Dot Patterns Perceptual Grouping: RDG vs K-means

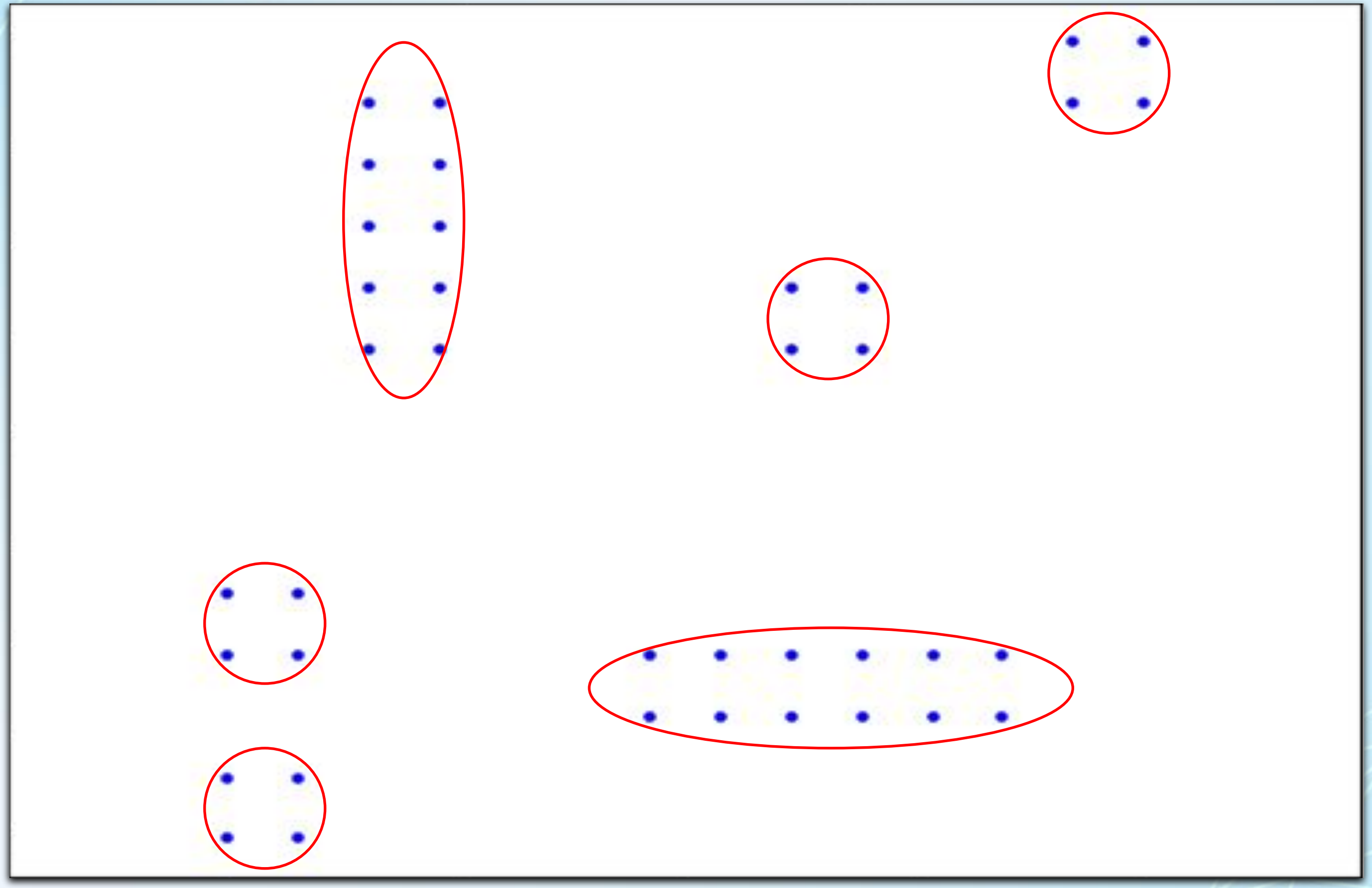
What will we talk about?

1. Dot Patterns, perception and Gestalt
2. Clustering and K-means
3. Algorithm based on the Reduced Delaunay Graph
4. Comparison: K-means vs RDG

How can the perception of Dot Patterns be modeled?

1. Dot Patterns, perception and Gestalt

How many groups do we perceive?



