The Death Map: Reading Nature's Geometry

John Burkardt,
Information Technology Department,
Virginia Tech.
.....
Mathematics Department,
Ajou University,
Suwon, Korea,
12 May 2009
.....
https://people.sc.fsu.edu/~jburkardt/presentations/death_map_2009_ajou.pdf

THE DEATH MAP

1. **A CASEBOOK**
2. Death in Golden Square
3. The Voronoi Diagram
4. Euler's Formula
5. Voronoi Computation
6. Centered Systems
7. Conclusion

Casebook: The Giant's Causeway

Casebook: The Giant's Causeway

On the northeast coast of Ireland, there is a "paved" area of 40,000 interlocking (mostly) hexagonal stone pillars, roughly the same size.

Some of the pillars reach up to a cliff, and form a staircase that disappears into the sea.

How can any physical process create such patterns?

Casebook: The Giant's Causeway

Casebook: The Spitting Fish

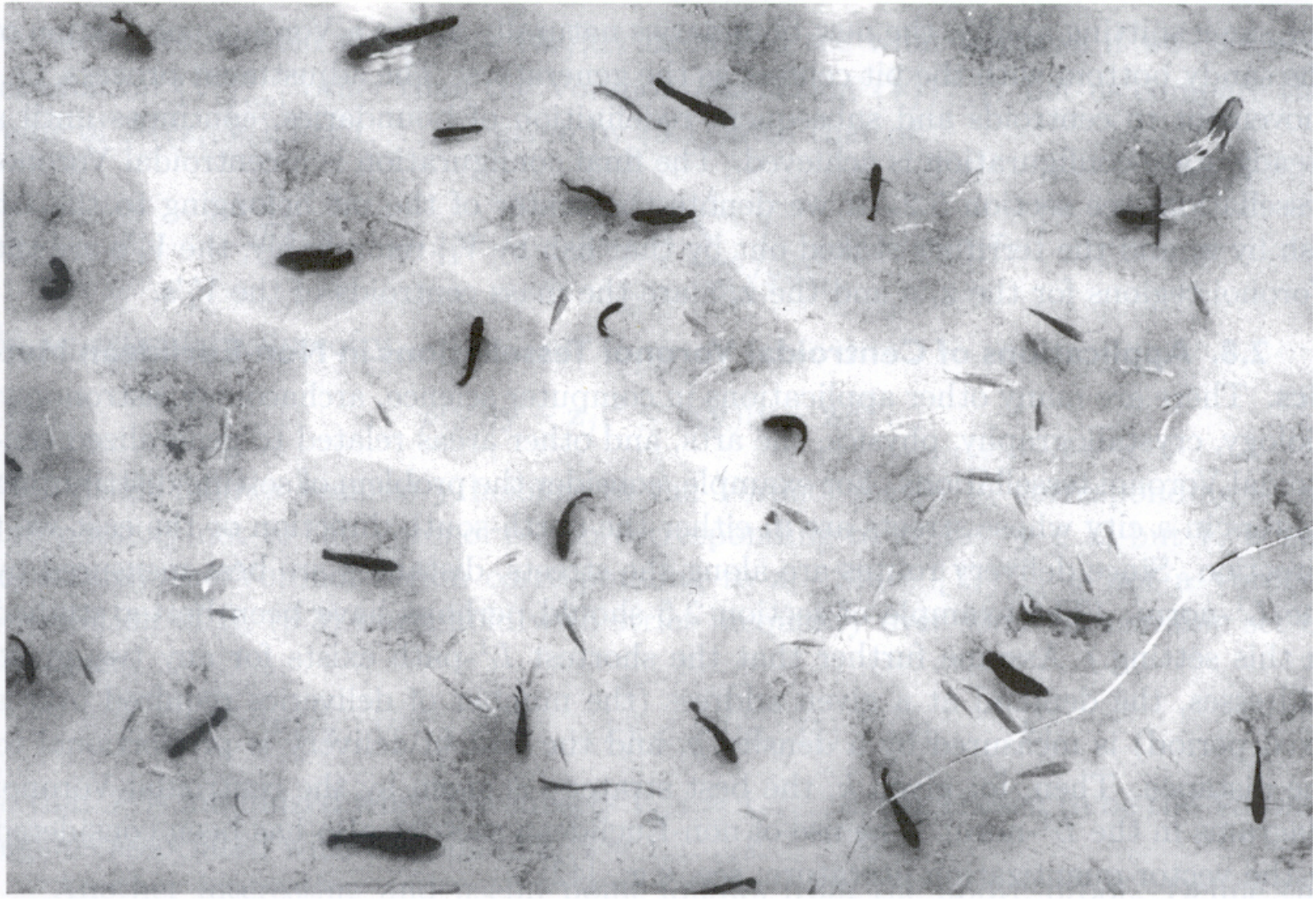
Casebook: The Spitting Fish

The *Tilapia mossambica* is sometimes called the spitting fish.

1. It raises its young in its mouth, spitting them out when they're ready.
2. It breeds by building a nest on the river bottom, picking up stones and spitting them away

Without taking geometry class, these fish build polygonal networks of nests

Casebook: The Spitting Fish

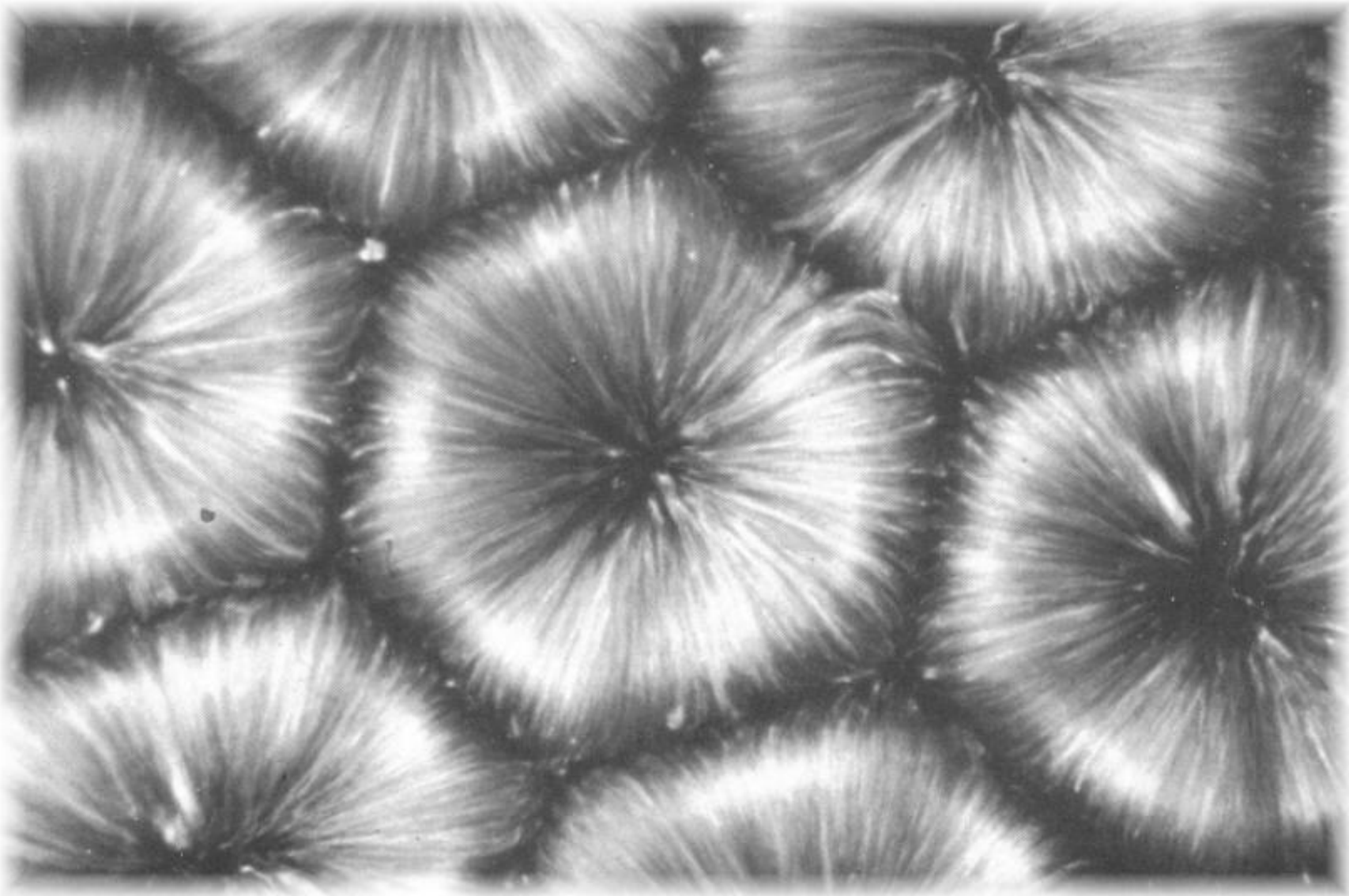


Casebook: The Spitting Fish

- Each cell is about the same size;
- Each cell is about the same shape;
- The cells are "centered".

As far as we know, the fish do not use a blueprint, nor do they have a planning committee meeting!

Casebook: Rayleigh-Benard Convection Cells



Casebook: Rayleigh-Benard Convection

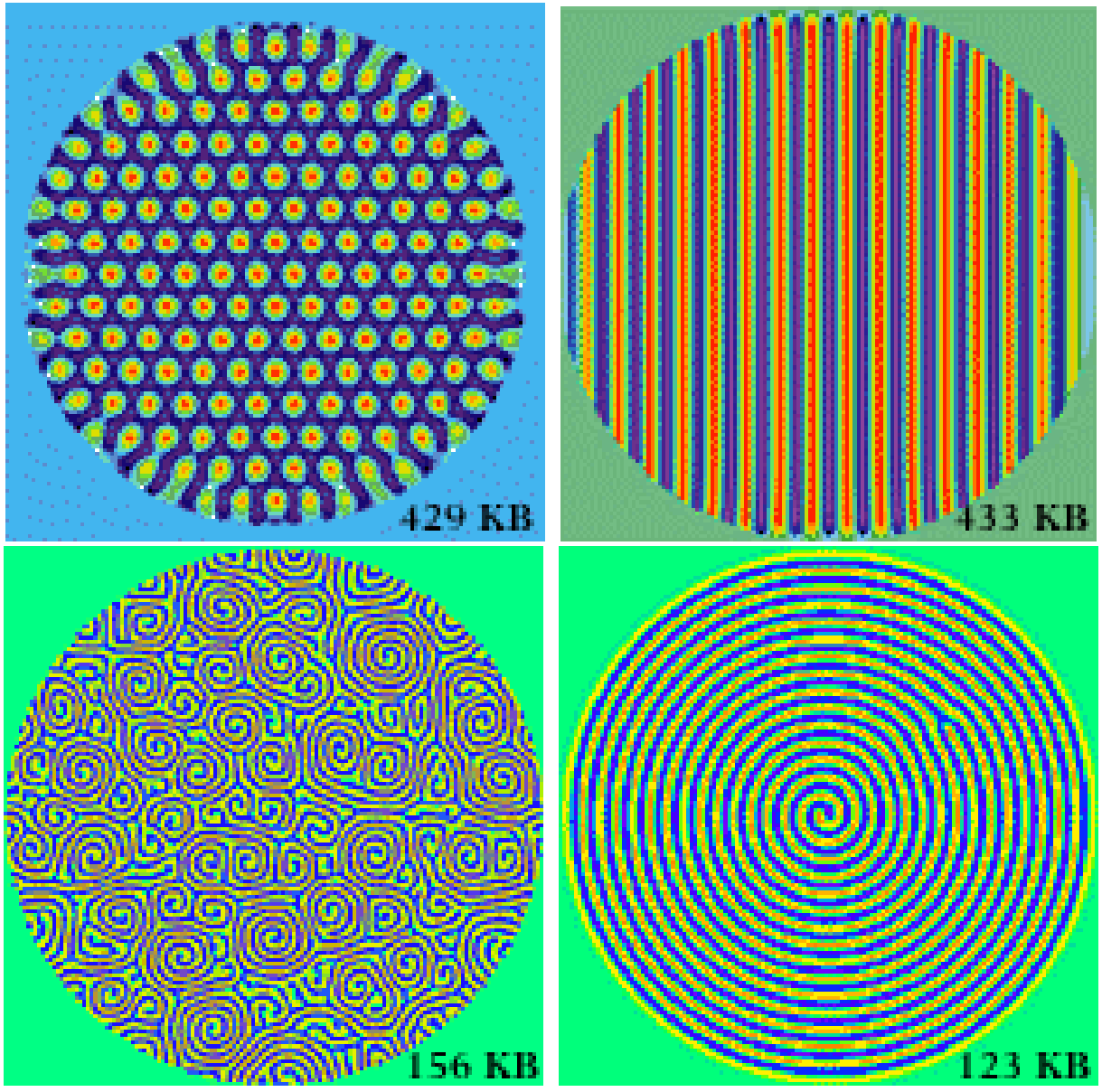
As water boils in a shallow, the hot water on the bottom of the pan rises to the cooler surface.

At a certain rate of heating, a uniform pattern of cells emerges.

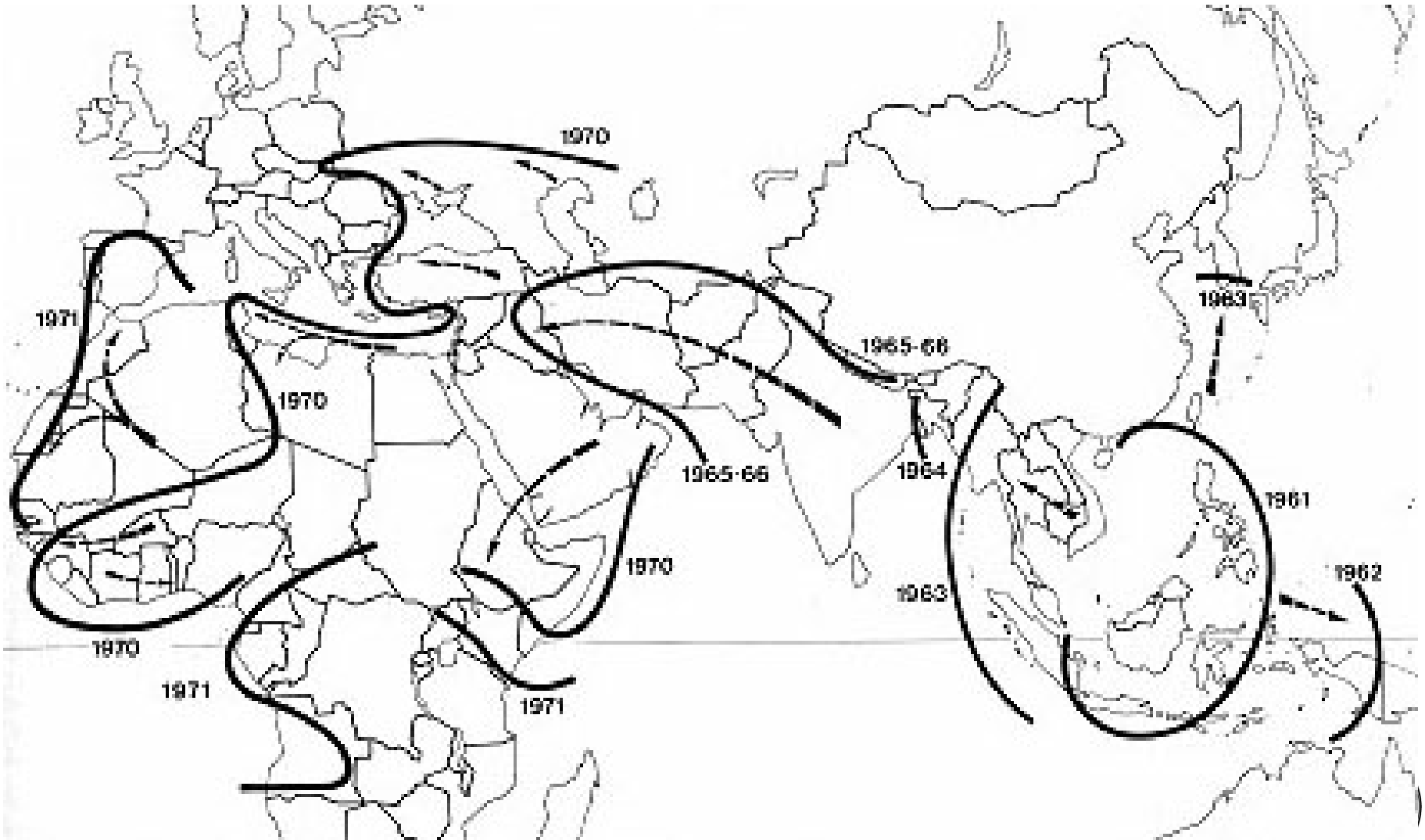
The cells are also somewhat stable. You can jiggle some cells with a spoon, and they will return to their original shape.

