Evenly sampling on a sphere: finding almost-regular polyhedra

Dr. Nicolás Passarelli

April 6, 2024

Abstract

This project is about evenly distributing samples/points in a sphere surface. The iterative solution presented is based in the relaxation of Voronoi diagrams asociated with the samples. The workflow of the python code that implements the solution starts with a popular approximation and converges to a state of maximum separation of the points. From this final state, two polyhedra are derived: one with a number of faces equals to the number of sampled points, and the other with the same number of vertices. The code naturally arrive to the platonic solids, wich have exactly equal faces and edges.

Intro

Good sampling of a sphere surface is critical; lots of scientific models are built on spheres, and evaluating a given quantity on a spherical contour is a common procedure. Meshes of the sphere are widely used as buiding blocks in 3D design software, and an even sampling can give a better spherical appereance using less points.

Regular polyhedra, can be used to sample evenly, but a small set of numbers of samples have to offer. Other numbers of points will give irregular polyhedra, and the coordinates must be found numerically. A popular method of sampling points in the sphere is to sample with spirals that fit in the surface, this is very commmon even in design sofware, but it is demonstrated is not a proper solution. The ammount of rotation along the main axis of the spiral is chosen to be multiples of the golden ratio, to obtain a well spread distribution. This "Fibonacci" method gives a spread distribution of points, but only on the eye.

Other approach is to define a 'potential energy' to the points and simulate their evolution with classical laws of motion. Routines of optimization can also been applied, for which a well defined cost function or figure of merit is needed. This methods are computationally mouch more expensive than the Fibonacci one, but gives better results. One can always make a library of spheres and import from memory.

This code make use of Sperical Voronoi diagrams, which divides the sphere surface in regions by proximity to a set of points. The regions are polygons and its area is a good measure of the isolation of the point. In a even distribution, all polygons should have the same area, and must be centered in their corresponding point. Thus, in a iterative process of moving the points towards the centroid of the corresponding Voronoi region, the diagram "relaxes" to a more even and regular distribution. As a starting point we use the Fibonacci method so that both solutions can be compared.

Keywords

Sphere surface meshing, Sampling in spherical coordinates, Spherical Voronoi diagrams, Spherical polygons.

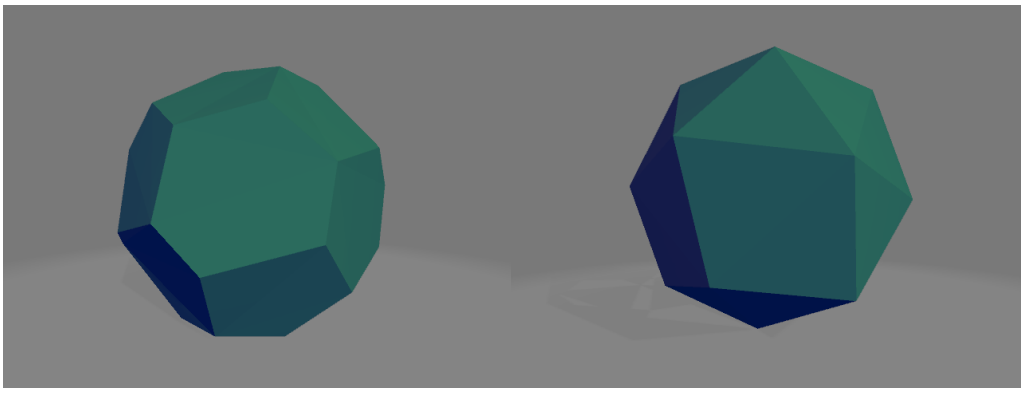

Figure 1: Two polyedra created with sampling 19 points. Left has 19 faces, Right has 19 vertices.

Disclaimer

There is maybe thousand of papers addressing this issue, this is not a new problem. I haven't read any, sorry if a short of technical vocabulary since I am not a mathematician nor a professional developer, so my codes for sure is up to improve. I just want to do my part towards a world with a more equalitarian sampling of spheres surfaces.

