the distance function to the generators is discontinuous. The generators consist of three curves (in bold) and two points.*

with ϕ. However, at no point do we need the explicit calculation of $\nabla\phi$. The discontinuity set of $\nabla\phi$ is not a priori known and can be larger than the generalized Voronoi diagram. Figure 1 illustrates this point, as the set along which $\nabla\phi$ is discontinuous includes points where the closest distance is attained by more than one point on the *same* generator.

Thus the main novelties of our algorithm are as follows:

- It solves a fundamental geometric problem: how to efficiently integrate functions over generalized Voronoi regions.
- It applies in any space dimension and to any finite collection of generators consisting of sets of arbitrary codimension.
- It requires one solution ϕ of an eikonal equation associated with all the generators, and never requires the computation of the generalized Voronoi diagram. It also does not require the explicit calculation of the gradient $\nabla\phi$.
- The generators themselves are not discretized on the computational grid; instead the *distance* to the generators (the eikonal solution) is discretized on the grid. This geometric information can be precisely calculated (to machine precision using [46]) even for coarse grids.
- The method encompasses one of the basic steps needed to find CVTs of rigid bodies in $\mathbb{R}^d$.

The paper is organized as follows: We first discuss the direct approach to this problem, that is, computing the Voronoi diagram explicitly, and then integrating over the obtained regions. In particular, we write out an explicit algorithm to calculate these boundaries and describe exactly where the computational inefficiencies arise. Next, we introduce our integration algorithm and address how we approximate the true measures of the generalized Voronoi regions. We then introduce the computational algorithm and provide error estimates. For notational purposes alone it is often convenient to write out the details of the numerical scheme in $\mathbb{R}^2$; however, we stress that the computational method and the analysis of the method are valid in any spatial dimension. We end with several applications in $\mathbb{R}^2$ and $\mathbb{R}^3$ (sections 5.2 and 5.4), including a few results for CVTs of circles (in $\mathbb{R}^2$) and spheres (in $\mathbb{R}^3$). However, we emphasize that we show these examples simply to motivate and showcase our method—the detailed and comprehensive application of this integration scheme to compute the CVT of rigid bodies is not the purpose of this article, and will be addressed elsewhere.

2. The direct approach.

We first motivate our approximation algorithm by presenting a more direct approach, and by detailing key inefficiencies we aim at circumventing. To calculate the measures directly, the boundaries of the generalized Voronoi regions are calculated, and then the input density is integrated over each of these regions to obtain the measure of each generator.

2.1. Computing the boundaries.

There are only very special cases where the generalized Voronoi diagram, $\cup_{i=1}^{n} \partial V_i$, is available explicitly.[1] In most cases, the Voronoi diagram must be computed numerically. To compute the diagram, we first obtain a distance function, ϕ_i, corresponding to each generator Γ_i, $i = 1, \ldots, n$. Using these distance functions, we present an algorithm below (Algorithm 1) to find n level set functions $\phi_{\partial V_i}$, $i = 1, \ldots, n$, whose zero contours give the Voronoi diagram. For each $i \in \{1, \ldots, n\}$, the distance functions, ϕ_i, can be obtained by solving the eikonal equation with Γ_i as the initial contour. The generalized Voronoi boundaries are simply the sets where $\phi_{\partial V_i} = 0$ for $i = 1, \ldots, n$.

Algorithm 1 Calculate boundary level set functions $\phi_{\partial V_i}$, $i = 1, \ldots, n$.

```
for $i = 1 : n$ do
    Solve $|\nabla \phi_i| = 1$ in $\Omega$,
    such that $\phi_i(\Gamma_i) = 0$.
end for
for $i = 1 : n$ do
    $\phi_{\partial V_i}(x) \leftarrow \min_{j \in \{1,\ldots,n\} \setminus \{i\}} \phi_j(x) - \phi_i(x)$;
end for
```

2.1.1. The algorithm.

Let us look a little closer at this algorithm. For any fixed generator, Γ_i, the zeros of the functions ϕ_{ji} (where $\phi_{ji} = \phi_i - \phi_j$), $j \in \{1, \ldots, n\} \setminus \{i\}$, are all candidates for the Voronoi boundaries. These functions are zero at any point that is equidistant to both Γ_i and Γ_j. For each i, the algorithm picks the true solution by computing a minimum among all candidates, ϕ_{ji}, $j \in \{1, \ldots, n\} \setminus \{i\}$. A one-dimensional example of this algorithm is presented in Figure 2. For $i = 1$, the candidates for the Voronoi boundary are ϕ_{21} and ϕ_{31}. By taking the minimum of these two functions, we calculate the Voronoi bisector to be at $x = 0.3$, which is the correct solution. The same characterization is true in higher dimensions.

2.1.2. Computational complexity.

There are very few generators that admit an explicit distance function. One example is circular generators: the distance function is simply a cone (see Appendix A). For more general generators, it is necessary to solve an eikonal equation, using the generator as the initial contour. For each generator, one distance function is required. Assume there are M gridpoints in each of d dimensions. That results in n computations of $\mathcal{O}(M^d)$ complexity (using a fast sweeping method [47, 52, 53, 5]), or $\mathcal{O}(M^d \log(M^d))$ (using a fast marching method [44, 43, 37, 20]). For each i, all distance functions may be required to calculate ϕ_{ij} for every j (a curve in $\mathbb{R}^2$ or surface in $\mathbb{R}^3$ can intersect *all* other generators). Then the computational time (assuming the fast sweeping method) is

$$\mathcal{O}(M^d)(n^2 + n).$$

As the dimension and number of generators grow, this becomes prohibitively slow.

[1]See Appendix A for the equation of the bisectors in the case of circular and spherical generators.

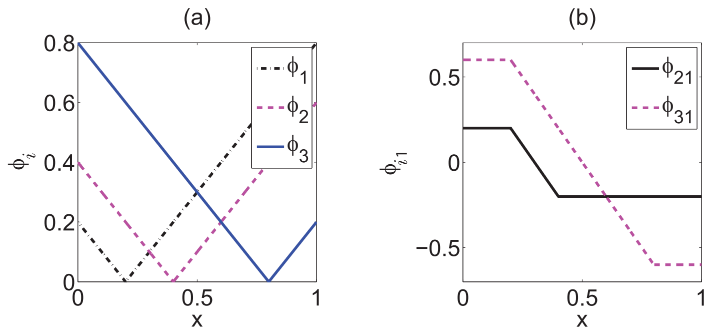


FIG. 2. (a) *Level set functions* $\phi_1(x)$*,* $\phi_2(x)$*,* $\phi_3(x)$ *with generators at* $x = 0.2, 0.4, 0.8$*.* (b) ϕ_{j1}*,* $j = 2, 3$*, corresponding to the level sets in panel* (a).

2.2. Integration. The calculation is not complete until we integrate the density function over the generalized Voronoi regions. First find the points where $\phi_{\partial V_i} = 0$ for all $i = 1, \ldots, n$. Then connect the points using interpolation, and integrate over each region. This can be done in a variety of ways, and the accuracy of the approach will depend on the accuracy of the eikonal solver that was used to obtain the distance functions ϕ_i, and the method used for obtaining the zeros of the $\phi_{\partial V_i}$.

3. Our approach. We now introduce a method for approximating the measures which does not require explicit knowledge of region boundaries. This avoids the computational challenges addressed in the previous section. We call any function a density if it is nonnegative and integrates to one over the domain Ω. The main idea is simple: instead of computing an integral over each generalized Voronoi region, V_i, to obtain the measure, accumulate the input density to a neighborhood of the generator, and then integrate the accumulated mass over this neighborhood. This shifts the problem from integrating over an unknown region to integrating over a known region. The key to this approach is to accumulate the input density in the correct way: we do this by iterating a Markov operator until a stationary density is reached.[2] The Markov operator accurately moves the input density towards the closest generator, albeit with some controlled error in a neighborhood of the region boundaries.

Our goal is to integrate any $\mu \in L^1$. To this end, we use the decomposition

$$\mu = c^+\mu^+ - c^-\mu^-,$$

where $c^+ = \int_\Omega \max(\mu(x), 0)\, dx$, $c^- = \int_\Omega \max(-\mu(x), 0)\, dx$, $\mu^+(x) = \frac{1}{c^+}\max(\mu(x), 0)$, and $\mu^-(x) = \frac{1}{c^-}\max(-\mu(x), 0)$. Without loss of generality, we perform the analysis on densities, and all results carry over to integrable functions under the above decomposition by the linearity of the constructed operator.

3.1. The Markov kernel operator. Let D denote the set of densities on Ω, that is, all nonnegative $\mu \in L^1(\Omega)$ such that $\|\mu\|_{L^1(\Omega)} = 1$. A linear operator $P : L^1(\Omega) \to L^1(\Omega)$ is called a *Markov operator* if $PD \subset D$. A *Markov kernel operator* has even more structure. It is defined in terms of a *stochastic kernel*, which is a nonnegative function $k : \Omega \times \Omega \to \mathbb{R}$ that satisfies

$$\int_\Omega k(x, y)\, dx = 1, \quad \text{a.e. } y \in \Omega.$$

[2]See [27, 28] for more information on Markov operators and stochastic kernels.

The stochastic kernel is deterministic; it is termed *stochastic* in the same spirit as a stochastic matrix. In fact, by discretizing a stochastic kernel appropriately, a stochastic matrix is obtained. We define the *Markov kernel operator*[3] P by its action on $\mu \in D$:

$$P[\mu](x) = \int_\Omega k(x,y)\mu(y)\,dy. \tag{3.1}$$

We iterate such a Markov kernel operator to accumulate the input density to neighborhoods around the generators, and finally integrate over these known regions to obtain the associated measure. In the following we show how this accumulation is encapsulated in the stochastic kernel and present its construction.

3.2. The stochastic kernel. The kernel is constructed by first solving for the minimum distance from points $x \in \Omega$ to any generator—this is the solution to the eikonal equation with the generators as initial contours. In particular, we first solve the following eikonal equation: Find the function $\phi : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} |\nabla\phi| &= 1 \quad \text{in } \Omega, \\ \phi &= 0 \quad \text{on } \cup_{i=1}^{n} \Gamma_i. \end{aligned} \tag{3.2}$$

We will transform this solution, ϕ, by changing the sign and scale, to put the generators at a local maximum and convert the "ridges" into "valleys." We also scale the solution to be in the range of $[0, 1]$. In doing so, the kernel will ensure that the input density moves towards the generators. To this end, we define the transformed eikonal solution $\bar{\phi} : \Omega \to [0, 1]$ as

$$\bar{\phi}(x) := 1 - \frac{|\phi(x)|}{\max_{y\in\Omega}(|\phi(y)|)}. \tag{3.3}$$

Let $\alpha > 0$. We may now define the function that assigns highest value to the generators and converts the ridges of the eikonal solution to valleys. We call this function $q : \Omega \to [0, 1 - \alpha]$ and define it as follows:

$$q(x) = \begin{cases} \bar{\phi}(x) & \text{if } \bar{\phi}(x) < 1 - \alpha, \\ 1 - \alpha & \text{otherwise.} \end{cases} \tag{3.4}$$

The parameter α is introduced to guarantee that the stochastic kernel is not singular. This will be discussed further in the following sections. The construction of $q(x)$ is depicted in Figure 3 for one spatial dimension.

To define the stochastic kernel we require some notation. Let $\overline{B_\varepsilon(x)}$ be the closed ball of radius $\varepsilon > 0$ around a point $x \in \Omega$. Moreover, let $\beta > 0$; this is another parameter that ensures the regularity of the kernel. We will first define $\tilde{k}(x, y)$ and then normalize it to be stochastic (to preserve the L^1 norm of the input density, μ). The form of $\tilde{k}(x, y)$ is

$$\tilde{k}(x,y) = \begin{cases} q(x) - q(y) + \beta & \text{for } y \in \overline{B_\varepsilon(x)} \quad \text{and} \quad q(x) \geq q(y), \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

[3]This equation was obtained by looking at a discrete-time Markov pure jump process whose initial position is generated by μ. $P[\mu](x)$ describes the probability that the process is at the point x in Ω after one time step.

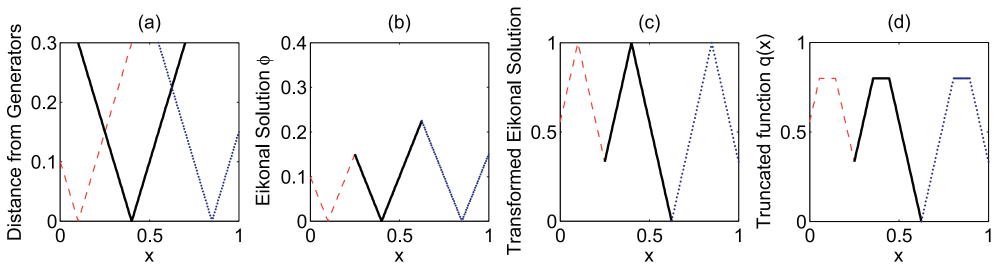


FIG. 3. (a) *Distance from generators at* $x = 0.1, 0.4, 0.85$. (b) *Minimum of distances from generators, i.e., the eikonal solution* $\phi(x)$. (c) *Transformed eikonal solution,* $\bar{\phi}(x)$. (d) *Truncated function* $q(x)$ *with* $\alpha = 0.2$.