As heating increases, more complicated patterns arise; then disappear as the flow becomes turbulent.

Casebook: Rayleigh-Benard Convection Cells



Casebook: The Death Map



Casebook: The Death Map

In the early nineteenth century, European doctors working in India reported a strange new disease called **cholera**.

Cholera was agonizing and fatal.

No one knew how it was transmitted. No one had any idea how to treat or prevent it.

Yearly reports showed that cholera had begun to move across Asia, and into southern Europe. Soon it showed up in England.

Casebook: The Death Map

Casebook: The Death Map

This picture, illustrating a cholera epidemic, displays one common theory of cholera's transmission: **transmission by miasm**.

A miasm was a hypothetical airborne cloud, smelling terrible, and carrying the disease.

Miasm explained why cholera victims often lived in poor areas full of tanneries, butcher shops, and general dirty conditions.

It also explained why cholera did not simply spread out across the entire population, but seemed to travel, almost like the wind, settling down to devastate a neighborhood, then moving on.

The Miasm Theory

The miasm theory was accepted by doctors and public officials.

They assumed there was nothing they could do, except perhaps to drive out the bad air, using smoke or scented handkerchiefs, and to tell people not to live in low-lying areas near swamps and marshes.

This was still 40 years before Pasteur and Koch and other scientists were able to identify the bacteria associated some diseases, to show that bacteria caused those diseases, and that killing bacteria could prevent disease.

The Cholera Bacteria: Vibrio Cholerae

Dr John Snow

Dr John Snow argued that the miasm theory was unscientific.

Even though he could not show what was causing the disease, he felt strongly that it was transmitted **by water**.

Without knowing about bacteria, this is somewhat like trying to do physics without really understanding atoms!

But the important thing was that Dr Snow was trying to make an improved model of disease, one that explained the facts he knew.

If the model was good enough, it might also tell him how to prevent disease.

Dr John Snow

Evidence That Miasms Were a Bad Model

The miasm theory didn't satisfy Dr Snow because it didn't try to explain the patterns he had noticed.

In one small outbreak of cholera, only people living near the Thames river got sick. To Dr Snow, this suggested something in the river water. To the miasm defenders, there must have been a disease cloud floating along the low-lying river areas.

A second outbreak occurred away from the river, in an area with piped-in water. People getting water from one company were healthy, while others got sick. The intake pipes for that one company were located upstream from London. Again, the miasm defenders were not impressed by the small amount of data.

THE DEATH MAP

1. A Casebook
2. **DEATH IN GOLDEN SQUARE**
3. The Voronoi Diagram
4. Euler's Formula
5. Voronoi Computation
6. Centered Systems
7. Conclusion

Golden Square: Cholera Outbreak of 1854

Golden Square:

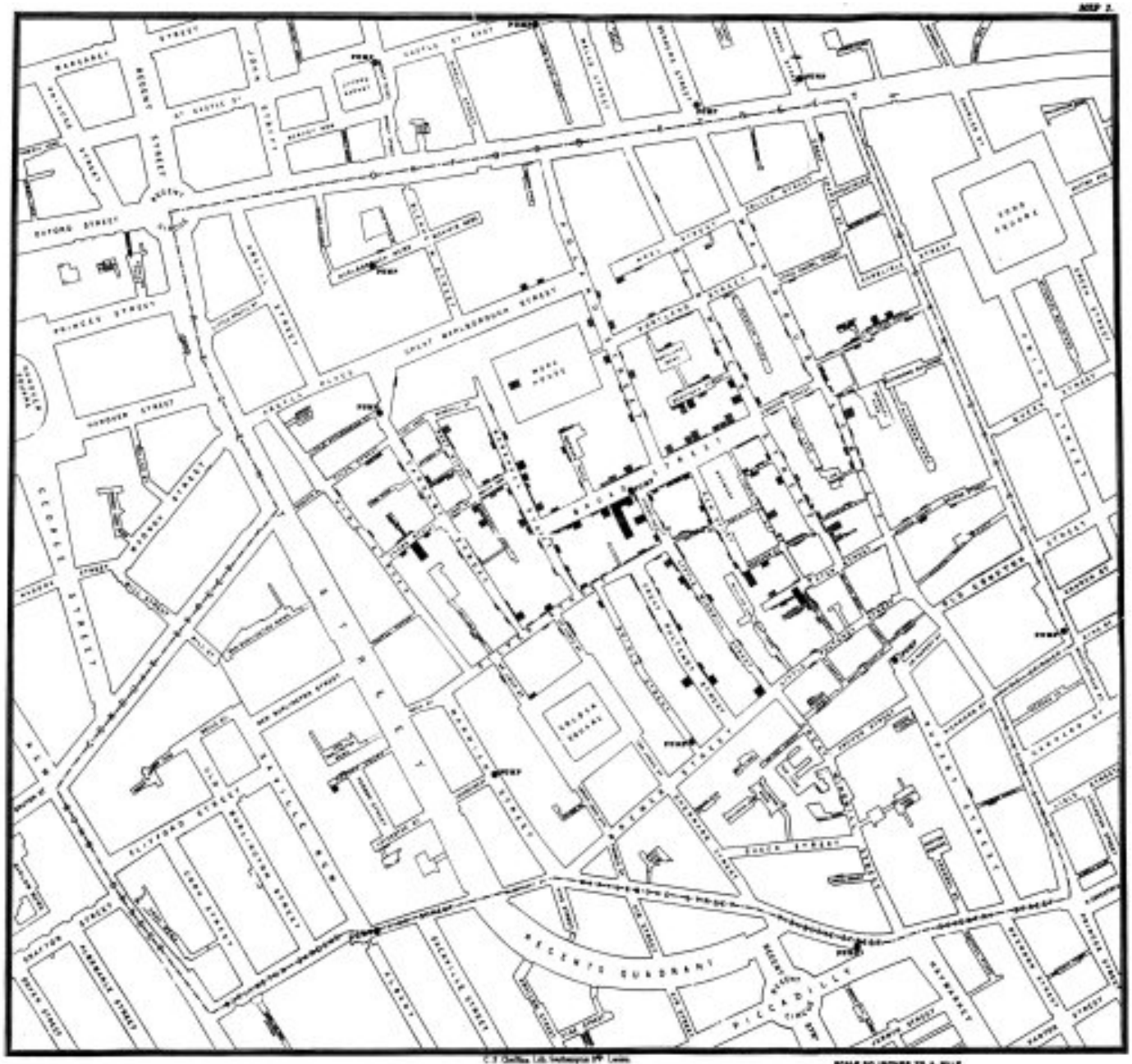
For most people in London, there was **no running water**. For drinking, cooking, cleaning and bathing, they went to one of the town pumps which drew water directly from the ground.

Most people had **no sewage system**. People used chamberpots and buckets, which were emptied into cisterns and carted off by an informal network of *night-soil carters*.

People were used to getting sick from the water now and then.

But when cholera bacteria arrived in London, and seeped from a cistern into the water, people died by the hundreds.

Golden Square: A Map



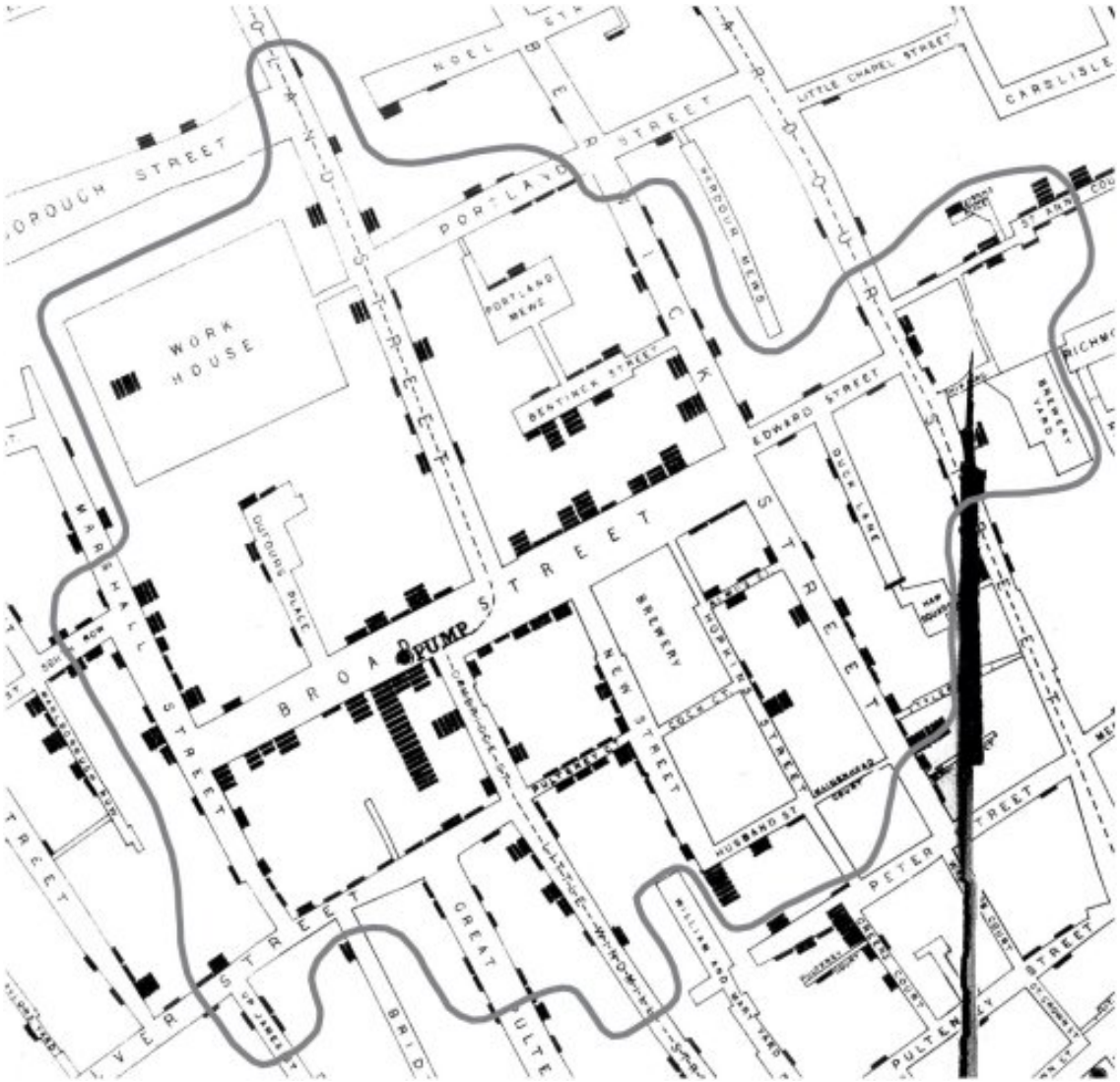
Golden Square: John Snow's Investigation

Dr John Snow suspected the water pump on Broad Street, but he needed evidence (no one knew about germs):

- He made a map of the district.
- He marked every house where cholera victims had lived;
- He paced the distance to the nearest pumps;
- The houses closest to the Broad Street pump were circled by a black line.

This was the **Death Map**.

Golden Square: The Pump and its Neighborhood

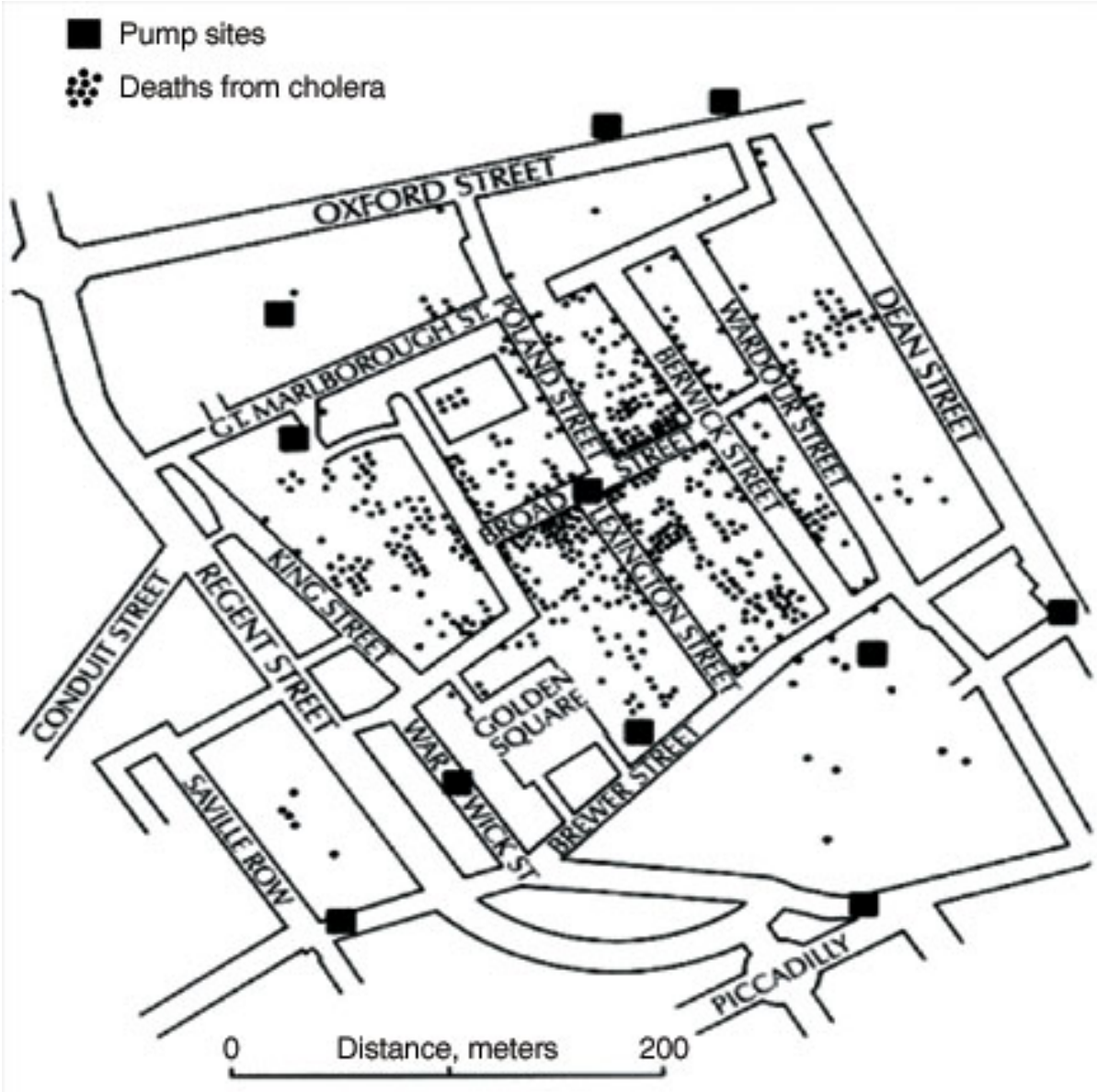


Golden Square: John Snow's Investigation

He personally interviewed people to explain exceptions to the pattern he found:

- healthy workers at a beer factory didn't drink any water!;
- some sick children lived outside the district, but came to school there;
- a woman who lived miles away had her son deliver "fresh" water from the Broad Street pump because she liked its taste.

