Discrete & Computational Geometry (2019) 61:91–119
https://doi.org/10.1007/s00454-018-0037-6

Analyzing the Squared Distance-to-Measure Gradient Flow System with *k*-Order Voronoi Diagrams

Patrick O'Neil[1] · **Thomas Wanner**[1]

Received: 24 March 2017 / Revised: 4 March 2018 / Accepted: 3 August 2018 /
Published online: 17 October 2018

Abstract
Point cloud data arises naturally from 3-dimensional scanners, LiDAR sensors, and industrial computed tomography (i.e. CT scans) among other sources. Most point clouds obtained through experimental means exhibit some level of noise, inhibiting mesh reconstruction algorithms and topological data analysis techniques. To alleviate the problems caused by noise, smoothing algorithms are often employed as a preprocessing step. Moving least squares is one such technique, however, many of these techniques are designed to work on surfaces in $\mathbb{R}^3$. As interesting point clouds can naturally live in higher dimensions, we seek a method for smoothing higher dimensional point clouds. To this end, we turn to the distance to measure function. In this paper, we provide a theoretical foundation for studying the gradient flow induced by the squared distance to measure function, as introduced by Chazal, Cohen-Steiner, and Mérigot. In particular, we frame the gradient flow as a Filippov system and find a method for solving the squared distance to measure gradient flow, induced by the uniform empirical measure, using higher order Voronoi diagrams. In contrast to some existing techniques, this gradient flow provides a smoothing algorithm which computationally scales with dimensionality.

Keywords Distance-to-measure function · Point cloud · Data smoothing · Voronoi diagrams · Topological data analysis

Editor in Charge: Kenneth Clarkson

Patrick O'Neil
ponl@ponl.io

Thomas Wanner
twanner@gmu.edu

[1] Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, USA

1 Introduction

There has been substantial research effort recently (for example [8,18,20,22,27,31]) concerned with analyzing point clouds drawn from an underlying set M in $\mathbb{R}^N$. Primarily, the focus has been on topologically and/or geometrically faithful manifold reconstructions using a point cloud (i.e. a finite set of points in $\mathbb{R}^N$) sampled from the manifold. Examples of such techniques include the moving least squares method (see [3,26,28], and [2]), spectral methods (see [25,30]), and many more (see [14] for a thorough exposition). Since most applications utilizing point clouds involve a noisy data collection process, the sampled points must be assumed to lie near the underlying set, instead of directly on the set. Therefore, algorithms which are topologically and geometrically robust to varying levels of noise are required.

One approach to reducing noise is through the use of smoothing gradient flows, where we evolve a point cloud according to some specified gradient flow. In the continuous case, several examples of smoothing flows come to mind, including mean curvature flow [33], Willmore flow [5], and Ricci flow [29]. Such smoothing methods maintain the topology of the set while varying its geometry. When dealing with a point cloud, one could envision constructing an approximation of the mean curvature flow and applying such a flow to the point cloud.

In this paper, we study the squared distance to measure gradient flow and its application to point cloud denoising. The distance to measure function gives a notion of the distance between a point in $\mathbb{R}^N$ and a probability measure defined on $\mathbb{R}^N$. Through the use of this gradient flow, we aim to reduce the level of noise present in a point cloud sampled with noise. The distance to measure function and its connection to denoising has been studied extensively. For example, in [7], Buchet et al. use the distance to measure function to remove outliers and resample the data, producing a denoised point cloud. Other works use the distance to measure function directly to study the topology of a set via a point cloud sampled from the set. For example, in [11], Chazal et al. explore statistical properties of the distance to measure function and its ability to estimate the persistent homology of the set S, from which a point cloud has been sampled. Recently, the distance to measure function was used by Brécheteau in [6] to build a hypothesis test for rejecting whether two point cloud samples came from the same underlying metric-measure spaces.

In this work, our goal is to develop a theoretical foundation for the gradient flow defined by the square of the distance to measure function introduced by Chazal et al. in [9]. In particular, we investigate the behavior of this gradient flow when the distance to measure function is induced by the uniform empirical measure. The gradient flow evolves a point cloud so as to minimize the distance between the points in the cloud and an approximation of the underlying sampling distribution. This has the effect of smoothing the point cloud.

In order to provide a better understanding of the squared distance to measure gradient flow, we recognize it as a Filippov system, i.e., as a piece-wise smooth flow. Systems of this type have been widely studied (see for example [4,17,23], and [16]) and can lead to interesting dynamical phenomena. One of the main results of the present paper is the determination of the specific structure of the squared distance to measure gradient flow. We will show that on a set of full measure this flow is linear, and in fact given by motion along lines towards differing points. Moreover, in contrast to general Filippov systems, our considered smoothing flow does not exhibit attractive sliding. In general Filippov systems, such behavior leads to the merging of sets of orbits, and necessitates explaining dynamical behavior on subsets of the domain which are of co-dimension two or higher. In addition to simplifying the study of the flow, the lack of attractive sliding motion allows for a streamlined simulation of the flow. Finally, we give a detailed description of the overall qualitative features of the flow, such as a characterization of invariant sets, as well as a description of unstable separatrices along interfaces with repulsive sliding.

Existing approaches for characterizing the squared distance to measure gradient flow, such as [21], recognize the squared distance to measure function as being equivalent to finding the squared Euclidean distance to a set of points in $\mathbb{R}^{N+1}$. From this perspective, some of our results characterizing the gradient flow, such as Theorem 3.7 which proves trajectories cannot "slide" along certain co-dimension one interfaces, can also be seen using sets of paraboloids. However, unlike the approach in [21], we recognize the system as a Filippov system (see [19]) and build our results under the Filippov framework. Moreover, using the subsequent results, we are able to produce a directed graph which captures the motion of the points in the system.

In the next section, we will recall the definition of the distance to measure function and provide some results from [9] which give motivation for its use. In Sect. 3, we will define the gradient flow system induced by the squared distance to measure function and develop a theoretical framework to describe the dynamics of the system. In particular, we will show that the system defines a piecewise-smooth flow as defined in [16], and we will use higher order Voronoi diagrams to characterize the system. Finally, in Sect. 4, we will provide numerical results obtained by running the gradient flow on some simple geometries.

2 The Distance to Measure Function

Point clouds are often sampled from physical objects. For example, the surface of the Earth can be sampled using LiDAR, yielding a point cloud. Often, the point cloud is then used to construct a manifold which approximates the sampled object. Many manifold reconstruction techniques rely on the use of the distance function d_K to a set K defined by

$$d_K(y) = \inf_{x \in K} \|x - y\| \quad \text{for all} \quad y \in \mathbb{R}^N,$$

where $\|\cdot\|$ refers to the Euclidean distance. For example, given a point cloud $X \subset \mathbb{R}^N$, the Čech complex can be formed using a sublevel set of this function. In particular, one specifies a parameter $\varepsilon > 0$ and uses the sublevel set $d_X^{-1}(0, \varepsilon/2)$ to construct a simplicial complex. This amounts to constructing spheres of radius $\varepsilon/2$ centered around the points of X and building simplices out of intersecting spheres. Although the distance function is conceptually simple, it has its issues. Of primary concern for mesh reconstruction, the presence of a single outlying point causes the topology of

$d_X^{-1}(0, \varepsilon/2)$ to change. Therefore, it is clear that the distance function is not robust to noise in the data.

We turn to the distance to measure function, introduced by [9], for a distance like function which is robust to noise. The distance to measure function is built on the pseudo-distance to a probability measure, as defined below.

Definition 2.1 Let μ be a probability measure on $\mathbb{R}^N$. For $m \in [0, 1]$, the *pseudo-distance* to the measure μ is defined for all $x \in \mathbb{R}^N$ as

$$\delta_{\mu,m}(x) = \inf\left\{r > 0 : \mu(\overline{B}(x,r)) > m\right\},$$

where $\overline{B}(x, r) \subset \mathbb{R}^N$ denotes the closed ball of radius r centered at $x \in \mathbb{R}^N$.

Note that for $m = 0$, the definition of $\delta_{\mu,m}$ reduces to the distance function to the support of μ. We can now turn to the definition of the distance to measure function, the primary tool for inducing our gradient flow.

Definition 2.2 Let μ be a probability measure on $\mathbb{R}^N$ and let $m_0 \in (0, 1]$. The *distance to measure μ with parameter m_0* is the function $d_{\mu,m_0} \colon \mathbb{R}^N \to \mathbb{R}^+$ defined by

$$d_{\mu,m_0}(x) = \left(\frac{1}{m_0}\int_0^{m_0} \delta_{\mu,m}(x)^2 dm\right)^{1/2}.$$

Notice that like the pseudo-distance function, the distance to measure function gives a precise notion of the distance of a point to a probability measure μ. However, unlike the pseudo-distance function, the distance to measure function is continuous with respect to the measure μ and the parameter m_0 (see [9]).

As mentioned previously, one of the primary benefits of using the distance to measure function is that this function is robust to noise. To make this precise, we first recall that given two Radon probability measures μ and ν on $\mathbb{R}^N$, a transport plan between μ and ν is another Radon probability measure π on $\mathbb{R}^N \times \mathbb{R}^N$ such that for all $A, B \subseteq \mathbb{R}^N$, we have $\pi(A \times \mathbb{R}^N) = \mu(A)$ and $\pi(\mathbb{R}^N \times B) = \nu(B)$. With this, we can define a distance metric between two probability measures.

Definition 2.3 Given $p \geq 1$, the p-cost of a transport plan π is defined as

$$C_p(\pi) = \left(\int_{\mathbb{R}^N\times\mathbb{R}^N} \|x-y\|^p d\pi(x,y)\right)^{1/p}.$$

The *Wasserstein distance* (of order p), as originally introduced in [24], denoted $W_p(\mu, \nu)$, between two Radon probability measures μ and ν with finite p-moments is defined as

$$W_p(\mu,\nu) = \inf_{\pi} C_p(\pi),$$

where the infimum is taken over all transport plans π between μ and ν.

It can be shown that the Wasserstein distance is in fact a metric and turns the space of all Radon probability measures over $\mathbb{R}^N$, with finite p-moments, into a metric space. Returning to the distance to measure function, Chazal et al. also show the following in [10].