This function is positive only for x and y within a small distance ε of one another. We do not allow the input density to evolve nonlocally, as we would like the density to accumulate around the closest generator. We constrain the function $\tilde{k}(x, y)$ to be positive only where $q(x)$ is greater than or equal to $q(y)$: that is, $\tilde{k}(x, y) > 0$ when point x is closer to the generator than point y is. In this way $\tilde{k}(x, y)$ will help to push the input density towards the closest generator. The stochastic kernel is

$$k_\Gamma(x, y) = \frac{\tilde{k}(x, y)}{\int_{u \in \Omega} \tilde{k}(u, y)\, du} \tag{3.6}$$

$$= \frac{(q(x) - q(y) + \beta)\, \mathbb{1}_{\{q(x) \geq q(y)\}} \mathbb{1}_{\overline{B_\varepsilon(x)}}(y)}{\int_{u \in \Omega} (q(u) - q(y) + \beta)\, \mathbb{1}_{\{q(u) \geq q(y)\}} \mathbb{1}_{\overline{B_\varepsilon(y)}}(u) du}. \tag{3.7}$$

By construction the kernel in (3.7) is stochastic and the associated Markov kernel operator sends mass towards the closest generator. In fact, we show in the next section that when P is iterated, mass will accumulate in a neighborhood of the closest generator. The constant β ensures there is positive density associated to moving to a location equally close to a generator. This parameter should be chosen to be much smaller than ε. If α and ε are chosen such that $2\alpha < \varepsilon$, then the set $A := \{x : q(x) \geq 1 - \alpha\}$ is invariant. This means that once mass moves to A, it may never move outside of this set. We will give these properties a detailed look in the next section. In Figure 4, we show two different stochastic kernels for the one-dimensional case. In (a), the generator is at $x = 0.5$. To the left of the generator, the density will move only towards $x = 0.5$. To the right of the generator, the density moves back towards $x = 0.5$. Moreover, once the density has jumped to a point of distance α or less from the generator, the point will stay in the invariant region $\{y : |0.5 - y| \leq \alpha\}$.

In Figure 4(b), we return to the example from Figure 3, where we have three generators at $x = 0.1$, 0.4, 0.85. This is to show how the kernel behaves near the boundary of the Voronoi region. Similar to part (a), we have density moving in the positive direction to the immediate left of a generator, and in the negative direction to the immediate right of a generator. The α-neighborhoods of each generator are invariant sets, where the density gets trapped. Moreover, at the Voronoi boundary (at $x = 0.25$, 0.625), there is symmetric probability of moving towards either generator. Table 1 summarizes the role of each parameter in k_Γ.

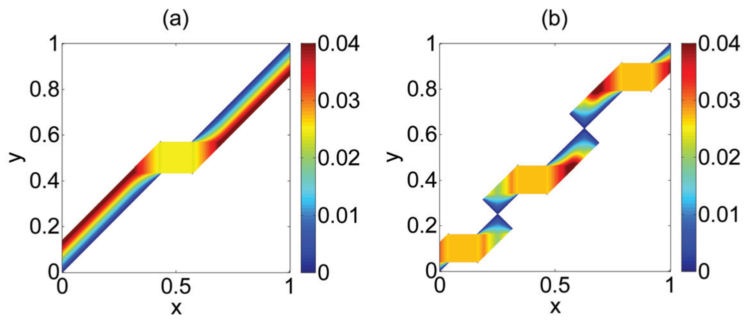


FIG. 4. *Stochastic kernels for a one-dimensional problem on* $\Omega = [0,1]$*: The probability of moving from a point* x *in the domain to another point* y *in the domain.* (a) *One generator at* $x = 0.5$, $\alpha = 0.067$, $\varepsilon = 0.14$. (b) *Three generators, as in Figure* 3, $\alpha = 0.06$, $\varepsilon = 0.1267$. *In both cases,* $\beta = 5.5 \times 10^{-4}$.

TABLE 1
Summary of parameters from stochastic kernel.

Parameter	Description	Values
ε	The maximum distance mass can move in one iteration	$\varepsilon > 0$
α	Controls the size of the invariant set around each generator	$0 \le \alpha \le \varepsilon/2$
β	Allows density to move to areas equally close to a generator	$0 \le \beta \ll \varepsilon$

3.3. Stationary densities and convergence results. In this section, we prove some results concerning the kernel operator. In particular, we discuss invariant sets, how the Markov kernel operator accumulates the input density, and how this allows us to calculate these measures.

3.3.1. Invariant sets and stationary densities. We define a set $A \subset \Omega$ to be *invariant* under P if for any $\mu \in D$ concentrated on A, $\int_{x\in A} P[\mu](x)dx = 1$. Let $\Gamma_i^\alpha := \left\{x \in \Omega \,\middle|\, |x-y| \le \alpha \text{ for some } y \in \Gamma_i\right\}$ denote the α-neighborhood of the generator Γ_i for $i = 1, \dots, n$.

PROPOSITION 3.1. *The sets* Γ_i^α *are invariant under* P.

Proof. Let μ be a density that is concentrated on Γ_i^α. Then

$$P[\mu](x) = \int_{y\in\Omega} k_\Gamma(x,y)\mu(y)\,dy = \int_{y\in\Gamma_i^\alpha} k_\Gamma(x,y)\mu(y)\,dy.$$

For any $y \in \Gamma_i^\alpha$, the stochastic kernel reduces to

$$k_\Gamma(x,y) = \begin{cases} \left(\int_{u\in\Gamma_i^\alpha\cap\overline{B_\varepsilon(y)}} du\right)^{-1} & \text{for } x \in \Gamma_i^\alpha \cap \overline{B_\varepsilon(y)}, \\ 0 & \text{otherwise.}\end{cases}$$

So we have

$$
\begin{aligned}
\int_{x\in\Gamma_i^\alpha} P[\mu](x)\,dx &= \int_{x\in\Gamma_i^\alpha}\left(\int_{y\in\Gamma_i^\alpha} k_\Gamma(x,y)\mu(y)\,dy\right)dx\\
&= \int_{x\in\Gamma_i^\alpha}\left(\int_{y\in\Gamma_i^\alpha} \frac{\mathbb{1}_{x\in\Gamma_i^\alpha\cap\overline{B_\varepsilon(y)}}}{\int_{u\in\Gamma_i^\alpha\cap\overline{B_\varepsilon(y)}}du}\mu(y)\,dy\right)dx\\
&= \int_{y\in\Gamma_i^\alpha} \underbrace{\frac{\int_{x\in\Gamma_i^\alpha\cap\overline{B_\varepsilon(y)}}dx}{\int_{u\in\Gamma_i^\alpha\cap\overline{B_\varepsilon(y)}}du}}_{=1}\,\mu(y)\,dy \qquad (3.8)\\
&= \int_{y\in\Gamma_i^\alpha}\mu(y)\,dy = 1.
\end{aligned}
$$

In (3.8), Fubini's theorem was applied.

We define a *stationary density* of P to be any density μ such that $P[\mu](x) = \mu(x)$. Before looking at how the operator P moves density to the invariant sets, we first consider the special case of point generators. As the following theorem shows, in this case the operator P has stationary densities of the following form:

$$\gamma_i(x) = v_{\alpha,i}\,\mathbb{1}_{\Gamma_i^\alpha},$$

where $v_{\alpha,i} = \left(\int_{\Gamma_i^\alpha} dx\right)^{-1}$ and $\mathbb{1}_{\Gamma_i^\alpha}$ denotes the indicator function of the set Γ_i^α.

THEOREM 3.2. *Let $\Gamma_i = \{z_i\}$ for some point $z_i \in \Omega$ for all $i = 1,\dots,n$. Further let $2\,\alpha \le \varepsilon$, and for $i = 1,\dots,n$, let $\tilde{w}_i \ge 0$ and $\sum_{i=1}^n \tilde{w}_i = 1$. Then*

$$P\Big[\sum_{i=1}^n \tilde{w}_i\gamma_i(y)\Big](x) = \sum_{i=1}^n \tilde{w}_i\gamma_i(x).$$

Proof. Since P is a linear operator, it suffices to show

$$P[\,\mathbb{1}_{\Gamma_i^\alpha}(y)](x) \;=\; \int_\Omega k_\Gamma(x,y)\mathbb{1}_{\Gamma_i^\alpha}(y)\,dy \;=\; \mathbb{1}_{\Gamma_i^\alpha}(x). \qquad (3.9)$$

We consider separately the cases $x \notin \Gamma_i^\alpha$ and $x \in \Gamma_i^\alpha$. For $x \notin \Gamma_i^\alpha$ and $y \in \Gamma_i^\alpha$, $k_\Gamma(x,y) = 0$. This is because the stochastic kernel $k_\Gamma(x,y)$ never sends mass from a higher $q(y)$ to a lower $q(x)$: $k_\Gamma(x,y) = f(x,y)\mathbb{1}_{\{q(y)\le q(x)\}}(x,y)$. On the other hand, if $y \in \Gamma_i^\alpha$, then $q(y) = 1-\alpha$. However, for $x \notin \Gamma_i^\alpha$, $q(x) < 1-\alpha$. Thus for $x \notin \Gamma_i^\alpha$, $k_\Gamma(x,y) = 0$, and hence $P[\,\gamma_i(y)](x) = 0$ on this set.

Suppose $x \in \Gamma_i^\alpha$. Then for any $y \in \Gamma_i^\alpha$,

$$|x-y| \;=\; |x-z_i+z_i-y| \;\le\; |x-z_i| \;+\; |z_i-y| \;\le\; 2\alpha \;\le\; \varepsilon.$$

That is, $x \in \Gamma_i^\alpha$ and $y \in \Gamma_i^\alpha$ imply $y \in \overline{B_\varepsilon(x)}$, and in particular the kernel is nonzero. For $x \in \Gamma_i^\alpha$,

$$
\begin{aligned}
P\big[\,\mathbb{1}_{\Gamma_i^\alpha}(y)\big](x) &= \int_\Omega k_\Gamma(x,y)\mathbb{1}_{\Gamma_i^\alpha}(y)\,dy\\
&= \int_\Omega \frac{(q(x)-q(y)+\beta)\,\mathbb{1}_{\{q(y)\le q(x)\}}\mathbb{1}_{\overline{B_\varepsilon(x)}}(y)}{\int_{u\in\Omega}(q(u)-q(y)+\beta)\,\mathbb{1}_{\{q(y)\le q(u)\}}\mathbb{1}_{\overline{B_\varepsilon(y)}}(u)du}\mathbb{1}_{\Gamma_i^\alpha}(y)\;dy\\
&= \int_{\Gamma_i^\alpha}\frac{\beta}{\int_{\Gamma_i^\alpha}(u)\;\beta\;du}\;dy = 1.
\end{aligned}
$$

In the last line, we used the fact that for $x, y \in \Gamma_i^\alpha$, $q(x) = q(y) = 1 - \alpha$, and that $q(y) \leq q(u)$ if and only if $u \in \Gamma_i^\alpha$. Note that for $u \in \Gamma_i^\alpha$, u is at most a distance of ε from y.

Remark. Theorem 3.2 holds for *any* positive constants $\tilde{w}_i$ that sum to one. We will show that for a given input density μ and a given ε, the Markov operator P will approach a stationary density such that the constants $\tilde{w}_i$ approximate the generalized Voronoi measures, w_i. For nonpoint generators, we will demonstrate that the measures are similarly approximated by

$$\int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx$$

for m large enough so that the density has concentrated on the invariant sets. Here P^m denotes the mth iterate of P.

3.3.2. Convergence to invariant sets. In this section we will show that given any starting density, $\mu \in L^1(\Omega)$, P will accumulate μ onto the invariant sets. That is,

$$\lim_{m\to\infty} \int_{\Omega\setminus\cup\Gamma_i^\alpha} P^m[\mu](x)dx \to 0.$$

First we show that any density concentrated on an annulus around the invariant set Γ_i^α will have some mass transported to the invariant set in one iteration of P. To this end, denote the annulus around the invariant set by $R := \Gamma_i^{\alpha+\varepsilon} \setminus \Gamma_i^\alpha$, that is,

$$R = \{x \in V_i \mid \alpha < |x - y| \leq \alpha + \varepsilon \text{ for } y \in \Gamma_i\}.$$

LEMMA 3.3. *Let $\mu \in D$ be supported on R and strictly positive. Then*

$$\int_{\Omega|\Gamma_i^\alpha} P[\mu](x)dx < \int_R \mu(x)dx.$$

Proof. Consider how P acts on μ:

$$\begin{aligned}
P[\mu](x) &= \int_{y\in\overline{B_\varepsilon(x)}} k_\Gamma(x,y)\mu(y)dy \\
&= \int_{y\in\overline{B_\varepsilon(x)}\cap R} k_\Gamma(x,y)\mu(y)dy \\
&= \mathbb{1}_{\{x\in\Gamma_i^\alpha\}} \int_{y\in\overline{B_\varepsilon(x)}\cap R} k_\Gamma(x,y)\mu(y)dy + \mathbb{1}_{\{x\notin\Gamma_i^\alpha\}} \int_{y\in\overline{B_\varepsilon(x)}\cap R} k_\Gamma(x,y)\mu(y)dy \\
&= \mathbb{1}_{\{x\in\Gamma_i^\alpha\}} \underbrace{\int_{y\in\overline{B_\varepsilon(x)}\cap R} k_\Gamma(x,y)\mu(y)dy}_{=:I} + \mathbb{1}_{\{x\in R\}} \int_{y\in\overline{B_\varepsilon(x)}\cap R} k_\Gamma(x,y)\mu(y)dy.
\end{aligned}$$

In the last line, we used the fact that for $x \notin \Gamma_i^\alpha$, the kernel is zero outside R (in that region $q(y) > q(x)$). We claim $I > 0$. To this end, note that $\overline{B_\varepsilon(x)} \cap R \neq \emptyset$ since R is adjacent to Γ_i^α. For $y \in R$ and $x \in \Gamma_i^\alpha$, $q(y) < 1 - \alpha = q(x)$ and hence $k_\Gamma(x,y) > 0$. On the other hand, the support of μ is R, and for $y \in R$, $\mu(y) > 0$. Thus $I > 0$ and hence

$$\int_{\Omega\setminus\Gamma_i^\alpha} P[\mu](x)dx \;=\; 1 - \int_{\Gamma_i^\alpha} P[\mu](x)dx \;<\; 1 \;=\; \int_R \mu(x)dx.$$

The lemma is proved.