Golden Square: See the Pumps as "Suspects"



Golden Square: Conclusion

Snow's map strongly suggested that people who died were precisely those for whom the Golden Square pump was the closest.

The miasmic cloud theory required the miasm, purely by chance, to settle only over people who used the Golden Square pump.

Dr Snow's map destroyed the miasm theory, because it could show **where** deaths occurred in a way that suggested **why**.

It took another 30 years before the exact cause of the disease, the cholera bacterium, was actually identified.

Golden Square: Conclusion

There were two organizing principles in Dr Snow's map:

- the **pumps** were "centers" of the geometry
- the **distance** to the nearest pump organized the inhabitants into "cells"

Dr Snow imagined every dead person walking back to the pump that was the nearest to them, and found most of them met at Golden Square.

Dr Snow was doing epidemiology, but also mathematics.
His map is an interesting example of a **Voronoi diagram**.

THE DEATH MAP

1. A Casebook
2. Death in Golden Square
3. **THE VORONOI DIAGRAM**
4. Euler's Formula
5. Voronoi Computation
6. Centered Systems
7. Conclusion

Voronoi: Definition

Suppose that

- we have a space $\mathcal{P}$ of points;
- we choose a subset $\mathcal{G}$ of points in $\mathcal{P}$, called "generators";
- we measure point distance by $d(p_1, p_2)$.

A **Voronoi diagram** $\mathcal{V}(\mathcal{G}, d)$ assigns points to nearest generators.

In Dr Snow's map,

- $\mathcal{P}$ was all the locations in Golden Square;
- $\mathcal{G}$ was the location of pumps;
- $d(p_1, p_2)$ was the walking distance between two points.

Voronoi: Names of Things

We can abstract the ideas from Dr Snow's diagram, and think about a simple situation involving points in a plane.

We can randomly pick a small number of points as our generators, and ask how these generators, plus a distance function, organize all the points in the plane.

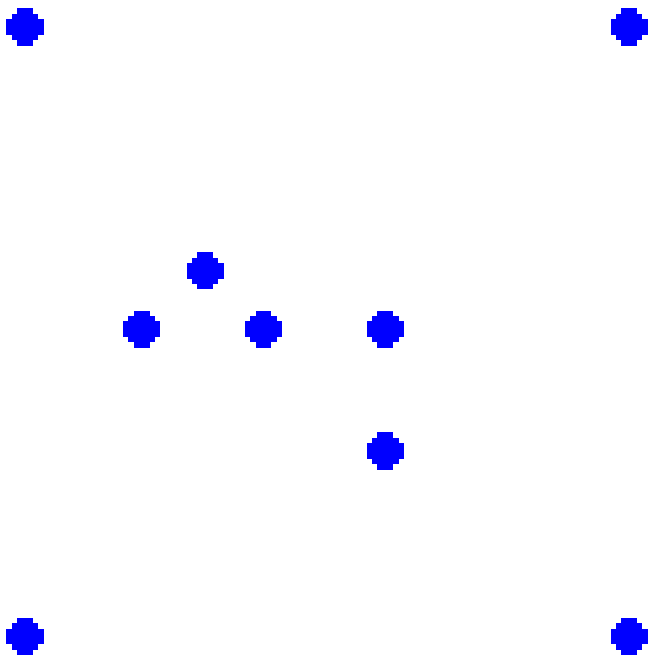
Mathematically, the distance that Dr Snow used was similar to something called the ℓ_1 distance (if streets only ran north-south or east west). We will usually prefer to use the ℓ_2 or "Euclidean" distance.

Voronoi: Names of Things

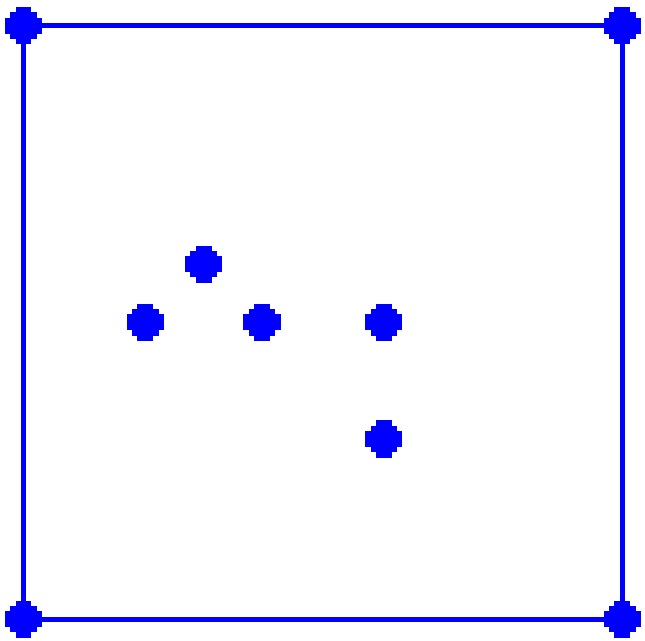
For a set of points or "generators" in the plane, there are three common geometric structures:

- **convex hull** – the smallest polygon containing the points;
- **Delaunay triangulation** – "best" connection of all points;
- **Voronoi diagram** – each generator creates a "country".

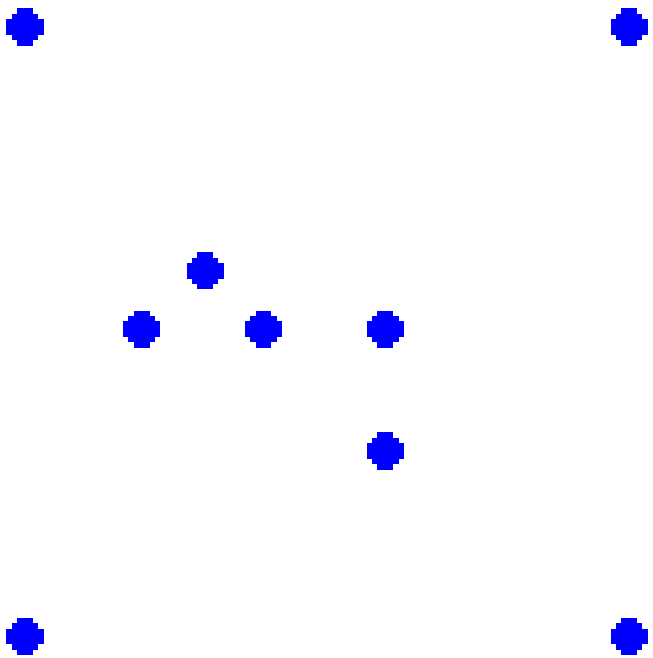
Voronoi: 9 points



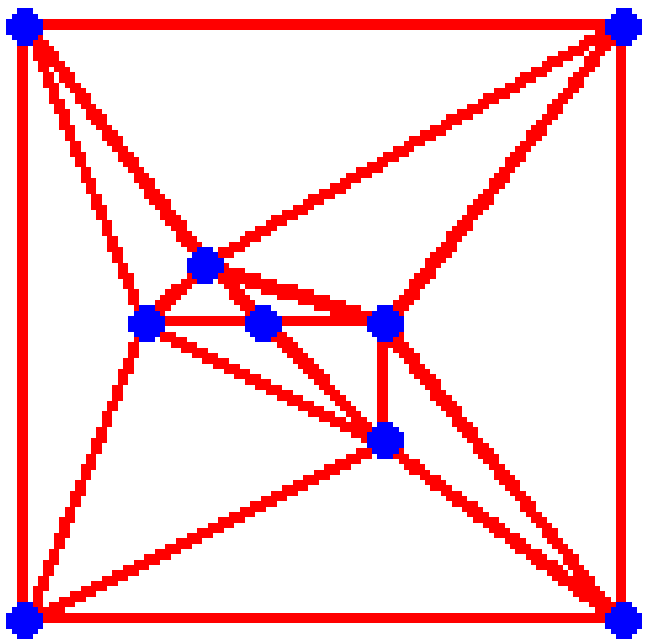
Voronoi: 9 point convex hull



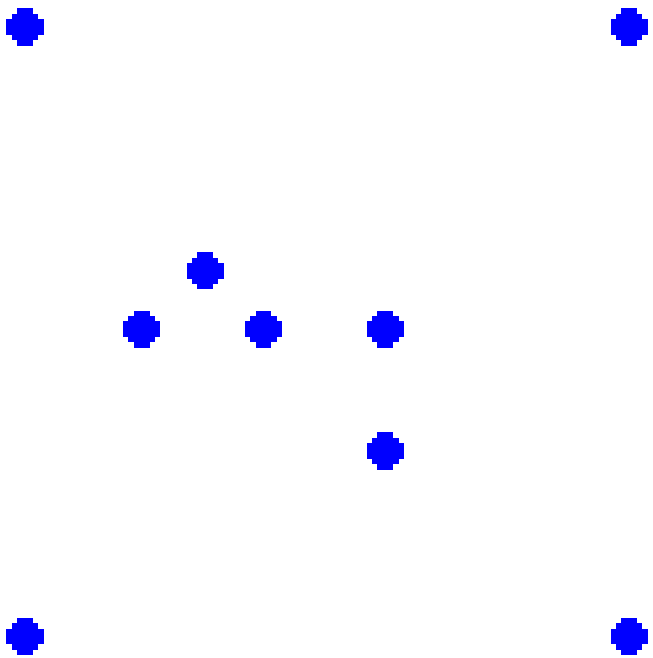
Voronoi: 9 points



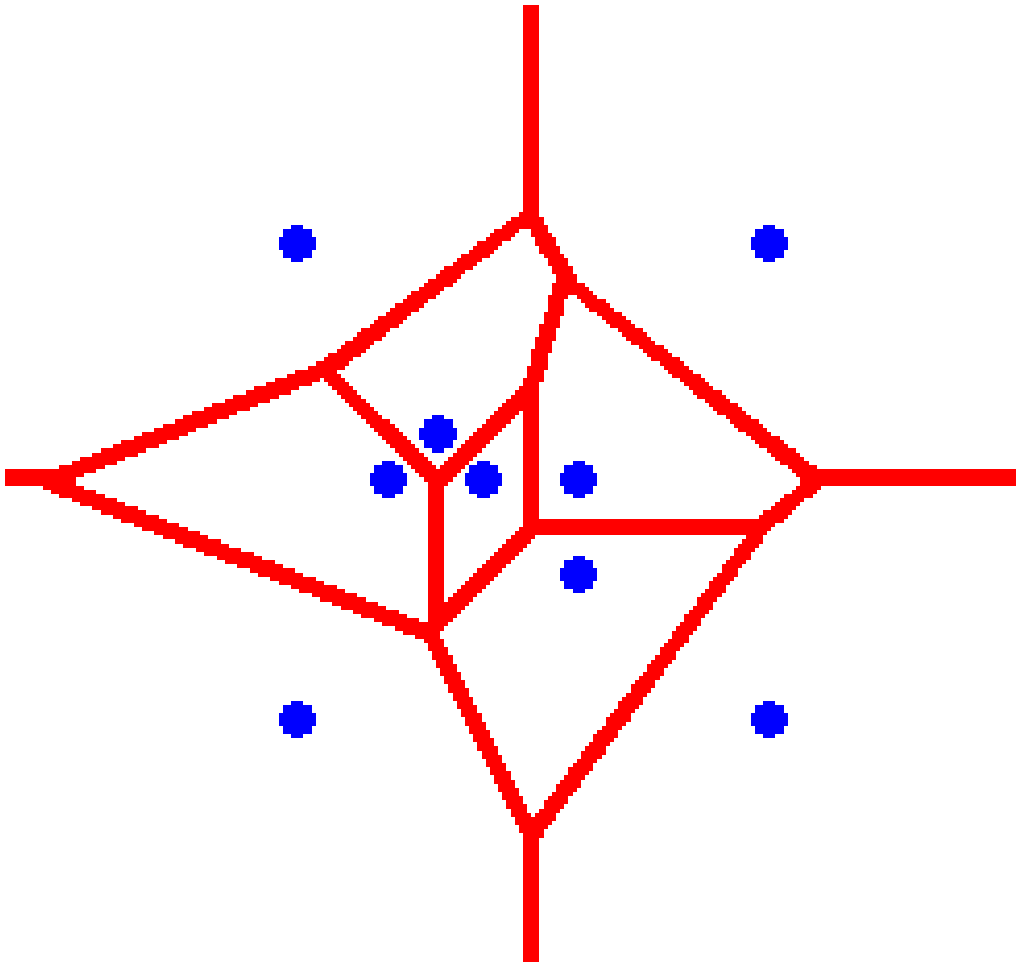
Voronoi: 9 point Delaunay triangulation



Voronoi: 9 points



Voronoi: 9 point Voronoi diagram



Voronoi: Properties of the Voronoi Regions

Assuming we are using the ℓ_2 distance function,

- All regions are convex (no indentations);
- All regions are polygonal;
- Each region is the intersection of half planes;
- The infinite regions have generators on the convex hull.