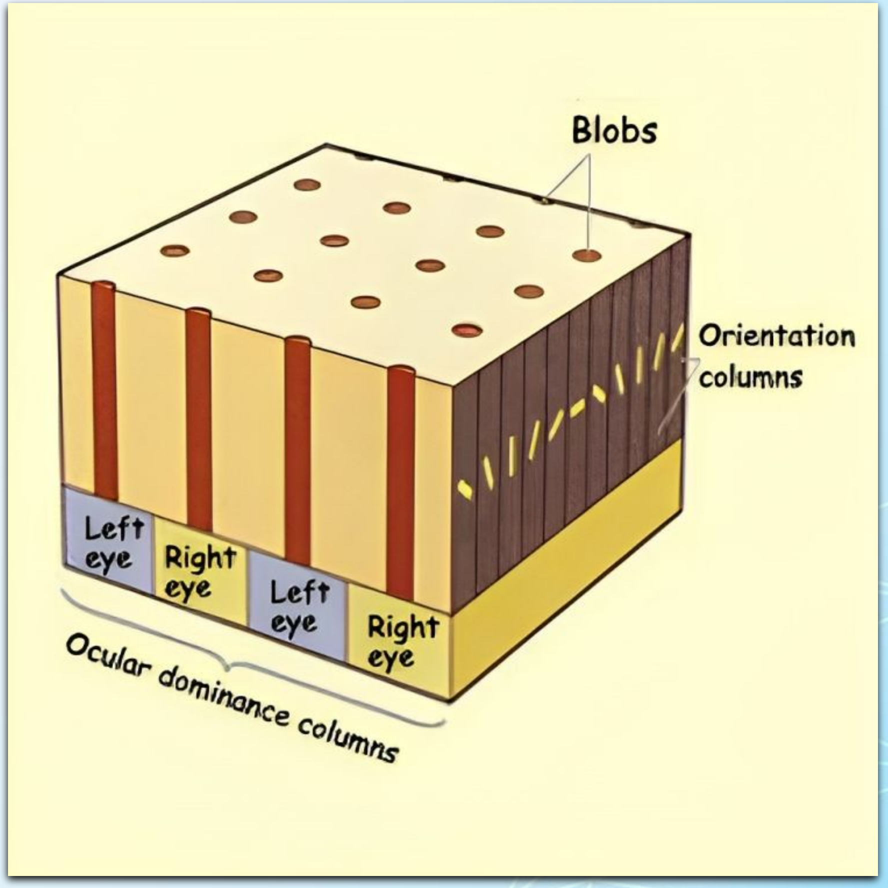
Location and Orientation Columns

Neurons in a location column have **close receptive fields**.

Inside a location column there are orientation columns with certain **preferred orientations**.

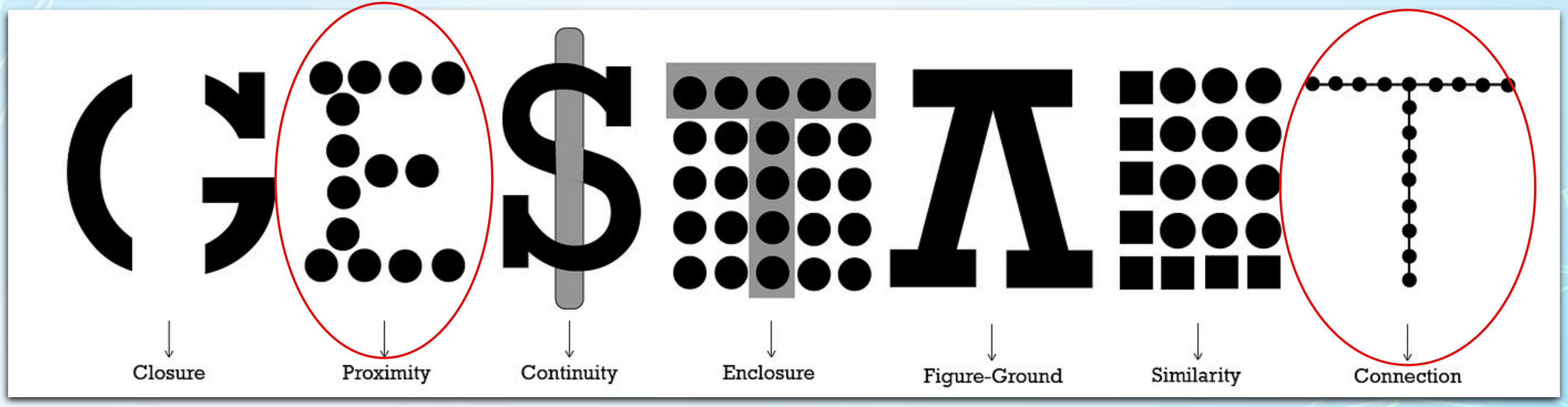
The orientation columns are grouped in **pinwheels** alternating with **blobs**.