Proposition 2.4 *If μ and ν are two Radon probability measures on $\mathbb{R}^N$ and $0 < m_0 \leq 1$, then*

$$\|d_{\mu,m_0} - d_{\nu,m_0}\|_{L^\infty(\mathbb{R}^N)} \leq \frac{1}{\sqrt{m_0}} W_2(\mu, \nu).$$

Recall that as we send $m_0 \to 0$, the distance to measure function reduces to the traditional distance function. In this proposition, as we send $m_0 \to 0$, the upper bound approaches infinity. This reflects the issue previously discussed that minor changes in a set K drastically change the distance function d_K. The importance of this proposition becomes clear when we consider μ to be a distribution induced by a noisy sampling of a set and ν to be some noiseless probability measure for sampling the set. In this context, if μ and ν are close in the Wasserstein sense, the distance to measure functions induced by μ and ν will be close in the L^∞ sense. Many additional error bounds and convergence results may be found in [12].

Since point clouds are often sampled through a noisy process, we would like to model the noise. For symmetric noise, we can set $M \subset \mathbb{R}^N$ to be the set from which we wish to sample, and let ν be the uniform probability measure on the surface of M, induced by surface area. If we were to sample directly from ν, we would not encounter any noise. However, if we let $\mathcal{N}(0, \sigma^2)$ denote an N-dimensional Gaussian distribution with covariance matrix $\sigma^2 I_N$ (where I_N is the N-dimensional identity matrix), and we set $\mu = \nu \star \mathcal{N}(0, \sigma^2)$ where $\star$ denotes the convolution of probability measures, then sampling from the resulting distribution, μ, would yield a noisy sampling of M. Under this sampling model, Chazal et al. establish the following in [10].

Proposition 2.5 *Let μ_X denote the empirical measure on X, that is*

$$\mu_X(A) = \frac{|A \cap X|}{|X|} \quad \text{for all Borel subsets } A \subseteq \mathbb{R}^N.$$

Also, assume μ_X is constructed from a point cloud X sampled according to $\mu = \nu \star \mathcal{N}(0, \sigma^2)$ where $\mathcal{N}(0, \sigma^2)$ is a Gaussian distribution with mean 0 and covariance matrix $\sigma^2 I_N$. Then

$$\lim_{|X| \to \infty} W_2(\mu_X, \mu) \leq \sigma \sqrt{N} \quad \text{with probability } 1.$$

Therefore, in the limit of the sampling size, the Wasserstein distance between the empirical distribution and the sampled distribution is bounded by the bandwidth (i.e. σ) of the Gaussian. Combining the results of Propositions 2.4 and 2.5, we can establish the following corollary.

Corollary 2.6 *Let $X \subset \mathbb{R}^N$ be a noisy point cloud sampled from the probability measure $\mu = \nu \star \mathcal{N}(0, \sigma^2)$ and let μ_X denote the empirical distribution on X, then we have the following estimate:*

$$\lim_{|X| \to \infty} \|d_{\mu_X, m_0} - d_{\mu, m_0}\|_{L^\infty(\mathbb{R}^N)} \le \sqrt{\frac{N}{m_0}}\, \sigma.$$

The result established by Corollary 2.6 shows that the distance to measure function d_{μ_X, m_0} induced by the point cloud sample provides a good approximation for the distance to measure function d_{μ, m_0} of the sampled distribution. We would like to move the points of X so that they lie closer (in the distance to measure sense) to the underlying distribution ν. To do so, we first note that since $d_{\mu,m}$ and $d_{\mu_X,m}$ are close, their squares are close. Thus, by moving the sampled points in a manner which decreases $d^2_{\mu_X,m}$, we hope to decrease $d^2_{\mu,m}$. Of course, by Proposition 2.4, we know $d_{\mu,m}$ and $d_{\nu,m}$ are close (and so are there squares) if μ and ν are close in the Wasserstein sense. Thus, through decreasing $d^2_{\mu_X,m}$, we aim to move the points closer to μ and in turn, closer to ν. In the following section, we investigate a gradient flow system that does just that.

3 Squared Distance-to-Measure Gradient Flow

In this section, we present the squared distance to measure gradient flow and establish a few properties of the flow. We then present our new theoretical framework for solving the gradient flow system. Although we are able to find exact solutions of the gradient flow using this technique, it is computationally expensive. In practice, directly inducing the gradient flow onto a point cloud through discretization is preferred due to its lower computational cost. However, our theoretical framework will give some insight into how inaccuracies may arise during this discretization. This will be discussed more in Sect. 4. We now turn our attention to finding solutions of the squared distance to measure gradient flow.

3.1 Smoothing Gradient Flow System

As we saw in the previous section, if two probability measures are close in the Wasserstein sense, then their respective distance to measure functions will be close, and thus, their squared distance to measure functions will be close. Using this idea, we will evolve a point cloud by moving each point along a trajectory which maximally reduces its squared distance to measure. To make this notion precise, we note that if we use the empirical measure μ_X, as defined in Definition 2.6, the distance to measure μ_X with parameter $m_0 = k/n$, for some non-negative integer $k \le |X|$, reduces to the following function:

$$E_X^k(x) := d^2_{\mu_X, m_0}(x) = \frac{1}{k} \sum_{y \in \mathrm{NN}_X^k(x)} \|x - y\|^2 \tag{1}$$

with $n = |X|$ and where $\mathrm{NN}_X^k(x)$ is the set of the k-nearest neighbors of x in the point cloud X. For the derivation of this result, see [9]. Since the distance to measure

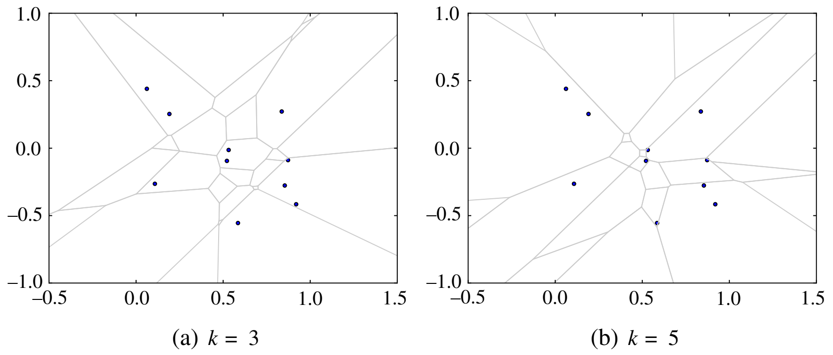


Fig. 1 k-Order Voronoi diagrams for 10 points

function in the above equation depends only on the integer value k, we will ignore the parameter m_0 and simply refer to $d^2_{\mu_X, m_0}$ as E^k_X for the remainder of the paper.

Note that there is some ambiguity above in the definition of $\mathrm{NN}^k_X(x)$. In particular, for a point $x \in \mathbb{R}^N$, if we let $y_1, y_2, \ldots, y_n$ denote all the points of X ordered so that $\|x - y_1\| \le \|x - y_2\| \le \cdots \le \|x - y_n\|$, then the natural definition of $\mathrm{NN}^k_X(x)$ is simply $\mathrm{NN}^k_X(x) = \{y_1, y_2, \ldots, y_k\}$. However, if $\|x - y_k\| = \|x - y_{k+1}\|$, then the points y_k and y_{k+1} are equidistant from x. Thus, we have more than k candidates for the set $\mathrm{NN}^k_X(x)$. While this is not an issue in (1) since we only need the distance to the k-nearest neighbors (which is the same for all the ambiguous points), it will be an issue when we take the gradient of this function. As it turns out, points for which there is ambiguity in the definition of E^k_X hold special significance for the gradient flow induced by the squared distance to measure function.

From (1), we can compute the gradient of E^k_X,

$$\nabla E^k_X(x) = \frac{2}{k} \sum_{y \in \mathrm{NN}^k_X(x)} (x - y). \tag{2}$$

As just discussed, this equation is valid almost everywhere. However, it is not valid at points of $\mathbb{R}^N$ where there is more than one valid k-nearest neighbor set. If we consider the set of all points for which $\mathrm{NN}^k_X(x)$ is ambiguous, then we have constructed the skeleton of the k-order Voronoi diagram induced by the point cloud X. Recall that the k-order Voronoi diagram induced by a finite set of points P, denoted $V^k(P)$, is a decomposition of $\mathbb{R}^N$ into regions, called k-order Voronoi cells, in which every interior point of a Voronoi region V shares the same k nearest neighbors as every other interior point of the region V. In Fig. 1, we show two k-order Voronoi diagrams, with $k = 3$ and $k = 5$, for a point cloud consisting of 10 points drawn uniformly on $[0, 1]^2$. Notice that as expected, the two diagrams are quite different.

Given a k-order Voronoi region V, we call the common k-nearest neighbors, of the interior points of V, the *generators* of V, denoted $\mathrm{gen}(V)$. Note that $\mathrm{gen}(V)$ is unique and well defined for all $V \in V^k(P)$ since it is defined using the interior of V, where there is no ambiguity in the nearest neighbor set. We will denote the skeleton of the k-order Voronoi diagram by $\partial V^k(P)$.

As a quick aside, it is a well known fact that higher order Voronoi diagrams can be constructed recursively. In particular, given a $(k-1)$-order Voronoi diagram, we can construct the k-order Voronoi diagram following a simple procedure. For every region $V \in V^{k-1}(X)$, we construct the first-order Voronoi diagram of the set $X \setminus \operatorname{gen}(V)$. Then, we intersect this new diagram with the region V, producing several subregions of V. Each of these subregions will contain the same k-nearest neighbors. We do this for every regions $V \in V^k(X)$. Once all the subregions are formed, a simple check must be performed to ensure that two adjacent subregions do not contain the same k-nearest neighbors. If they do, these subregions must be merged. Thus, every face of a Voronoi region in $V^k(X)$ arises from an intersection with a first order Voronoi diagram. Since all the faces of a first order Voronoi diagram are bisecting hyperplanes between two points, we can see that all co-dimension all co-dimension one faces one faces of the k-order Voronoi diagram arise as bisecting hyperplanes. In particular, for any adjacent k-order Voronoi regions V_i and V_j sharing a co-dimension one Voronoi face, we must have $|\operatorname{gen}(V_i) \cap \operatorname{gen}(V_j)| = k-1$ (the generators of the regions only disagree on one point since the interface between the regions is co-dimension one). Furthermore, since the co-dimension one Voronoi faces arise from bisecting hyperplanes between a finite number of points, we of course have that the union of all the co-dimension one Voronoi faces will be measure zero in $\mathbb{R}^N$. Therefore, the probability, following the model outlined in Sect. 2, that a point cloud X will be sampled in which a point $x \in X$ lies on a co-dimension one Voronoi face is zero. This can be seen by considering the sampling of X to be incremental. When sampling x_i, the points $x_1, \ldots, x_{i-1}$ induce a higher-order Voronoi diagram whose co-dimension one faces, when unioned, have zero measure in $\mathbb{R}^N$. Thus, the probability that x_i would fall on one of these faces is zero. These results will be important in the proof of Theorem 3.7.