Voronoi: Properties of the Edges and Vertices

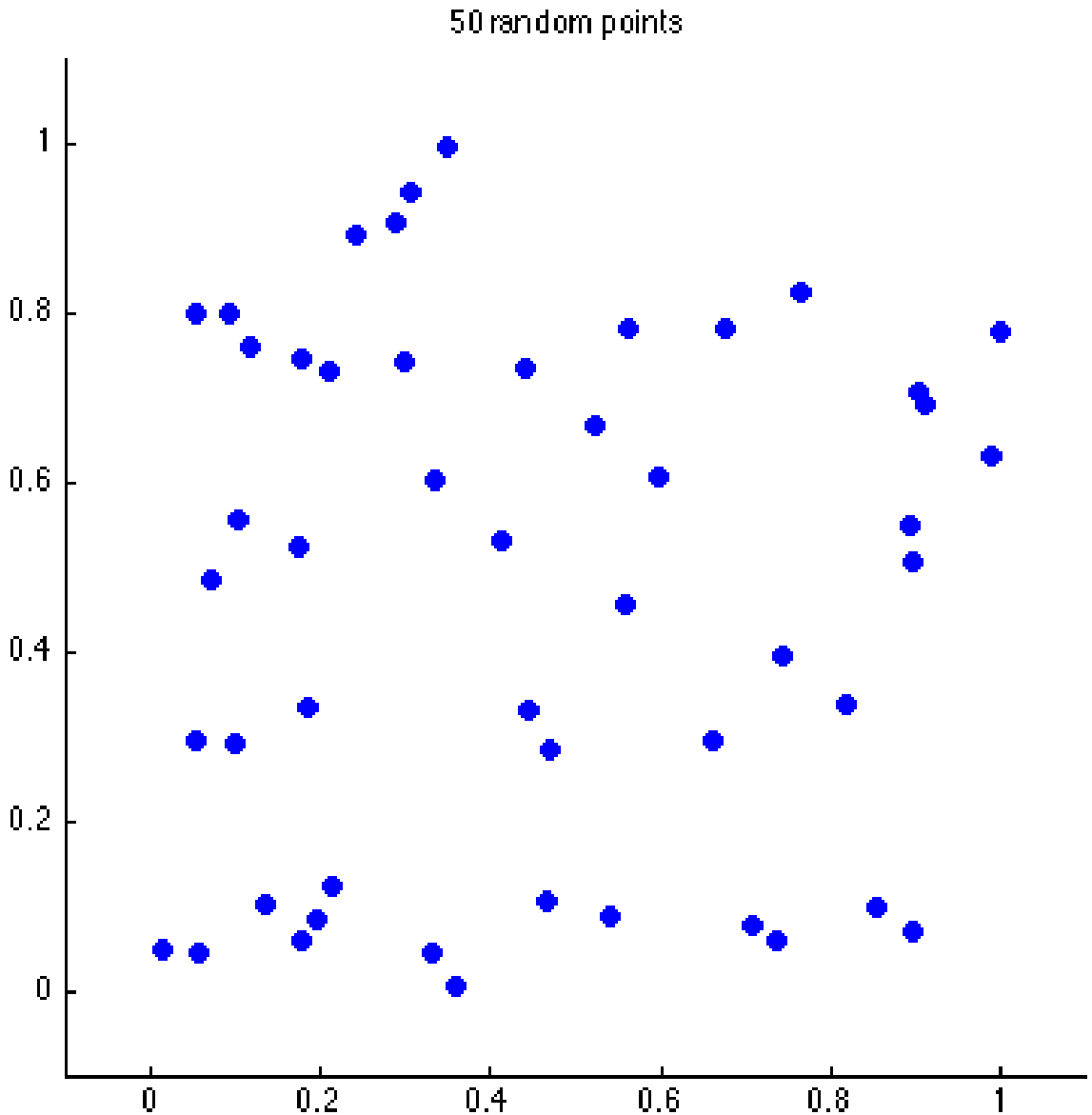
The Voronoi **edges** are the line segments of the boundary;

- Edges are perpendicular bisectors of neighbor lines; they contain points equidistant from 2 generators;
- If two generators share an edge, then they are connected in the Delaunay triangulation.

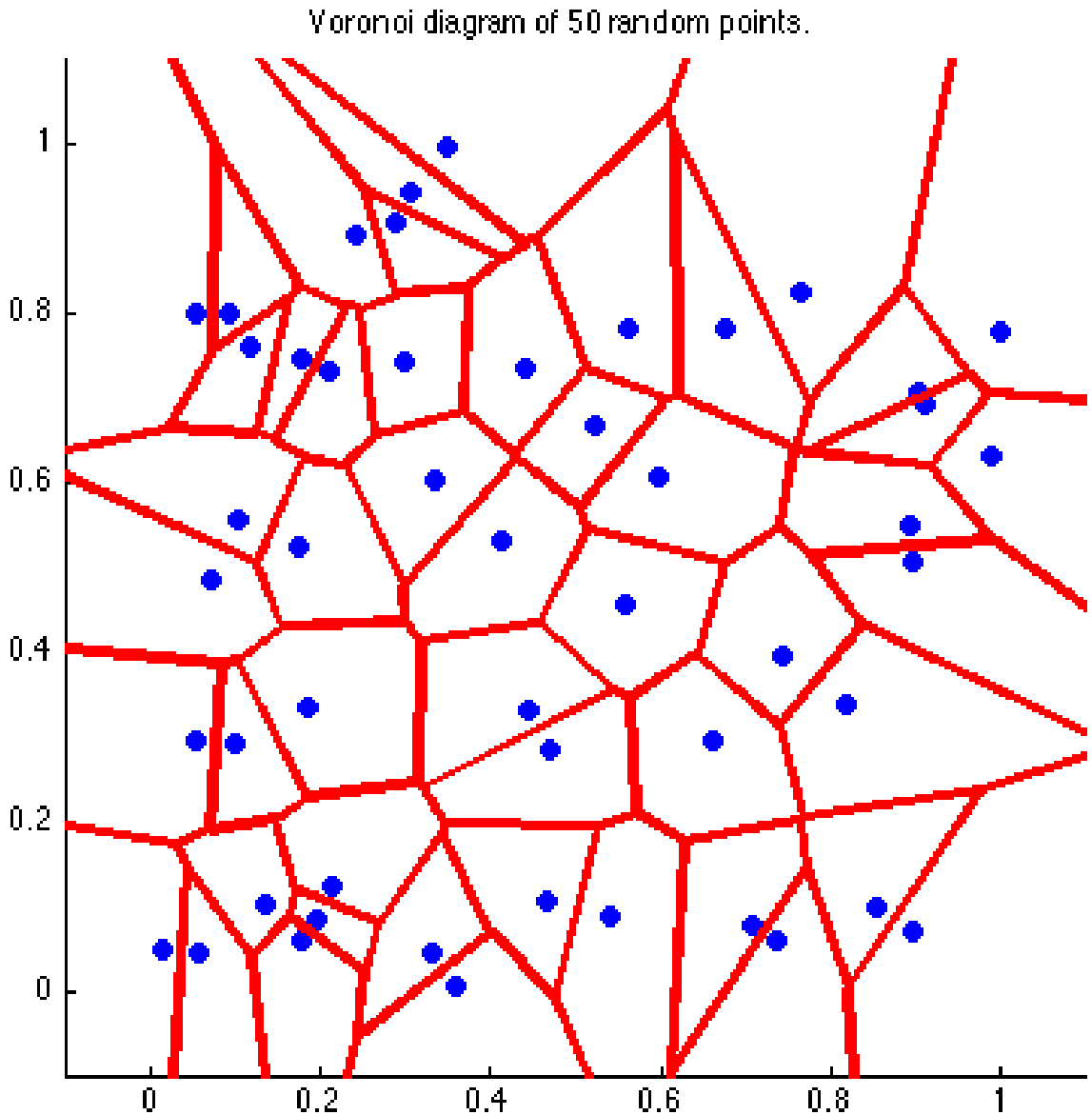
The Voronoi **vertices** are "corners" of the boundary.

- Vertices are the endpoints of 3 (possibly more) edges; they are equidistant from 3 (possibly more) generators;
- Each vertex is the center of a circle through 3 generators;
- Each vertex circle is empty (no other generators).

Voronoi: 50 points



Voronoi: 50 points Voronoi diagram



Voronoi: Related Problems

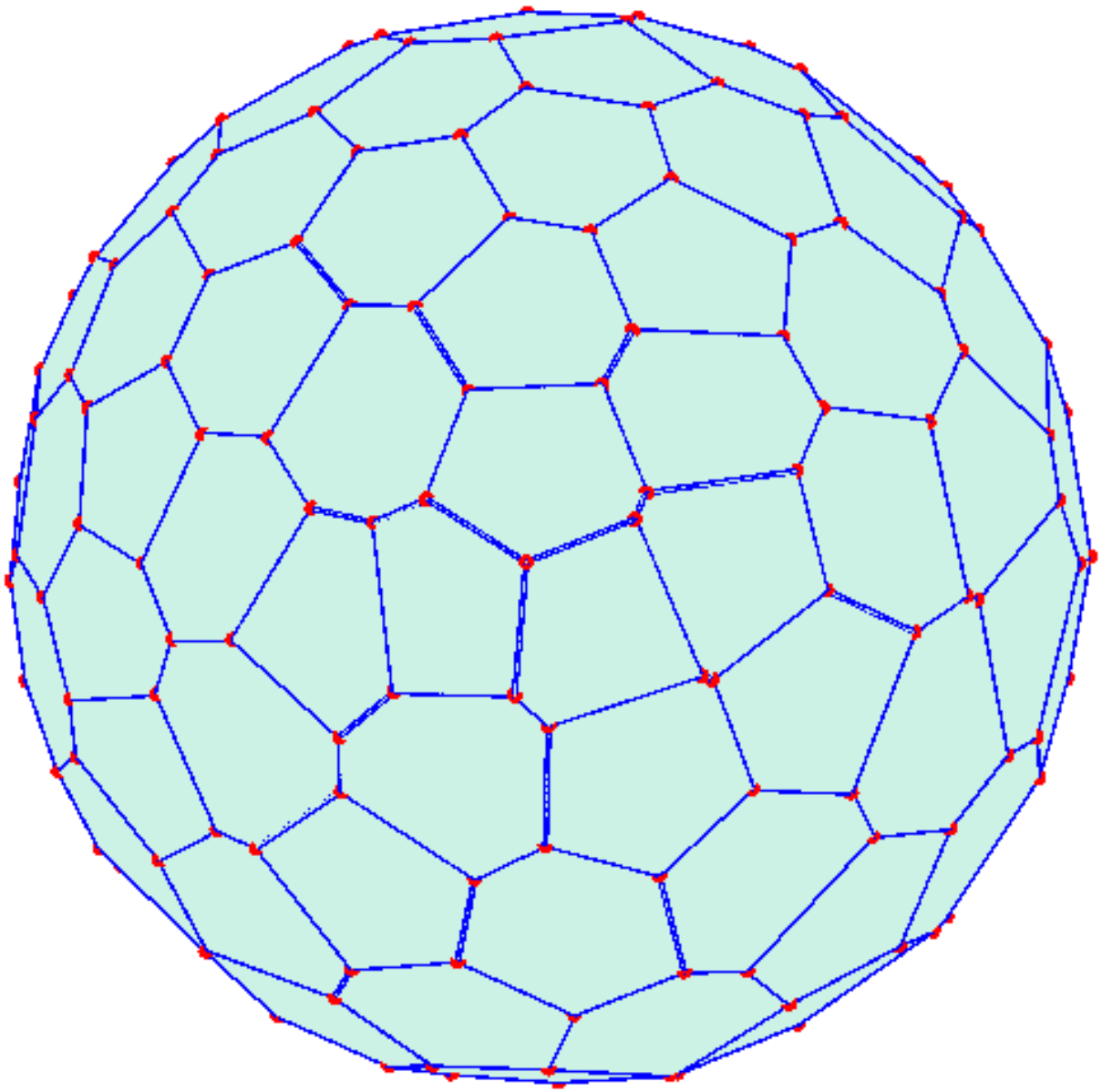
Most algorithms and software for the Voronoi diagram only work in the (infinite) plane with the Euclidean distance.

It's easy to want to extend these ideas to related problems:

- Restrict to some finite shape (perhaps with holes);
- "Wrap-around" regions (like a torus, or Asteroid);
- Define Voronoi diagrams on a surface;
- Problems in higher dimensions.
- Include a "density" function;

The sphere is an important region for which exact and approximate methods have been developed.

Voronoi: Spheres and Other Shapes



Voronoi: Spheres and Other Shapes

The previous image on a sphere was computed by exact techniques.

We will now run an interactive program that suggests how a Voronoi diagram on a sphere can be drawn by sampling.

sphere_voronoi_display_open_gl 100

Voronoi Computation: Sampling Method for Spheres

THE DEATH MAP

1. A Casebook
2. Death in Golden Square
3. The Voronoi Diagram
4. **EULER'S FORMULA**
5. Voronoi Computation
6. Centered Systems
7. Conclusion

Euler's Formula: Counting Faces, Edges, Vertices

If we only know the **number** of generators involved, that's not enough to tell us how complicated the Voronoi diagram will be.

If we think about storing the information in a computer, we would probably need to know the number of vertices and edges, **before we try to compute the diagram**, so that we have enough storage.

Since these numbers also depend on the **arrangement** of the generators, the best we can hope is that we might be able to compute an overestimate for the number of edges and vertices, so we are guaranteed to have enough storage.

Euler's Formula: Counting Faces, Edges, Vertices

Leonhard Euler developed a formula that related the number of faces, edges and vertices in a 3-dimensional polyhedron.

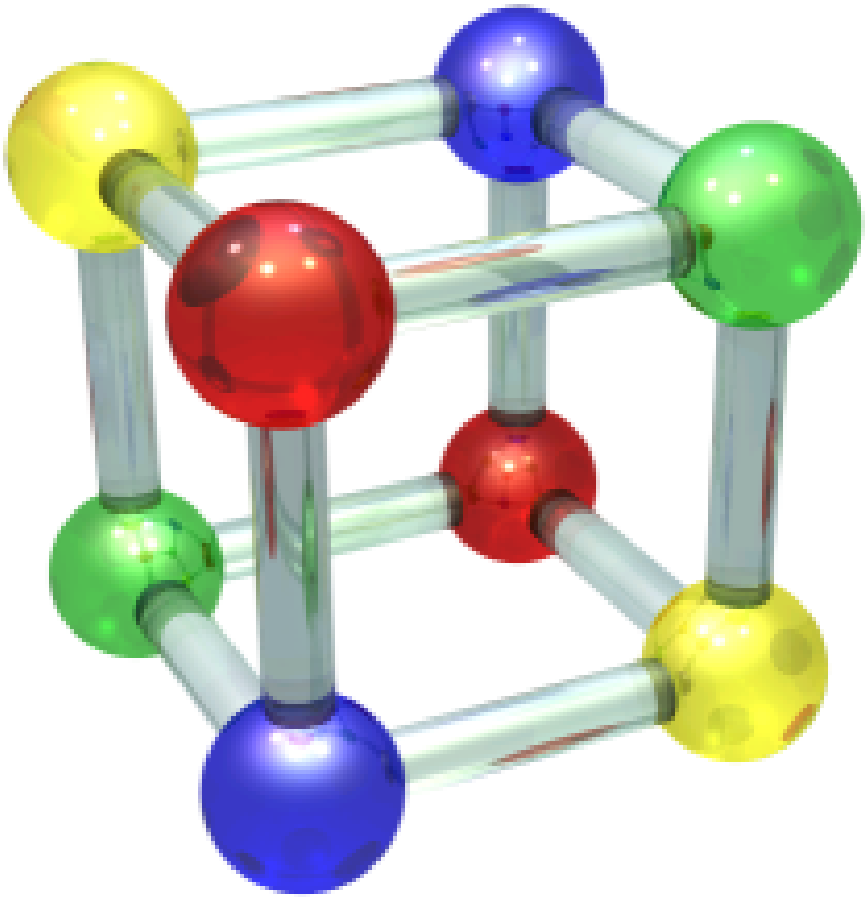
This formula can estimate the "size" of a Voronoi diagram, that is, the number of edges and faces for a given number of points.