The **ocular dominance columns** receive input from an eye or the other via the **Lateral Geniculate Nucleus**.



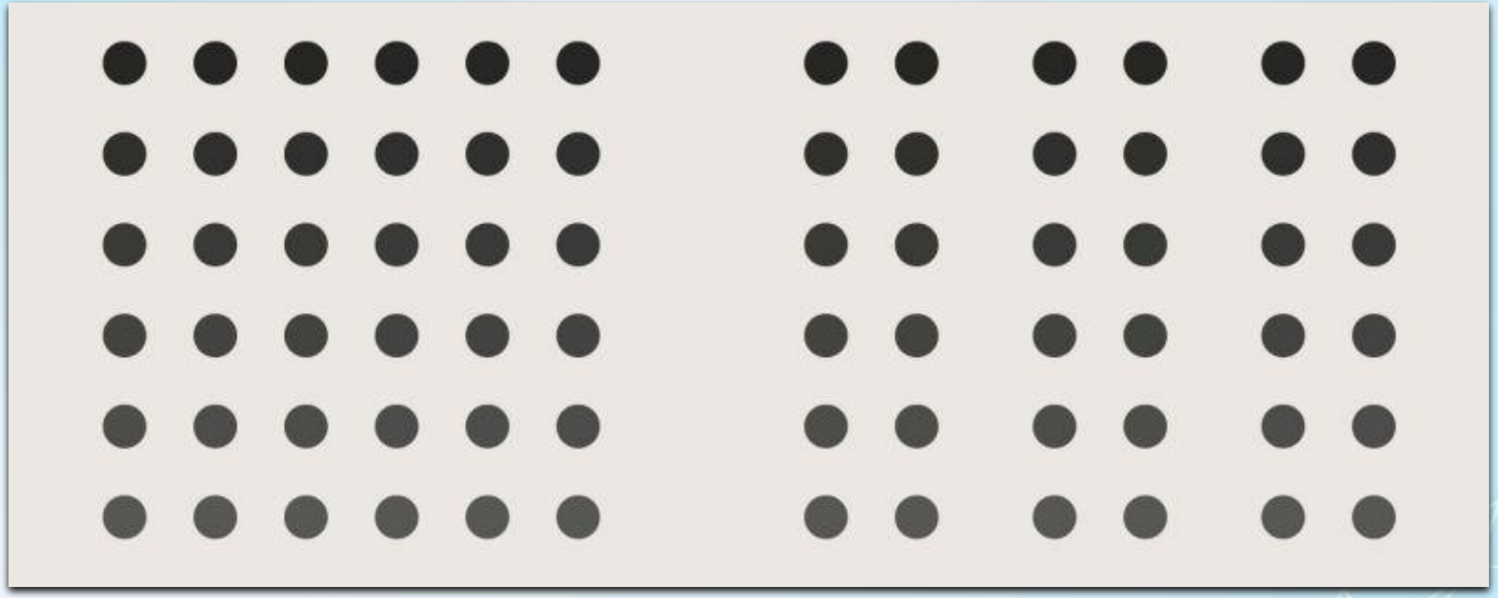
Gestalt Laws

The psychology of Gestalt concerns **form** and **representation** in the context of the Human Visual System.
Elements in a group can have properties that emerge from **mutual relationships**.