Returning to the discussion at hand, due to the ambiguities of the gradient $\nabla E_X^k(x)$ for points x on the boundary $\partial V^k(X)$, we cannot simply define a gradient flow using (2). Instead, we would like to define a piecewise-smooth flow, as defined by M. di Bernardo et al. in [16]. We reproduce this definition below.

Definition 3.1 A piecewise-smooth flow is a finite set of ordinary differential equations,

$$\dot{x} = F_i(x) \quad \text{for} \quad x \in S_i,$$

where the sets $S_i \subset \mathbb{R}^N$ each have non-empty interior and $\bigcup_i S_i \subseteq \mathbb{R}^N$. Additionally, the intersection $\Sigma_{ij} = \overline{S_i} \cap \overline{S_j}$ is either a co-dimension one submanifold included in the boundaries ∂S_i and ∂S_j, or is the empty set. Finally, each vector field F_i is smooth in both x, and defines a smooth flow $\phi_i(x,t)$ with any open set $U \supset S_i$. In particular, each flow ϕ_i is well defined on both sides of the boundary ∂S_i.

Using the language of [16], for $V_i, V_j \in V^k(X)$, we will refer to the boundary $\Sigma_{ij} = \partial V_i \cap \partial V_j$ as a *switching manifold*. Additionally, we define

$$\Sigma = \bigcup \Sigma_{ij}.$$

For the remainder of this paper, as in [16], when we discuss switching manifolds, we only concern ourselves with switching manifolds of co-dimension one. Lower dimensional submanifolds are of little concern since the set of initial points whose trajectories will pass through these manifolds has measure zero. This fact is a result of Theorem 3.7 which states that trajectories cannot "slide" along the Voronoi faces in positive time.

Putting the gradient flow system in the context of piecewise-smooth flows, we let E_i denote the function

$$E_i(x) = \frac{1}{k} \sum_{y \in \mathrm{gen}(V_i)} \|x - y\|^2$$

defined over all $x \in \mathbb{R}^N$. Then, in Definition 3.1, the gradient flow induced by $\nabla E_i(x)$ plays the role of $\phi_i(x, t)$. That is, we can set

$$\frac{\partial \phi_i}{\partial t}(x, t) = -\nabla E_i(\phi_i(x, t))$$

for $t > 0$ and $\phi_i(x, 0) = x$.

Now, consider a co-dimension one switching manifold Σ_{ij} and suppose we had $\nabla E_i(x) = \nabla E_j(x)$ for some $x \in \Sigma_{ij}$. Then we have

$$\frac{2}{k} \sum_{v \in \mathrm{gen}(V_i)} (x - v) = \frac{2}{k} \sum_{w \in \mathrm{gen}(V_j)} (x - w)$$

and therefore

$$\sum_{v \in \mathrm{gen}(V_i)} v = \sum_{w \in \mathrm{gen}(V_j)} w.$$

However, since we know $k - 1$ points of $\mathrm{gen}(V_i)$ and $\mathrm{gen}(V_j)$ must agree (since they share a co-dimension one face), it is impossible to have this equality since this would imply all k points of $\mathrm{gen}(V_i)$ and $\mathrm{gen}(V_j)$ must agree and hence V_i and V_j cannot be separate k-order Voronoi regions, contrary to our assumption.

Thus, along the switching manifold, we have $\nabla E_i(x) - \nabla E_j(x) \neq 0$ for $x \in \Sigma_{ij}$. As noted in [16], at these points the switching manifold Σ_{ij} is said to have degree of smoothness one. This means that the first non-zero partial derivative with respect to t of $[\phi_i(x, t) - \phi_j(x, t)]|_{t=0}$ is of order one. Since our system exhibits switching manifolds with degree of smoothness one, we recognize the gradient flow system as a Filippov type system.

Defining the gradient along the switching manifolds becomes important in Filippov systems. In this work, we follow Filippov's convex method [19]. To utilize Filippov's convex method, we must define the *sliding region*, $\widehat{\Sigma}_{ij}$ of the switching manifold

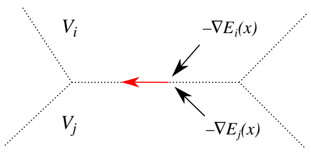


Fig. 2 Attractive sliding along the interface $\partial V_i \cap \partial V_j$

$\Sigma_{ij} = \partial V_i \cap \partial V_j$. First, let η_x denote the normal vector of the co-dimension one hyperplane $\partial V_i \cap \partial V_j$. In particular, we choose the normal vector which points into V_i. Let $E_i(x)$ and $E_j(x)$ be the function $E^k_X(x)$ as defined for V_i and V_j respectively (i.e. induced by $\mathrm{gen}(V_i)$ and $\mathrm{gen}(V_j)$). Then we can define the *sliding region*, $\widehat{\Sigma}_{ij}$ of the switching manifold Σ_{ij} to be

$$\widehat{\Sigma}_{ij} = \big\{x \in \Sigma_{ij} : \langle \eta_x, \nabla E_i(x)\rangle \cdot \langle \eta_x, \nabla E_j(x)\rangle < 0\big\}.$$

Thus, points in the sliding region $\widehat{\Sigma}_{ij}$ consist of points where the gradients on either side of Σ_{ij} point into different k-order Voronoi regions.

For points $x \in \Sigma_{ij} \backslash \widehat{\Sigma}_{ij}$, the definition of the gradient is straightforward since both gradients point into the same k-order Voronoi region. In particular, we will just follow the flow and if a trajectory leaves V_i and enters V_j, then we define the gradient on $\partial V_i \cap \partial V_j$ to be the gradient induced by the generators of V_j (i.e. $\mathrm{gen}(V_j)$). However, on $\widehat{\Sigma}_{ij}$, we define the gradient $\nabla E^k_X(X)$ to be a convex combination of the gradients on either side of $\partial V_i \cap \partial V_j$. Thus, for a point $x \in \widehat{\Sigma}_{ij}$, we set

$$\nabla E^k_X(x) = (1 - \alpha(x))\nabla E_i(x) + \alpha(x)\nabla E_j(x) \quad \text{for all} \quad x \in \Sigma_{ij}. \tag{3}$$

We then define $\alpha : \mathbb{R}^N \to \mathbb{R}$ as in [16], setting

$$\alpha(x) = \frac{\langle \eta_x, \nabla E_i(x)\rangle}{\langle \eta_x, \nabla(E_i - E_j)(x)\rangle},$$

where $\langle \cdot, \cdot \rangle$ is the Euclidean inner product. This choice of $\alpha(x)$ causes the gradient $\nabla E^k_X(x)$ to lie orthogonal to η_x. This is illustrated in Fig. 2, which shows the resulting gradient (red arrow) for two regions V_i and V_j which exhibit attractive sliding along the interface.

In [16], Bernardo et al. show that the above definition of $\widehat{\Sigma}_{ij}$ is identical to the following,

$$\widehat{\Sigma}_{ij} = \big\{x \in \Sigma_{ij} : 0 \le \alpha \le 1\big\}.$$

There are two modes of sliding which can occur in the sliding region. Recall that in the region V_i (resp. V_j), the induced flow follows the vector field given by $-\nabla E_i$ (resp. $-\nabla E_j$). Note that if $\langle \eta_x, -\nabla E_i\rangle < 0$ and $\langle \eta_x, -\nabla E_j\rangle > 0$, then the system exhibits *attractive sliding* at the point x, since η_x was chosen to point into V_i. Conversely, if $\langle \eta_x, -\nabla E_i\rangle > 0$ and $\langle \eta_x, -\nabla E_j\rangle < 0$, then the system exhibits *repulsive sliding* at x.

Repulsive sliding does not pose any problems evolving the point cloud in forward time since these interfaces cannot be reached in forward time. Additionally, as previously mentioned, the probability that a point cloud X is sampled for which there exists a point $x \in X$ which lies on a co-dimension one Voronoi face is zero. On the other hand, attractive sliding is important since regardless of whether a point starts on the interface, it may be drawn to the interface in forward time, at which point it will begin sliding. If points can slide along an interface, then two points can enter the sliding regions at different points and exit the sliding region at the same point. Thus, trajectories are not well defined in reverse time. However, as we will see in the next section, attractive sliding does not occur for any point cloud undergoing this flow. Repulsive sliding, on the other hand, does occur. In fact, repulsive sliding regions provide the unstable regions of the gradient flow system. In particular, a repulsive sliding region acts as a separatrix, with points on opposing sides following entirely different trajectories.

For now, we put off analyzing the sliding region dynamics since we have all we need to define the gradient flow system for the entire domain $\mathbb{R}^N$. To define the trajectory $u(t)$ for a point $x \in \mathbb{R}^N$, we evolve using the following differential equation:

$$\begin{cases} \dfrac{du}{dt}(t) = -\nabla E_X^k(u(t)), & t > 0, \\ u(t) = x, & t = 0, \end{cases} \tag{4}$$

where ∇E_X^k follows the form given in (2) if x does not lie on a sliding region and the form given in (3) when x lies on some sliding region. Now, let $\phi(x,t) = u(t)$ where u is the trajectory defined above for a given $x \in \mathbb{R}^N$. Given a point cloud X, we define $X_t = \{\phi(x,t) : x \in X\}$ for all $t \geq 0$. As we increase t, every point in the point cloud will move in the direction which minimizes its squared distance to measure.

It is important to note a subtle issue that we side stepped when defining the gradient flow system. If we were to evolve a point cloud X according to the system as described in (4), updating the nearest neighbor function as we go, all the points would begin to cluster together. This is because the points are drawn to their nearest neighbors, which are in turn drawn to them. To help alleviate this issue, we fix the nearest neighbor function NN_X^k at time $t = 0$. That is, at any time $t > 0$, instead of computing $\nabla E_{X_t}^k$ we compute $\nabla E_{X_0}^k$, using the original positions of the points, and use this gradient in (4).

3.2 Qualitative Description of the Flow

The discussion of the squared distance to measure gradient flow system made it clear that there is a strong connection between higher order Voronoi diagrams and the system described in (4). We will now make the connection clear and use the higher order Voronoi diagram to solve the squared distance to measure gradient flow system. First, we must establish a few properties of k-order Voronoi diagrams.

Proposition 3.2 *Let $X \subset \mathbb{R}^N$ be a point cloud. Then $V^1(X)$ is a set of convex polytopes.*

To see why this proposition holds, note that each Voronoi diagram is simply an intersection of half-spaces. In particular, the Voronoi region corresponding to x is precisely the intersection of all half-spaces H_x^y (for $x \neq y \in X$) where the separating hyperplane of H_x^y consists of all points equidistant from x and y. With this in mind, we can prove the following.