Euler's Formula: for 3D Polyhedrons

The formula relates **F**aces, **V**ertices, and **E**dges of a polyhedron.

$$F + V = E + 2$$

$$6 + 8 = 12 + 2$$



Euler's Formula: Adapted to 2D

We can apply Euler's Formula to a 2D figure in the plane

Just imagine the surface is stretchable. Puncture the surface at one point and flatten it out.

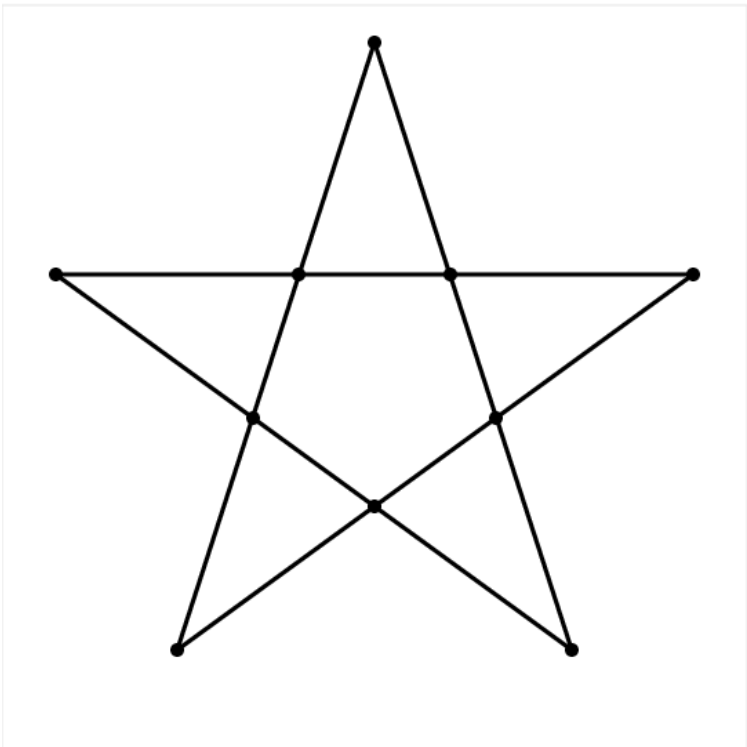
Our puncture will make one face disappear. Things will add up correctly in Euler's formula if we increase the number of faces we can see by 1, to account for the face we destroyed.

Euler's Formula: for Bounded 2D Figures

In 2D, and a bounded figure, add one infinite face.

$$(F + 1) + V = E + 2$$

$$(6 + 1) + 10 = 15 + 2$$



Euler's Formula: Unbounded 2D Figures

We can also apply Euler's Formula to a Voronoi diagram.

A Voronoi diagram is unbounded, some edges go to infinity.

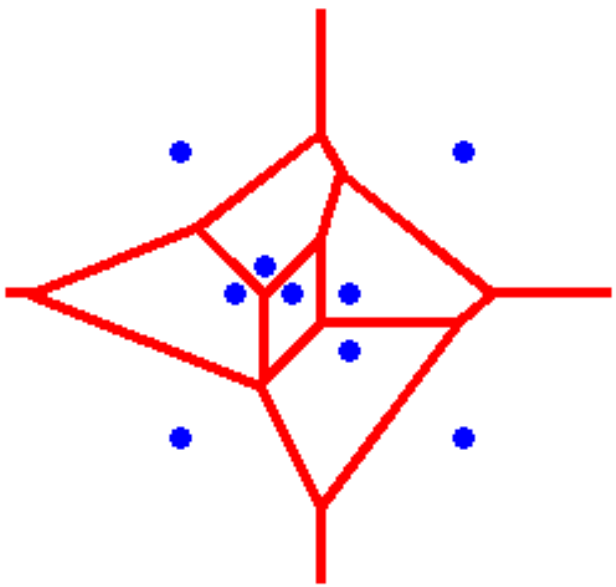
We can think of the infinite regions as just "very big" faces;
we can imagine the infinite edges all meet at a vertex at infinity.

So the original 3D polyhedron formula will work for us, as long as we add one fictitious vertex at infinity.

Euler's Formula: Unbounded 2D Figures

In 2D, and unbounded diagram, add vertex at infinity.

$$\begin{aligned} F + (V + 1) &= E + 2 \\ 9 + (11 + 1) &= 19 + 2 \end{aligned}$$



Euler's Formula: Estimating Voronoi Size

Now we use our formula $F + V = E + 1$ to bound the maximum sizes of E and V.

First, we can figure out a limit for V if we know E.

Each edge is defined by the 2 vertices that are its endpoints, and each vertex belongs to (at least) 3 edges.

Therefore:

$$3V \leq 2E$$

Euler's Formula: Estimating Voronoi Size

Substituting for E in Euler's equation gives:

$$\begin{aligned} F + V &= E + 1 \quad \textit{(Euler)} \\ 2F + 2V &= 2E + 2 \quad \textit{(double)} \\ 2F + 2V &\geq 3V + 2 \quad \textit{(because } 3V \leq 2E\textit{)} \\ 2(F - 1) &\geq V \end{aligned}$$

So we have a limit on Voronoi vertices V in terms of the Voronoi generators F.

Euler's Formula: Estimating Voronoi Size

Now we seek a bound on E:

$$\begin{aligned} F + V &= E + 1 && \textit{(Euler)} \\ F + 2(F-1) &\geq E + 1 && \textit{(by our limit on } V\textit{)} \\ 3(F-1) &\geq E \end{aligned}$$

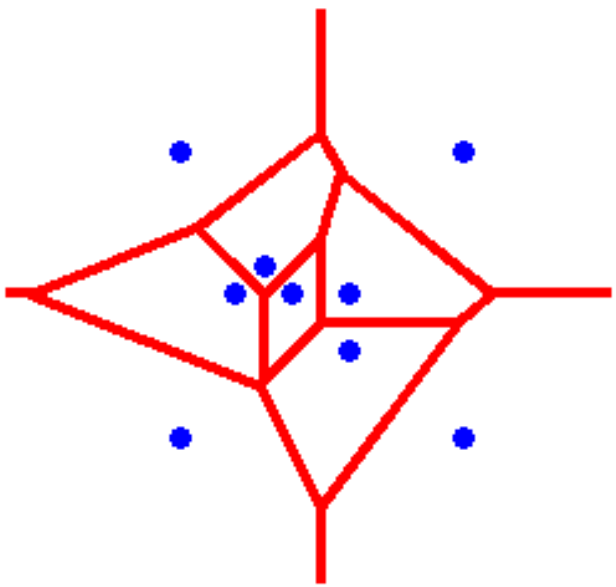
And thus, if we know the number of generators F, we can bound V and E.

This allows us to set aside memory for arrays when computing Voronoi diagrams.

Euler's Formula: Unbounded 2D Figures

For our example Voronoi diagram, $F = 9$, so

$$\begin{aligned} 16 = 2(F-1) &\geq V = 11 \\ 24 = 3(F-1) &\geq E = 19 \end{aligned}$$

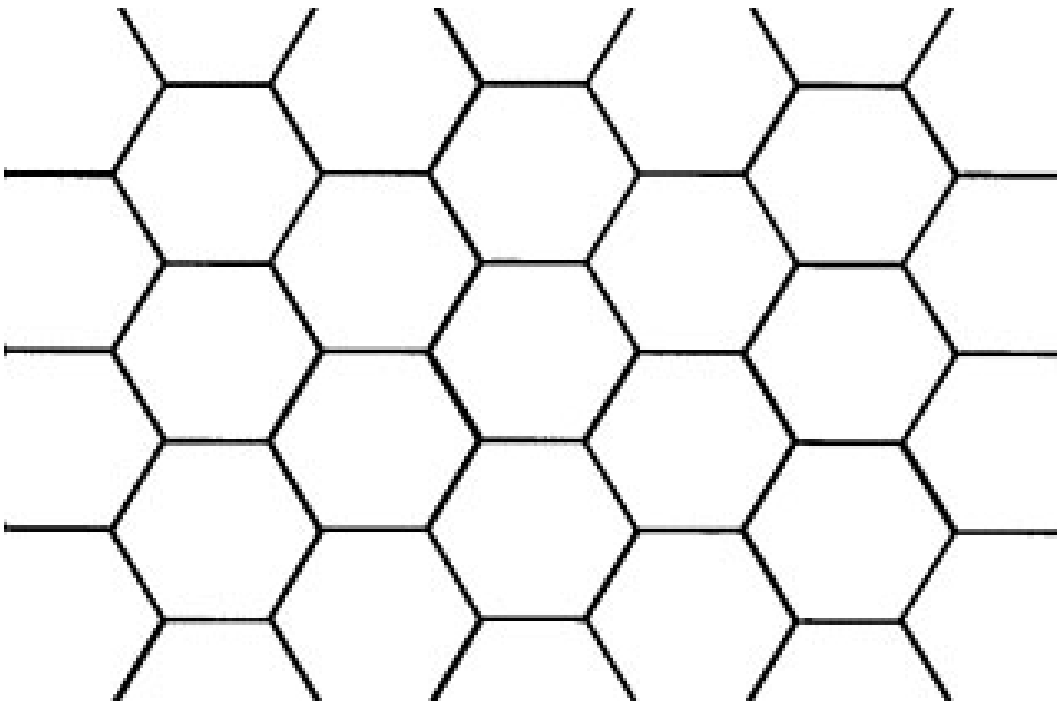


Euler's Formula: Estimating Voronoi Size

The limit on edges gives us one more interesting fact.
Each edge separates two neighboring faces.

If the number of edges is roughly its maximum $3F$, then the average number of neighbors would be about 6.

This corresponds nicely with an optimal hexagonal tiling pattern!



THE DEATH MAP

1. A Casebook
2. Death in Golden Square
3. The Voronoi Diagram
4. Euler's Formula
5. **VORONOI COMPUTATION**
6. Centered Systems
7. Conclusion

Voronoi Computation

In some ways, a Voronoi diagram is just a **picture**; in other ways, it is a **data structure** which stores geometric information about boundaries, edges, nearest neighbors, and so on.

How would you compute and store this information in a computer?

It's almost a bunch of polygons, but of different shapes and sizes and orders - and some polygons are "infinite".

Euler's formula lets us know in advance a limit of the number of edges and vertices.

A convex hull calculation lets us know how many infinite regions we will have.

Voronoi Computation: A Picture Calculation

If you just want a picture of the Voronoi diagram, then you will be satisfied with a list of the line segments that make up the edges.

The semi-infinite edges just have to be drawn to the edges of the picture.

For the picture option, we don't store any extra information.

MATLAB: **voronoi (xVector, yVector)** ;

Mathematica: **DiagramPlot[xyList]** .

Voronoi Computation: A Geometric Calculation

A more detailed calculation would need to compute lists:

- the location of the vertices;
- the pairs of vertices that form an edge;
- the sequence of edges that form the boundary of a region.
- some way to indicate whether the region is finite or infinite.

MATLAB: **[v,c] = voronoin (xyArray)** ;

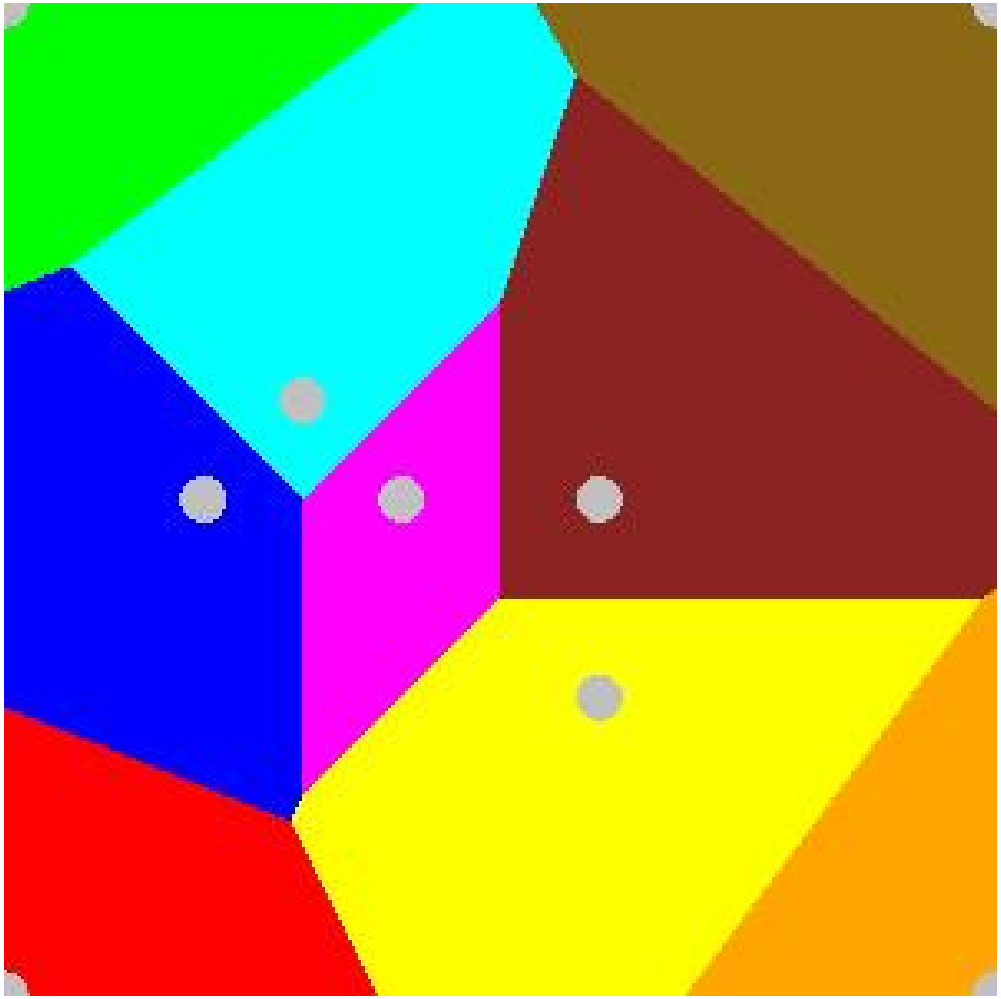
Mathematica: **VoronoiDiagram[xyList]** .

