



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

Print/export  
[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)


Languages  
[Français](#)  
[Українська](#)  
[اردو](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Search 

# Lloyd's algorithm

From Wikipedia, the free encyclopedia

In [computer science](#) and [electrical engineering](#), **Lloyd's algorithm**, also known as **Voronoi iteration** or relaxation, is an algorithm named after Stuart P. Lloyd for finding evenly-spaced sets of points in subsets of [Euclidean spaces](#), and partitions of these subsets into well-shaped and uniformly sized convex cells.<sup>[1]</sup> Like the closely related [k-means clustering](#) algorithm, it repeatedly finds the [centroid](#) of each set in the partition, and then re-partitions the input according to which of these centroids is closest. However, Lloyd's algorithm differs from *k*-means clustering in that its input is a continuous geometric region rather than a discrete set of points. Thus, when re-partitioning the input, Lloyd's algorithm uses [Voronoi diagrams](#) rather than simply determining the nearest center to each of a finite set of points as the *k*-means algorithm does.

Although the algorithm may be applied most directly to the [Euclidean plane](#), similar algorithms may also be applied to higher-dimensional spaces or to spaces with other [non-Euclidean](#) metrics. Lloyd's algorithm can be used to construct close approximations to [centroidal Voronoi tessellations](#) of the input,<sup>[2]</sup> which can be used for [quantization](#), [dithering](#), and [stippling](#). Other applications of Lloyd's algorithm include smoothing of [triangle meshes](#) in the [finite element method](#).

## Contents

[hide]

- [1 Algorithm description](#)
- [2 Convergence](#)
- [3 Applications](#)
- [4 Different distances](#)
- [5 See also](#)
- [6 References](#)

## Algorithm description [edit]

Lloyd's algorithm starts by an initial placement of some number *k* of point sites in the input domain. In mesh smoothing applications, these would be the vertices of the mesh to be smoothed; in other applications they may be placed at random, or by intersecting a uniform triangular mesh of the appropriate size with the input domain.

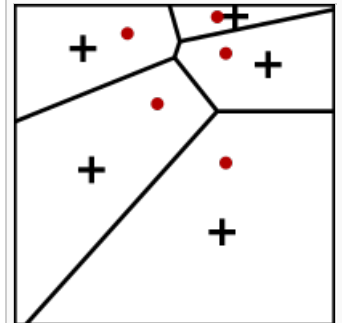
It then repeatedly executes the following relaxation step:

- The [Voronoi diagram](#) of the *k* sites is computed.
- Each cell of the Voronoi diagram is integrated and the centroid is computed.
- Each site is then moved to the centroid of its Voronoi cell.

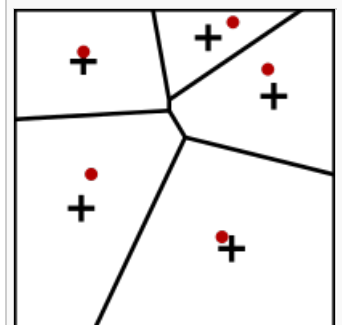
Because Voronoi diagram construction algorithms can be highly non-trivial, especially for inputs of dimension higher than two, the steps of calculating this diagram and finding the centroids of its cells may be approximated by a suitable discretization in which, for each cell of a fine grid, the closest site is determined, after which the centroid for a site's cell is approximated by averaging the centers of the grid cells assigned to it. Alternatively, [Monte Carlo methods](#) may be used, in which random sample points are generated according to some fixed underlying probability distribution, assigned to the closest site, and averaged to approximate the centroid for each site.

### Example of Lloyd's algorithm.

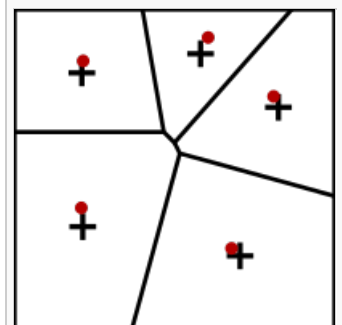
The Voronoi diagram of the current points at each iteration is shown. The plus signs denote the centroids of the Voronoi cells.



