Title:
On the Uniform Resampling of a Point Cloud

Authors:
John K. Johnstone, jkj@uab.edu, Computer Science, UAB

Keywords:
Data cleaning, uniform sampling.

DOI: 10.14733/cadconfP.2026.135-139

Introduction

This paper considers a data cleaning problem: building a uniform sampling from a non-uniform sampling. To motivate the problem, consider the highly non-uniform sampling of Fig. 1a. It is a section of the inner limiting membrane from a dataset of the optic nerve head [7], resulting from an earlier image segmentation. A uniform sampling is desired for the calculation of mean statistics. A uniform sampling is also useful for multiresolution analysis [5].

A natural approach to this problem seems to be to fit a curve to the sampling, then uniformly sample this curve. However, there are two fundamental problems with this approach: the initial fit may fail (and does for the sampling of Fig. 1a). Second, getting a uniform sampling from the curve is not simply a matter of uniformly sampling in parameter space, since the curve will not be arc-length parameterized.

Our strategy to solve this problem is as follows. The first problem is to define the underlying shape by curve fitting. However, a highly non-uniform and dense sampling prevents a reliable fit. To discover the underlying curve, the first step is to downsample. Then the downsampled cloud is fit by a cubic B-spline. This fitting uses a centripetal parameterization, which is established to yield better shapes than uniform parameterization [2, 8, 10]. At this stage, fitting using a uniform parameterization can lead to undesirable behaviour such as cusps, self-intersections, and corners, since it ignores distance between points [2, 6, 8]. The next step is to upsample, using a dense uniform sampling in parameter space. The goal of this step is to define the underlying shape as a polygon. This will be oversampled, and it will not be uniform in Euclidean space since the curve's parameterization is not arc-length [4]. The last step is to uniformly sample this polygon in Euclidean space at the desired rate, another form of downsampling. After defining terminology in the next section, this algorithm is outlined in Algorithm 1, then in more detail in Algorithm 2 at the end of the paper.

The structure of the paper is as follows. We start by defining the problem and providing a general algorithm for its solution. The next sections discuss the initial downsampling, the normalization of the point cloud before this downsampling to simplify the choice of ϵ threshold during downsampling, the upsampling in parameter space, and the final downsampling, again in Euclidean space. We finish with conclusions and directions for future work.

Problem definition and algorithm

To define the problem, we first need some terminology.

Definition 0.1 *A* **sampling** *is a sequence of points in Euclidean d-space (Fig. 1a).*

Definition 0.2 *A* **uniform sampling** *is a sequence of points in Euclidean d-space where the Euclidean distance between any two consecutive points is the same (Fig. 1f).*

Definition 0.3 *Given a sampling that admits a fitting, the* **underlying shape** *of the sampling is its interpolating centripetally-parameterized B-spline curve (Fig. 1d for the sampling of Fig. 1c).*

Definition 0.4 *Given a sampling S, its* **uniform (ϵ-) resampling** *is a uniform sampling R where the Euclidean distance of S from the underlying shape of R is bounded above by a threshold $\epsilon > 0$ (Fig. 1f).*

We can now define the problem and our approach to solving it. Algorithm 1 defines the problem and our general algorithm for its solution. Figure 1 illustrates the algorithm.

Algorithm 1 Uniform resampling: general algorithm

Input: a sampling S, a size n, and a threshold $\epsilon > 0$
Output: a uniform ϵ-resampling of S of size n

1: Downsample S, while preserving its shape within ϵ.
2: Build the underlying shape I.
3: Uniformly and densely sample I in parameter space, yielding a polygonal chain C.
4: Uniformly sample C in Euclidean space.

Polygon decimation is a related problem. Indeed, decimation is the first step of our algorithm. However, polygon decimation aims to build a *small* resampling that preserves the shape, while Algorithm 1 aims to build a *uniform* resampling that preserves the shape. A polygon decimation will typically not be uniform. Note that Algorithm 1 allows the size of the uniform sampling to be controlled.

Downsampling

Since the fundamental reason that fitting fails is the oversampling of the cloud (compare the failure regions of the fitting in Fig. 1b to the oversampled regions of the original point cloud in Fig 1a), the first step is to downsample the point cloud while preserving its shape.

This uses polygon decimation, using the polygon defined by the sampling. There are many choices for decimation algorithm. A classic choice would be the Douglas-Peucker algorithm [1]. Given a sampling that begins at A and ends at B, it finds the farthest point F from the line segment AB and recursively decimates the two halves (from A to F, from F to B) if F is further from AB than ϵ. However, since this algorithm works globally, we use a different algorithm, which we call the **advancing-foot algorithm** [1], that changes the sampling locally and exerts more control over the sampling distance.