Voronoi Computation: Algorithms

Algorithms for computing a Voronoi diagram include:

1. **The Method of Half Planes** - add one generator at a time;
2. **Byer's Method** - bisect the "neighbor" lines;
3. **Fortune's Method** - scans the data from left to right, building the diagram in stages;
4. **Contour method** - at each point, compute the distance to the nearest generator; then use graphics software to draw contour lines.
5. **Sampling method** -choose many sample points from the region. Assign each sample point to nearest generator, and estimate properties by sampling.
6. **Pixel method** - (sampling method, using pixels) give each generator a color. Assign each pixel the color of the nearest generator.

Voronoi Computation: Pixel Method (Euclidean distance)



Voronoi Computation: The Sampling Method

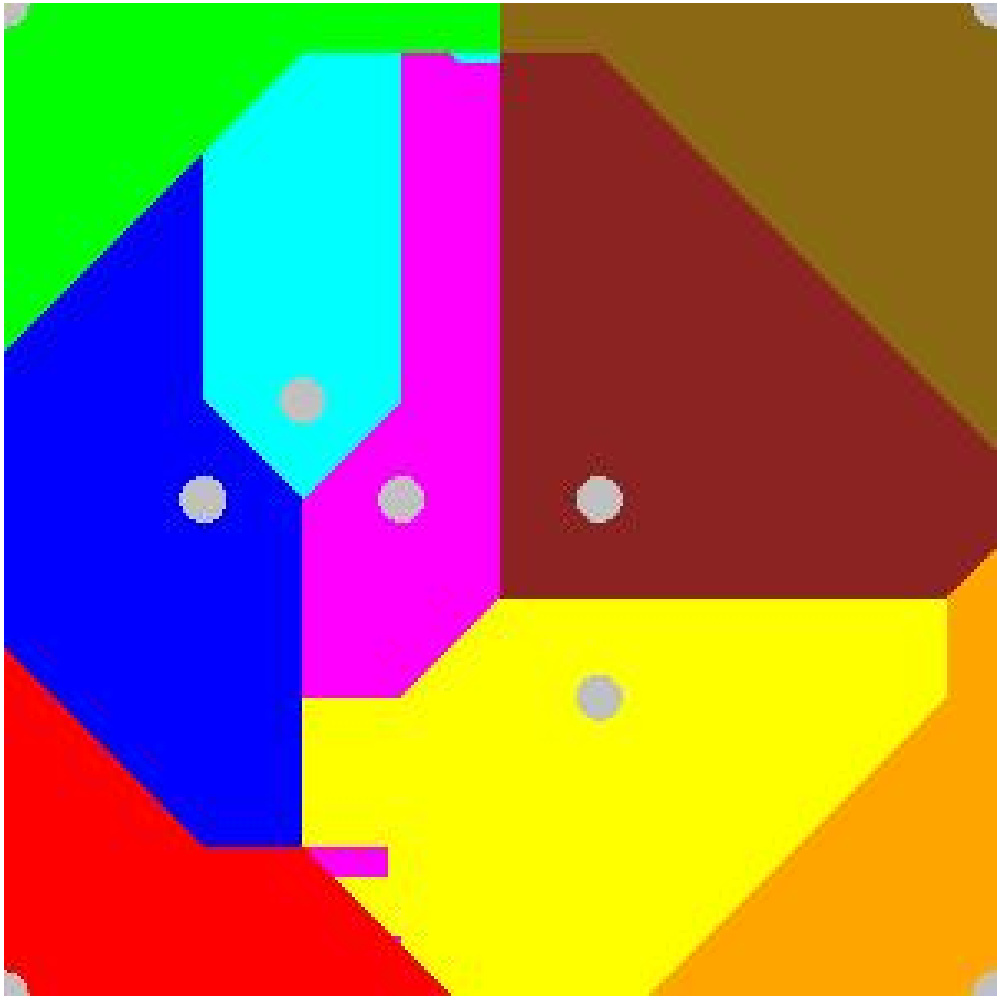
Exact methods are slow, complicated and hard to modify.

What happens if you want to work:

- inside a strangely shaped region?
- with a different distance function?
- on the surface of a sphere?
- in 3D?

If **approximate** information is acceptable,
the sampling method can handle these problems and more.

Voronoi Computation: Sampling Method (L1 distance)

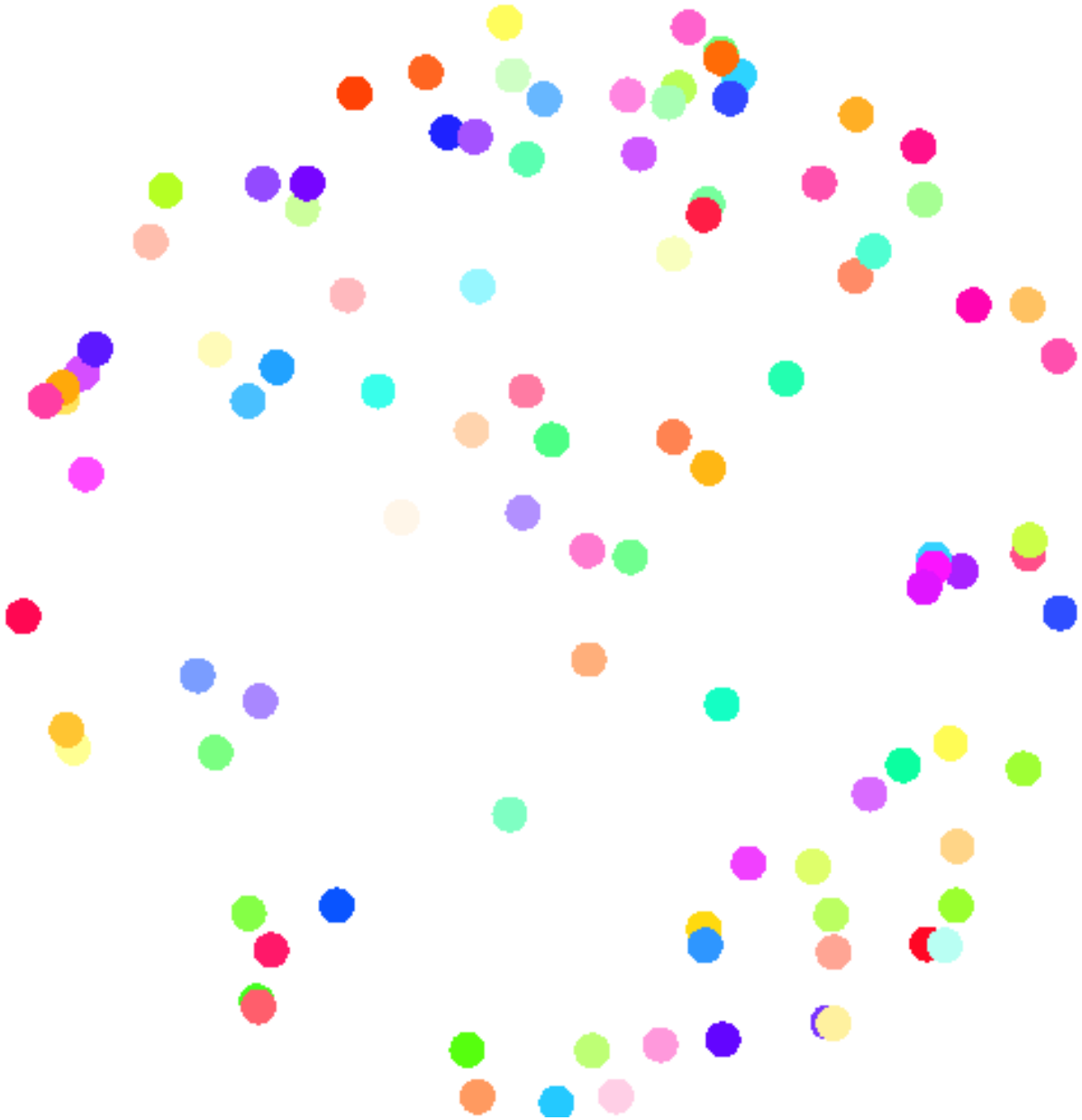


Voronoi Computation: Sampling Method for Spheres

To use the sampling method on a sphere, you may want to do a few things:

- be able to choose random points on a sphere for generators;
- set up some kind of grid on the sphere's surface;
- figure out the formula for distance along the surface (it's essentially an angle)

Voronoi Computation: Sampling Method for Spheres



Voronoi Computation: Sampling Method for Spheres

THE DEATH MAP

1. A Casebook
2. Death in Golden Square
3. The Voronoi Diagram
4. Euler's Formula
5. Voronoi Computation
6. **CENTERED SYSTEMS**
7. Conclusion

Centered Systems: Natural Patterns?

The Voronoi diagram seems to create some order out of scattered points.

We might compare the Giant's Causeway, or the spots on leopards or giraffes, or the "spitting fish"" nests.

However, the generators are not "centered" and the shape and size of the regions seems to vary more than in some natural patterns. Can we explain or control this?

Centered Systems: More Realism?

Physical examples differ from our mathematics.

- Instead of the infinite plane, we have a finite area or surface.
- The regions are roughly the same size.
- The regions are roughly the same shape.

Centered Systems: Center the Generators

We started by choosing generators at random
and the generators gathered nearest points to them.

Why should our random initial generators be the "best" ones?

What would make a generator better?

Suppose...each generator **moved to the center** of each region.
This would reduce the average distance to all the points.

This is like moving the capital city to the center of the state (this happened in Iowa!)

Centered Systems: Center the Generators

We can *exactly* measure the improvement.

We define the **energy** of the **i-th** Voronoi cell C_i, whose generator is at (x_i, y_i), as the integral of the square of the distances to the center

$$\mathcal{E}_i = \int_{(x,y)\in C_i} (x - x_i)^2 + (y - y_i)^2 \, dx \, dy$$

Moving the generator to the center of cell C_i reduces the energy $\mathcal{E}_i$.

Centered Systems: Center the Generators

Moving the generator has another effect....
some points in C_i may now be closer to a different generator.

Suppose we transfer these points? A point that moves from cell C_i to cell C_j lowers the energy $\mathcal{E}_i$ and raises the energy $\mathcal{E}_j$.

However, we are moving the point because it is closer, so we must be decreasing the **total energy**:

$$\mathcal{E} = \sum_i \mathcal{E}_i$$

Transferring these points reduces the total energy $\mathcal{E}$.

Centered Systems: Algorithm

Now we have an *iterative* method to center our Voronoi diagram.

The energy gives us a way of measuring our progress.

1. Choose generators at random.
2. Move every point to nearest generator.
3. Compute new energy $\mathcal{E}$.
4. If energy did not decrease very much, exit.
5. Compute centroid of each cell.
6. Move generators to cell centroid.
7. Return to step 2.

Centered Systems: Algorithm

This algorithm needs the centroid $(\bar{x}_i, \bar{y}_i)$ of each cell C_i.

If C_i is a polygon, we use geometry to find the centroid.

If C_i is more complicated, we must use calculus:

$$\bar{x}_i = \frac{\int_{C_i} x \, dx \, dy}{\int_{C_i} dx \, dy}, \quad \bar{y}_i = \frac{\int_{C_i} y \, dx \, dy}{\int_{C_i} dx \, dy},$$

But exact calculations are limited to simple, small, regular problems!

Centered Systems: Algorithm

Instead of an exact integral, we can estimate the centroid using sampling.

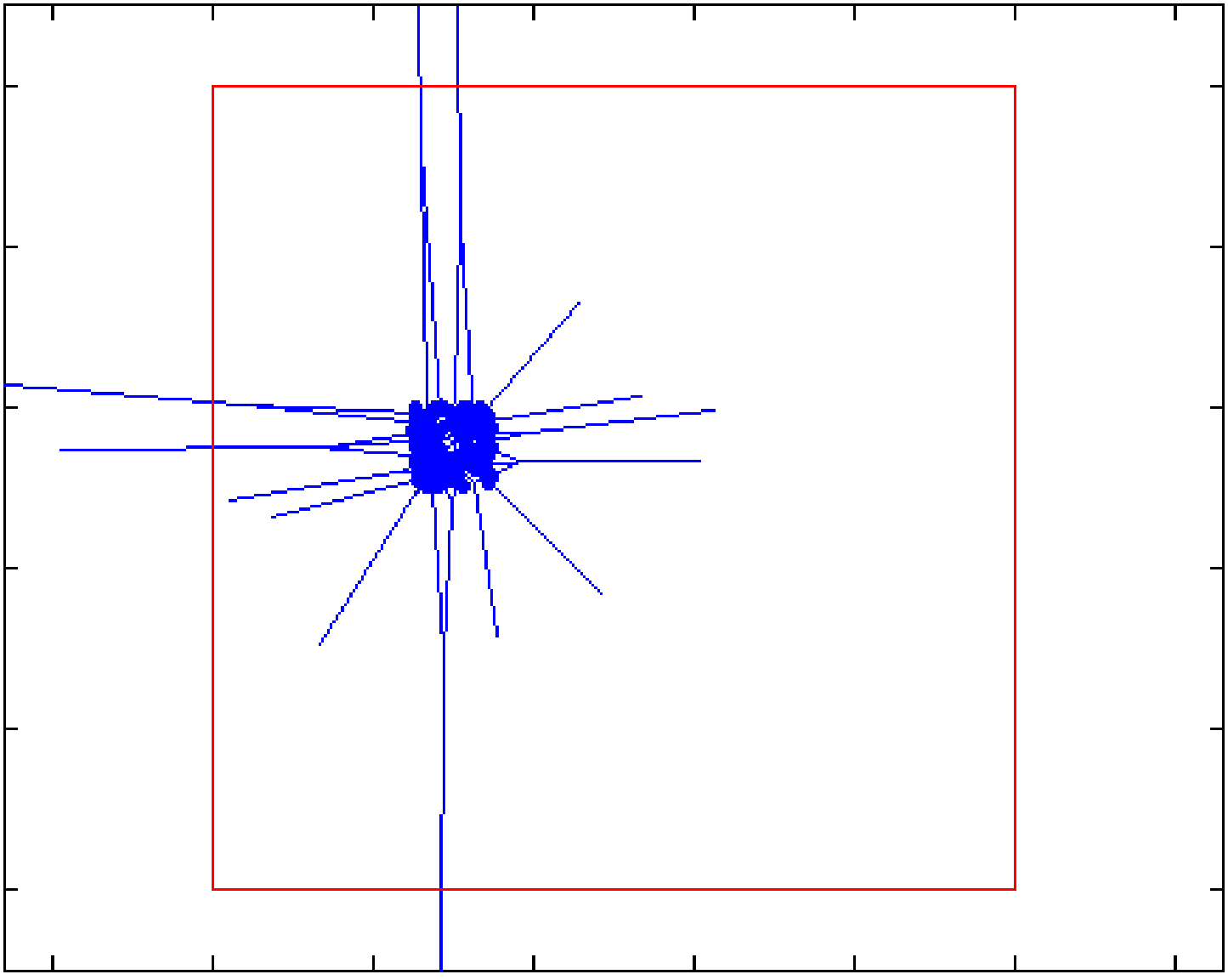
First, we generate a large number of sample points p in the entire region.

The centroid of cell C_i is **approximately** the average of the sample points p that belong to the C_i!