Proposition 3.3 *Let $X \subset \mathbb{R}^N$ be a point cloud. Then the k-order Voronoi diagram of X is a set of convex polytopes.*

Proof Consider a non-empty k-order Voronoi region $V \in V^k(X)$. Let the generator set of V be given by $\operatorname{gen}(V) = \{x_1, \ldots, x_k\}$. Recall that any point x in the interior of V is closer to the points $x_1, \ldots, x_k$ than to any other $y \in X \setminus \operatorname{gen}(V)$. Consider the subsets $X_i \subset X$ for $1 \le i \le k$ where $X_i = X \setminus \{x_1, \ldots, \widehat{x_i}, \ldots, x_k\}$ and $\widehat{x_i}$ denotes the absence of x_i. Then in the first order Voronoi diagram $V^1(X_i)$, the Voronoi region V_i corresponding to x_i is convex by Proposition 3.2. Additionally, V_i consists of all points whose distance to x_i is less than or equal to the distance to any other point in $X \setminus \operatorname{gen}(V)$. Repeating this for every $1 \le i \le k$ yields a set of convex polytopes $\{V_1, \ldots, V_k\}$ where for each i, V_i is convex. Thus, the set $V_1 \cap V_2 \cap \cdots \cap V_k$ is convex. We claim that $V_1 \cap V_2 \cap \cdots \cap V_k = V$ and so V is also convex.

To see this, let $x \in V_1 \cap V_2 \cap \cdots \cap V_k$. Since $x \in V_i$ for each $1 \le i \le k$, we know that $\|x - x_i\| \le \|x - y\|$ for all $y \in X \setminus \operatorname{gen}(V)$. Since this holds for each $1 \le i \le k$, we have

$$\max_{1 \le i \le k} \|x - x_i\| \le \min_{y \in X \setminus \operatorname{gen}(V)} \|x - y\|.$$

Thus, we have $x \in V$ and so $V_1 \cap \cdots \cap V_k \subseteq V$. Conversely, if $x \in V$, then by the definition of a k-order Voronoi region, for any $1 \le i \le k$, we know that the distance from x to x_i is less than or equal to the distance from x to any point in $X \setminus \operatorname{gen}(V)$. Hence, $x \in V_i$ for all $1 \le i \le k$. Thus, $x \in V_1 \cap V_2 \cap \cdots \cap V_k$ and so $V \subseteq V_1 \cap V_2 \cap \cdots \cap V_k$. Therefore, we have $V = V_1 \cap V_2 \cap \cdots \cap V_k$ as desired. Since V was an arbitrary k-order Voronoi region in $V^k(X)$, we see that all the k-order Voronoi regions of $V^k(X)$ are convex. □

Since all the k-order Voronoi regions are convex polytopes, we know there must be a finite number of k-order Voronoi regions since there are a finite number of points in X (and thus a finite number of k-combinations of points in X). Futhermore, we can easily identify the vertex barycenter of each bounded region. For our purposes, we will need to distinguish between the vertex barycenter of the polytope and the barycenter of the generators of the k-order Voronoi region. To make this difference precise, we use the following definition.

Definition 3.4 For a k-order Voronoi region V with $\operatorname{gen}(V) = \{x_1, \ldots, x_k\}$, let $\operatorname{Bar}_G(V)$, called the *generator barycenter*, be defined by

$$\operatorname{Bar}_G(V) = \frac{1}{k} \sum_{i=1}^{k} x_i.$$

On the other hand, let $\mathrm{Bar}_P(V)$ be the *vertex barycenter* of V. That is,

$$\mathrm{Bar}_P(V) = \frac{1}{n} \sum_{v \in \mathrm{Vert}(V)} v,$$

where $\mathrm{Vert}(V)$ denotes the vertices of the convex polytope V. This will be defined for all bounded k-order Voronoi regions.

It is important to note that for a k-order Voronoi region $V \in V^k(X)$, we must have $\mathrm{Bar}_P(V) \in V$ if $\mathrm{Bar}_P(V)$ exists (since V is the convex hull of $\mathrm{Vert}(V)$ and $\mathrm{Bar}_P(V)$ is a convex sum of points in $\mathrm{Vert}(V)$). However, we will not, in general, have $\mathrm{Bar}_G(V) \in V$. In fact, the regions V for which $\mathrm{Bar}_G(V) \in V$ are of particular interest.

Theorem 3.5 *Let $V^k(X)$ be the k-order Voronoi diagram for a point cloud $X \subset \mathbb{R}^N$. Under the gradient flow induced by (4), a region $V \in V^k(X)$ is a positively invariant set if and only if it contains its generator barycenter. That is,*

$$\phi(x,t) \in V \quad \textit{for all} \quad x \in V,\ t \geq 0 \quad \Leftrightarrow \quad \mathrm{Bar}_G(V) \in V.$$

Proof Let $\mathrm{gen}(V) = \{x_1, \ldots, x_k\}$. Since all points of V share the same k-nearest neighbors, we can solve the gradient flow exactly for all $x \in V$ (until the trajectory leaves V). In particular, for any $x \in V$, let $u(t) = \phi(x,t)$, then we have

$$\frac{du}{dt} = -\frac{2}{k}\sum_{i=1}^{k}(u - x_i) = -2u + \frac{2}{k}\sum_{i=1}^{k} x_i = -2u + 2\,\mathrm{Bar}_G(V),$$

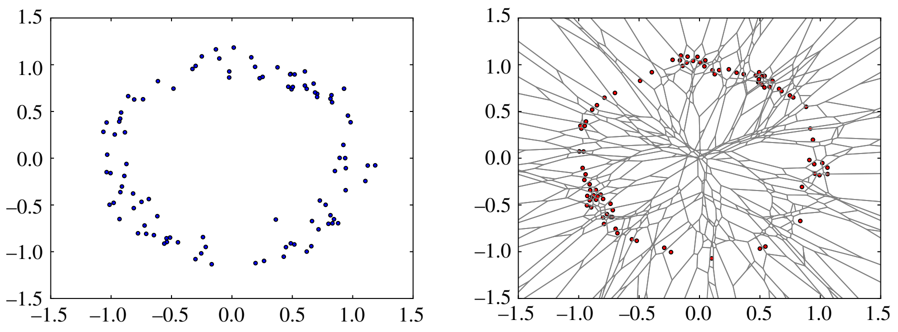
which has the solution

$$u(t) = \mathrm{Bar}_G(V) + c_1 e^{-2t}$$

for some $c_1 \in \mathbb{R}$. Thus, since $u(0) = x \in V$, we have $x = \mathrm{Bar}_G(V) + c_1$ and so $c_1 = x - \mathrm{Bar}_G(V)$. Hence, the solution is

$$\begin{aligned} u(t) &= \mathrm{Bar}_G(V) + (x - \mathrm{Bar}_G(V))e^{-2t} \\ &= xe^{-2t} + (1 - e^{-2t})\,\mathrm{Bar}_G(V) \\ &= x\lambda(t) + (1 - \lambda(t))\,\mathrm{Bar}_G(V), \end{aligned} \tag{5}$$

where $\lambda(t) = e^{-2t}$. This solution is only valid so long as $u(t) \in V$. Let $T = \max\{t \in \mathbb{R}^+ : u(t) \in V\}$. Note that $\lambda(t) \in (0,1]$ for $t \in [0,T)$. Therefore, $\{u(t)\}_{t=0}^{T}$ is a line segment from x to $\mathrm{Bar}_G(V)$. Since V is convex, this line segment never leaves V. Thus, if $\mathrm{Bar}_G(V)$ is contained in V, then the trajectory never leaves V and the point x flows toward $\mathrm{Bar}_G(V)$.



(a) Original Cloud (b) Barycentric Sinks

Fig. 3 Barycentric sinks for a point cloud sampled from S^1

Conversely, suppose $V \in V^k$ is a positively invariant set. Then for any $x \in V$ with $u(t) = \phi(x, t)$, we have $u(t) \in V$ for all $t \geq 0$. We know the trajectory of x must begin at x and follow a straight line [by (5)], ending at $\mathrm{Bar}_G(V)$. However, since $u(t) \in V$ for all $t \geq 0$, this line never leaves V. Thus, $\mathrm{Bar}_G(V)$ is either in the interior of V or it is a limit point of V. However, since V is closed, we must have $\mathrm{Bar}_G(V) \in V$. $\square$

For the remainder of this paper, we will refer to the condition $\mathrm{Bar}_G(V) \in V$ as the *barycentric sink condition*. The above theorem is quite powerful. In fact, we have established that the set

$$\mathscr{S} = \left\{V \in V^k(X) : \mathrm{Bar}_G(V) \in V\right\}$$

consists of all the positively invariant k-order Voronoi regions. Additionally, we know that the set

$$\mathscr{B} = \left\{\mathrm{Bar}_G(V) : V \in \mathscr{S}\right\}$$

contains all the equilibria of the gradient flow system defined by (4), and thus, all the convergent orbits of the system will converge to a point in $\mathscr{B}$.

As an example, in Fig. 3a we show a point cloud X of 100 points noisily drawn from the unit circle S^1. Additionally, in Fig. 3b, we show the elements of $\mathscr{B}$ as red points and the fifth-order Voronoi diagram of the point cloud X. Notice that many points of $\mathscr{B}$, although close to the sampled circle, form tightly packed clusters around the circle. This can result in clustering of the data as the gradient flow progresses and points flow toward elements of $\mathscr{B}$.

In the proof of the last theorem, we established an additional result which we capture in the following lemma. In this lemma, we denote the interior of a set A as $\mathrm{int}(A)$.

Lemma 3.6 *Let $V \in V^k(X)$ and let $x \in \mathrm{int}(V)$. Then there exists $\tau > 0$ such that $\phi(x, t) \in V$ for all $t < \tau$. Furthermore, for any $x \in V$, the set $\{\phi(x, t) : t < \tau\}$ is a line segment from x toward $\mathrm{Bar}_G(V)$, terminating either at the boundary ∂V or at the point $\mathrm{Bar}_G(V)$ itself.*

Having established some positively invariant sets, we now return to analyzing the dynamics on a sliding region. As mentioned in the previous section, repulsive sliding motion is of little concern since it occurs with probability zero. However, attractive sliding motion deserves greater attention. In the following theorem, we show that attractive sliding cannot occur.