As a corollary we have that this is true for any annulus a certain distance d from the invariant set: In one iteration of P, the support will include a region at a distance $d-\varepsilon$ from the invariant set.

COROLLARY 3.4. *Let μ be a positive density with support on the (nontrivial) set $R := V_i \cap (\Gamma_i^{\alpha+d} \setminus \Gamma_i^{\alpha+d-\varepsilon})$. Then*

$$\int_{\Omega\setminus\Gamma_i^{\alpha+d-\varepsilon}} P[\mu](x)dx < \int_R \mu(x)dx.$$

In fact in each iteration, the support will include regions a distance ε closer to the invariant set. Therefore we have the following result.

PROPOSITION 3.5. *If μ is a density concentrated on a set whose minimum distance to Γ_i^α is d, then $\operatorname{supp}(P^m[\mu]) \cap \Gamma_i^\alpha \neq \emptyset$ for all $m \geq \lceil \frac{d}{\varepsilon} \rceil$.*

Before proving the main convergence result, we recall the following definitions.

DEFINITION 3.6. *A sequence of measures μ_m is called* tight *if for any $\epsilon > 0$ there is a compact subset $K_\epsilon \subset \Omega$ such that for all $m \in \mathbb{N}$, $\mu_m(K_\epsilon) > 1-\epsilon$.*

DEFINITION 3.7. *We say a sequence of measures μ_m* converges weakly *to a measure μ^* if for all bounded continuous functions g,*

$$\int_\Omega g(x)\mu_m(dx) \to \int_\Omega g(x)\mu^*(dx).$$

In this case we write $\mu_m \xrightarrow{w} \mu^*$.

We now wish to prove that as $m \to \infty$, $P^m[\mu]$ loses its support on $\Omega \setminus \cup\Gamma_i^\alpha$. For the proof presented below, we will require the application of P to potentially singular measures. Our current definition of P is based upon the kernel k_Γ which is discontinuous because of the cut-off $\mathbb{1}_{\{|x-y|\leq\varepsilon\}}$. By including an annular region of thickness $\lambda \ll \varepsilon$ around $B_\varepsilon(y)$ which linearly decreases in the radial direction to 0, we obtain an analogous kernel $\bar{k}_\Gamma(x,y)$ which is continuous in x and y. Recall $\tilde{k}(x,y)$ from (3.5). We define this linearly decreasing function to be

$$k^{\text{lin}}(x,y) = \frac{\tilde{k}(y+\varepsilon\frac{x-y}{|x-y|}, y)}{\lambda}\left|(x-y)\Big(1-\frac{\varepsilon+\lambda}{|x-y|}\Big)\right|.$$

Then the modified kernel has the following form:

$$\bar{k}_\Gamma(x,y) = \frac{\tilde{k}(x,y)\mathbb{1}_{\{|x-y|\leq\varepsilon\}} + k^{\text{lin}}(x,y)\mathbb{1}_{\{\varepsilon<|x-y|\leq\varepsilon+\lambda\}}}{\int_{u\in\Omega}\tilde{k}(u,y)\mathbb{1}_{\{|u-y|\leq\varepsilon\}} + k^{\text{lin}}(u,y)\mathbb{1}_{\{\varepsilon<|u-y|\leq\varepsilon+\lambda\}}\,du}.$$

By taking λ to be smaller than the grid size, P and $\bar{P}$ yield the same algorithm. Hence we may carry out the convergence analysis for $\bar{P}$ instead of P. In particular, Lemma 3.3, Corollary 3.4, and Proposition 3.5 hold for $\bar{P}$ and any probability measure μ.

THEOREM 3.8. *Let $\mu \in L^1(\Omega)$ be a density. Then*

$$\lim_{m\to\infty}\int_{\Omega\setminus\cup\Gamma_i^\alpha} \bar{P}^m[\mu](x)dx \to 0.$$

Proof. Let

$$a_0 = \int_{\Omega\setminus\cup\Gamma_i^\alpha}\mu(x)dx \qquad \dots \qquad a_m = \int_{\Omega\setminus\cup\Gamma_i^\alpha}\bar{P}^m[\mu](x)dx.$$

By Lemma 3.3, Corollary 3.4, and Proposition 3.5, $\{a_m\}_{m=0}^{\infty}$ is a nonincreasing sequence: Indeed, if $\bar{P}^m[\mu]$ has support in an $\alpha+\varepsilon$-neighborhood of any generator, then by Lemma 3.3, $a_{m+1} < a_m$. If $\bar{P}^m[\mu]$ doesn't have support in an $\alpha+\varepsilon$-neighborhood, but has support in an $\alpha+\varepsilon+d$-neighborhood of some generator, then $a_{m+\lceil d/\varepsilon \rceil} < a_m$ by Corollary 3.4 and Proposition 3.5. If $\bar{P}^m[\mu]$ has no support outside $\cup\Gamma_i^\alpha$, we're done, because $a_m \equiv 0$.

Since $1 \geq a_m \geq 0$ for all m, the sequence $\{a_m\}_{m=0}^{\infty}$ converges. Our goal is to show that this sequence converges to zero. Suppose this is not the case. Then there is some $c \in (0,1)$ such that

$$\lim_{m\to\infty} a_m = c.$$

Because $\bar{P}$ is a Markov operator on a closed, bounded domain, the sequence of measures $\bar{P}^m[\mu]$ is tight. By Prokhorov's theorem [6], for every tight sequence of measures, there is a weakly convergent subsequence. So for some m_k, $k \in \mathbb{N}$,

$$\bar{P}^{m_k}[\mu] \overset{w}{\to} \mu^*.$$

Note here that a priori we cannot guarantee that the measure μ^* is absolutely continuous with respect to Lebesgue measure. For this reason we are working with $\bar{P}$. Since $\lim_{m\to\infty} a_m = c$, we know that $\int_{\Omega\setminus\cup\Gamma_i^\alpha} \mu^*(dx) = c$. So there is some set in $\Omega \setminus \cup\Gamma_i^\alpha$ to which μ^* assigns mass. We claim that for any $l \in \mathbb{N}$,

$$\bar{P}^{m_k+l}[\mu] \overset{w}{\to} \bar{P}^l[\mu^*]. \tag{3.10}$$

First let $l = 1$. Then for any $g \in C(\Omega \setminus \cup\Gamma_i^\alpha)$, it suffices to prove that

$$\int g(x)\bar{k}_\Gamma(x,y)dx \quad \text{is a continuous function of } y, \tag{3.11}$$

as then

$$\begin{aligned}
\int g(x)\bar{P}[\bar{P}^{m_k}[\mu]](x)dx &= \int g(x)\int \bar{k}_\Gamma(x,y)\bar{P}^{m_k}[\mu](y)dy\,dx \\
&= \int \Bigg(\underbrace{\int g(x)\bar{k}_\Gamma(x,y)dx}_{\in C(\Omega\setminus\cup\Gamma_i^\alpha)}\Bigg)\bar{P}^{m_k}[\mu](y)dy \\
&\to \int \left(\int g(x)\bar{k}_\Gamma(x,y)dx\right)\mu^*(dy) \\
&= \int g(x)\bar{P}[\mu^*](dx).
\end{aligned}$$

But (3.11) holds true by the continuity of $\bar{k}_\Gamma$. Since the induction step is analogous, we have (3.10). But now μ^* assigns positive probability on $\Omega \setminus \cup\Gamma_i^\alpha$. By Proposition 3.5 (applied to $\bar{P}$ and any probability measure μ^*), for $l \geq \lceil \frac{d}{\varepsilon+\lambda} \rceil + 1 \in \mathbb{N}$, $\bar{P}^l[\mu^*]$ will send some of this mass to the generators. Therefore $\int_{\Omega\setminus\cup\Gamma_i^\alpha} \bar{P}^l[\mu^*](dx) < c$. But this is a contradiction and therefore $c = 0$.

3.3.3. Approximating the measures. In this section we demonstrate the error induced from iterating the Markov operator.

THEOREM 3.9. *Given a positive function $\mu \in L^{1+\delta}(\Omega)$, then*

$$\lim_{m\to\infty} \int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx = w_i + \mathcal{O}\left(\varepsilon^{\frac{\delta}{1+\delta}}\right).$$

Proof. Consider the generalized Voronoi region V_i. Define

$$V_i^{\varepsilon/2} = \{x \in \Omega \mid |x-y| < \epsilon/2,\ y \in \partial V_i\}.$$

Furthermore for $m = 1, 2, \ldots,$ define $k_m(x,y)$ inductively by

$$k_1(x,y) \;=\; k_\Gamma(x,y), \qquad \ldots \qquad k_m(x,y) = \int k_\Gamma(x,z)k_{m-1}(z,y)dz,$$

and note that $P^m[\mu] = \int_\Omega k_m(x,y)\mu(y)\,dy$. Moreover, let $\mu(x) = \mu_1(x)+\mu_2(x)+\mu_3(x)$, where $\mathrm{supp}(\mu_1) \subset V_i \setminus V_i^{\varepsilon/2}$, $\mathrm{supp}(\mu_2) \subset V_i^{\varepsilon/2}$, and $\mathrm{supp}(\mu_3) \subset (V_i \cup V_i^{\varepsilon/2})^c$. Then for any m,

$$\begin{aligned}\int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx &= \int_{\Gamma_i^\alpha} P^m[\mu_1](x)\,dx + \int_{\Gamma_i^\alpha} P^m[\mu_2](x)\,dx + \underbrace{\int_{\Gamma_i^\alpha} P^m[\mu_3](x)\,dx}_{=0}\\ &= \underbrace{\int_{\Gamma_i^\alpha} P^m[\mu_1](x)\,dx}_{=:I_1} + \underbrace{\int_{\Gamma_i^\alpha} P^m[\mu_2](x)\,dx.}_{=:I_2}\end{aligned}$$

For $y \in \mathrm{supp}(\mu_1)$, $k_\Gamma(x,y)$ will only be nonzero for $x \in \mathrm{supp}(\mu_1)$. Thus, $\mathrm{supp}(P^m[\mu_1]) \subset V_i \setminus V_i^{\varepsilon/2}$. But we know that all mass converges to the invariant sets Γ_i^α, so $\mathrm{supp}(P^m[\mu_1]) \to \Gamma_i^\alpha$ as $m \to \infty$. Since this mass is conserved,

$$\lim_{m\to\infty} I_1 = \int_{V_i\setminus V_i^{\varepsilon/2}} \mu_1(x)\,dx.$$

Now consider I_2:

$$\begin{aligned}I_2 &= \int_{\Gamma_i^\alpha}\left(\int_\Omega k_m(x,y)\,\mu(y)\,\mathbb{1}_{V_i^{\varepsilon/2}}(y)\,dy\right)dx\\ &= \int_\Omega \mu(y)\,\mathbb{1}_{V_i^{\varepsilon/2}}(y)\int_{\Gamma_i^\alpha} k_m(x,y)\,dx\,dy\\ &\le \int_\Omega \mu(y)\,\mathbb{1}_{V_i^{\varepsilon/2}}(y)\,dy\\ &\le \|\mu\|_{1+\delta}\left(\int_\Omega\left(\mathbb{1}_{V_i^{\varepsilon/2}}(y)\right)^{\frac{1+\delta}{\delta}}\,dy\right)^{\frac{\delta}{1+\delta}}\\ &\le \|\mu\|_{1+\delta}\,\mathrm{measure}(V_i^{\varepsilon/2})^{\frac{\delta}{1+\delta}} \;=\; \mathcal{O}\left(\varepsilon^{\frac{\delta}{1+\delta}}\right).\end{aligned}$$

Here we use Fubini's theorem, Hölder's inequality, and the facts that k_m is a stochastic kernel and that the measure of the set $V_i^{\varepsilon/2}$ is $\mathcal{O}(\varepsilon)$. Similarly,

$$\int_{V_i\cap V_i^{\varepsilon/2}} \mu(x)dx \leq \|\mu\|_{1+\delta} \text{ measure}\left(V_i \cap V_i^{\varepsilon/2}\right)^{\frac{\delta}{1+\delta}} = \mathcal{O}\left(\varepsilon^{\frac{\delta}{1+\delta}}\right).$$

Therefore,

$$\begin{aligned}\lim_{m\to\infty}\int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx &= \lim_{m\to\infty}(I_1+I_2)\\ &= \int_{V_i\setminus V_i^{\varepsilon/2}} \mu(x)\,dx + \mathcal{O}(\varepsilon^{\frac{\delta}{1+\delta}}) + \int_{V_i\cap V_i^{\varepsilon/2}} \mu(x)\,dx - \int_{V_i\cap V_i^{\varepsilon/2}} \mu(x)\,dx\\ &= \int_{V_i} \mu(x)\,dx + \mathcal{O}\left(\varepsilon^{\frac{\delta}{1+\delta}}\right)\\ &= w_i + \mathcal{O}\left(\varepsilon^{\frac{\delta}{1+\delta}}\right).\end{aligned}$$

The theorem is proved.