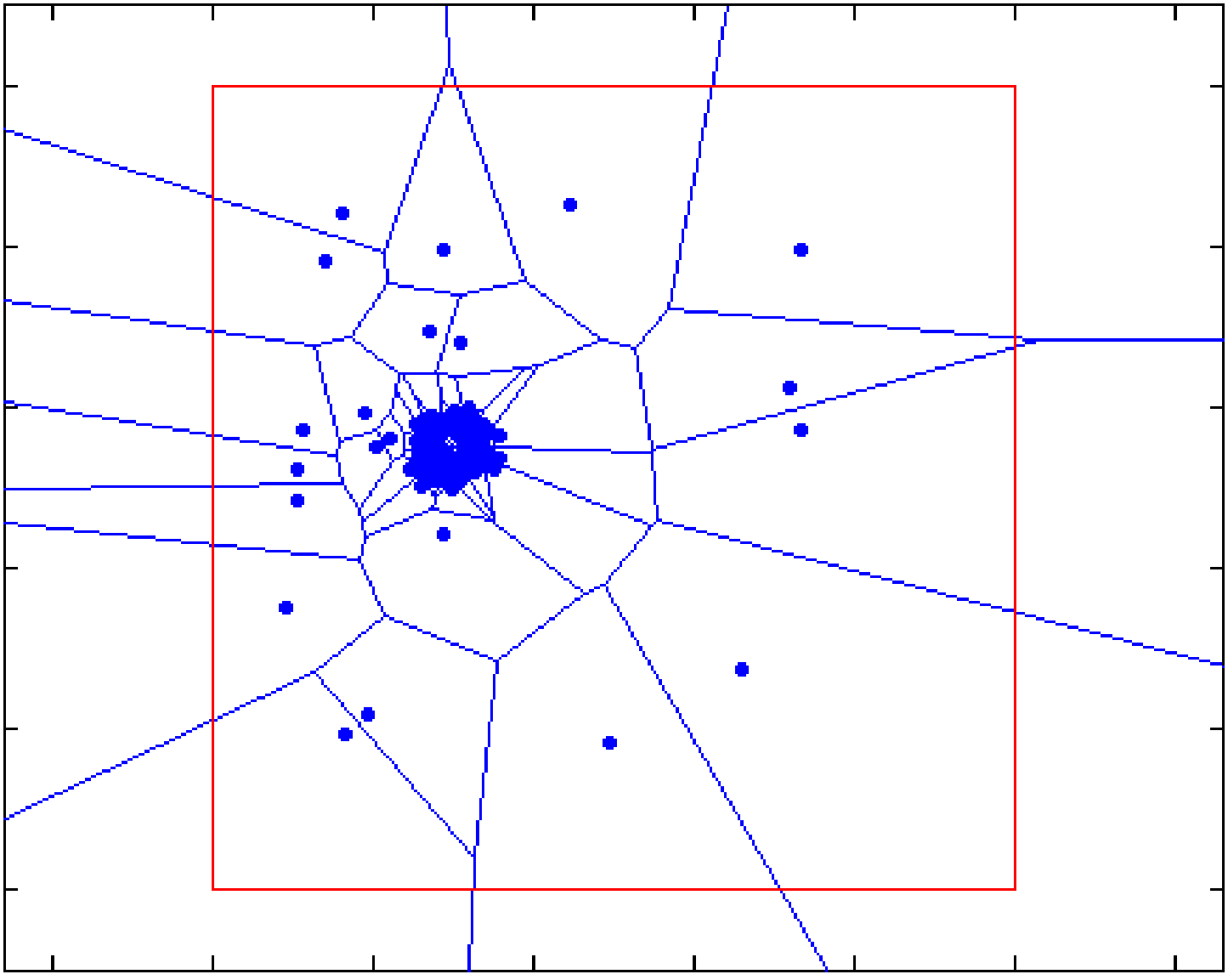
$$\bar{x}_i = \frac{\sum_{p \in C_i} x_p}{\sum_{p \in C_i} 1}, \quad \bar{y}_i = \frac{\sum_{p \in C_i} y_p}{\sum_{p \in C_i} 1}$$

(This allows us to work with complicated shapes, spheres, higher dimensions and so on!)