Theorem 3.7 *Attractive sliding does not occur when the gradient flow system given in* (4) *is applied to a point cloud.*

Proof For sake of contradiction, suppose that a co-dimension one interface between $V_i \in V^k(X)$ and $V_j \in V^k(X)$ exhibits attractive sliding motion. Since this interface, $\partial V_i \cap \partial V_j$, is between two k-order Voronoi regions, we know it is a co-dimension one hyperplane. Thus, we can extend the interface to decompose $\mathbb{R}^N$ into two regions H_i and H_j, such that $V_i \subseteq H_i$ and $V_j \subseteq H_j$. Following our procedure in the previous section, we define η_x to be the normal vector to $\partial V_i \cap \partial V_j$ at the point $x \in \partial V_i \cap \partial V_j$ pointing into V_i, and thus into H_i as well. Of course, the normal direction of $\partial V_i \cap \partial V_j$ is constant along $\partial V_i \cap \partial V_j$ (due to $\partial V_i \cap \partial V_j$ being a subset of a hyperplane), therefore, we will simply write η when referring to η_x.

The boundary of $\partial V_i \cap \partial V_j$ may intersect other k-order Voronoi regions. However, points in the interior of $\partial V_i \cap \partial V_j$ will only be members of V_i and V_j (since $\partial V_i \cap \partial V_j$ is co-dimension one). Thus, given a point x in the interior of $\partial V_i \cap \partial V_j$, we have

$$r_i = \max_{y \in \text{gen}(V_i)} \|x - y\| = \max_{z \in \text{gen}(V_j)} \|x - z\| = r_j.$$

To see why this is true, suppose without loss of generality that $r_i < r_j$. Then there exists $z \in \text{gen}(V_j)$ such that $\|x - y\| < \|x - z\|$ for all $y \in \text{gen}(V_i)$. Since there are k points in $\text{gen}(V_i)$, the point z cannot be one of the k-nearest neighbors of x, contradicting the fact that $x \in V_j$. Thus, the statement must hold. Next, we let

$$G_i = \{y \in \text{gen}(V_i) : \|x - y\| = r_i\},$$
$$G_j = \{y \in \text{gen}(V_j) : \|x - y\| = r_j\}$$

and see that $|G_i| = |G_j| = 1$ and $G_i \cap G_j = \emptyset$. This can be seen immediately from the recursive construction of the k-order Voronoi diagram. During its construction, all co-dimension one Voronoi faces in the k-order Voronoi diagram arise as bisecting hyperplanes (between two points). This is due to taking the intersection of a lower order Voronoi region with a first order Voronoi diagram, which itself consists of bisecting hyperplanes. Finally, since x lies on the interior of $\partial V_i \cap \partial V_j$, it cannot lie on the intersection of two or more bisecting hyperplanes (since this would result in a face of the k-order Voronoi diagram with co-dimension greater than one). Thus, the sets G_i and G_j must consist of a single point each, in particular the points for which $\partial V_i \cap \partial V_j$ act as a bisecting hyperplane.

For the system to exhibit attractive sliding motion at the point $x \in \partial V_i \cap \partial V_j$, we must have

$$\langle \eta, -\nabla E_i(x) \rangle < 0 \qquad \text{and} \qquad \langle \eta, -\nabla E_j(x) \rangle > 0.$$

Since G_i and G_j each consist of a single element, we can let $G_i = \{y\}$ and $G_j = \{z\}$ with $y \neq z$. By definition, since every other point of $\mathrm{gen}(V_i)$ and $\mathrm{gen}(V_j)$ is closer to x than the points y and z are to x, we must have that the sets $\mathrm{gen}(V_i)$ and $\mathrm{gen}(V_j)$ intersect in every element except y and z. That is,

$$\mathrm{gen}(V_i) \cap \mathrm{gen}(V_j) = \{c_1, \dots, c_{k-1}\},$$

where $c_i \in X \backslash \{y, z\}$ for every $1 \leq i \leq k-1$. Now define

$$B = \frac{1}{k-1} \sum_{i=1}^{k-1} c_i$$

and notice that

$$\mathrm{Bar}_G(V_i) = \left(\frac{k-1}{k}\right) B + \frac{1}{k}\, y,$$
$$\mathrm{Bar}_G(V_j) = \left(\frac{k-1}{k}\right) B + \frac{1}{k}\, z.$$

That is, $\mathrm{Bar}_G(V_i)$ and $\mathrm{Bar}_G(V_j)$ are both convex combinations of B with y and z, respectively. Now by the assumption that Σ_{ij} exhibits sliding motion, we must have $\mathrm{Bar}_G(V_i) \in H_j$ and $\mathrm{Bar}_G(V_j) \in H_i$. We also know $y \in H_i$ since points of V_i are closer to y than they are to z. Then since $\mathrm{Bar}_G(V_i) \in H_j$ is a convex combination of B and $y \in H_i$, we must have $B \in H_j$ following from the fact that H_i is also convex. However, we also have $z \in H_j$ and since $\mathrm{Bar}_G(V_j) \in H_i$ is a convex combination of B and $z \in H_j$, we must similarly have $B \in H_i$. The only way this can occur is if $B \in H_i \cap H_j$. But then if $B \in H_i \cap H_j$, any convex combination of B with a point from $\mathrm{int}(H_i)$ will fall in H_i and not in H_j. Thus we have come to a contradiction since we assumed $\mathrm{Bar}_G(V_i) \in H_j$. Therefore, we cannot have attractive sliding motion. □

Another way to see this result can be obtained using sets of paraboloids defined via the barycenters of the points of X. As shown in [21], the squared distance to measure function is equivalent to

$$E_X^k(x) = \min_{c \in C} \|x - c\|^2 - w_c,$$

where C is the set of all barycenters of k points in X and w_c is a weighting term. Thus, the distance to measure function can be seen as arising from a set of paraboloids. Hence, at the interface of two k-order Voronoi regions, it can be shown, using these paraboloids, that there is no attractive sliding motion.

Finally, we turn our attention to periodic orbits. Since the squared distance to measure gradient flow system is in fact a gradient flow system, it should come as no surprise that there are no periodic orbits. This follows from the fact that the squared distance to measure gradient flow is piecewise-smooth.

Theorem 3.8 *The gradient flow system defined in this section contains no periodic orbits.*

Proof Suppose there were a point $x \in \mathbb{R}^N$ such that for some $t > 0$, we had $\phi(x, 0) = \phi(x, t)$. Then obviously $E_X^k(\phi(x, 0)) = E_X^k(\phi(x, t))$. The trajectory $\phi(x, \cdot)$ may cross the k-order Voronoi skeleton many times during the flow. However, it can never stay in the k-order Voronoi skeleton for some non-zero amount of time since there is no attractive sliding in the system (see Theorem 3.7). Suppose the crosses happen at times $0 \le t_1 < \cdots < t_n \le t$. Then

$$\begin{aligned} 0 = E_X^k(\phi(x,0)) - E_X^k(\phi(x,t)) &= \sum_{i=1}^{n-1} E_X^k(\phi(x,t_{i+1})) - E_X^k(\phi(x,t_i)) \\ &= \sum_{i=1}^{n-1} \int_{t_i}^{t_{i+1}} \frac{dE_X^k(\phi(x,t))}{dt}\, dt. \end{aligned}$$

However, since $\frac{dE_X^k(\phi(x,t))}{dt} < 0$ for all $t \in I_i$ and each $i \in \{1, \ldots, m\}$, we see that the right hand side of the above derivation must be strictly less than 0. This contradicts the assumption that $E_X^k(\phi(x, t)) = E_X^k(\phi(x, 0))$ and hence the assumption that $\phi(x, t) = \phi(x, 0)$ for some $x \in \mathbb{R}^N$ and some $t > 0$. □

3.3 The Flow Diagram

Having demonstrated a few properties of the gradient flow system described by (4), we are now ready to provide a systematic method for solving such a system. To this end, we will construct a directed graph which encapsulates the dynamics of the gradient flow system. The graph will be a directed subgraph of the k-order Delaunay triangulation, whose definition is given below.

Definition 3.9 Let $V^k(X)$ be the k-order Voronoi diagram of a finite point cloud $X \subset \mathbb{R}^N$. The *k-order Delaunay triangulation*, $D^k(X) = (\mathscr{V}, E)$, is the graph whose vertex set $\mathscr{V}$ is given by

$$\mathscr{V} = \left\{ \mathrm{Bar}_P(V) : V \in V^k(X) \right\},$$

where we set $\mathrm{Bar}_P(V)$ to be some point of the interior of V if V is unbounded. An edge exists between vertices $v_i = \mathrm{Bar}_P(V_i)$ and $v_j = \mathrm{Bar}_P(V_j)$ if the corresponding boundaries of the k-order Voronoi regions intersect,

$$(v_i, v_j) \in E \quad \Leftrightarrow \quad \partial V_i \cap \partial V_j \neq \emptyset.$$

Of course this graph is the natural extension of the traditional Delaunay triangulation and is the dual graph of the k-order Voronoi diagram. We now seek to direct some of the edges of $D^k(X)$ and remove other edges. The remaining edges will indicate when points from one region V_i will flow into another region V_j. Thus, after the orientation

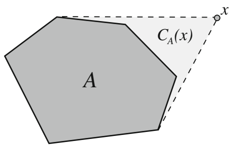


Fig. 4 Example polyhedral cone $C_A(x)$

and removal process, any k-order Voronoi region V satisfying the barycentric condition will have zero edges leaving V. Conversely, any region V not satisfying the barycentric sink condition will have at least one edge leaving V. To determine which edges to keep and which edges to remove, we require the following tool.

Definition 3.10 Let A be a closed, convex polytope in $\mathbb{R}^N$ and let $x \in \mathbb{R}^N$. The *polyhedral cone*, $C_A(x)$, from x to A is given by

$$C_A(x) = \big\{\lambda x + (1-\lambda)y : y \in A,\ \lambda \in [0,1]\big\}. \tag{6}$$

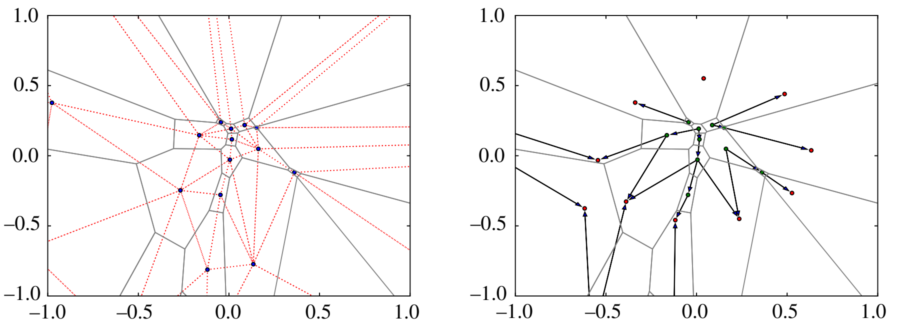
Note that this is simply the union of all line segments originating in a closed, convex polytope A and terminating at some $x \in \mathbb{R}^N$. If the point x is in fact contained in the polytope A, then $C_A(x) = A$. For our purposes, we will set the polytope to be one of our k-order Voronoi regions, $V \in V^k(X)$, and let $x = \mathrm{Bar}_G(V)$. An example polyhedral cone $C_A(x)$ for a set A is given in Fig. 4.