COROLLARY 3.10. *For $\mu \in L^\infty(\Omega)$,*

$$\lim_{m\to\infty}\int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx = w_i + \mathcal{O}(\varepsilon),$$

that is, bounded densities will yield a linear approximation of the true measure.

Remark. For $d = 2$, $\mu = |\Omega|^{-1}$, and point generators,

$$\lim_{m\to\infty}\int_{\Gamma_i^\alpha} P^m[\mu](x)\,dx = w_i + \mathcal{O}(\varepsilon^2).$$

For point generators, the region boundaries ∂V_i are piecewise linear. In the region $V_i^{\varepsilon/2}$, the operator P sends mass to the "wrong" generator. However, when the region boundaries are linear and $\mu = 1$, this error is canceled symmetrically. For any point $y \in V_i^{\varepsilon/2}$, there is a symmetric point $y' \in V_i^{\varepsilon/2}$ such that $y' \in V_j$ and $\int_{V_j} P[\delta_y]dx = \int_{V_i} P[\delta_{y'}]dx$. To find y', simply reflect y perpendicularly in ∂V_i. This symmetry degrades in ε-neighborhoods of the corners of ∂V_i. However, the corners are $\mathcal{O}(\varepsilon^2)$. For point generators, this symmetry is apparent in higher dimensions as well.

3.4. Remark on non-Euclidean distances.

In the development of the Markov operator, Euclidean distance was used (through $\phi(x)$). The Euclidean distance function can be replaced by any distance function. To use a different notion of distance, one need only change ϕ, for example, by solving a different eikonal equation. Using the algorithm of [46], ℓ^p distance and multiplicatively weighted distances—i.e., $\phi(x) = \min_i \left(a_i \operatorname{dist}(x, \Gamma_i)\right)$, where a_i are positive constants—can be used.

4. Numerical scheme. In this section we discuss the numerical discretization of the Markov operator, the algorithm to compute the measures, and error estimates.

4.1. Computational domain. We now discuss how to calculate these measures on a discrete computational grid. Because we are interested in the *integral* of $P^m[\mu]$ on the invariant set, we construct a finite volume method. By using a finite volume approach to discretize the Markov operator, we can explicitly deal with the singularities in the kernel, as we are no longer working with the density, but rather the integrated density. Therefore, we take $\alpha = 0, \beta = 0$, and we set $\varepsilon = h$, for example. In the previous sections, we carefully defined the stochastic kernel in terms of the regularization parameters α and β, though we now set them to zero. This was for ease of analysis: we first demonstrate that the operator P has invariant sets Γ_i^α. Then it is clear that as $\alpha \to 0$, the operator concentrates mass along singular sets. Because we simply need the measure that $\lim_{m\to\infty} P^m[\mu]$ assigns to the invariant sets, computationally, we can set α, $\beta = 0$. Numerically, the discretized kernel is a left-stochastic matrix: for any column that sums to zero, there must be a corresponding diagonal value of one; these are exactly the grid points where mass accumulates, the invariant sets. We present the discretization of the Markov operator in two dimensions for ease of notation; however, we emphasize that the method is first order in any dimension by the analogous calculations.

Because $\Omega \subset \mathbb{R}^2$ is a bounded domain, there exist $R_1, R_2, R_3, R_4 \in \mathbb{R}$ such that $\Omega \subset [R_1, R_2] \times [R_3, R_4]$. We take R_1, R_2, R_3, R_4 such that $[R_1, R_2] \times [R_3, R_4]$ is the smallest square covering Ω. Let $h = \frac{R_4 - R_3}{N}$. Then the computational domain Ω_h is defined to be all pairs (x_i, y_j) of the following form:

$$\begin{aligned} x_i &= R_1 + (i-1)h, \quad i = 1, \dots, N+1, \\ y_j &= R_3 + (j-1)h, \quad j = 1, \dots, N+1. \end{aligned}$$

For all functions previously defined on the bounded domain Ω, including the density $\mu \in L^1(\Omega)$ and the kernel k_Γ, we extend them by zero to all of the computational domain.

4.2. Discretization and error analysis: One iteration. To obtain an iterative scheme, we first approximate the kernel by a piecewise constant function, and subsequently discretize all remaining integrals using the trapezoidal rule. We analyze the error associated with one iteration, and then with multiple iterations.

4.2.1. Approximating the kernel. We approximate $k_\Gamma(x, y, u, v)$ by a piecewise constant function along each grid cell in (u, v). In particular, for $(u, v) \in [x_i, x_{i+1}) \times [y_j, y_{j+1})$, we approximate the kernel $k_\Gamma(x, y, u, v)$ by its value at the left lower endpoint,

$$k_\Gamma(x, y, u, v) = k_\Gamma(x, y, x_i, y_j) + \mathcal{O}(h). \tag{4.1}$$

Let $\varepsilon = ch$ for $c \geq 1 \in \mathbb{Z}$ (but $c \ll N$). We let $Q_{ij}^{(1)}$ denote the mass moved to the region $[x_i, x_{i+1}) \times [y_j, y_{j+1})$ after one iteration and $Q_{ij}^{(0)}$ the density integrated over the same gridbox:

$$\begin{aligned}
Q_{ij}^{(1)} &:= \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} P[\mu](x,y)dy\,dx \\
&= \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \left(\int_{x_{i-c}}^{x_{i+c}} \int_{y_{j-c}}^{y_{j+c}} k_\Gamma(x,y,u,v)\,\mu(u,v)\,dv\,du \right) dy\,dx \qquad (4.2)\\
&= \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \Big(k_\Gamma(x,y,x_k,y_l) + \mathcal{O}(h) \Big) Q_{kl}^{(0)}\,dy\,dx \\
&= \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \underbrace{\int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} k_\Gamma(x,y,x_k,y_l)\,dy\,dx}_{=:K_{ijkl}}\; Q_{kl}^{(0)} \\
&\qquad + \underbrace{\mathcal{O}(h) \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \int_{x_i-c}^{x_{i+c}} \int_{y_{j-c}}^{y_{j+c}} \mu(u,v)\,dv\,du\,dy\,dx}_{\le \|\mu\|_\infty (2c+1)^2 h^4 \mathcal{O}(h)} \qquad (4.3)\\
&= \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl}^{(0)} + \mathcal{O}(h^5). \qquad (4.4)
\end{aligned}$$

The summations and integrals above run from $k = \max(i-c, 1)$ to $k = \min(i + c-1, N-1)$, and similarly for l, to remain in the computational domain. The kernel is only nonzero for (x, y) and (u, v) within a distance $\varepsilon = ch$ of each other. For $(x, y) \in [x_i, x_{i+1}) \times [y_j, y_{j+1})$, (u, v) is restricted to $[x_{i-c}, x_{i+c}) \times [y_{j-c}, y_{j+c})$, which is precisely the range of the inner integration.

In (4.4), K_{ijkl} is an element of a sparse tensor, with at most $(2c+1)^2N^2$ nonzero elements (out of N^4). By (4.4), we see that iterating the discretized Markov operator reduces to multiplying a sparse $N^2 \times N^2$ matrix ($\mathbf{K}$) by an $N^2 \times 1$ vector ($\mathbf{Q}$).

The error from assuming $k_\Gamma(x, y, u, v)$ is piecewise constant in (u, v) is $\mathcal{O}(h^5)$ under one iteration of the Markov operator. We must still discretize the integral of the kernel and the integral of the input density; these integrals are discretized using the trapezoidal rule.

4.2.2. Integrating the kernel. We construct the discretized kernel with elements

(4.5)

$$\begin{aligned}
K_{ijkl} &= \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} k_\Gamma(x,y,x_k,y_l)\;dy\,dx \\
&= \Big(\underbrace{\int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} k_0(x,y,x_k,y_l)\;dy\,dx}_{:=I_1}\Big)\Big(\underbrace{\int_{x_{k-c}}^{x_{k+c}} \int_{y_{l-c}}^{y_{l+c}} k_0(x,y,x_k,y_l)\;dy\,dx}_{:=I_2}\Big)^{-1}
\end{aligned}$$

for $i, j, k, l \in \{1, \ldots, N\}$. These integrals are computed using the trapezoidal rule. Let $\hat{K}_{ijkl}$ denote the discretized tensor element and define

$$\begin{aligned}
\mathrm{Trap}_1 &= \frac{h^2}{4}\Big(k_0(x_i, y_j, x_k, y_l) + k_0(x_i, y_{j+1}, x_k, y_l)\Big)\\
&\quad + \frac{h^2}{4}\Big(k_0(x_{i+1}, y_j, x_k, y_l) + k_0(x_{i+1}, y_{j+1}, x_k, y_l)\Big),\\
\mathrm{Trap}_2 &= \sum_{r=k-c}^{k+c-1}\sum_{p=l-c}^{l+c-1} \frac{h^2}{4}\Big(k_0(x_r, y_p, x_k, y_l) + k_0(x_r, y_{p+1}, x_k, y_l)\Big)\\
&\quad + \sum_{r=k-c}^{k+c-1}\sum_{p=l-c}^{l+c-1} \frac{h^2}{4}\Big(k_0(x_{r+1}, y_p, x_k, y_l) + k_0(x_{r+1}, y_{p+1}, x_k, y_l)\Big).
\end{aligned}$$

Then $I_1 = \mathrm{Trap}_1 + \mathcal{O}(h^3)$ (see Appendix C).[4] Similarly, $I_2 = \mathrm{Trap}_2 + \mathcal{O}(h^3)$. The error from discretizing K_{ijkl} using the trapezoidal rule is $\mathcal{O}(h^3)$, that is,

$$\underbrace{\Big|\sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl} - \sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}\Big|}_{=:E} \le \mathcal{O}(h^3).$$

Observe that

$$\begin{aligned}
&(4.6) \qquad E \le \sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} \Big|K_{ijkl} - \hat{K}_{ijkl}\Big| Q_{kl}\\
&(4.7) \qquad = \sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} \Big|\frac{I_1}{I_2} - \frac{\mathrm{Trap}_1}{\mathrm{Trap}_2}\Big| Q_{kl}\\
&(4.8) \qquad = \sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} \Big|\frac{I_1\,\mathrm{Trap}_2 - \mathrm{Trap}_1 I_2}{\mathrm{Trap}_2\, I_2}\Big| Q_{kl}\\
&(4.9) \qquad \le \sum_{k=i-c}^{i+c-1}\sum_{l=j-c}^{j+c-1} \frac{\overbrace{|I_1 - \mathrm{Trap}_1|}^{=\mathcal{O}(h^3)}\,\overbrace{|\mathrm{Trap}_2|}^{=\mathcal{O}(h^2)} + \overbrace{|\mathrm{Trap}_1|}^{=\mathcal{O}(h^2)}\,\overbrace{|I_2 - \mathrm{Trap}_2|}^{=\mathcal{O}(h^3}}{\underbrace{|\mathrm{Trap}_2\, I_2|}_{=Ch^4}} \underbrace{Q_{kl}}_{=\mathcal{O}(h^2)}\\
&(4.10) \qquad = \mathcal{O}(h^3).
\end{aligned}$$

Once we have discretized the final integral and the integral of the input density, the Markov operator will be fully discretized, and we can analyze the error accumulated in the first step.

4.2.3. Integrating the input density. We must take the input density μ and compute the mass matrix $\mathbf{Q}^{(0)}$,

$$Q_{ij}^{(0)} := \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \mu(u, v)\, dv\, du. \tag{4.11}$$

[4]The error is different from the standard trapezoidal rule error because the limits of integration depend on the grid size; see Appendix C for details.

These integrals are also computed using the trapezoidal rule. Denote by $\hat{Q}_{ij}^{(0)}$ the discretized mass function. Then

$$\hat{Q}_{ij}^{(0)} = \frac{h^2}{4}\Big(\mu(x_i, y_j) + \mu(x_i, y_{j+1}) + \mu(x_{i+1}, y_j) + \mu(x_{i+1}, y_{j+1})\Big)$$

and $Q_{ij}^{(0)} = \hat{Q}_{ij}^{(0)} + \mathcal{O}(h^4)$ (see Appendix C). Therefore we have that

$$\Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}^{(0)} - \underbrace{\sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, \hat{Q}_{kl}^{(0)}}_{=\hat{Q}_{ij}^{(1)}} \Big| \leq \mathcal{O}(h^4).$$

4.2.4. Error summary for the first iteration. For the first iteration, the error obtained by discretizing the Markov operator is as follows:

$$\begin{aligned}
\Big|Q_{ij}^{(1)} - \hat{Q}_{ij}^{(1)}\Big| &\leq \underbrace{\Big|Q_{ij}^{(1)} - \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl}^{(0)}\Big|}_{=\mathcal{O}(h^5)} \\
&+ \underbrace{\Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl}^{(0)} - \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}^{(0)}\Big|}_{=\mathcal{O}(h^3)} \\
&+ \underbrace{\Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}^{(0)} - \hat{Q}_{ij}^{(1)}\Big|}_{=\mathcal{O}(h^4)} \\
&= \mathcal{O}(h^3).
\end{aligned}$$

After one iteration we retain third order accuracy.