Imagine walking along the sampling. Each foot is always at a point of the sampling. We maintain the back foot planted at all times, advancing the front foot while all interior points between the two feet are within ϵ of the line between the two feet. That is, if the back foot is at p_i and the front foot is at p_{i+k}, the invariant is that all points $p_{i+1}, \ldots, p_{i+k-1}$ are within ϵ of the line $p_i p_{i+k}$. When the invariant fails, the front foot backs up to the previous point $q = p_{i+k-1}$, and all intermediate points $(p_{i+1}, \ldots, p_{i+k-2})$ are decimated. To restart the process, the back foot now replaces the front foot at q, and the front foot again starts edging forward through the sampling. The decimation stops when the walk reaches the last point (or the first point again).

The downsampling is shown in Fig. 1c.

Cube normalization

As in all decimation, an ϵ parameter is introduced that allows the decimated shape to differ from the original shape by up to ϵ. To give semantics to ϵ, the sampling is cube-normalized before downsampling.

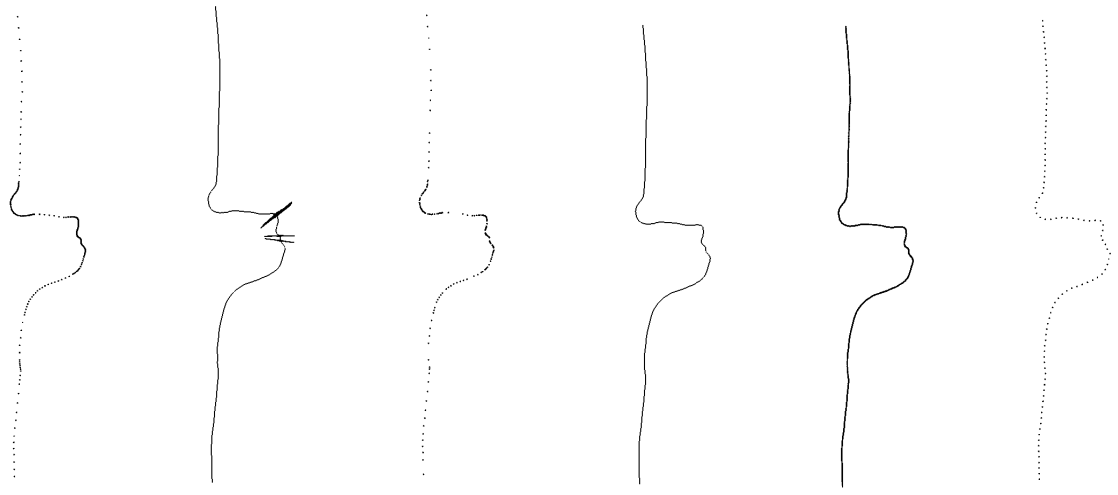

Fig. 1: Building a uniform sampling.
(a) A highly non-uniform sampling.
(b) A faulty cubic B-spline fit of the original sampling.
This fails in areas of oversampling:
see the 3 high-frequency oscillations in the right concavity.
(c) Downsampling.
(d) Cubic B-spline fit of the downsampling.
(e) Dense sampling of (d).
(f) Uniform resampling, with n=100 samples.

The **cube normalization** of a point cloud is its transform (by a translation and isotropic scaling) so that its axis-aligned bounding cube is exactly the unit cube $[-1, 1]^d$. Using vectorization, the C++/Eigen code for cube normalization is only 4 lines long. After cube normalization, all samplings are expressed in the same units, so that the same ϵ can be used for each sampling, at least within the same application.

Our default choice for ϵ is .0001 ($\frac{1}{20000}^{th}$ of the shape size). If desired, ϵ can be locked at .0001, and removed as an algorithm parameter, but we leave it as a parameter for additional flexibility. The value of ϵ can be informed by the accuracy of both the input and the output. The accuracy of the input sampling may be known (say based on the accuracy of the algorithm, such as image segmentation, that generated it). The size of ϵ may also be informed by the desired precision of the downstream task that uses the uniform sampling.

Cube normalization is similar to mean normalization, a preprocessing step in principal component analysis. However, the important step in cube normalization is the uniform scaling, which preserves shape, while mean normalization uses nonuniform scaling by the standard deviation of each coordinate. The translation of the center of the axis-aligned bounding cube to the origin is also different than the translation of the mean of the point cloud to the origin.