Let $V \in V^k(X)$ and let E_V denote all the edges of $D^k(X)$ which include $\mathrm{Bar}_P(V)$ as a vertex. We wish to keep an edge e from V to a neighbor region V' if any point $y \in V$ passes into the interior of V' during the gradient flow. Of course, if V satisfies the barycentric sink condition, then none of the points of V will leave V during the flow and thus we will not orient any of the edges in E_V. Let us assume that V does not satisfy the barycentric sink condition. Since we know that the point y will follow the line segment from y to $\mathrm{Bar}_G(V)$, if this line segment intersects the interior of V', then we know that y will pass into V' during the flow. By definition, this will happen exactly when

$$C_V(\mathrm{Bar}_G(V)) \cap \mathrm{int}(V') \neq \emptyset. \tag{7}$$

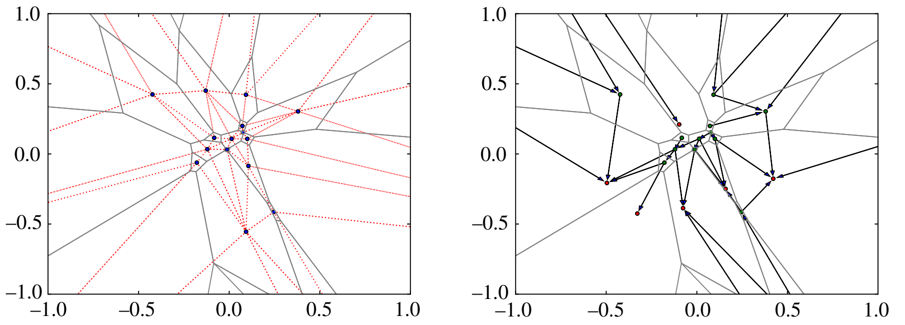
Therefore, to determine whether to keep an edge from V to V', we simply check whether $\mathrm{int}(V')$ intersects the polyhedral cone from $\mathrm{Bar}_G(V)$ to V. If the intersection does occur, then we orient an edge from V to V'. Doing this for all k-order Voronoi regions in V^k, we complete the process by removing any undirected edges. This produces the flow diagram $F^k(X)$, a directed graph describing the dynamics of the system induced by (4). Note that this flow graph does not fully describe the dynamics, but instead gives a higher level view of the dynamics. To fully characterize the flow, we would need to further decompose the k-order Voronoi diagrams according to where the points flow. In this graph however, each k-order Voronoi region may have many edges pointing to adjacent k-order Voronoi regions. This indicates that some points in the region flow to one neighbor while other points flow to another neighbor.

Examples of $D^k(X)$ and $F^k(X)$ for $k = 3$ and $k = 5$, computed on a small point cloud $X \in \mathbb{R}^2$, are given in Figs. 5 and 6, respectively. Of course, the dynamics will



(a) $D^3(X)$ (b) $F^3(X)$

Fig. 5 Delaunay triangulation (left) and flow diagram (right)



(a) $D^5(X)$ (b) $F^5(X)$

Fig. 6 Delaunay triangulation (left) and flow diagram (right)

change when using different values of k, as can be seen in the two figures. In all the figures, we show the k-order Voronoi diagram for the point cloud X using black, solid lines. Then, in the first figures of Figs. 5 and 6, the third and fifth order Delaunay triangulations are shown using red, dotted lines. In (b) of each figure, we show the flow diagram. Here, the vertices of $F^3(X)$ and $F^5(X)$ are color coded, where a red vertex indicates a sink of the flow and a green vertex indicates that the region does not satisfy the barycentric sink condition. The arrows indicate the direction of the flow out of the green regions.

To emphasize the fact that regions satisfying the barycentric sink condition contain a sink of the gradient flow, we have not only shown the sinks in red in Figs. 5b and 6b, we have actually opted to move the vertices of these regions to their generator barycenters. That is, in these figures, whenever a region V satisfies the barycentric sink condition, instead of showing $\mathrm{Bar}_P(V)$ as the vertex, we show $\mathrm{Bar}_G(V)$ as the vertex. This gives a better visual sense of where points will end up after running the gradient flow for a long time. For the higher order Delaunay triangulations, we still use the polytope generator since this is common practice.

4 Numerical Results

In this section, we present some examples which help illustrate the result of inducing the squared distance to measure gradient flow on a point cloud. Since computing the k-order Voronoi diagram is expensive for large point clouds and large values of k, we instead opt to simply induce the gradient flow without first computing the k-order Voronoi diagram. Additionally, we discretize the system by setting $T = (t_1, t_2, \ldots, t_n)$ and

$$\phi(x, t_i) = \phi(x, t_{i-1}) - \lambda \nabla E_X^k(\phi(x, t_{i-1})), \tag{8}$$

where λ is often referred to as the *step-size*. Of course, discretizing the dynamical system may result in loss of accuracy. Through our theoretical framework, we know that this will happen if the magnitude of $\lambda \nabla E_X^k(\phi(x, t_{i-1}))$ is large enough that we skip over one of the k-order Voronoi regions we would pass through under the continuous time gradient flow. Therefore, properly setting the value of the step-size λ is important. The most effective setting will depend on the point cloud and, in particular, the induced k-order Voronoi diagram.

Similarly, setting the appropriate value of k depends on the point cloud. In general, larger values of k will induce greater smoothing. This can be desired if the sampled object is itself smooth and lacks edges. However, if edges are present in the object, the gradient flow will smooth these edges and destroy these features. This is of particular concern for buildings in LiDAR data. To preserve the edges, a lower value of k would need to be used. In follow on work, we explore allowing k to adapt to the approximated local geometry of the point cloud as the gradient flow progresses.

Finally, we must determine how long to perform the gradient flow smoothing. This again depends on the data set. In our experiments, we see this dependence: smoothing the point clouds sampled from simpler geometries takes more iterations than the more complex point clouds. In the examples that follow, we set $\lambda = 0.05$ and experiment with various values of k and the duration of the gradient flow.

4.1 Unit Circle in $\mathbb{R}^2$

We begin our numerical analysis using simple point clouds noisily sampled from the unit circle, S^1. We choose S^1 since it has a simple geometry which allows us to compute the geometric error associated with the smoothed point cloud. In particular, given a smoothed point cloud X_t, we can compute the geometric error as

$$E(Y) = \sum_{y \in Y} (1 - \|y\|)^2, \tag{9}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^2$. To test the k-nearest neighbor algorithm, we sampled 1000 points from S^1 and added symmetrically distributed noise to each point in the point cloud. For the noise, we used a Gaussian distribution with mean 0 and bandwidth $\sigma = 0.1$. An example point cloud is shown in Fig. 7.

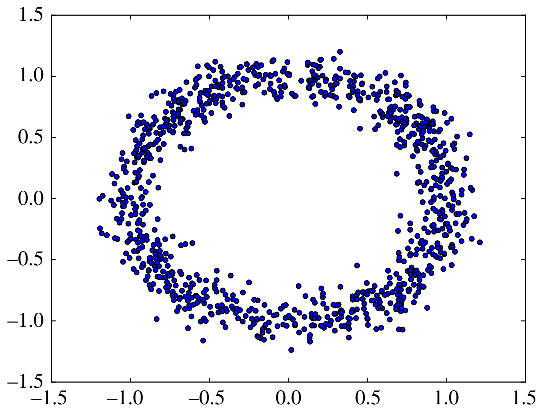


Fig. 7 Sample point cloud drawn from S^1

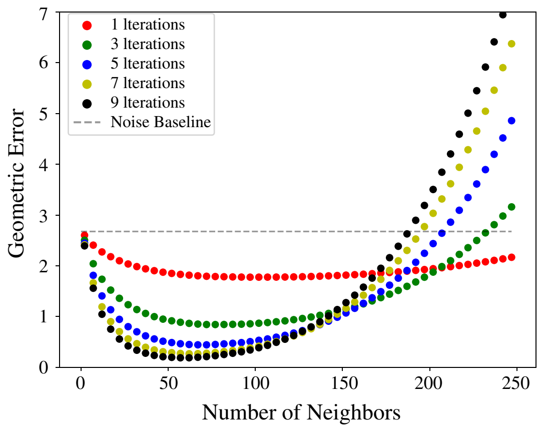


Fig. 8 Average geometric error for S^1

We generated 100 point clouds in this fashion and ran the k-nearest neighbor gradient flow for varying values of k. Thus, for each value of k, we have 100 point clouds. We then compute the geometric error of these point clouds and take the average, resulting in a single average geometric error for each value of k. Furthermore, we do this at every iteration t_i. The results of these calculations are shown in Fig. 8.

As a baseline, the average error present in a noisy point cloud X was $E(X) = 9.89$. Thus, all of the shown error rates are lower than the average original error in the point cloud. Even a single iteration removes a great deal of noise. Any of the displayed parameter values would reduce the geometric error in the point cloud, however it is clear that the optimal value for k is around $k = 75$ and the optimal number of iterations to run the gradient flow is around $T = 20$. After this, the point cloud begins to converge and the average error does not change much.