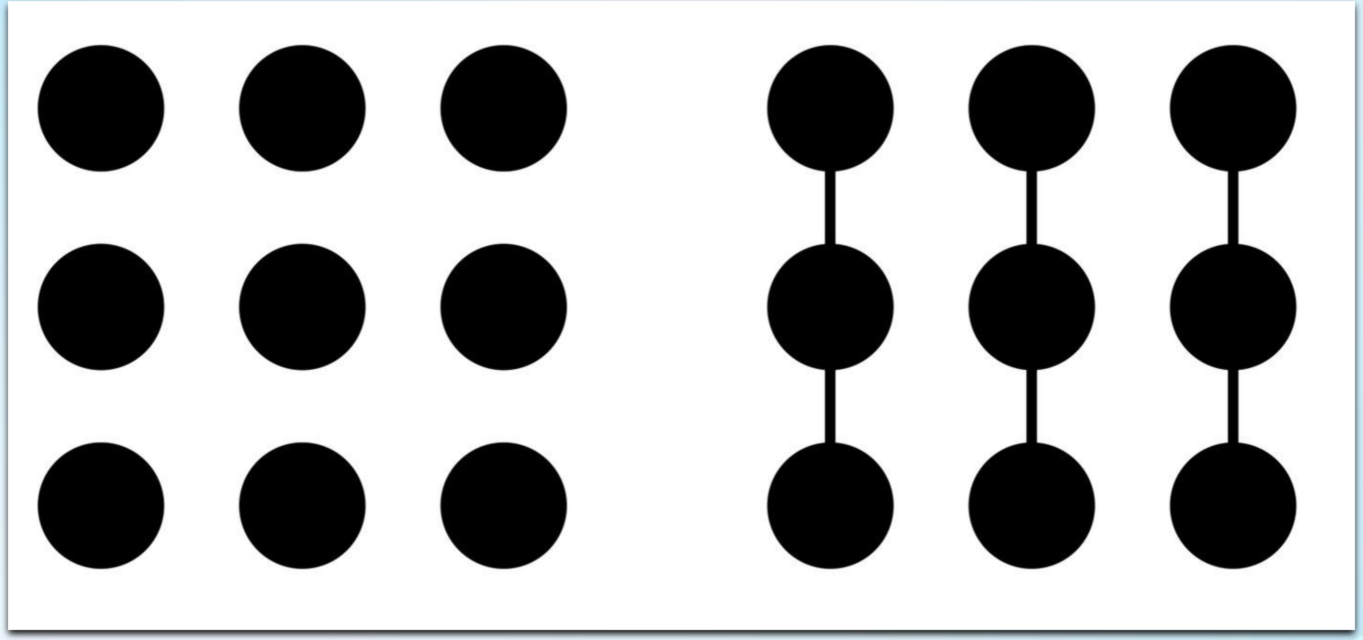
Gestalt Law: proximity

The human eye tends to **group nearby elements**, separating them from those farther away.



Gestalt Law: connection

The human eye tends to **group elements connected by lines**.



We want to reproduce this mechanisms **algorithmically**!

2. Clustering and K-means

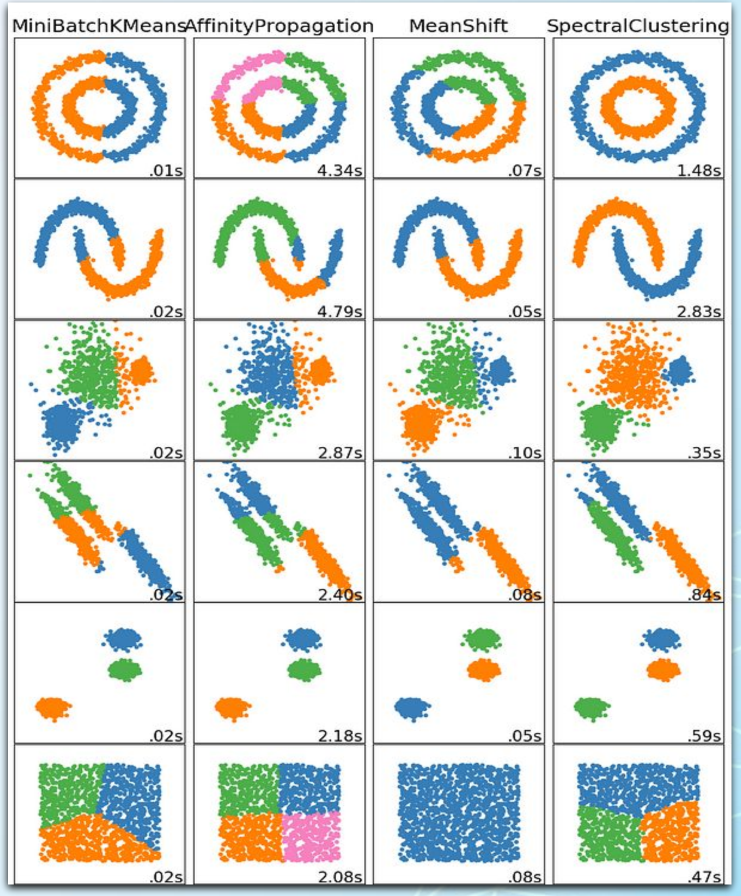
A way to model Gestalt: Clustering

Clustering is a set of unsupervised techniques aimed at selecting and **grouping homogeneous elements** in a data set.