4.3. Error analysis for *m* iterations. Assume that after m iterations of the numerical scheme, the input density has concentrated on the invariant sets. Computationally, we find m to be finite, and in fact $\mathcal{O}(N)$. See Appendix B for more information on the number of iterations until convergence. The error after m iterations is bounded as follows:

$$\begin{aligned}
\Big|Q_{ij}^{(m)} - \hat{Q}_{ij}^{(m)}\Big| &\leq \underbrace{\Big|Q_{ij}^{(m)} - \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl}^{(m-1)}\Big|}_{=:E_1} \\
&+ \underbrace{\Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} K_{ijkl}\, Q_{kl}^{(m-1)} - \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}^{(m-1)}\Big|}_{=:E_2} \\
&+ \underbrace{\Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl}\, Q_{kl}^{(m-1)} - \hat{Q}_{ij}^{(m)}\Big|}_{E_3}.
\end{aligned}$$

The error E_1 comes from assuming the kernel is piecewise constant in (u, v). As in (4.3),

$$E_1 = \mathcal{O}(h) \int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} \int_{x_{i-c}}^{x_{i+c}} \int_{y_{j-c}}^{y_{j+c}} P^{m-1}[\mu] \, dv \, du \, dy \, dx.$$

However, $P^{m-1}[\mu]$ should be almost concentrated along the generators. In the case of point generators, the maximum of the iterated density should be $\mathcal{O}(N^2)$, as the mass should be concentrated on n grid boxes of area h^2 (recall that n is the number of generators, which is independent of h). In the case of curved generators, the iterated density should be $\mathcal{O}(N)$, as for each generator, the mass should be concentrated on $\mathcal{O}(N)$ grid cells of area h^2. Therefore,

$$\int_{x_{i-c}}^{x_{i+c}} \int_{y_{j-c}}^{y_{j+c}} P^{m-1}[\mu] \, dv \, du \le \begin{cases} \mathcal{O}(h) & \text{for curved generators,} \\ \mathcal{O}(1) & \text{for point generators.} \end{cases}$$

And in particular,

$$E_1 \le \begin{cases} \mathcal{O}(h^4) & \text{for curved generators,} \\ \mathcal{O}(h^3) & \text{for point generators.} \end{cases}$$

The second error comes from discretizing the kernel using the trapezoidal rule. This calculation is also analogous to the error in the first iteration, adjusting for the concentrated nature of $P^{m-1}[\mu]$. As in (4.9),

$$E_2 \le \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \frac{\overbrace{|I_1 - \text{Trap}_1|}^{=\mathcal{O}(h^3)} \overbrace{|\text{Trap}_2|}^{=\mathcal{O}(h^2)} + \overbrace{|\text{Trap}_1|}^{=\mathcal{O}(h^2)} \overbrace{|I_2 - \text{Trap}_2|}^{=\mathcal{O}(h^3)}}{\underbrace{|\text{Trap}_2 \, I_2|}_{=Ch^4}} \underbrace{Q_{kl}}_{\mathcal{O}(h) \text{ or } \mathcal{O}(1)}$$
$$\le \begin{cases} \mathcal{O}(h^2) & \text{for curved generators,} \\ \mathcal{O}(h) & \text{for point generators.} \end{cases}$$

The third error is as follows:

$$\begin{aligned} E_3 &= \Big| \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl} \, Q_{kl}^{(m-1)} - \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \hat{K}_{ijkl} \hat{Q}_{kl}^{(m-1)} \Big| \\ &\le \sum_{k=i-c}^{i+c-1} \sum_{l=j-c}^{j+c-1} \left| \hat{K}_{ijkl} \right| \left| Q_{kl}^{(m-1)} - \hat{Q}_{kl}^{(m-1)} \right| . \end{aligned}$$

This error is the same order as the error of the previous iteration. Since that error will always be dominated by the error from integrating the kernel using the trapezoidal rule, in the final step, this error will be

$$E_3 \le \begin{cases} \mathcal{O}(h^2) & \text{for curved generators,} \\ \mathcal{O}(h^1) & \text{for point generators.} \end{cases}$$

4.3.1. Error summary for *m* iterations. After m iterations, when the mass has concentrated on the generators, the error is

$$\left| Q_{ij}^{(m)} - \hat{Q}_{ij}^{(m)} \right| \le \begin{cases} \mathcal{O}(h^2) & \text{for curved generators,} \\ \mathcal{O}(h^1) & \text{for point generators.} \end{cases}$$

4.4. Obtaining the measures. Given $\hat{\mathbf{Q}}^{(m)}$ and the indices corresponding to each generator, it only remains to sum the values of $\hat{\mathbf{Q}}^{(m)}$ along each generator to obtain the approximated weights. Define $\mathcal{I}_i := \{(j,k) \mid d((x_j, y_k), \Gamma_i) < h\}$. These are the computational points which lie closest to the generator Γ_i. Then $\tilde{w}_i = \sum_{(j,k)\in\mathcal{I}_i} \hat{Q}_{jk}^{(m)}$. Let the union of these sets be denoted $\mathcal{I} = \cup_{i=1}^n \mathcal{I}_i$.

4.4.1. Measures associated with curves. In the case of curved generators, this summation will be over $\mathcal{O}(N)$ points, so an order of accuracy will be lost. The error we obtain by discretizing the Markov operator is

$$\begin{aligned}\left|\tilde{w}_i - \sum_{(j,k)\in\mathcal{I}} \hat{Q}_{jk}^{(m)}\right| &\leq \sum_{(j,k)\in\mathcal{I}} |Q_{jk}^{(m)} - \hat{Q}_{jk}^{(m)}| \\ &= \mathcal{O}(N)\mathcal{O}(h^2) \\ &= \mathcal{O}(h).\end{aligned}$$

Finally, recall that the true measure was denoted w_i. The error for curves between the true measure and that obtained by numerically iterating the Markov operator is first order:

$$\begin{aligned}\left|w_i - \sum_{(j,k)\in\mathcal{I}_i} \hat{Q}_{jk}^{(m)}\right| &\leq \underbrace{|w_i - \tilde{w}_i|}_{=\mathcal{O}(h)} + \underbrace{\left|\tilde{w}_i - \sum_{(j,k)\in\mathcal{I}} \hat{Q}_{jk}^{(m)}\right|}_{\leq\mathcal{O}(h)} \\ &= \mathcal{O}(h).\end{aligned}$$

Therefore the method is first order for curved generators.

4.4.2. Measures for points. In the case of point generators, $\mathcal{I}_i$ consists of one or at most four gridpoints, which is a size $\mathcal{O}(1)$ set. Then

$$\left|\tilde{w}_i - \sum_{(j,k)\in\mathcal{I}_i} \hat{Q}_{jk}^{(m)}\right| \leq 4 \max_{(j,k)\in\mathcal{I}_i} |Q_{jk}^{(m)} - \hat{Q}_{jk}^{(m)}| \leq \mathcal{O}(h).$$

Moreover,

$$\left|w_i - \sum_{(j,k)\in\mathcal{I}_i} \hat{Q}_{jk}^{(m)}\right| \leq \underbrace{|w_i - \tilde{w}_i|}_{\leq\mathcal{O}(h^2)} + \underbrace{\left|\tilde{w}_i - \sum_{(j,k)\in\mathcal{I}_i} \hat{Q}_{jk}^{(m)}\right|}_{\leq\mathcal{O}(h)} \leq \mathcal{O}(h).$$

Therefore the numerical method is first order for point generators as well. These error rates are demonstrated in the next section on numerical results.

Remark. By the same reasoning, it is clear that this scheme is first order accurate in any spatial dimension for generators of any codimension. So far in our discussion on errors, we have not addressed the error induced by solving the eikonal equation numerically. The error of the numerical scheme determines the error in the placement of the Voronoi diagram. A first order numerical scheme should therefore result in a first order error for the area of the generalized Voronoi regions, which does not degrade the error rate of our scheme. The eikonal equation can be solved numerically via the fast sweeping method or fast marching method (see [47, 52, 53, 5, 43, 44, 37]; see also [46, 20]).

4.5. Algorithm. Implementing this iterative scheme to compute Voronoi measures involves five steps. The first is to solve the eikonal equation numerically, and for that we use the fast sweeping method [52]. In the case of points and circles, the eikonal solution is the minimum of conical surfaces, which can be computed exactly (see Appendix A). Next, the probability transition tensor $\hat{\mathbf{K}}$ must be populated using the trapezoidal discretization. Third, the input density must be integrated numerically (via the trapezoidal rule, exactly, or by any higher order quadrature scheme). Fourth, the scheme must be iterated until the mass is accumulated on the generators. For this, iteration is terminated after the mass outside the set $\mathcal{I}$ (all gridpoints in an h-neighborhood of the generators) is less than some tolerance level. The tolerance should be less than the accuracy divided by the number of gridpoints in $\mathcal{I}$. For an idea of the accuracy, see Figure 5(d). Finally, the measures are calculated by summing the mass along the gridpoints closest to each generator. These five steps are summarized in Algorithm 2.

Algorithm 2 Calculate approximate measures.

```
**Given:** a bounded domain $\Omega \subset \mathbf{R}^d$, a density $\mu$, and generators $\{\Gamma_i\}_{i=1}^n$.
**Set** TOL, $h$.
Find gridpoints $\mathcal{I}_i := \{(j_1, \dots, j_d) \mid d((x_{j_1}, \dots, x_{j_d}), \Gamma_i) < h\}$.
1. **Solve an eikonal equation:**
   a. Initialize $\phi(\mathcal{I})$, by interpolation, and set $\phi(\Omega_h \setminus \mathcal{I}) = \infty$.
   b. Solve eikonal equation for $\phi$ on $\Omega_h$ (equation (3.2)).
   c. Obtain the function $q$ on $\Omega_h$ (equation (3.4)).
2. **Compute probability transition tensor $\hat{\mathbf{K}}$** from $q$ (equation (4.5)).
3. **Integrate** $\mu$ along grid cells to obtain $\hat{\mathbf{Q}}$ (equation (4.11)).
4. **Accumulate mass to $\mathcal{I}$:**
**while** $\|\hat{\mathbf{Q}}\|_{L^1(\mathcal{I})} < 1-$ TOL **do**
   $\hat{\mathbf{Q}} \leftarrow \hat{\mathbf{K}}\hat{\mathbf{Q}}$
**end while**
5. **Obtain the measures:**
**for** $i = 1 : n$ **do**
   $\tilde{w}_i = \sum_{(j_1,\dots,j_d)\in\mathcal{I}_i} \hat{Q}_{j_1,\dots,j_d}$.
**end for**
```

4.6. Algorithm complexity. The algorithm that has been developed here to evaluate integrals over generalized Voronoi regions is a convergent algorithm that circumvents explicitly identifying the typically nonconvex generalized Voronoi regions. The efficiency of the algorithm is in fact *independent of the number of generators* (though h must be small enough for the asymptotic regime to be reached).

As for the algorithm's complexity, Appendix B contains detailed numerical results showing the computational time required to populate the discretized kernel, and the number of iterations until convergence for the Voronoi and generalized Voronoi cases. It is useful to emphasize that the basic idea behind evaluating these integrals is the same regardless of dimension. Recall the procedure: Once the operator has been iterated long enough (see criteria in Algorithm 2), the mass has accumulated on gridpoints that lie in a neighborhood of generators. It is trivial to find these points, so one only needs to perform a straight sum of these values. The task in three dimensions is the same as that in two dimensions, with the difference being that the gridpoints surrounding the generators live on a cubic lattice, and so on for higher dimensions. The overall complexity is then controlled primarily by the matrix-vector multiplication required to iterate the discretized Markov operator. As demonstrated in Appendix B, for N gridpoints in each of d spatial dimensions, the complexity of the matrix-vector multiplication is $\mathcal{O}(N^d)$, and $\mathcal{O}(N)$ iterations are required for convergence, yielding an overall complexity of $\mathcal{O}(N^{d+1})$.

5. Numerical results.

In this section we first provide numerical verification that the numerical scheme is first order for the case of points, curves, and surfaces. In each case, the density function is identically 1. For the case of curves and surfaces, we use the calculations in Appendix A to compute these measures exactly. We then give an application of the method with nonuniform density to the Los Angeles County highway system. The method computes the fraction of population that lives closest to each highway. Then we show moment computations in two and three spatial dimensions, and we conclude with two examples of CVT, one in $\mathbb{R}^2$ for circular generators, and one in $\mathbb{R}^3$ for spherical generators.

5.1. Error.

Below we present the error rates for the numerical scheme in the case of point, circular, and spherical generators.

The point generators are the white dots shown in Figure 5(a). We include the standard Voronoi case (point generators) to complement the theoretical error rates previously presented. The circular generators are the solid white lines in Figure 5(c), and the spherical generators are the spheres in Figure 5(e). We use circular and spherical generators to demonstrate the theoretical error rate in a case where the generalized Voronoi diagram can be found and the measure of each region explicitly calculated. The Voronoi diagram for the circular and spherical generators was calculated using the formulas of Appendix A. The scheme results in first order convergence for both points, circles, and spheres.