An example smoothed point cloud is shown in Fig. 9. In this figure, we can see that the k-nearest neighbor gradient flow tends to induce clustering in the point cloud during the smoothing process. This is because points will flow until they reach a region satisfying the barycentric sink condition. Once they enter this region, they will remain in this region. This can cause many points to flow toward the same point. However,

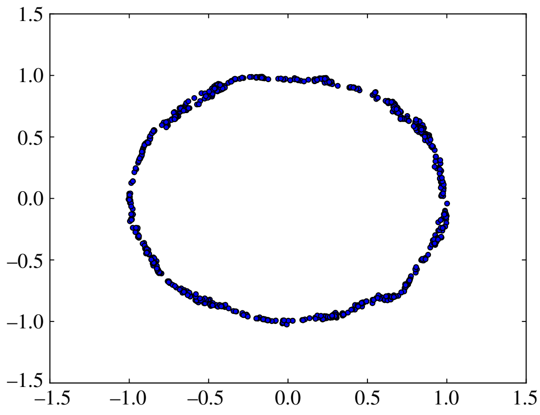


Fig. 9 Smoothed point cloud

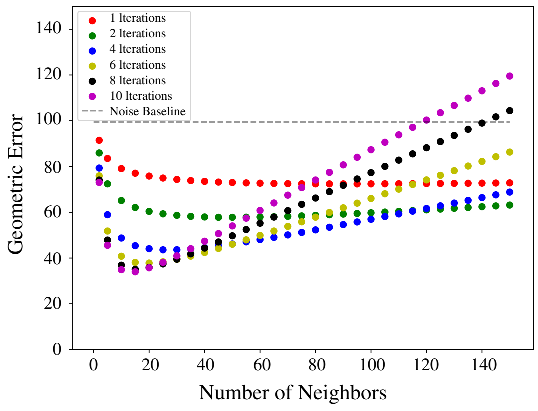


Fig. 10 Geometric error for a point cloud drawn from S^5 under the k-nearest neighbors flow

it should be noted that even points which lie far from the circle in Fig. 7 have been moved substantially closer to the circle in Fig. 9.

4.2 Five-Dimensional Sphere

To demonstrate the effectiveness of the smoothing gradient flow in higher dimensions, we investigate the effect of applying the flow to a point cloud sampled from the five dimensional unit sphere embedded in $\mathbb{R}^6$, i.e. the sphere S^5. Since we have increased the dimensionality, we also increase the number of points in the point cloud, to avoid issues with sampling density. In particular, we drew a point cloud of 10,000 points from S^5. Since we are using S^5, the geometric error function $E(Y)$ is nearly the same as (9), we just take the norm to be the Euclidean norm in $\mathbb{R}^6$. Similar to the previous case, we sampled 100 random point clouds using Gaussian noise and computed the geometric error for these point clouds across a range of parameter values.

The results of running the flow are shown in Fig. 10. From this image, we see that the gradient flow was able to drastically reduce the original geometric error, despite the problem being posed in $\mathbb{R}^6$. Thus, we see that the gradient flow still works as expected for higher dimensions. Interestingly, here we see that the optimal number of nearest neighbors is substantially lower for S^5 than it was for S^1. In particular, the k-nearest neighbors flow performs best with $k = 15$ and 10 iterations. The smallest error obtained under this flow is $E(Y) = 33.99$. Thus, the average geometric error, per point, was 0.0034 after the optimal number of iterations and under the optimal value of k (on the other hand, the average error in the S^1 case was 0.0099). The baseline error was $E(X) = 99.46$. Therefore, as we see in Fig. 10, running the gradient flow for any number of iterations from 1 to 10 and using any value of k between around $k = 3$ and $k = 110$, reduces the overall geometric error, in the sense of (9), in the point cloud.

From these results, we see that the gradient flow algorithms can be effective in higher dimensions. As for computational cost, let X_2 and X_5 be two point clouds drawn from S^2 and S^5, respectively. Let $|X_2| = |X_5| = 10,000$ and let $k = 1000$. Then a single iteration of the k-nearest neighbor gradient flow for X_2 takes approximately 5.706 seconds while a single iteration of the gradient flow for X_5 takes approximately 8.910 seconds. We obtained these numbers by running the gradient flow on several thousand point clouds sampled from both S^2 and S^5. We used a multi-threaded implementation[1] of the k-nearest neighbor gradient flow on an Intel Core-i7 6700k CPU. As expected, the high dimensional point cloud takes longer to process for the same point cloud size and number of neighbors. However, the extra expense is not large given the ability to smooth these higher dimensional point clouds. Furthermore, the computational cost is dominated by the size of the point cloud and the value of k, not the dimensionality of the point cloud. These factors drive the most expensive part of the algorithms, the determination of the k-nearest neighbors for each point in X.

4.3 Stanford Bunny

We now turn our analysis toward a point cloud sampled from the classic Stanford Bunny model, which has a rich history of use in testing various 3-dimensional modeling algorithms. The Bunny was created by Greg Turk and Marc Levoy in [32] while at Stanford University in 1994. Here, we analyze the performance of the smoothing algorithm on the resulting sampled point clouds. Later in the section, we will analyze the performance of the k-nearest neighbor algorithm on the Stanford Dragon as well.

For our evaluation using the Stanford Bunny, we will take a point cloud sampled from the surface of the Bunny and add artificial noise to the point cloud. Following this approach will allow us to measure the geometric error of the point cloud precisely, since we know the surface from which the point cloud was sampled. The version of the Stanford Bunny we use is a point cloud X of 35,947 points. The point cloud is contained in a rectangular cuboid R which measures approximately

$$R = (-0.0972, 0.0631) \times (0.0304, 0.1889) \times (-0.0633, 0.0607).$$

Given these dimensions, we add a randomly sampled 3-dimensional Gaussian distribution of mean 0 and bandwidth $\sigma = 0.001$ to each point of X, thereby producing

[1] Available at https://github.com/ponl/GradSmooth.

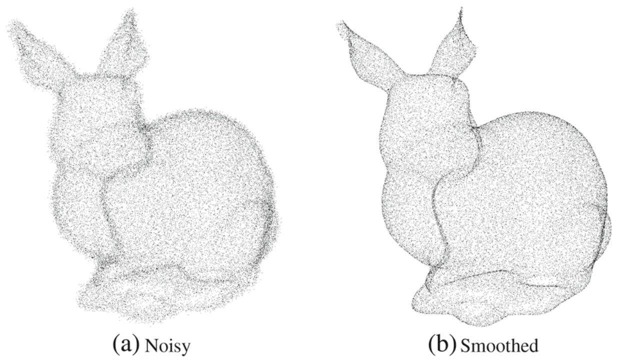
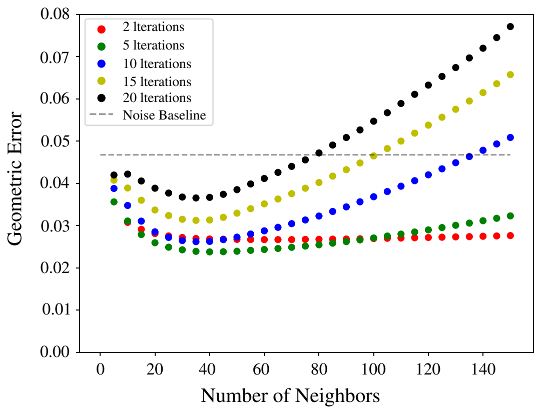


Fig. 11 Example point clouds for Stanford Bunny

Fig. 12 Geometric error for Stanford Bunny



a noisy point cloud X_σ. We then apply the k-nearest neighbor flow of Sect. 3.1 for varying values of k and for a varying number of iterations. For a given value of k and a given number of iterations T, we then measure the geometric error $L(X, Y)$ of the resulting smoothed point cloud Y using the following equation:

$$L(X, Y) = \sum_{y \in Y} \min_{x \in X} \|x - y\|^2. \tag{10}$$

Note that the above equation approximates the exact geometric error at each point. While ideally we would measure the error by taking the distance to the manifold, since the points of X all lie on the surface of the true manifold and provide a dense sampling of the manifold, the point cloud based error function L provides an accurate approximation of the true geometric error (Fig. 11).

The geometric error rates for the two algorithms are presented in Fig. 12. Here, we show the value of L obtained by setting $k = 5i$ where $i \in \mathbb{N}$ with $2 \leq i \leq 30$. We reported the error rate after T iterations, where T is set to be 2, 5, 10, 15, and finally 20. The *noise baseline* is also reported. This is the value of L computed directly on the pair X and X_σ (i.e. the initial error of the noisy point cloud).

Some immediate conclusions can be drawn from these two figures. In the k-nearest neighbor flow in Fig. 12, it is clear that the optimal value of k lies somewhere around $k = 35$. Additionally, the optimal number of iterations to run the algorithm was $T = 5$.

It is worth noting that the increasing error demonstrated by the k-nearest neighbor flow after the optimal parameters have been passed is due to volumetric shrinking of the point cloud. That is, when the point cloud is smoothed too much, its volume begins to shrink. Thus, while the shape of the point cloud may still be an accurate representation of the original point cloud, its volume has shrunken enough that the smoothed point cloud is much smaller than the original point cloud, hence the increasing geometric error.

Finally, notice that while the error rates for the k-nearest neighbor flow with $T = 5, 10, 15$ and 20 never cross each other as k increases, the error rates for $T = 2$ cross both the $T = 5$ and $T = 10$ error rates. This makes sense since during the initial evolution of the flow, the biggest decrease in error is produced. Then, once the optimal value of k is passed (i.e. around $k = 35$), the volume begins to shrink, an effect felt less when there are only two iterations. Also, notice that the error rates produced by $T = 20$ are the worst for every value of k. This indicates the k-nearest neighbor smoothing should be run for fewer iterations. In fact, other than for $T = 2$, the error rates decrease at every value of k for decreasing values of T.

4.4 Stanford Dragon

For our next experiment, we analyze the performance of the k-nearest neighbors flow when applied to a noisy version of the Stanford Dragon point cloud. The Stanford Dragon is a point cloud created by the Stanford University Computer Graphics Laboratory using the Cyberware 3030 MS scanner. The point cloud first appeared in [13]. To obtain the point cloud, approximately seventy scans were taken of the dragon, producing 437,645 points.

Similar to the Stanford Bunny in Sect. 4.3, the points lie directly on the sampled manifold. Therefore, to analyze the smoothing quality of our algorithms, we will artificially add noise to the point cloud. This allows us to precisely measure the geometric error of the resulting point cloud. As in the previous section, we will use notation X to represent the original point cloud, X_σ to represent the noisy point cloud, and Y to represent the smoothed point cloud. Additionally, we once again use the geometric error function L, given by (10).