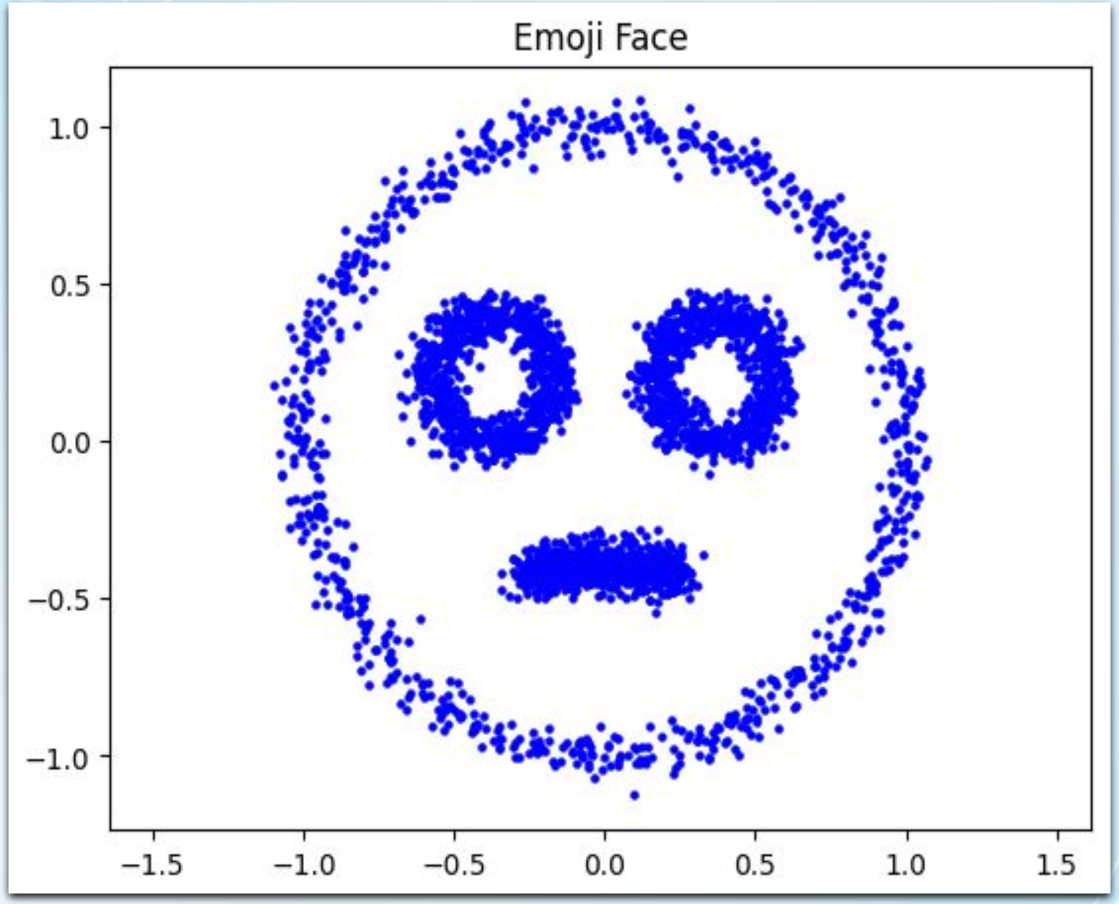
Tipologie:

- **Hard clustering**: each element is assigned to one and only one cluster.
- **Soft/fuzzy clustering**: an element can belong to multiple clusters with different degrees of membership.

- **Partitioning clustering** (or not hierarchical or k-clustering): group membership is determined by the distance from a representative point. Example: **k-means**.
- **Hierarchical clustering**: uses a hierarchy of partitions characterized by an increasing or decreasing number of groups.