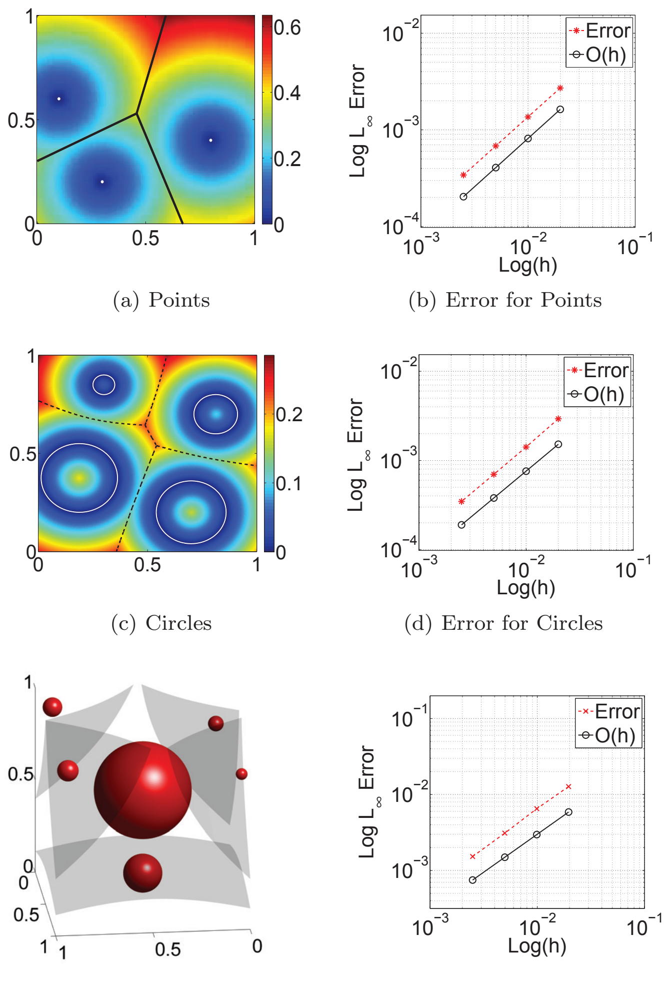
5.2. Generalized Voronoi example in $\mathbb{R}^2$ with nonuniform density: The Los Angeles County Highway System.

Next we compute the population influences (measures) of the highways in Los Angeles (L.A.) County. The influence describes the fraction of population living closest to each highway in L.A. County, and hence μ is simply the population density of L.A. The population density data and influences are depicted in Figures 6(a) and 6(b).

The map tiles are from [45, 38]. The highway data and geographic boundary data for L.A. County zip codes are from the U.S. Census Bureau [49] and is accurate as of January 1, 2010. The population density for each L.A. county zip code is also from the U.S. Census Bureau [48]. The highway data came in the form of latitude and longitude coordinates. These coordinates were used as the initial contour for solving the eikonal equation. The solution was obtained in a square domain, where the population density was set to zero outside L.A. County; the grid size was $h = 0.005$. From a purely qualitative perspective, the results are intuitive: there are more people living near the larger interstate freeways, and fewer people living in the domain of influence of the state highways. The numerical results are in Table 2.

5.3. Moments of generalized Voronoi regions in $\mathbb{R}^2$ and $\mathbb{R}^3$.

To demonstrate the algorithm further, we present examples in $\mathbb{R}^2$ and $\mathbb{R}^3$ where the generators are nonconvex, are nonsymmetric, and have C^0 boundaries. Let m_0 denote the area or volume of each region, and let m_1 denote the center of mass. In the following figures (Figures 7, 8, and 9), the dots represent the center of mass (m_1) of their corresponding Voronoi region. The underlying contour plot is the minimum distance to the generators.



(a) Points (b) Error for Points

(c) Circles (d) Error for Circles

(e) Spheres (f) Error for Spheres

FIG. 5. *Two- and three-dimensional convergence plots.*

From this, it is possible to infer information about each generalized Voronoi region. The example in Figure 7 shows overlapping generators. This, coupled with the nonconvexity of the shapes, creates generalized Voronoi regions that are quite complicated. The color scale of the background distance plot highlights the shape of the generalized Voronoi regions in the most crucial regions, while sacrificing resolution in the corners of the domain.

The areas (zeroth moment) of each generalized Voronoi region along with the center of mass (the first moments) are presented in Table 3; these correspond to the shapes in Figure 7. The grid size used was $h = 0.0033$. Here, we present only the zeroth and first moments, because they are more easily verified in the figures; however, there is no obstacle to computing any higher moment.

In Figure 8, we present an example where the centers of mass of the *generalized Voronoi regions* are located at the center of mass of each shape. The moments for these regions are presented in Table 3. The grid size used here was $h = 0.0033$.

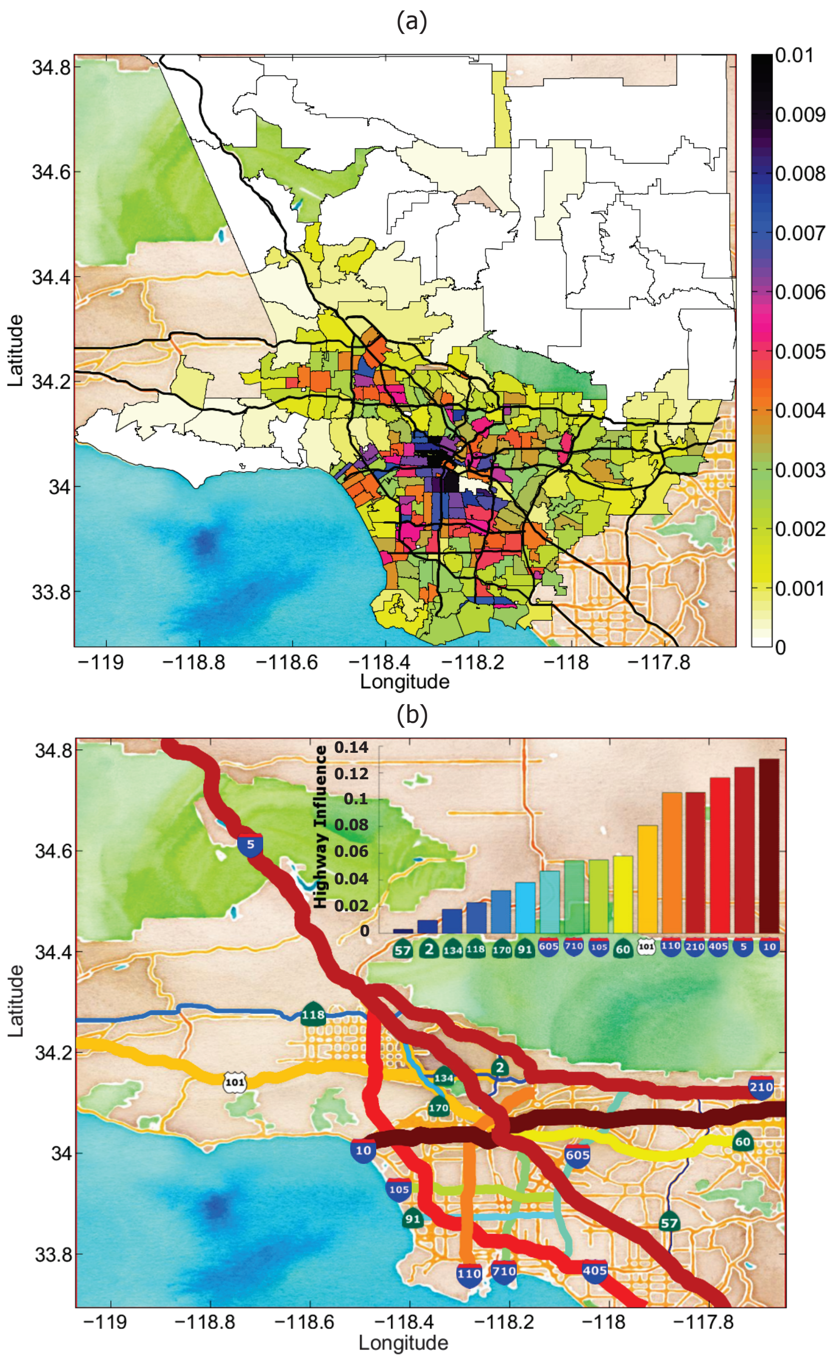


FIG. 6. (a) *Population density in L.A. County.* (b) *Fraction of L.A. County population living closest to each highway.*

TABLE 2
Percent of population living closest to major L.A. highways.

Highway name	Influence	Highway name	Influence
I-10	13.05%	I-710	5.46%
I-5	12.41%	I-605	4.67%
I-405	11.64%	State Rte. 91	3.81%
I-210	10.53%	State Rte. 170	3.21%
I-110	10.52%	State Rte. 118	2.30%
U.S. 101	8.07%	State Rte. 134	1.80%
State Rte. 60	5.78%	State Rte. 2	0.97%
I-105	5.49%	State Rte. 57	0.30%

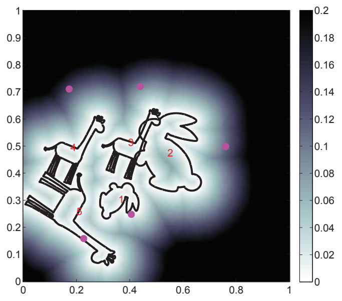


FIG. 7. *Overlapping* Γ_i.

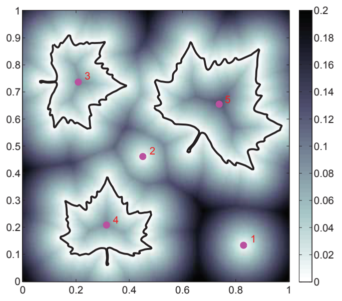


FIG. 8. *Nonconvex* Γ_i.

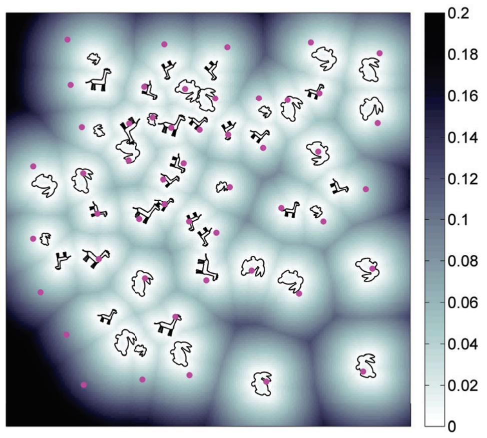


FIG. 9. *Integration example in* $\mathbb{R}^2$ *with* 50 *generators.*

In Figure 9, we present an example with 50 generators in $\Omega = [0, 1]^2$. The centers of mass are displayed, and the distance function is displayed in the background as a contour plot. The grid size was $h = 0.005$.

In $\mathbb{R}^3$, an example of the volume integration in the case of spheres was already presented (see Figure 5(e)). Another example with nonspherical generators is presented in Figures 10 and 11. In these examples, the grid size was $h = 0.005$, and the domain was $\Omega = [0, 1]^3$.

TABLE 3
Areas and centers of mass for Figures 7 and 8.

	Moments for Figure 7		Moments for Figure 8	
Gen.	m_0	m_1	m_0	m_1
1	0.0556	$(0.4060, 0.2476)$	0.0908	$(0.8295, 0.1334)$
2	0.4677	$(0.7607, 0.4983)$	0.0293	$(0.4508, 0.4614)$
3	0.1156	$(0.4396, 0.7205)$	0.2329	$(0.2083, 0.7365)$
4	0.2042	$(0.1724, 0.7106)$	0.2705	$(0.3144, 0.2079)$
5	0.1569	$(0.2271, 0.1582)$	0.3765	$(0.7381, 0.6550)$

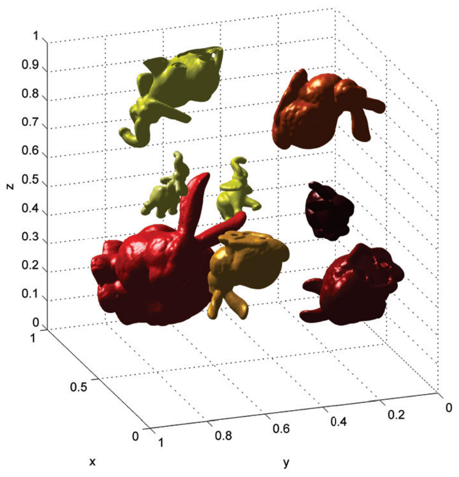
This example uses five bunnies of different sizes and three elephants. The generalized Voronoi regions generated by these shapes were produced using the direct method (see Algorithm 1) for illustration purposes.

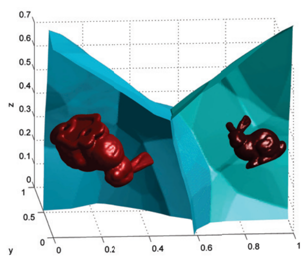
5.4. Application: Three-dimensional CVT of spheres. To extend the notion of a CVT from the point-generator case, one must find a suitable energy to minimize. In fact, we have already shown the form of this energy in (1.2). For an elegant survey of algorithms to compute CVTs for points, see [11, 12]. Included in these references, one will find many useful algorithms, including Lloyd's method [32], probabilistic algorithms [34, 21], a Newton–Lloyd method [10], multigrid methods [9], as well as quasi-Newton algorithms [31] that offer superlinear convergence to a minimum of the energy.

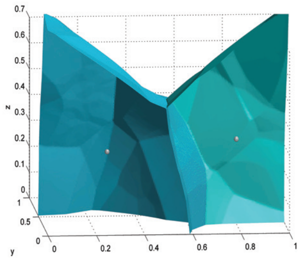
Historically, the problem of computing CVTs of points has been approached from both implicit and explicit algorithmic perspectives. On the explicit side, the Voronoi region of the point generators is computed exactly and used directly [32, 33]. However, there are also very efficient *implicit* algorithms to compute CVTs, where no computation of the Voronoi regions is needed [21]. In this spirit, our current algorithm can be likened to the latter algorithms, because the explicit calculation of the generalized Voronoi regions is not required. Interestingly, the method described here is also very much an extension of the Lloyd and quasi-Newton minimization algorithms for points [32, 33]—explicit methods. To accommodate a wide class of generators, the exact computation of the generalized Voronoi regions is avoided. However, for some of the curve primitives mentioned in the introduction, very accurate algorithms exist for computing the generalized Voronoi diagram. In the case that CVT should be required for such generators, the trade-off between the accuracy of using explicit Voronoi regions and the computational efficiency and flexibility obtained through the implicit approach can be studied in detail. This is left for future study.