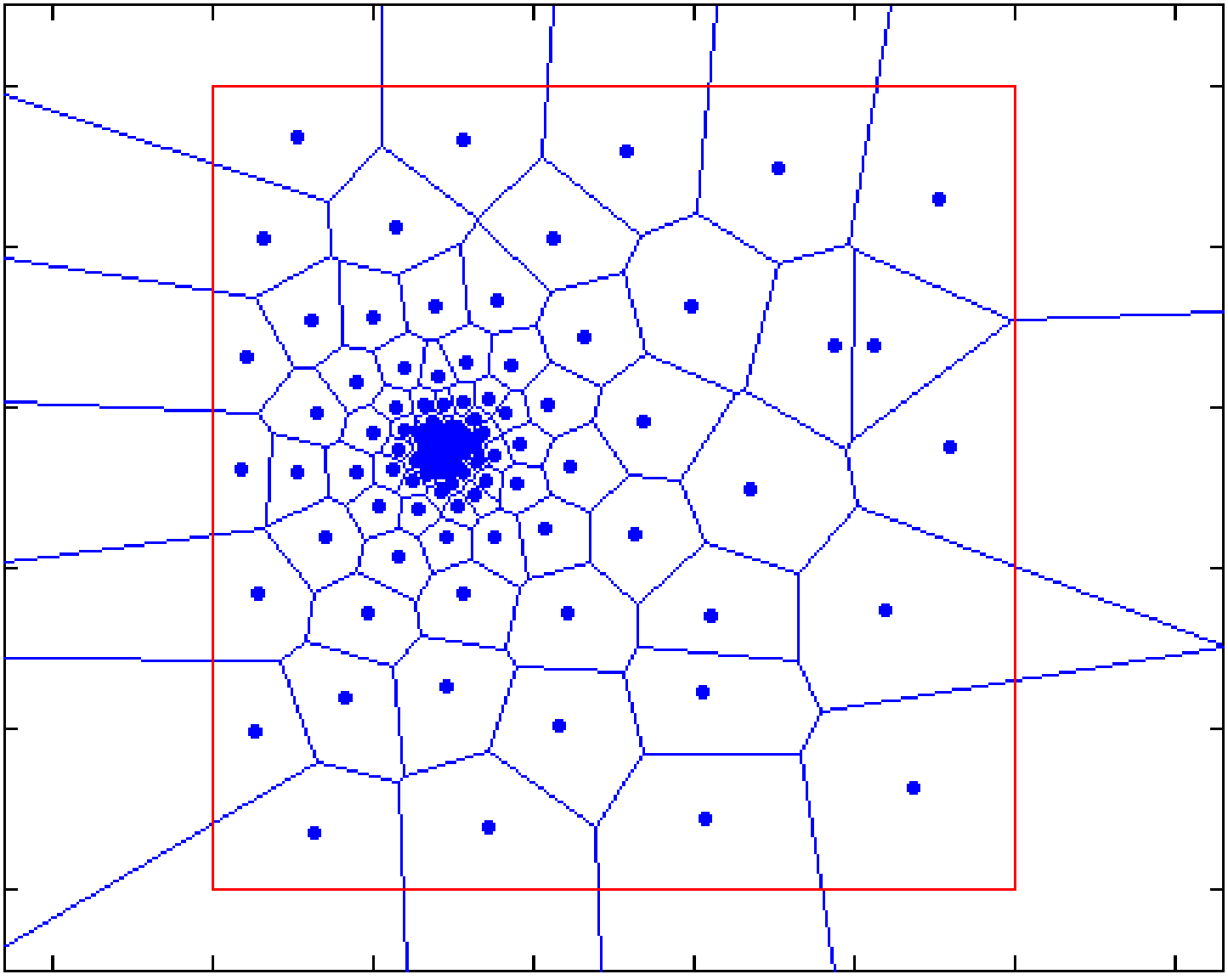
Centered Systems: Step 001



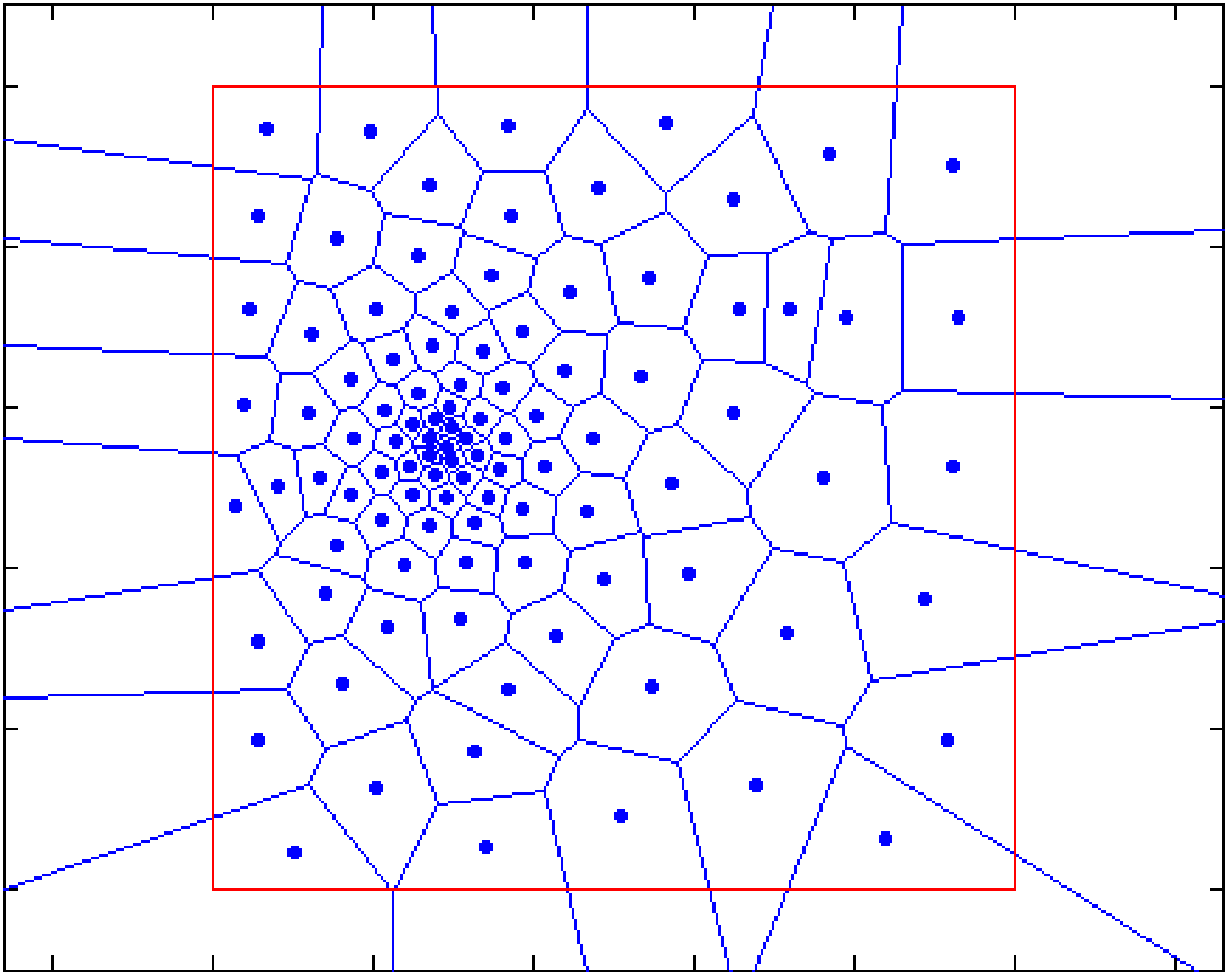
Centered Systems: Step 002



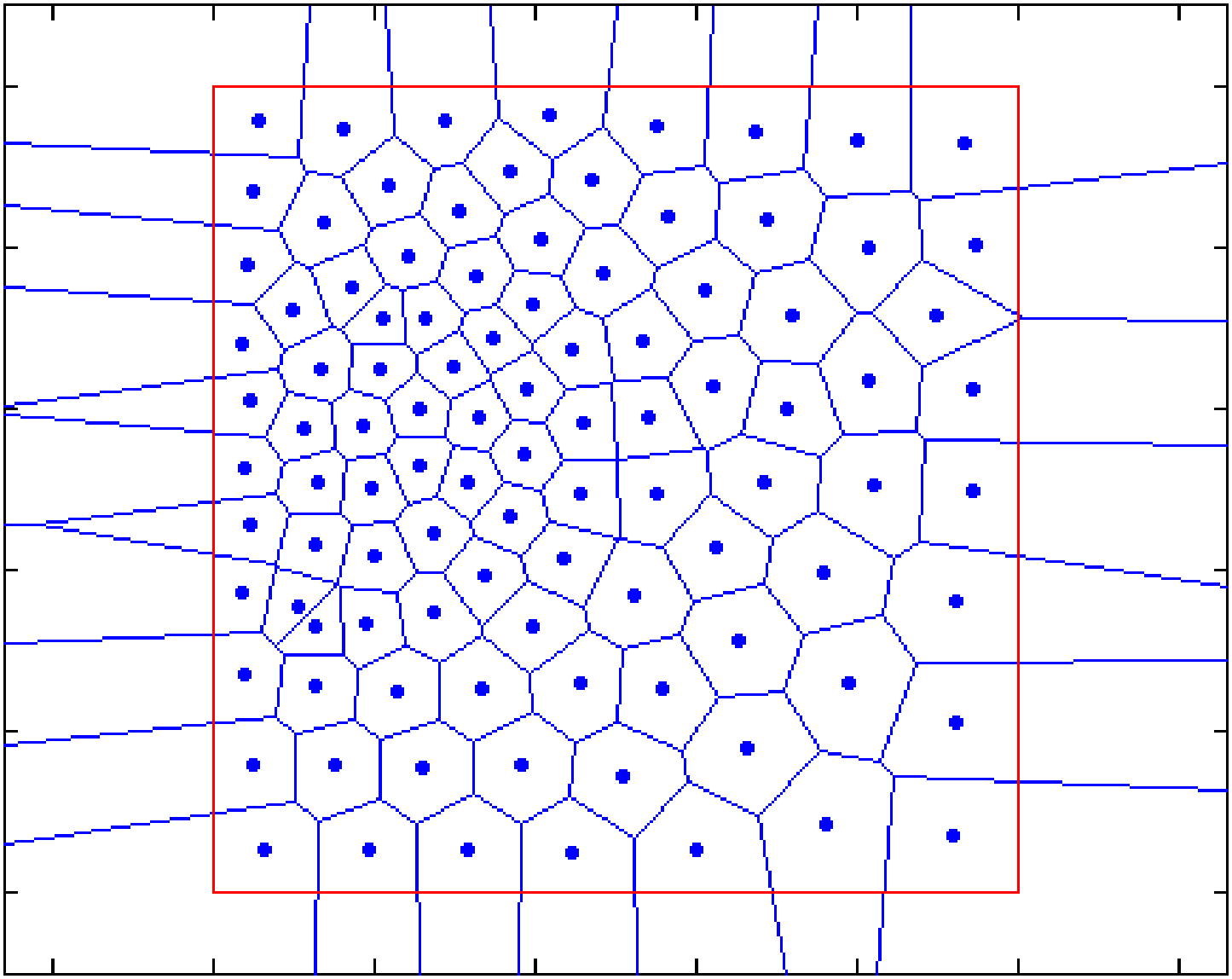
Centered Systems: Step 010



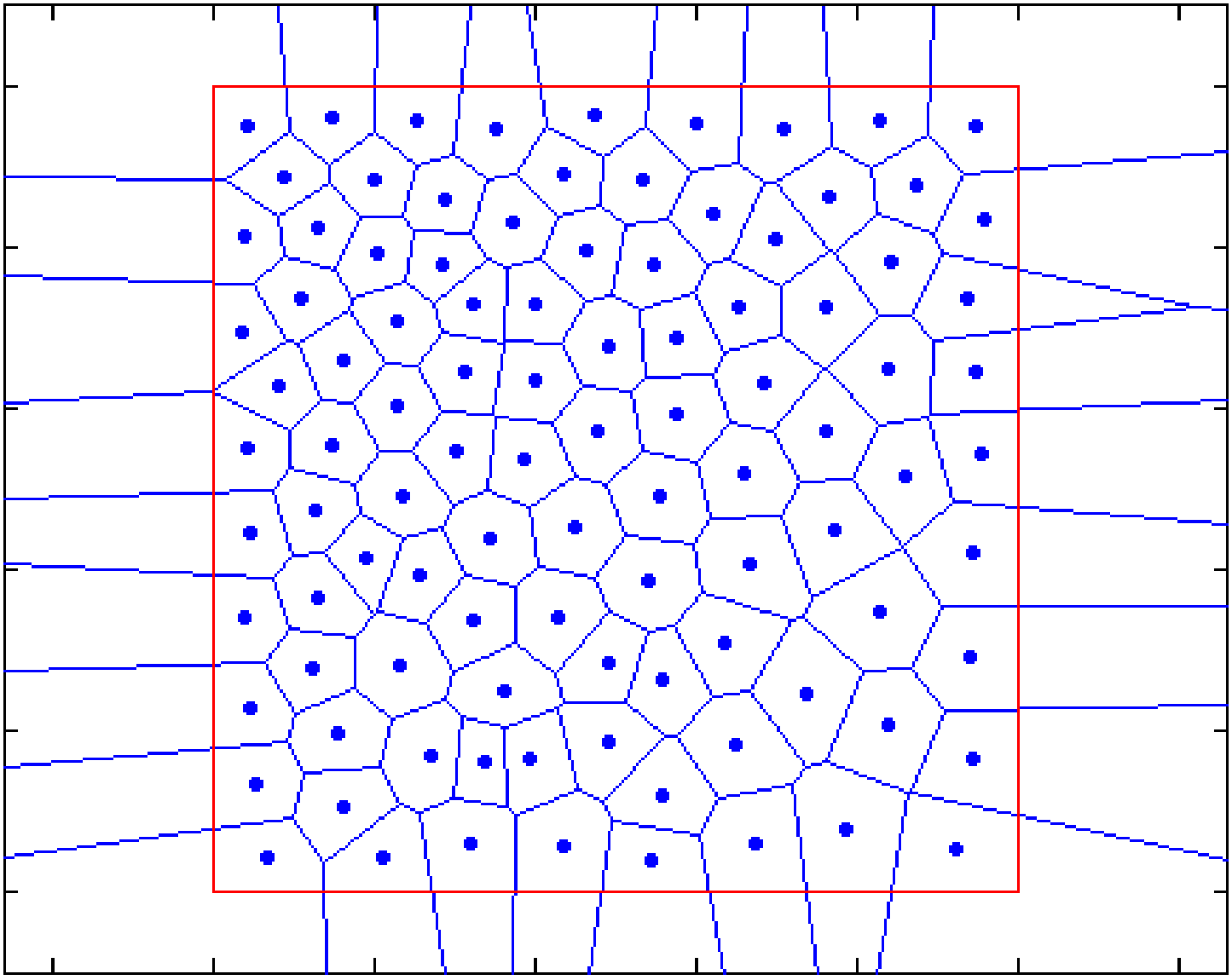
Centered Systems: Step 020



Centered Systems: Step 040

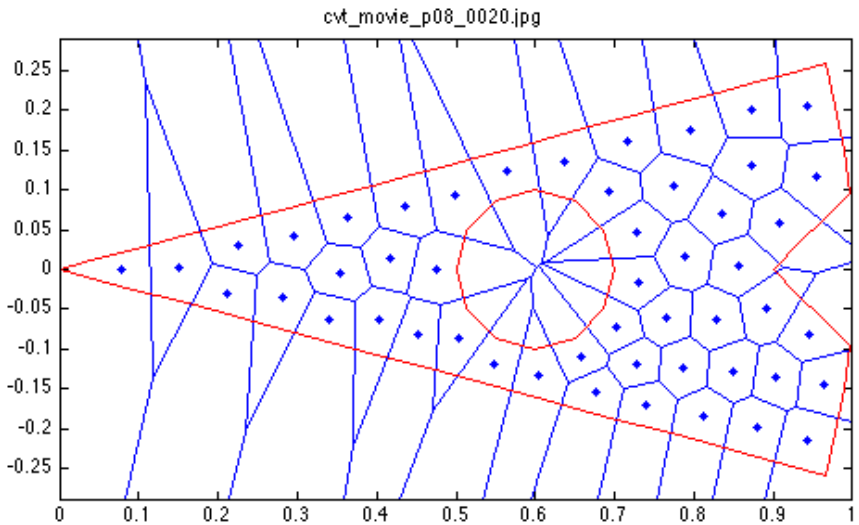


Centered Systems: Step 080



Centered Systems: Regions with Holes and Corners

To get a better feeling for how these computations work, let's look at one or two **animations**, set in a pie shaped region.



- **cvt_movie_p08.mov**, random initial points
- **cvt_movie_p08_cramped.mov**, all points start in one place

Centered Systems: Density Functions

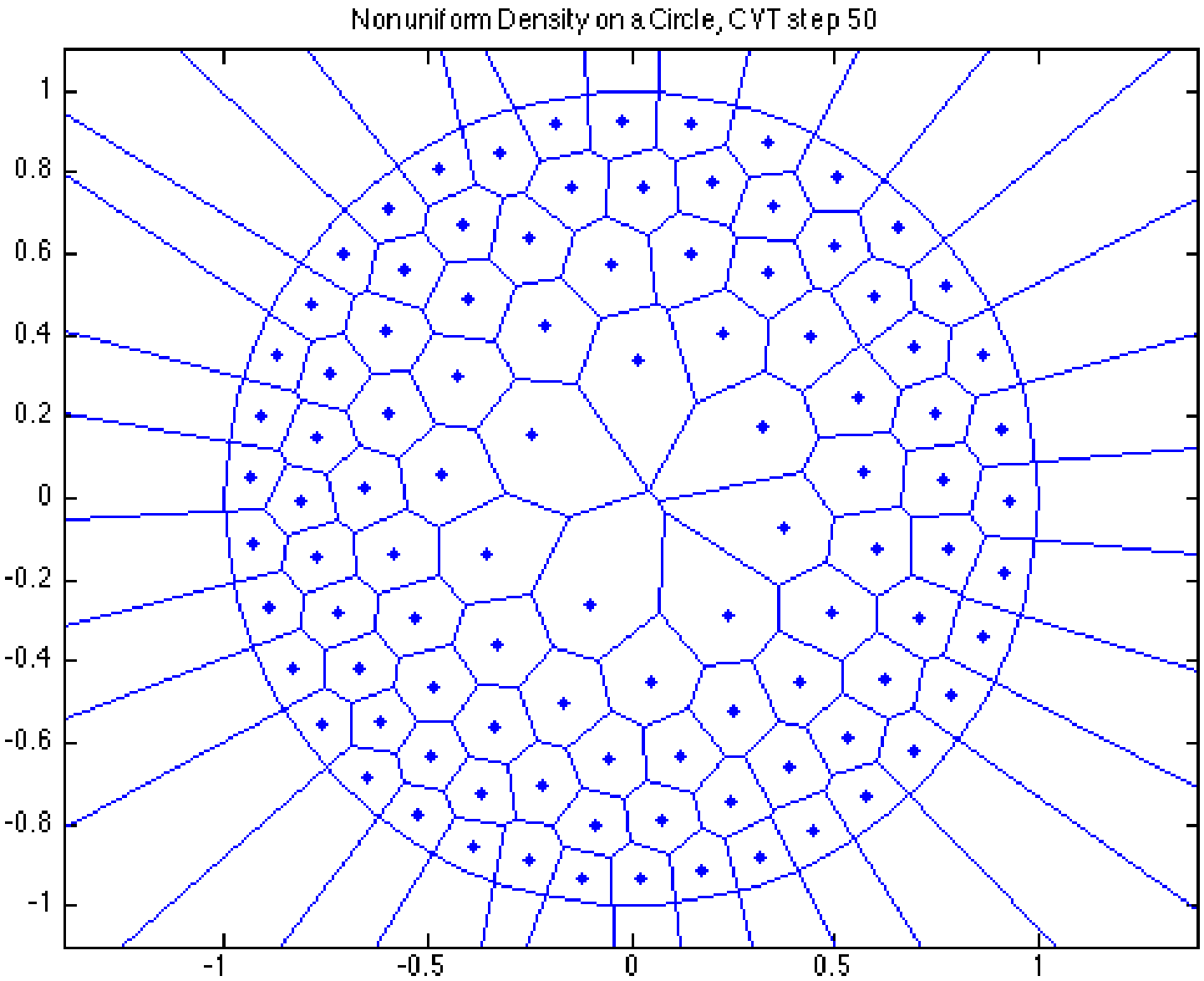
By using **density functions**, we can vary the size of the regions.

Two regions will not have the same area, but rather the same "weight", defined as the integral of the density.

This is like dividing a country into provinces of equal population.

Using this idea, for instance, you can generate a mesh for a computational region, and force the mesh to be fine near the boundaries, and very fine near corners or transition zones.

Centered Systems: Nonuniform Density, Curved Region



THE DEATH MAP

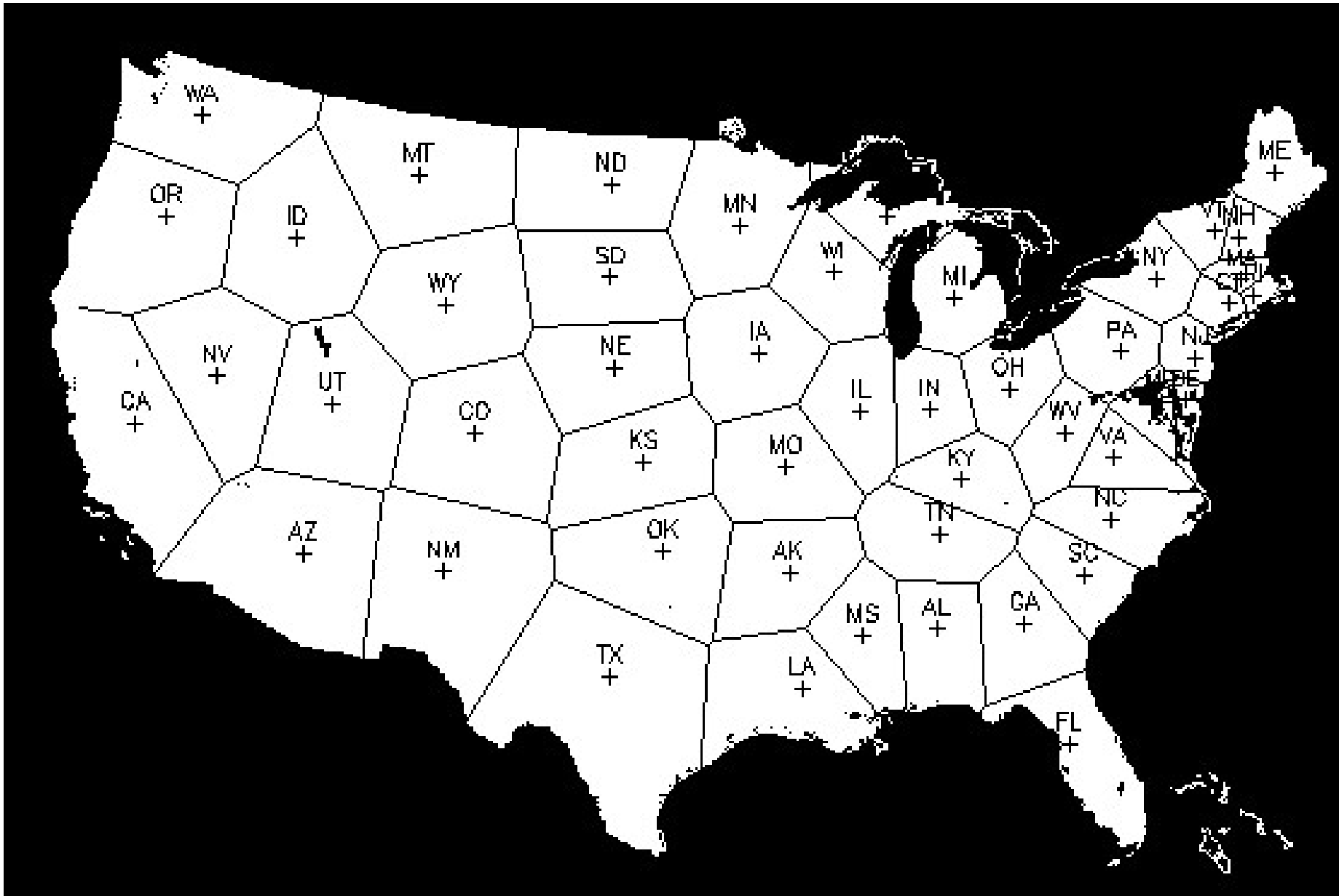
1. A Casebook
2. Death in Golden Square
3. The Voronoi Diagram
4. Euler's Formula
5. Voronoi Computation
6. Centered Systems
7. **CONCLUSION**

Conclusion: Voronoi Diagrams

The Voronoi diagram is a mathematical tool for analyzing geometric processes with many local centers.

- disease outbreaks
- patterns on animal skin or sea shells
- territories of ant colonies, prairie dogs
- how can an attacking plane best avoid all enemy bases?
- if we have already chosen capitals for the states, where should state boundaries be?

Conclusion: The Voronoi States of America



Conclusion: Centroidal Voronoi Diagrams

The Centroidal Voronoi diagram helps us to simulate, model, and sometimes to understand cell structures in which the generator tends to be in the center, or to move there:

- how do fish adjust their nesting sites?
- how do rotating cells form in a cooling liquid?
- placement of mailboxes
- arrangement of sonar receivers on ocean floor
- what arrangement of bases makes it hardest for planes to attack?

Conclusion: Computational Geometry

Computational Geometry: geometric algorithms for:

- distances, areas, volumes, angles of complicated shapes;
- nearest neighbors;
- shortest paths, and best arrangements of points;
- hulls, triangulations, Voronoi diagrams;
- meshes, decomposition of shapes into triangles or tetrahedrons;
- motion of an object through a field of obstacles.

CGAL is a computational geometry library;
Mathematica includes a computational geometry package;
MATLAB has many computational geometry routines.

Conclusion: The John Snow Memorial