Dot Patterns

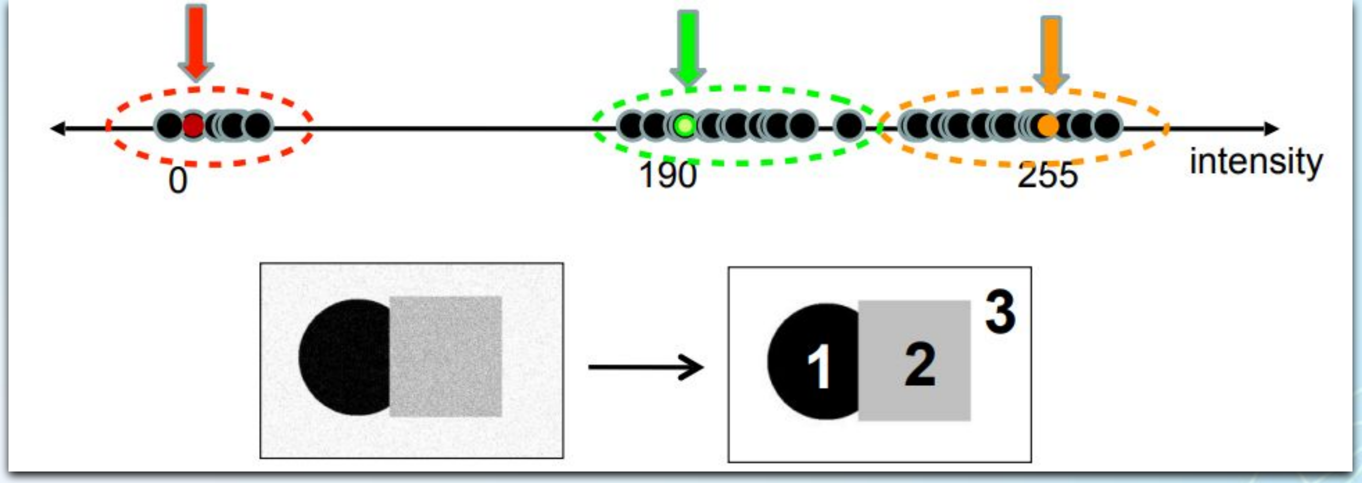


K-means clustering

First idea: using the **k-means** algorithm.

Group identification → selecting representative centroids that minimize a certain **cost function**: $\sum_{j=1}^{k} E(C_j)$

For example the **Sum of Square Distances** (SSD): $\sum_{cluster\ i} \sum_{points\ p\ in\ cluster\ i} ||p - c_i||^2$



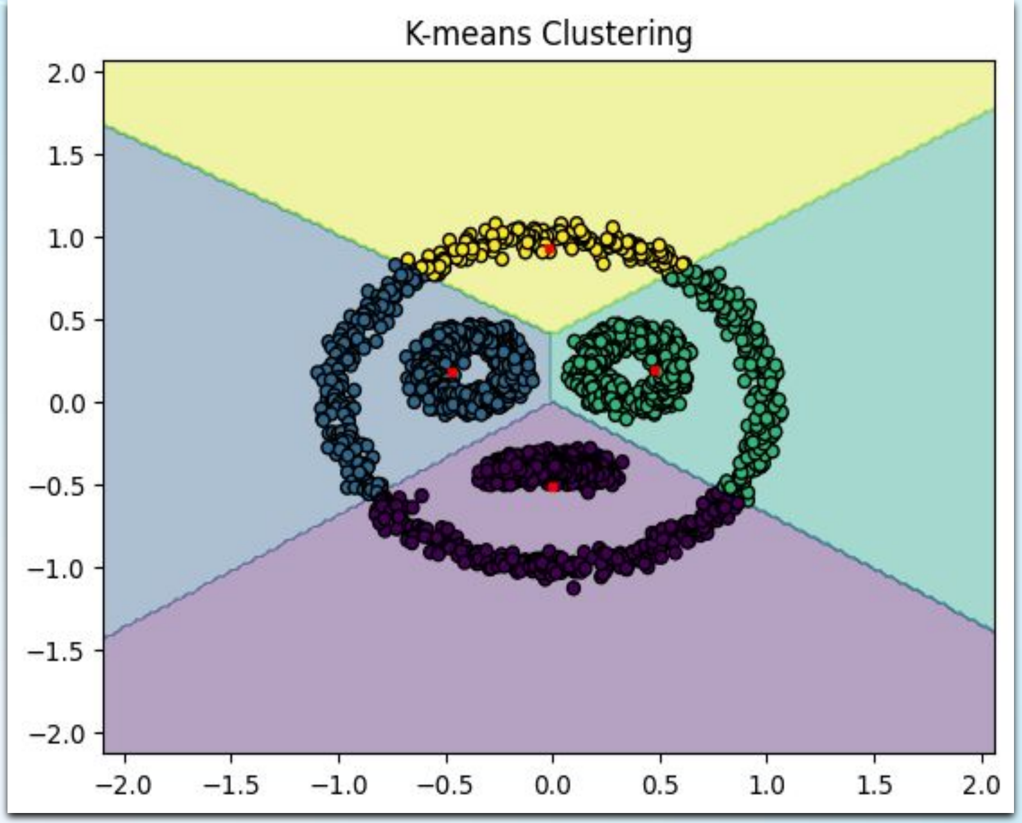
Chicken and egg problem with the groups and corresponding centroids!

K-means clustering

Input: an array of points in the plane characterized by a pair of coordinates *[x, y]* and the number *k* of clusters.
Output and effects (typically): the coordinates of centroids for each cluster and the assignment of each input point to a cluster.

1. **Randomly** initialize *k* centroids $c_1, \ldots, c_k$
2. Given the centroids find points for each cluster:
   a) For each point *p* find the nearest c_i
   b) Put *p* in cluster *i*
3. Given the found points in each cluster, find c_i and set c_i as the mean of the points in cluster *i*
4. If c_i has changed from the current one go back to 2. otherwise terminate

K-means clustering



We want to model **perception** more accurately!