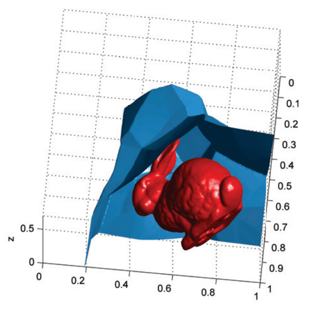
The primary focus of this paper is for more general generators, and there has been recent work in this direction. One extension of Lloyd's algorithm in $\mathbb{R}^2$ for points was in the area of nonphotorealistic rendering [17]. The authors modified Lloyd's method to allow for the rotation of lines and polygons, and used the method to generate stippled drawings. Their algorithm first projects the center of the shape to the center of mass of the Voronoi region generated by the shape (analogous to the original Lloyd's method). Then the angle of inertia of the shape is rotated to match the angle of inertia of the Voronoi region. This algorithm computes the zeroth, first, and second moments of the Voronoi regions via a discrete summation over pixels using the algorithm of [18]. The method gives an approximate CVT for shapes, although the variational foundations of the algorithm were not explored.

Another algorithm to compute the CVT of line segments (and graphs) was pro-


(a) Three-dimensional integration example


(b) Γ_1 & Γ_2


(c) m_1 for Γ_1 & Γ_2


(d) Γ_3

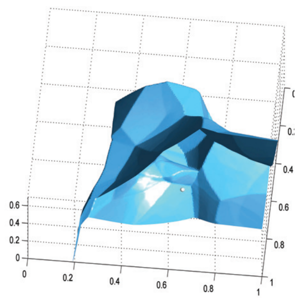
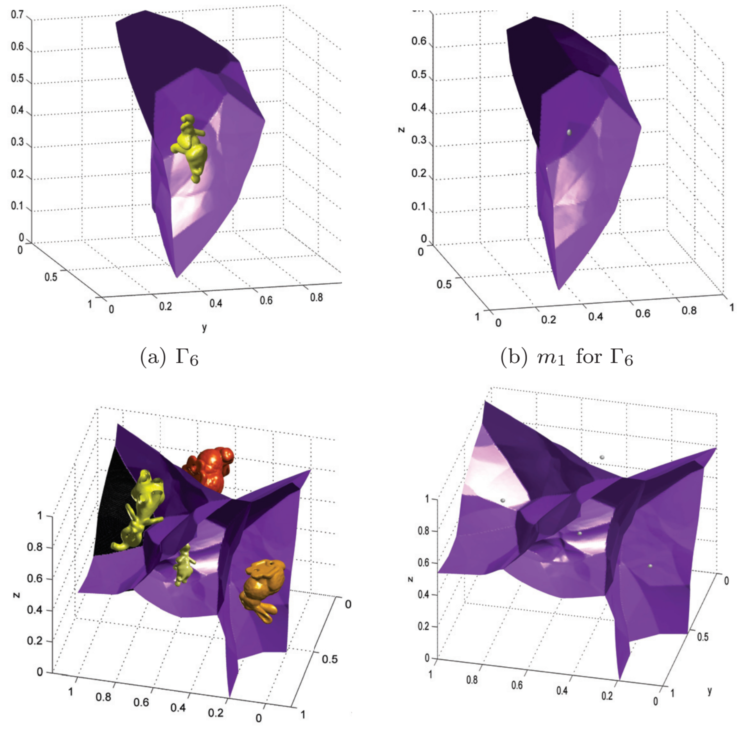

(e) m_1 for Γ_3

FIG. 10. *Three-dimensional generators for the integration example, part* I.

posed in [33]. Here, the line segments are not constrained to have a fixed length; instead, both endpoints of each line segment are allowed to move to minimize the CVT energy (with a regularization term). The energy is simplified by approximating each line by a sequence of points and using the point-based CVT energy as an approximation. This gives a good approximation for a dense point sampling of each segment, and reduces the integration to a summation of integrals over polygons. Our example is more closely related to previous work in nonphotorealistic rendering [17], as the shapes we optimize are not deformed as a result of the energy minimization. In



(a) Γ_6 (b) m_1 for Γ_6

(c) $\Gamma_4, \Gamma_5, \Gamma_7, \Gamma_8$ (d) m_1 for $\Gamma_4, \Gamma_5, \Gamma_7$, & Γ_8

FIG. 11. *Three-dimensional integration example, part* II.

the upcoming example, we can minimize the CVT energy for spheres directly, without using a point-based approximation of the energy.

5.4.1. CVT of spheres. We now describe the CVT framework to optimize the placement of spheres. Assume we are working in a convex domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$. Moreover, we are given a fixed set of n shapes $\{\Gamma_i\}_{i=1}^n$ which depend on the sphere center $\mathbf{x}_i$ (spheres are rotationally invariant, so there is no angle to account for). Let $\mathbf{X}$ denote the $(dn \times 1)$-dimensional vector of location coordinates. Each sphere is parametrized by a radius, r_i. Let μ be a density in $L^1(\Omega)$.

To obtain a *CVT of spheres* $\{\Gamma_i\}_{i=1}^n$, one must find the locations $\mathbf{X}$ that minimize the following energy:

$$F(\mathbf{X}) = \sum_{i=1}^{n} \int_{V_i(\mathbf{X})} \operatorname{dist}^2(\mathbf{y}, \Gamma_i)\, \mu(\mathbf{y})\, d\mathbf{y}. \tag{5.1}$$

The Voronoi region V_i depends on the location of Γ_i *and* all neighbors of Γ_i. We emphasize this by writing $V_i(\mathbf{X})$.

5.4.2. Energy gradient. To minimize the CVT energy (equation (5.1)), we use a quasi-Newton method [35, 30]. Although it is possible to write down the second derivatives of the energy function F, they include boundary integrals along the curved Voronoi diagram generated by $\{\Gamma_i\}_{i=1}^n$. By using a quasi-Newton method, we only need to compute first order derivatives. The L-BFGS algorithm is guaranteed to converge when the energy F is twice continuously differentiable in a neighborhood of the minimizer, though it is known to converge even in cases where F is less regular [29]. An analysis of the smoothness of the energy for spheres is beyond the scope of this example.

To implement the quasi-Newton algorithm, we need to compute the first derivatives of the energy F with respect to the locations, $\mathbf{X}$. For each term in the summation that comprises F, the domain of integration as well as the integrand depend on these locations. The derivative of an integral with respect to its domain of integration is given in Lemma 6.1 of [11], and it can be shown that the first derivative of F with respect to $x_i^{(k)}$ (the kth component of the location vector $\mathbf{x}_i$) is

$$\frac{\partial F}{\partial x_i^{(k)}} = \int_{V_i} \frac{\partial}{\partial x_i^{(k)}} \text{dist}^2(\mathbf{y}, \Gamma_i)\, \mu(\mathbf{y})\, d\mathbf{y}. \tag{5.2}$$

The manner in which the boundary integrals vanish in the case of rigid bodies is analogous to the case of points. See [19] for the point-based energy derivative calculations.

To calculate the energy derivatives, the derivatives of the squared distance function $\text{dist}^2(\mathbf{y}, \Gamma_i)$ must be computed. The distance from any point $\mathbf{y} \in \Omega$ to a sphere Γ_i is given by $\text{dist}(\mathbf{y}, \Gamma_i) = \big||\mathbf{y} - \mathbf{x}_i| - r_i\big|$, where $\mathbf{x}_i$ is the center of the sphere, and r_i its radius. This distance function can be used to obtain explicit equations for the energy gradient with respect to the sphere's center, $\mathbf{x}_i$:

$$\frac{\partial F}{\partial x_i^{(k)}} = \int_{V_i} 2\Big(\mathbf{x}_i^{(k)} - \mathbf{y}^{(k)}\Big)\Big(1 - \frac{r_i}{|\mathbf{y} - \mathbf{x}_i|}\Big)\mu(\mathbf{y})\, d\mathbf{y}.$$

When the distance is not available explicitly, one must approximate the integrand numerically by appropriate derivatives of eikonal solutions. This requires additional analysis and will be presented in a separate paper on the CVT of rigid shapes in $\mathbb{R}^3$.

5.4.3. Numerical examples.

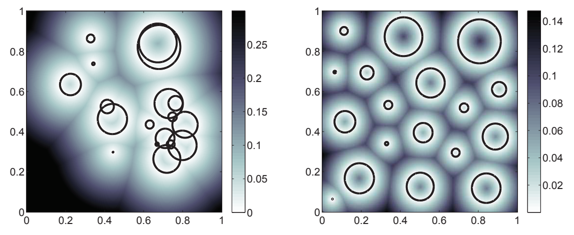
The numerical simulations presented here use a constant density, $\mu = 1$. Starting with the configuration of circles in Figure 12(a), L-BFGS was run with $M = 20$ for 30 iterations. Notice that this initial configuration has overlapping generators. This type of configuration makes for complicated Voronoi regions, but the integration remains simple using the method in this paper.

In each step of the optimization, a unit step length was tested first before the line search algorithm was run. After 30 iterations, the configuration of circles became distributed as in Figure 12(b). The energy and L^2 norm of the energy gradient per iteration have been plotted in Figures 12(c) and 12(d).

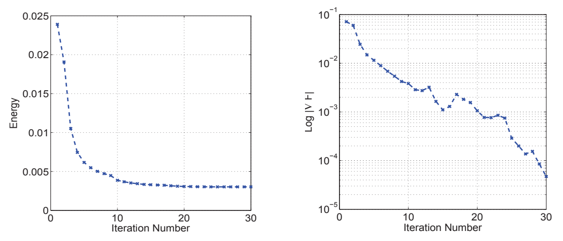
Similarly, in three dimensions, we began with the configuration of spheres in Figure 13. After 30 L-BFGS iterations, the spheres were distributed as in Figure 14. The beginning energy was 0.0424, and the final energy was 0.0144.

6. Summary.

We have presented an efficient numerical scheme to compute the measures of generalized Voronoi regions. The scheme is first order accurate and can deal with generators of arbitrary codimension. This algorithm computes a fundamental geometric quantity. The utility of the scheme was demonstrated on applications to urban planning and the CVT of spheres of different radii.



(a) Initial (b) 30 Iterations



(c) Energy (d) $|\nabla F|$

Fig. 12. *CVT of two-dimensional circles.*

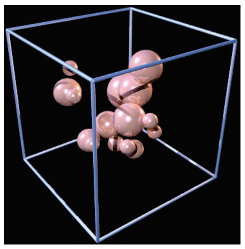


Fig. 13. *Initial configuration.*

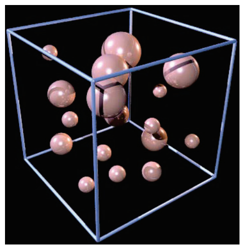


Fig. 14. *After* 30 *iterations.*

Appendix A. Example of analytically available region boundaries. It is worth noting that in some cases the formulas for the boundaries of the generalized Voronoi regions are analytically available. In this example we consider circular generators (see also [24, 25, 22]).

In the case of nonoverlapping circular generators in two dimensions, we can derive an explicit formula for $\phi_{ij} = 0$. In this case,

$$\phi_i(x,y) = \sqrt{(x-x_i)^2 + (y-y_i)^2} - r_i.$$

Solving for $\phi_{ij} = 0$ yields

$$y = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}, \tag{A.1}$$

where $A, B,$ and C are defined as

$$A = \left(\frac{y_j - y_i}{r_i - r_j}\right)^2 - 1,$$

$$B = \frac{(y_j - y_i)}{(r_i - r_j)^2}\left(y_i^2 - y_j^2 + (x - x_i)^2 - (x - x_j)^2\right) + y_i + y_j,$$

$$C = \left(\frac{y_i^2 - y_j^2 + (x - x_i)^2 - (x - x_j)^2}{2(r_i - r_j)} - \frac{(r_i - r_j)}{2}\right)^2 - (x - x_j)^2 - y_j^2.$$

In (A.1), the positive or negative root is chosen such that $\phi_i(x, y) = \phi_j(x, y)$. If there is not one unique y value for each x, one should use the analogous expression for x as a function of y. This equation was used to find the Voronoi diagram in Figure 5(c). There, the generators are shown in solid white and the region boundaries are shown in dashed black lines. The contour plot of the distance function is in the background.

In three dimensions, we have $\phi_i(x, y, z) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - r_i$. Solving for $\phi_{ij} = 0$ yields

$$z = \frac{(c_j^2 - c_i^2)(z_i - z_j) + K(z_i + z_j) \pm \sqrt{(r_i - r_j)^2\left((c_i^2 - c_j^2 + K)^2 - 4Kc_i^2\right)}}{2K},$$

where

$$c_j = \sqrt{(x - x_j)^2 + (y - y_j)^2},$$
$$K = (r_i - r_j + z_i - z_j)(r_i - r_j - z_i + z_j).$$

The equation $\phi_{ij} = 0$ can also be solved in terms of x or y. The curves that are equidistant to three spheres can be found analytically as the solution of a quartic polynomial.

Appendix B. CPU times.

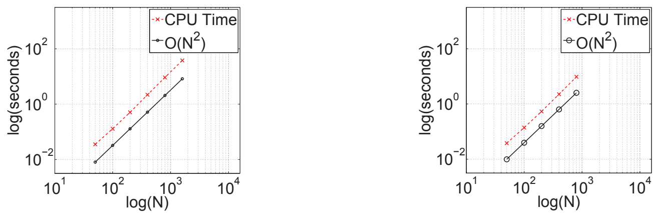
In this section, we report the CPU times for the two-dimensional integration algorithm. In the following sections, N refers to the number of gridpoints in *each* spatial dimension, so the total degrees of freedom here is N^2. These tests were run in MATLAB on a 3.0 GHz quad-core desktop with 8 GB of RAM.