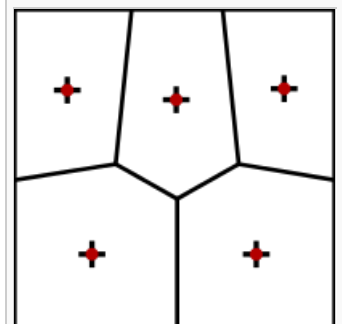
First iteration



Second iteration



Third iteration



Fifteenth iteration

In the last image, the points are very near the centroids of the Voronoi cells. A centroidal Voronoi tessellation has been found.

## Convergence [edit]

Each time a relaxation step is performed, the points are left in a slightly more even distribution: closely spaced points move farther apart, and widely spaced points move closer together. In one dimension, this algorithm has been shown to converge to a centroidal Voronoi diagram, also named a [centroidal Voronoi tessellation](#).<sup>[3]</sup> In higher dimensions, some slightly weaker convergence results are known.<sup>[4][5]</sup>

The algorithm converges slowly or, due to limitations in numerical precision, may not converge. Therefore, real-world applications of Lloyd's algorithm typically stop once the distribution is "good enough." One common termination criterion is to stop when the maximum distance moved by any site in an iteration falls below a preset threshold. Convergence can be accelerated by over-relaxing the points, which is done by moving each point  $\omega$  times the distance to the center of mass, typically using a value slightly less than 2 for  $\omega$ .<sup>[6]</sup>

## Applications [edit]

Lloyd's method was originally used for scalar quantization, but it is clear that the method extends for vector quantization as well. As such, it is extensively used in [data compression](#) techniques in [information theory](#). Lloyd's method is used in computer graphics because the resulting distribution has [blue noise](#) characteristics (see also [Colors of noise](#)), meaning there are few low-frequency components that could be interpreted as artifacts. It is particularly well-suited to picking sample positions for [dithering](#). Lloyd's algorithm is also used to generate dot drawings in the style of [stippling](#).<sup>[7]</sup> In this application, the centroids can be weighted based on a reference image to produce stipple illustrations matching an input image.<sup>[8]</sup>

In the [finite element method](#), an input domain with a complex geometry is partitioned into elements with simpler shapes; for instance, two-dimensional domains (either subsets of the Euclidean plane or surfaces in three dimensions) are often partitioned into triangles. It is important for the convergence of the finite element methods that these elements be well shaped; in the case of triangles, often elements that are nearly equilateral triangles are preferred. Lloyd's algorithm can be used to smooth a mesh generated by some other algorithm, moving its vertices and changing the connection pattern among its elements in order to produce triangles that are more closely equilateral.<sup>[9]</sup> These applications typically use a smaller number of iterations of Lloyd's algorithm, stopping it to convergence, in order to preserve other features of the mesh such as differences in element size in different parts of the mesh. In contrast to a different smoothing method, [Laplacian smoothing](#) (in which mesh vertices are moved to the average of their neighbors' positions), Lloyd's algorithm can change the topology of the mesh, leading to more nearly-equilateral elements as well as avoiding the problems with tangling that can arise with Laplacian smoothing. However, Laplacian smoothing can be applied more generally to meshes with non-triangular elements.

## Different distances [edit]

Lloyd's algorithm is usually used in a [Euclidean space](#). The Euclidean distance plays two roles in the algorithm: it is used to define the Voronoi cells, but it also corresponds to the choice of the centroid as the representative point of each cell, since the centroid is the point that minimizes the average squared Euclidean distance to the points in its cell. Alternative distances, and alternative central points than the centroid, may be used instead. For example, [Hausner \(2001\)](#) used a variant of the [Manhattan metric](#) (with locally-varying orientations) to find a tiling of an image by approximately-square tiles whose orientation aligns with features of an image, which he used to simulate the construction of tiled [mosaics](#).<sup>[10]</sup> In this application, despite varying the metric, Hausner continued to use centroids as the representative points of their Voronoi cells. However, for metrics that differ more significantly from Euclidean, it may be appropriate to choose the minimizer of average squared distance as the representative point, in place of the centroid.<sup>[11]</sup>