Voronoi Diagram and Delaunay Triangulation

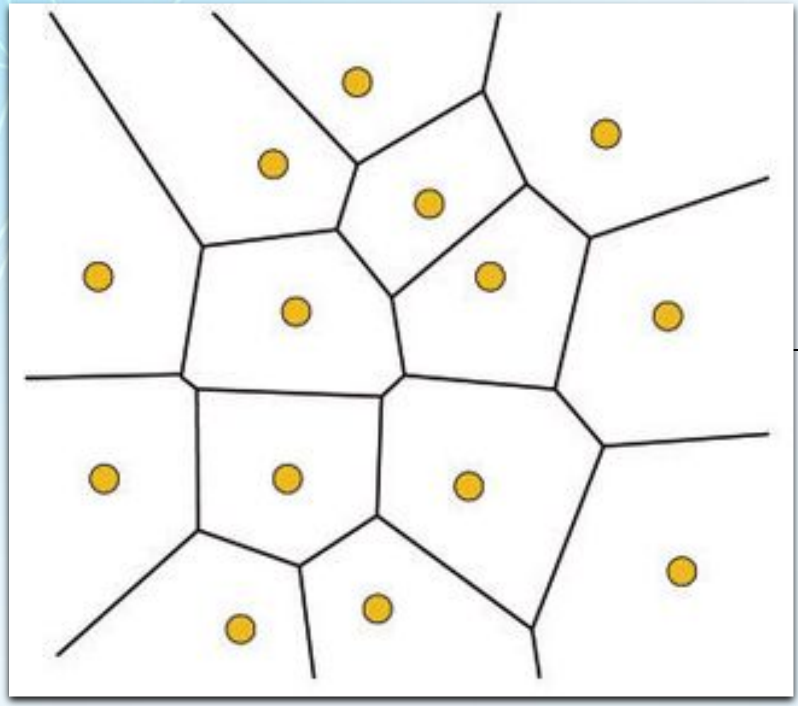
Given a set $S = \{p_1, \ldots, p_N\}$ of ***N* points** in the plane, **partition the plane in cells** $C_1, \ldots, C_N$ so that the points belonging to the cell C_j, associated with the point $p_j \in S$, are nearer to p_j than any other point $p_k \in S, k \neq j$:

$$q \in C_j \Leftrightarrow d(q,p_j) \leq d(q,p_k) \qquad \forall q, \ \forall p_j, p_k \in S$$

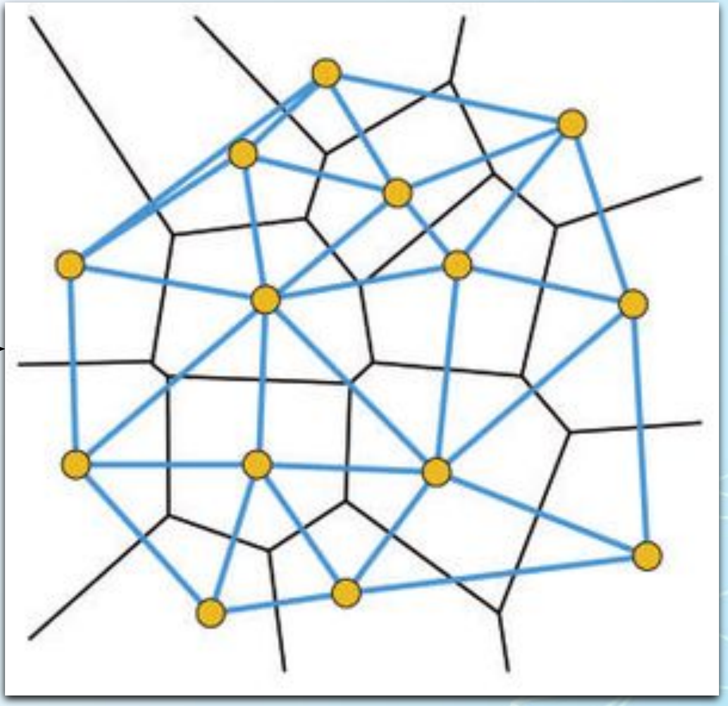
The dual of the Voronoi Diagram or Tessellation, called the Delaunay Triangulation or Graph, is obtained by **connecting all pairs of points** in set S whose Voronoi Diagram cells **share a boundary**.