B.1. Solving the eikonal equation.

The first step in our algorithm is to find the eikonal solution with the generators as the initial contours. This is done via a fast sweeping method, which is an $O(N^2)$ algorithm (see [52]).

B.2. Constructing the discretized kernel.

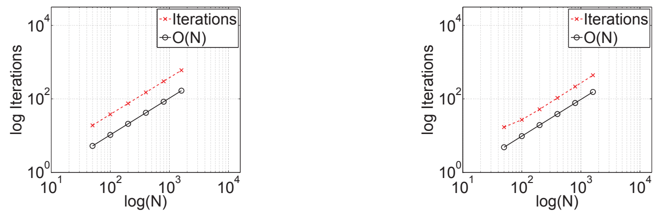
To construct the discretized kernel, we must populate a matrix with $O(N^2)$ nonzero elements. This takes $O(N^2)$ seconds, which is verified below in Figures 15(a) and 15(b).



(a) Points (b) Circles

FIG. 15. *CPU time for kernel population.*

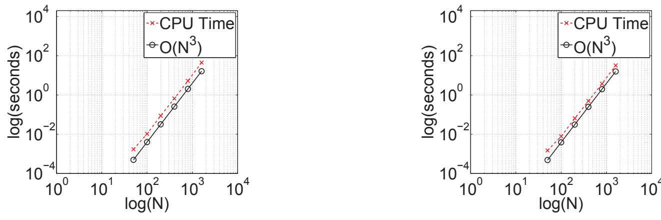
B.3. Number of iterations until convergence. We conjecture the number of iterations until the mass converges to be $O(N)$. Figures 16(a) and 16(b) confirm that the number of iterations until convergence is $O(N)$.



(a) Points (b) Circles

FIG. 16. *Number of iterations for operator convergence.*

B.4. CPU time for convergence to invariant sets. Each iteration of the discretized Markov operator requires the multiplication of a vector with a sparse $O(N^2)$ matrix.



(a) Points (b) Circles

FIG. 17. *CPU time for operator convergence.*

The CPU time required for this is $O(N^2)$. As $O(N)$ iterations are required to obtain convergence, we expect a CPU time that is $O(N^3)$. This is exactly what is observed in Figures 17(a) and 17(b).

Appendix C. Trapezoidal rule error. Let $f : \mathbb{R}^2 \to \mathbb{R}$ have two continuous partial derivatives in both x and y. Let $h = x_{i+1} - x_i = y_{j+1} - y_j$ for $i, j \in \{1, \ldots, N\}$.

Then by integration by parts

$$\begin{aligned}\int_0^h \int_0^h f(x_i+t, y_j+s)dt\,ds &= \underbrace{\frac{h^2}{4}(f(x_i,y_j)+f(x_i,y_{j+1})+f(x_{i+1},y_j)+f(x_{i+1},y_{j+1}))}_{=:\mathrm{Trap}_{ij}} \\ &+ \frac{h}{2}\int_0^h \Big(\frac{(s-\frac{h}{2})^2}{2} - \frac{h^2}{8}\Big) f_{yy}(x_{i+1}, y_j+s)ds \\ &+ \frac{h}{2}\int_0^h \Big(\frac{(s-\frac{h}{2})^2}{2} - \frac{h^2}{8}\Big) f_{yy}(x_i, y_j+s)ds \\ &+ \int_0^h\int_0^h \Big(\frac{(t-\frac{h}{2})^2}{2} - \frac{h^2}{8}\Big) f_{xx}(x_i+t, y_j+s)ds\,dt.\end{aligned}$$

Then

$$\max_{i,j\in\{1,\dots,N\}} \Big| \int_0^h\int_0^h f(x_i+t,y_j+s)dt\,ds - \mathrm{Trap}_{ij}\Big| \le \frac{h^4}{12}(\|f_{yy}\|_\infty + \|f_{xx}\|_\infty).$$

So for any function with two continuous partial derivatives, we expect the trapezoidal rule to yield fourth order accuracy. The stochastic kernel we define is only continuous on $\{(x,y)\,|\,|x-y|\le\varepsilon\}$ (this is precisely the domain of integration). In fact, there will be a gridpoint such that the left and right x derivatives (similarly for the y derivatives) will not agree. Numerically, we see that for some gridpoint (x_i, y_j)

$$\|f_{xx}\|_\infty = \Big|\frac{f_x(x_{i+1},y_j) - f_x(x_i,y_j)}{h}\Big| = \frac{c}{h},$$

and similarly for f_{yy}. Therefore in the case of the kernel we have defined, we expect the trapezoidal rule to yield $\mathcal{O}(h^3)$ convergence. Note that because the limits of integration depend on the grid size h, the above analysis is different from the standard trapezoidal rule, where the limits of integration are fixed, and the error depends upon the number of quadrature points.

Acknowledgments. The authors would like to thank participants at the 2013 Banff workshop on Partial Differential Equations in the Social and Life Sciences for their comments, and the editors and anonymous referees for their helpful suggestions.

REFERENCES

[1] H. Alt, O. Cheong, and A. Vigneron, *The Voronoi diagram of curved objects*, Discrete Comput. Geom., 34 (2005), pp. 439–453.

[2] H. Alt and O. Schwarzkopf, *The Voronoi diagram of curved objects*, in Proceedings of the Eleventh Annual Symposium on Computational Geometry, ACM, New York, 1995, pp. 89–97.

[3] F. Anton, D. Mioc, and M. Santos, *Exact computation of the topology and geometric invariants of the Voronoi diagram of spheres in* 3*D*, J. Comput. Sci. Tech., 28 (2013), pp. 255–266.

[4] F. Aurenhammer, *Voronoi diagrams—A survey of a fundamental geometric data structure*, ACM Comput. Surveys, 23 (1991), pp. 345–405.

[5] J.-D. Benamou, S. Luo, and H. Zhao, *A compact upwind second order scheme for the eikonal equation*, J. Comput. Math., 28 (2010), pp. 489–516.

[6] P. Billingsley, *Probability and Measure*, Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, NJ, 2012.

[7] J.D. Boissonnat, C. Wormser, and M. Yvinec, *Curved Voronoi diagrams*, in Effective Computational Geometry for Curves and Surfaces, Springer-Verlag, Berlin, 2007, pp. 67–116.

[8] J.J. Chou, *Voronoi diagrams for planar shapes*, IEEE Comput. Graphics Appl., 15 (1995), pp. 52–59.

[9] Z. Di, M. Emelianenko, and S.G. Nash, *Truncated Newton-based multigrid algorithm for centroidal Voronoi diagram calculation*, Numer. Math. Theory Methods Appl., 5 (2012), pp. 242–259.

[10] Q. Du and M. Emelianenko, *Acceleration schemes for computing centroidal Voronoi tessellations*, Numer. Linear Algebra Appl., 13 (2006), pp. 173–192.

[11] Q. Du, V. Faber, and M. Gunzburger, *Centroidal Voronoi tessellations: Applications and algorithms*, SIAM Rev., 41 (1999), pp. 637–676.

[12] Q. Du, M. Gunzburger, and L. Ju, *Advances in studies and applications of centroidal Voronoi tessellations*, Numer. Math. Theory Methods Appl., 3 (2010), pp. 119–142.

[13] G. Elber and M.-S. Kim, *Bisector curves of planar rational curves*, Computer-Aided Design, 30 (1998), pp. 1089–1096.

[14] I. Emiris, A. Mantzaflaris, and B. Mourrain, *Voronoi diagrams of algebraic distance fields*, Computer-Aided Design, 45 (2012), pp. 511–516.

[15] S. Fortune, *Voronoi Diagrams and Delaunay Triangulations in Computing in Euclidean Geometry*, World Scientific, River Edge, NJ, 1992.

[16] I. Hanniel and G. Elber, *Computing the Voronoi cells of planes, spheres and cylinders in* $R3$, Comput. Aided Geom. Design, 26 (2009), pp. 695–710.

[17] S. Hiller, H. Hellwig, and O. Deussen, *Beyond stippling—Methods for distributing objects on the plane*, Comput. Graphics Forum, 22 (2003), pp. 515–522.

[18] K.E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver, *Fast computation of generalized Voronoi diagrams using graphics hardware*, in Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, ACM Press/Addison-Wesley, New York, 1999, pp. 277–286.

[19] M. Iri, K. Murota, and T. Ohya, *A fast Voronoi-diagram algorithm with applications to geographical optimization problems*, in System Modelling and Optimization, P. Thoft-Christensen, ed., Lecture Notes in Control and Inform. Sci. 59, Springer-Verlag, Berlin, Heidelberg, 1984, pp. 273–288.

[20] M.W. Jones, J.A. Baerentzen, and M. Sramek, *3D distance fields: A survey of techniques and applications*, IEEE Trans. Visualization Comput. Graphics, 12 (2006), pp. 581–599.

[21] L. Ju, Q. Du, and M. Gunzburger, *Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations*, Parallel Comput., 28 (2002), pp. 1477–1500.

[22] D.-S. Kim, Y. Cho, and D. Kim, *Euclidean Voronoi diagram of 3D balls and its computation via tracing edges*, Computer-Aided Design, 37 (2005), pp. 1412–1424.

[23] D.-S. Kim, Y. Cho, D. Kim, S. Kim, J. Bhak, and S.-H. Lee, *Euclidean Voronoi diagrams of 3D spheres and applications to protein structure analysis*, Japan J. Indust. Appl. Math., 22 (2005), pp. 251–265.

[24] D.-S. Kim, D. Kim, and K. Sugihara, *Voronoi diagram of a circle set from Voronoi diagram of a point set:* I. *Topology*, Comput. Aided Geom. Design, 18 (2001), pp. 541–562.

[25] D.-S. Kim, D. Kim, and K. Sugihara, *Voronoi diagram of a circle set from Voronoi diagram of a point set:* II. *Geometry*, Comput. Aided Geom. Design, 18 (2001), pp. 563–585.

[26] R. Klein, *Concrete and Abstract Voronoi Diagrams*, Lecture Notes in Comput. Sci. 400, Springer-Verlag, Berlin, 1990.

[27] A. Lasota and M.C. Mackey, *Probabilistic Properties of Deterministic Systems*, Cambridge University Press, Cambridge, UK, 1985.

[28] A. Lasota and M.C. Mackey, *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*, Appl. Math. Sci. 97, Springer-Verlag, New York, 1994.

[29] A.S. Lewis and M.L. Overton, *Nonsmooth optimization via quasi-Newton methods*, Math. Program., 141 (2013), pp. 135–163.

[30] D.C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Programming, 45 (1989), pp. 503–528.

[31] Y. Liu, W. Wang, B. Lévy, F. Sun, D.M. Yan, L. Lu, and C. Yang, *On centroidal Voronoi tessellation—Energy smoothness and fast computation*, ACM Trans. Graphics, 28 (2009), 101.

[32] S. Lloyd, *Least squares quantization in PCM*, IEEE Trans. Inform. Theory, 28 (1982), pp. 129–137.

[33] L. Lu, B. Levy, and W. Wang, *Centroidal Voronoi tessellation of line segments and graphs*, Comput. Graphics Forum, 31 (2012), pp. 775–784.

[34] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, Berkeley, CA, 1967, pp. 281–297.

[35] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comp., 35 (1980), pp. 773–782.

[36] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons, Chichester, UK, 1992.

[37] S.J. Osher and R.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, 2002.

[38] OSM, *Openstreepmap*, http://openstreetmap.org, 2012.

[39] J. Ryu, R. Park, and D.-S. Kim, *Molecular surfaces on proteins via beta shapes*, Computer-Aided Design, 39 (2007), pp. 1042–1057.

[40] R.I. Saye and J.A. Sethian, *The Voronoi implicit interface method for computing multiphase physics*, Proc. Natl. Acad. Sci. USA, 108 (2011), pp. 19498–19503.

[41] R.I. Saye and J.A. Sethian, *Analysis and applications of the Voronoi implicit interface method*, J. Comput. Phys., 231 (2012), pp. 6051–6085.

[42] J.K. Seong, E. Cohen, and G. Elber, *Voronoi diagram computations for planar NURBS curves*, in Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, ACM Press/Addison-Wesley, New York, 2008, pp. 67–77.

[43] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595.

[44] J.A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.

[45] Stamen Maps, *Watercolor Map Tiles*, http://maps.stamen.com, 2012.

[46] Y.-H.R. Tsai, *Rapid and accurate computation of the distance function using grids*, J. Comput. Phys., 178 (2002), pp. 175–195.

[47] Y.-H.R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694.

[48] U.S. Census Bureau, *Profile of General Population and Housing Characteristics:* 2010 *Census Summary File*, 2010.

[49] U.S. Census Bureau, *Topologically Integrated Geographic Encoding and Referencing System*, 2010.

[50] C.K. Yap, *An O(n log n) algorithm for the Voronoi diagram of a set of simple curve segments*, Discrete Comput. Geom., 2 (1987), pp. 365–393.

[51] Z. Yuan, G. Rong, X. Guo, and W. Wang, *Generalized Voronoi diagram computation on GPU*, in Proceedings of the Eighth IEEE International Symposium on Voronoi Diagrams in Science and Engineering (ISVD), 2011, pp. 75–82.

[52] H. Zhao, *A fast sweeping method for Eikonal equations*, Math. Comp., 74 (2005), pp. 603–627.

[53] H. Zhao, *Parallel implementations of the fast sweeping method*, J. Comput. Math., 25 (2007), pp. 421–429.