The routine last one minute or two in my i7 laptop (note I studied the smaller numbers), offering a nice visualisation of the process of relaxation. The code stores all the data and some plots and a .obj file to avoid re-running. The output for the first 16 samples, which results in 32 diffent polyhedra, are also included with the code.

Implementation

The code is completely written on python. The os library is imported for managing files, numpy is used for all the calculations, matplotlib for plotting, and scipy for the Spherical Voronoi and Convex Hull classes. The evolution of the process can be seen in console. An animation can be made of it, but calling a trustable external software Image Magick (I had problems to implement 3D animations full within python).

The workflow starts with the Fibonacci distribution, easily implemented in a function call. There is two ways to do it, generating angles with the golden relation between x and y coordinates or between x and z, and then calculate the remaining coordinate. The first one generate spiral that 'rotates' around the poles ($\phi = 0, \pi$), the other rotates around $\theta = -\pi, \pi$, as can be seen in Figure 2.

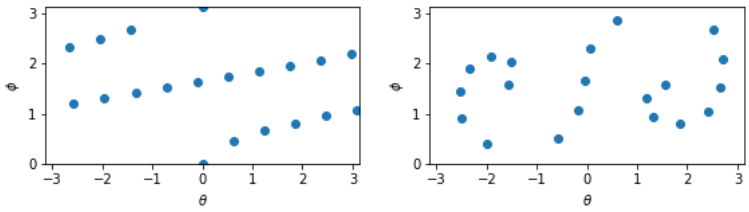

Figure 2: Fibonacci angular distributions of 20 points.

Then a loop starts, the Voronoi diagram is calculated using scipy routine, the sperical ploygons are obtained. The centroids are calculated using

$$\boldsymbol{C} = R \sum_{j=1}^{N} \hat{\boldsymbol{n}}_{j,j+1} d_{j,j+1}$$

Where the sum runs over the pair of vertices that make the closed polygon. R is the radius of the sphere. $\hat{\boldsymbol{n}}$ are the unitarian normals to the big circle generated by a points. d is the angle between the vector vertices.

$$\hat{\boldsymbol{n}}_{i,j} = \frac{\hat{\boldsymbol{v_1}} \times \hat{\boldsymbol{v_2}}}{\|\hat{\boldsymbol{v_1}} \times \hat{\boldsymbol{v_2}}\|}$$

$$d_{i,j} = \arccos\left(\hat{\boldsymbol{v_1}} \cdot \hat{\boldsymbol{v_2}}\right)$$

where $\hat{\boldsymbol{v_1}}$ and $\hat{\boldsymbol{v_2}}$are the normalized vertices coordinates. In the code the angle is calculated using the half angle version of this common formula, for computational reasons.

Once with the centroids coordinates, the sampling points moves towars them in a step that represents a given fraction of the way: a 20% worked fine. The loop then continues plotting everything, and calculating the new Voronoi distribution, centroids, moving points, plotting, etc...

The code ends when one of three criteria is satisfied: 1- a given number of maximum iteration is reached, or the relative standar deviation RSTD of the polygons area close to zero, 2- the RSTD of the RSTD of the areas is close to zero in the last iterations, 3- number of iterations exeeded. The Platonic solids converge with the first criteria, the rest ideally with the second before maximum iterations are reached. Once converged it is saved, so the postprocess can be done in a different script.

In the end of the process the centroids and sample points coincides in a good proportion. The last Voronoi diagram is the first polyhedra one obtain, the number of faces coincide with the number of sampling points s. But one can go further and take this evenly spaced points are the vertices of another polyhedron. For this the convex hull calculation is done, calling the scipy class. This polyhedron have $2s - 4$ faces for some reason. Finally both objects are exported in .obj format for importing it in most 3D software.