Fitting, upsampling, and another round of downsampling

The downsampled cloud is now ready for fitting, to define its underlying shape. We use a standard cubic B-spline interpolation with centripetal parameterization [3, 9]. As noted above, a centripetal parameterization is optimal for defining the shape [2, 6, 8]. The B-spline fit is shown in Fig. 1d.

The next step is to upsample, densely sampling the B-spline using a dense uniform sampling in parameter space. The goal of this step is to define the underlying shape as a polygon. The number of samples n_d in the dense sampling can be normalized by the perimeter of the downsampled polygon, so that it has consistent density across datasets. Since there is no harm in choosing a larger n_d, a conservative choice is possible. Upsampling allows the next step, uniform sampling, to work easily in Euclidean space. The upsampling is shown in Fig. 1e.

The upsampling is uniform in the spline's parameter space, but not uniform in Euclidean space, since the curve's parameterization is not arc-length [4]. It is also (purposely) oversampled. The last step is to uniformly downsample this polygon in Euclidean space at the desired rate. In this last downsampling, the polygon defined by the upsampling is uniformly sampled in Euclidean space. If the polygonal chain associated with the dense sampling is of perimeter p and a uniform sampling of size n is desired, the uniform sampling is gathered by stepping along the polygon using a step size of $\frac{p}{n}$. Note that this is a quite different downsampling from the earlier decimation, since it is inherently uniform.

In summary, the complete detailed algorithm is given in Algorithm 2.

Algorithm 2 Uniform resampling: detailed algorithm

Input: a sampling S, a size n, and a threshold $\epsilon > 0$
Output: S'''', a uniform ϵ-resampling of S of size n

1: $S' =$ cube-normalization(S)
2: Define ϵ (default: .0001)
3: $S'' =$ downsampling(S', ϵ), using an advancing-foot algorithm
4: $C =$ fit(S''), a cubic B-spline with centripetal parameterization
5: $S''' =$ dense-sampling(C, n_d), sampling uniformly in parameter space
6: $S'''' =$ uniform-sampling(S''', n), sampling uniformly in Euclidean space
7: Transform S'''' back to its location and scale, by inverting the cube-normalization.

Conclusions:

This paper has presented a data cleaning algorithm: an algorithm for uniform resampling from an arbitrary sampling. This is particularly attractive when the initial sampling is highly nonuniform and fitting algorithms fail, as is often the case in point clouds from image segmentation. The algorithm involves downsampling and upsampling, both in the ambient Euclidean space and in parameter space. Although modest, we have found this technique to be valuable across many applications. A natural evolution of these ideas is to the uniform resampling of other geometric structures.

References:

[1] Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10(2), December 1973, 112-122.

[2] Epstein, M.: On the Influence of Parameterization in Parametric Interpolation. SIAM Journal on Numerical Analysis, 13(2), 1976, 261-268. https://doi.org/10.1137/0713025

[3] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide. Academic Press, 3rd edition, 1993. https://doi.org/10.1016/C2009-0-22351-8

[4] Farouki, R., Sakkalis, T.: Real rational curves are not 'unit speed'. Computer Aided Geometric Design, 8(2), 1991, 151-157. https://doi.org/10.1016/0167-8396(91)90040-I

[5] Finkelstein, A., Salesin, D.: Multiresolution Curves. SIGGRAPH 1994, 261–268. https://doi.org/10.1145/192161.192223

[6] Floater, M.: On the deviation of a parametric cubic spline interpolant from its data polygon. Computer Aided Geometric Design, 25(3), 2008, 148-156. https://doi.org/10.1016/j.cagd.2007.08.001

[7] Johnstone, J., Fazio, M., Rojananuangnit, K., Smith, B., Clark, M., Downs, J.C., Owsley, C., Girard, M., Mari, J.-M., Girkin, C.: Variation of the Axial Location of Bruch's Membrane Opening with Age, Choroidal Thickness and Race. Investigative Ophthalmology and Visual Science 55:3, 2014, 2004–2009. https://doi.org/10.1167/iovs.13-12937

[8] Lee, E.: Choosing nodes in parametric curve interpolation. Computer Aided Design, 21(6), 1989, 363-370. https://doi.org/10.1016/0010-4485(89)90003-1

[9] Piegl, L., Tiller, W.: The NURBS Book. Springer, 1997. https://doi.org/10.1007/978-3-642-59223-2

[10] Yuksel, C., Schaefer, S., Keyser, J.: On the Parameterization of Catmull-Rom Curves. SIAM/ACM Joint Conference on Geometric and Physical Modeling, 2009, 47-53. https://doi.org/10.1145/1629255.1629262