## See also [edit]

- The [Linde–Buzo–Gray algorithm](#), a generalization of this algorithm for vector quantization
- [Farthest-first traversal](#), a different method for generating evenly-spaced points in geometric spaces
- [Mean shift](#), a related method for finding maxima of a density function

## References [edit]

- ↑ Lloyd, Stuart P. (1982), "Least squares quantization in PCM", *IEEE Transactions on Information Theory* **28** (2): 129–137, doi:10.1109/TIT.1982.1056489 ↗.
- ↑ Du, Qiang; Faber, Vance; Gunzburger, Max (1999), "Centroidal Voronoi tessellations: applications and algorithms", *SIAM Review* **41** (4): 637–676, doi:10.1137/S0036144599352836 ↗.
- ↑ Du, Qiang; Emelianenko, Maria; Ju, Lili (2006), "Convergence of the Lloyd algorithm for computing centroidal

- Voronoi tessellations", *SIAM Journal on Numerical Analysis* **44**: 102–119, doi:[10.1137/040617364](https://doi.org/10.1137/040617364).
4. <sup>^</sup> Sabin, M. J.; Gray, R. M. (1986), "Global convergence and empirical consistency of the generalized Lloyd algorithm", *IEEE Transactions on Information Theory* **32** (2): 148–155, doi:[10.1109/TIT.1986.1057168](https://doi.org/10.1109/TIT.1986.1057168).
  5. <sup>^</sup> Emelianenko, Maria; Ju, Lili; Rand, Alexander (2009), "Nondegeneracy and Weak Global Convergence of the Lloyd Algorithm in  $\mathbb{R}^d$ ", *SIAM Journal on Numerical Analysis* **46**: 1423–1441, doi:[10.1137/070691334](https://doi.org/10.1137/070691334).
  6. <sup>^</sup> Xiao, Xiao. "Over-relaxation Lloyd method for computing centroidal Voronoi tessellations." (2010).
  7. <sup>^</sup> Deussen, Oliver; Hiller, Stefan; van Overveld, Cornelius; Strothotte, Thomas (2000), "Floating points: a method for computing stipple drawings", *Computer Graphics Forum* **19** (3): 41–50, doi:[10.1111/1467-8659.00396](https://doi.org/10.1111/1467-8659.00396), Proceedings of *Eurographics*.
  8. <sup>^</sup> Secord, Adrian (2002), "Weighted Voronoi stippling", *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, *ACM SIGGRAPH*, pp. 37–43, doi:[10.1145/508530.508537](https://doi.org/10.1145/508530.508537).
  9. <sup>^</sup> Du, Qiang; Gunzburger, Max (2002), "Grid generation and optimization based on centroidal Voronoi tessellations", *Applied Mathematics and Computation* **133** (2–3): 591–607, doi:[10.1016/S0096-3003\(01\)00260-0](https://doi.org/10.1016/S0096-3003(01)00260-0).
  10. <sup>^</sup> Hausner, Alejo (2001), "Simulating decorative mosaics", *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, *ACM SIGGRAPH*, pp. 573–580, doi:[10.1145/383259.383327](https://doi.org/10.1145/383259.383327).
  11. <sup>^</sup> Dickerson, Matthew T.; Eppstein, David; Wortman, Kevin A. (2010), "Planar Voronoi diagrams for sums of convex functions, smoothed distance and dilation", *Proc. 7th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2010)*, pp. 13–22, [arXiv:0812.0607](https://arxiv.org/abs/0812.0607), doi:[10.1109/ISVD.2010.12](https://doi.org/10.1109/ISVD.2010.12).

Categories: [Geometric algorithms](#) | [Mathematical optimization](#)

This page was last modified on 16 July 2015, at 01:26.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)



WIKIMEDIA  
project



Powered By  
MediaWiki