Fast Capacity Constrained Voronoi Tessellation

Hongwei Li* Diego Nehab† Li-Yi Wei† Pedro V. Sander* Chi-Wing Fu‡

*Hong Kong UST †Microsoft Research ‡Nanyang Tech. Univ.

Abstract

Capacity constrained Voronoi tessellation (CCVT) [Balzer et al. 2009] addresses a crucial quality issue of Lloyd relaxation but at the expense of slower computation, which could hinder its potential wide adoption. We present a fast capacity constrained Voronoi tessellation algorithm which is orders of magnitude faster than the original method proposed by Balzer et al. [2009] (and $10\times$ faster than a previous accelerated implementation of the same technique) while maintaining excellent distribution quality and scaling very well as the number of points increase.

Keywords: capacity constrained, point distribution, Voronoi tessellation, relaxation, blue noise, sampling, stippling

1 Introduction

Though it is widely employed to generate point distributions for a variety of applications, Lloyd relaxation [Lloyd 1982] has the tendency to settle into semi-regular structures, which harms distribution quality severely. Recently, Balzer et al. [2009] addressed this issue via capacity constrained Voronoi tessellation (CCVT). Their key idea is to maintain capacity of the Voronoi regions to discourage the distribution from settling into a semi-regular structure. Due to the importance of Lloyd's method, CCVT has the potential to become a popular replacement for many applications. Unfortunately, CCVT is computationally much more expensive than the original Lloyd's method, a potential hindrance for widespread adoption of this improved method. Even though accelerations exist for traditional relaxation, they are not applicable to [Balzer et al. 2009] due to the use of capacity constraints. To address this performance issue, we present a fast capacity constrained Voronoi tessellation (CCVT) algorithm. Our method is extended from CCVT [Balzer et al. 2009], but we make several algorithmic innovations to remove performance bottlenecks of the original method without affecting quality.

2 Algorithm

The most time consuming part of original CCVT algorithm is in swapping site affiliations among the heap points. It takes $O(\frac{m}{n}\log\frac{m}{n})$ between every two sites, where $\frac{m}{n}$ is the average number of points per site. Our new method for tackling this problem is to seek the optimal assignment of points between them by using median cut. For a given pair of sites s_i and s_j, we first move all points to site s_i, and decide which points to move to site s_j. Each time we move a point p from site s_i to site s_j, the change in energy is

$$\Delta\mathbf{E}(p, s_i \to s_j) = |p - s_j|^2 - |p - s_i|^2 \qquad (1)$$

Given that we have to move exactly $c(s_j)$ points from s_i to s_j to preserve the capacity constraints, the question now becomes which $c(s_j)$ out of a total $c(s_i)+c(s_j)$ points to move in order to minimize the above energy term. This can be achieved by splitting the points between s_i and s_j at the median point τ whose energy $\Delta\mathbf{E}(\tau, s_i \to s_j)$ lies on the median among all $c(s_i) + c(s_j)$ points. Since the median can be found in $O(\frac{m}{n})$ time (e.g. with quickselect), this method achieves the intended time complexity.

We also present several other accelerations, ranging from high level parallelization to low-level performance optimizations.

# sites	1024	4096	4096	16384
# points	256	256	1024	256
original CCVT	454.39	N/A	N/A	N/A
+ bounding circle and coherent init	5.96	34.99	192.71	270.63
our method				
+ median site swap	66.95	1523.41	22331.00	30599.17
+ bounding circle and coherent init	2.27	20.44	127.55	274.92
+ point dist prune	2.20	18.85	126.75	257.78
+ multi-level	1.75	25.78	114.48	400.78
+ OpenMP	1.63	12.86	67.26	136.43
+ 2-stage pair caching	0.70	3.80	28.58	27.49

Table 1: Performance of three relaxation algorithms. We measured the running time (in seconds) between [Balzer et al. 2009] and our fast CCVT on different combinations of # sites and points. We also break down the performance impact for each added acceleration technique. N/A indicates entries that are too slow for measurement. All performance numbers are measured on a PC with Windows 7 + Intel Xeon E5520 (quad core) @ 2.25GH + 12GB RAM.

Bounding circle pruning + coherent initialization These are accelerations from the original implementation of [Balzer et al. 2009]. The former skips swapping points between sites whose bounding circles (of all affiliated points) do not overlap. The latter initializes points to sites via a coherent assignment according to point-site distance as in [Lloyd 1982] instead of randomly as in the descriptions of [Balzer et al. 2009].

Parallel site swap Another main accleration is that the swapping operation can be performed in parallel on independent pairs of sites. We arrange the site pairs such that they are grouped in large clusters of independent site pairs.

Two-stage pair caching We partition the process into two stages. In the full stage we consider all pairs of site and in cached stage only process the cached pairs (passing the bounding circle test). Since only a subset of pairs will swap points in each iteration, by reducing the number of pairs in each iteration we can save a significant amount of computation.

Median cut pruning When considering swapping between two sites s_i and s_j we first compute $\Delta\mathbf{E}(p, s_i \to s_j)$ for all points belonging to s_i and s_j (see Equation 1). If $\max(\Delta\mathbf{E}(p_j, s_i \to s_j), p_j \in s_j) < \min(\Delta\mathbf{E}(p_i, s_i \to s_j), p_i \in s_i)$, then the points are already optimally allocated between these two sites and the median computation can be skipped.

Point distance pruning When the distance between a point and a site is bigger than a pre-computed value, we ignore the point in the median cut computation. This is a more aggressive optimization than bounding circle pruning and median cut pruning.

Multilevel optimization We start the simulation with a smaller number of points m and increase the capacity upon convergence. This reduces point swapping in some configurations.

References

BALZER, M., SCHLOMER, T., AND DEUSSEN, O. 2009. Capacity-constrained point distributions: A variant of lloyd's method. In *SIGGRAPH '09*, 86:1–8.

LLOYD, S. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2, 129–137.

I3D 2010, Washington, DC, February 19 – 21, 2010.
ISBN 978-1-60558-938-1/10/0002