Some plots

I explored the lowest 16 samples possible, ranging from 4 points to 20 and obtaining 32 polyhedra. In all cases the area of the Voronoi polygons are within a 5%. The two bodies that emerges from a given number of points are very distinct. The first may present different kind of polygons, but the second is always composed of triangles. There are some sample numbers, that resulted in shapes I recognize:

For the first class: $s = 4$ tetrahedron, $s = 5$ triangular prism, $s = 6$ cube, $s = 7$ pentagonal prism, $s = 12$ dodecahedron.

For the second class: $s = 4$ also tetrahedron, $s = 5$ trigonal bipyramid, $s = 6$ octahedron, $s = 7$ pentagonal bipyramid, $s = 12$ Icosahedron.

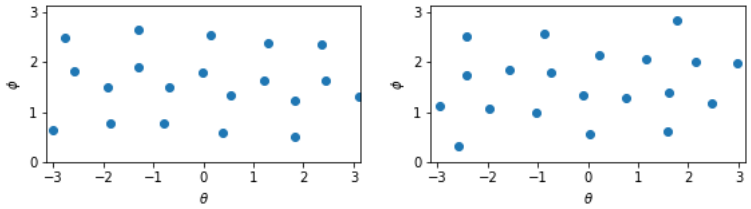

Figure 3: Angular distributions of 20 points. Left: 12 samples, second class. Right: 20 samples, first class.

Figure 3 shows the results for an angular distribution of 20 points. The letf panel is the Icosahedron, whereas the right one is an almost-regular distribution. Note how this two images tile the angular space much more evenly than the Fibonacci distribution of same number in Figure 1.

In Figure 4 there are compared the vertices of both kind of object for several samples. Note that the orange one are the polygons from the Voronoi diagram of the blue points. The areas seems off in this figure because we are looking angles; the closer to $\phi = 0$, the bigger seems the spacing between points, this would compensate the $\sin(\phi)$ in the diferential of area. For the low number of sampling points, the faces seems to have less vertices on average.

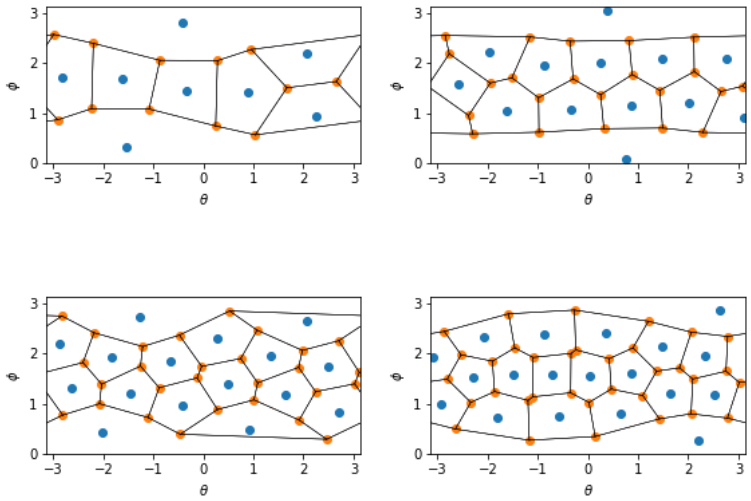

Figure 4: Angular distributions of 20 points. In orange first class object vertices, and in blue the second kind. Top Left: 8 samples. Top Right: 13. Bottom Left: 16. Bottom Right: 19.

Usage

There is two python executables:

-'relax-points.py' that makes the calculation, and show progress on display. It opens a folder, named with the number of samples, to save the data of the last voronoi diagram. Also, it allows to save the plots for making a .gif and plot like figure 4.

-'read_and_export.py' it reads the numbered directories, creates the convex hull polyhedron and write the two .obj files with the polyhedra obtained.

The codes are separated in sections, with notes inside. Only the first section contains the parameters of user control. In principle it is just enter the number of samples and run.