The Stanford Dragon is contained in a rectangular cuboid R of dimensions

$$R = (-0.109, 0.097) \times (-0.527, 0.198) \times (-0.050, 0.042)$$

since the size of R is different for the Stanford Bunny than for the Stanford Dragon, we will use a different bandwidth σ for the Gaussian noise. In particular, we set $\sigma = 0.01$. We then perform a similar parameter sweep to the sweep we did in Sect. 4.3. Since this point cloud has an order of magnitude more points than the Stanford Bunny, we perform the sweep against larger values of k. Due to the increasing computational demands as we increase k, we sampled the parameter space more densely for lower

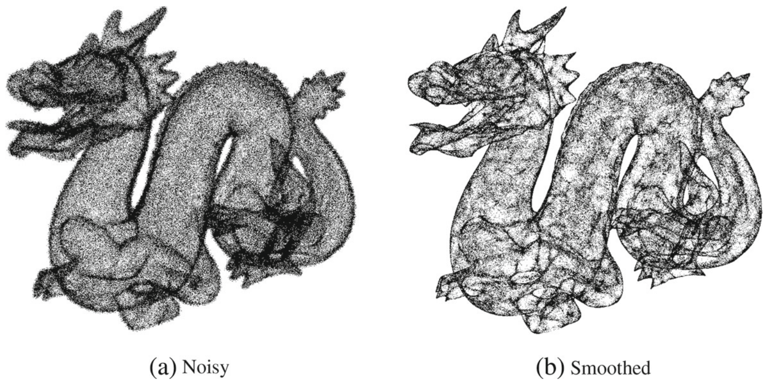
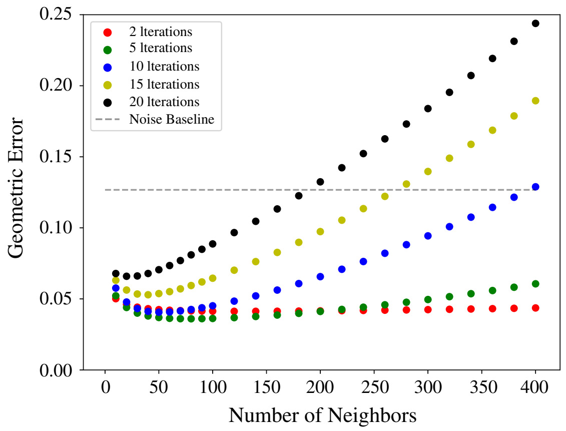


Fig. 13 Example point clouds for Stanford Dragon

Fig. 14 Geometric error for Stanford Dragon



values of k than for higher values of k. That is, we set $k = 10i$ for $i = 1, \dots, 10$ and then set $k = 20j$ for $j > 5$. We use the same iteration values as in the previous experiment.

The results of our parameter sweep are shown in Fig. 14. We show the value of $L(X, X_\sigma)$, represented by the dotted *noise baseline*. From these results, we can see that the optimal value for k lies somewhere around 50 when using 2, 5, or 10 iterations. When using more iterations, the optimal value of k shrinks to around 25. Once again, we see that fewer iterations is preferable for the k-nearest neighbors smoothing algorithm.

An example of a smoothed point cloud is shown in Fig. 13. Unlike the Stanford Bunny example, the clustering tendency of the gradient flow is on full display in this figure. Many regions of the dragon are over sampled. However, as seen in the geometric error plots (Fig. 14), these points are actually closer to the underlying surface, even if they are overly dense in some regions.

5 Conclusion

In this paper, we built a theoretical framework for studying the gradient flow induced by the squared distance to measure function under the empirical measure. This gradient flow, first described in [9], arises naturally when studying the squared distance to measure function. We saw that k-order Voronoi diagrams provide an effective geometric construction for studying the qualitative characteristics of the k-nearest neighbor flow. From this construction, we were able to determine fixed points, some positively invariant sets, and demonstrate that the flow does not give rise to any periodic orbits. Furthermore, we recognized the system as a Filippov system and were able to show that the flow does not exhibit any attractive sliding motion, an important result for numerical implementations.

In a future paper, we will investigate methods for reducing the clustering tendency of the k-nearest neighbor flow. A method for mitigating the clustering consists of adding a diffusive term to the flow so that points push against one another as they begin to cluster. A more sophisticated technique we will explore involves approximating the normal and tangent bundles of the sampled manifold and then projecting the k-nearest neighbor gradient flow onto the normal bundle of the manifold. This approach would mimic the surface reconstruction techniques of Alexa et al. in [1].

Furthermore, we will investigate appropriate modifications for making the flow adaptive to local conditions. This is desired since the optimal level of smoothing in one region of the point cloud may be suboptimal in another region. By allowing the gradient flow to adapt to the local geometry, this non-uniform smoothing can be performed. The difficulty arises in finding an appropriate formulation for the adaptivity which works in high dimensions. In [15], Dey and Sun created an adaptive surface reconstruction method, however this technique only works in $\mathbb{R}^3$ on two-dimensional surfaces. Our focus is on arbitrary dimensions, therefore this approach would not work. Instead, we seek a method which is computationally inexpensive in high dimensions, yet also effective at adapting the flow to local conditions.

References

1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Point set surfaces. In: Proceedings of the Conference on Visualization (VIS'01), pp. 21–28. IEEE, Washington, DC (2001)
2. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Vis. Comput. Graphics **9**(1), 3–15 (2003)
3. Amenta, N., Kil, Y.J.: Defining point-set surfaces. ACM Trans. Graphics **23**(3), 264–270 (2004)
4. Biák, M., Hanus, T., Janovská, D.: Some applications of Filippov's dynamical systems. J. Comput. Appl. Math. **254**, 132–143 (2013)
5. Bobenko, A.I., Schröder, P.: Discrete Willmore flow. In: Proceedings of the 3rd Eurographics Symposium on Geometry Processing (SGP'05), # 101. Eurographics Association, Aire-la-Ville (2005)
6. Brécheteau, C.: The DTM-signature for a geometric comparison of metric-measure spaces from samples (2017). arXiv:1702.02838
7. Buchet, M., Dey, T.K., Wang, J., Wang, Y.: Declutter and resample: towards parameter free denoising. In: Aronov, B., Katz, M.J. (eds.) Proceedings of the 33rd International Symposium on Computational Geometry (SoCG'17). Leibniz International Proceedings in Informatics, vol. 77, pp. 23:1–23:16. Schloss Dagstuhl. Leibniz-Zentrum für Informatik, Wadern (2017)

8. Chazal, F., Chen, D., Guibas, L.J., Jiang, X., Sommer, C.: Data-driven trajectory smoothing. Research Report RR-7754, INRIA (2011)
9. Chazal, F., Cohen-Steiner, D., Mérigot, Q.: Geometric inference for measures based on distance functions. Research Report RR-6930, INRIA (2010)
10. Chazal, F., Cohen-Steiner, D., Mérigot, Q.: Geometric inference for probability measures. Found. Comput. Math. **11**(6), 733–751 (2011)
11. Chazal, F., Fasy, B.T., Lecci, F., Michel, B., Rinaldo, A., Wasserman, L.A.: Robust topological inference: Distance to a measure and kernel distance (2014). arXiv:1412.7197
12. Chazal, F., Massart, P., Michel, B.: Rates of convergence for robust geometric inference. Electron. J. Stat. **10**(2), 2243–2286 (2016)
13. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96), pp. 303–312. ACM, New York (1996)
14. Dey, T.K.: Curve and surface reconstruction: algorithms with mathematical analysis. In: Cambridge Monographs on Applied and Computational Mathematics, vol. 23. Cambridge University Press, Cambridge (2007)
15. Dey, T.K., Sun, J.: An adaptive MLS surface for reconstruction with guarantees. In: Proceedings of the 3rd Eurographics Symposium on Geometry Processing (SGP'05), Art. No. 43. Eurographics Association, Aire-la-Ville (2005)
16. di Bernardo, M., Budd, C.J., Champneys, A., Kowalczyk, P.: Piecewise-Smooth Dynamical Systems: Theory and Applications. Applied Mathematical Sciences, vol. 163. Springer, London (2008)
17. di Bernardo, M., Liuzza, D.: Incremental stability of planar Filippov systems. In: Proceedings of the 2013 European Control Conference (ECC), pp. 3706–3711. IEEE, Washington, DC (2013)
18. Dong, M., Chou, W., Fang, B.: Underwater matching correction navigation based on geometric features using sonar point cloud data. Sci. Program. **2017**, 10 (2017)
19. Filippov, A.F.: Differential equations with discontinuous right-hand side. Mat. Sb. (N.S.) **51**(93), 99–128 (1960)
20. Gevaert, C., Persello, C., Sliuzas, R., Vosselman, G.: Informal settlement classification using point-cloud and image-based features from UAV data. ISPRS J. Photogramm. Remote Sens. **125**, 225–236 (2017)
21. Guibas, L.J., Mérigot, Q., Morozov, D.: Witnessed k-distance. In: Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG'11), pp. 57–64. ACM, New York (2011)
22. Hu, L., Xu, X., Wang, L., Guo, N., Xie, F.: 3D registration method based on scattered point cloud from B-model ultrasound image. In: Proceedings Volume 10245, International Conference on Innovative Optical Health Science, Art. No. 102450C (2017)
23. Ito, T.: A Filippov solution of a system of differential equations with discontinuous right-hand sides. Econ. Lett. **4**(4), 349–354 (1979)
24. Kantorovich, L., Rubinshtein, G.: On a space of totally additive functions. Vestn. Leningr. Univ. **13**(7), 52–59 (1958) (in Russian)
25. Kolluri, R., Shewchuk, J.R., O'Brien, J.F.: Spectral surface reconstruction from noisy point clouds. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'04), pp. 11–21. ACM, New York (2004)
26. Levin, D.: Mesh-independent surface interpolation. In: Brunnett, G., Hamann, B., Müller, H., Linsen, L. (eds.) Geometric Modeling for Scientific Visualization, pp. 37–49. Springer, Berlin (2004)
27. Malihi, S., Valadan Zoej, M.J., Hahn, M., Mokhtarzade, M., Arefi, H.: 3D building reconstruction using dense photogrammetric point cloud. In: Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLI-B3, pp. 71–74 (2016)
28. Mederos, B., Velho, L., de Figueiredo, L.H., de Figueirêdo, H.F.: Robust smoothing of noisy point clouds. In: Proceedings of the SIAM Conference on Geometric Design and Computing (2003)
29. Morgan, J.W., Tian, G.: Ricci flow and the Poincaré conjecture (2007). arXiv:math/0607607v2
30. Niyogi, P., Smale, S., Weinberger, S.: A topological view of unsupervised learning from noisy data. SIAM J. Comput. **40**(3), 646–663 (2011)
31. Steer, P., Lague, D., Gourdon, A., Croissant, T., Crave, A.: 3D granulometry: grain-scale shape and size distribution from point cloud dataset of river environments. In: Proceedings of the EGU General Assembly 2016, vol. 18, EGU2016-8514 (2016)

32. Turk, G., Levoy, M.: Zippered polygon meshes from range images. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'94), pp. 311–318. ACM, New York (1994)
33. White, B.: Evolution of curves and surfaces by mean curvature. In: Proceedings of the International Congress of Mathematics, vol. 1, pp. 525–538. Higher Education Press, Beijing (2002)