3. Algorithm based on the Reduced Delaunay Graph

Voronoi Diagram

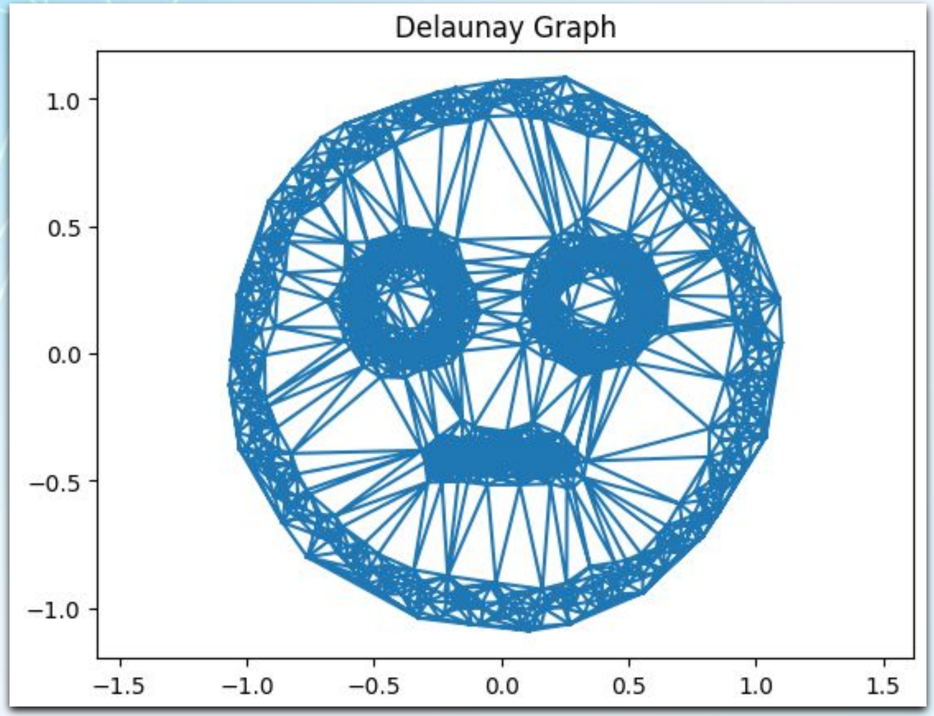


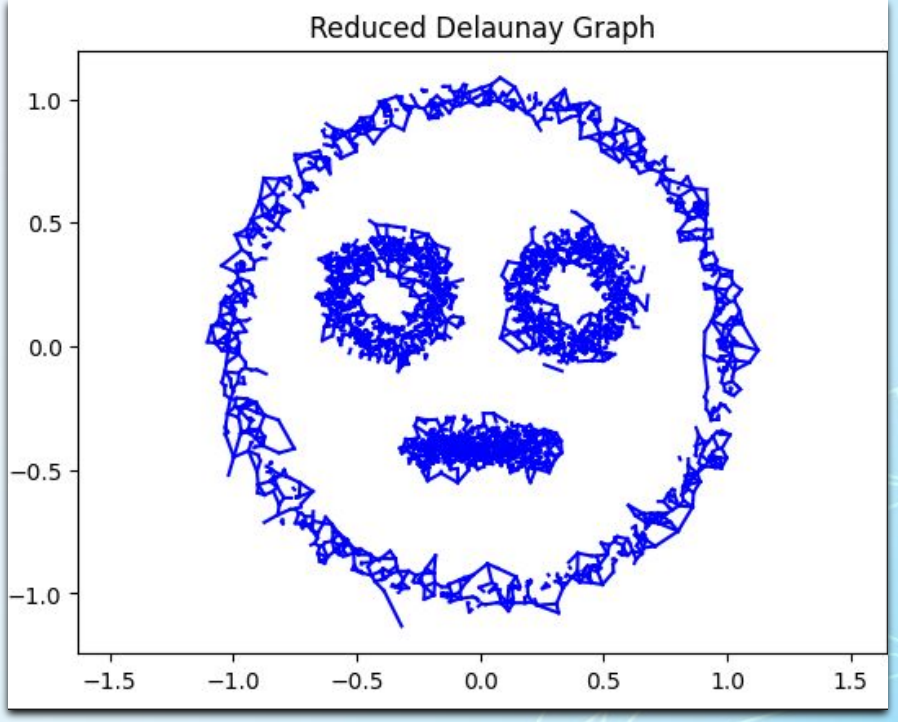
Delaunay Triangulation



Reduced Delaunay Graph

Idea: keep **only some edges** so to highlight nearby zones.





Reduced Delaunay Graph

To select which edges *pq* to remove, calculate a **normalized distance**:

$$\xi(p,q) = \frac{d(p,q)}{\min_{x \in S}\{d(p,x)\}}; \qquad \xi(q,p) = \frac{d(q,p)}{\min_{x \in S}\{d(q,x)\}}$$

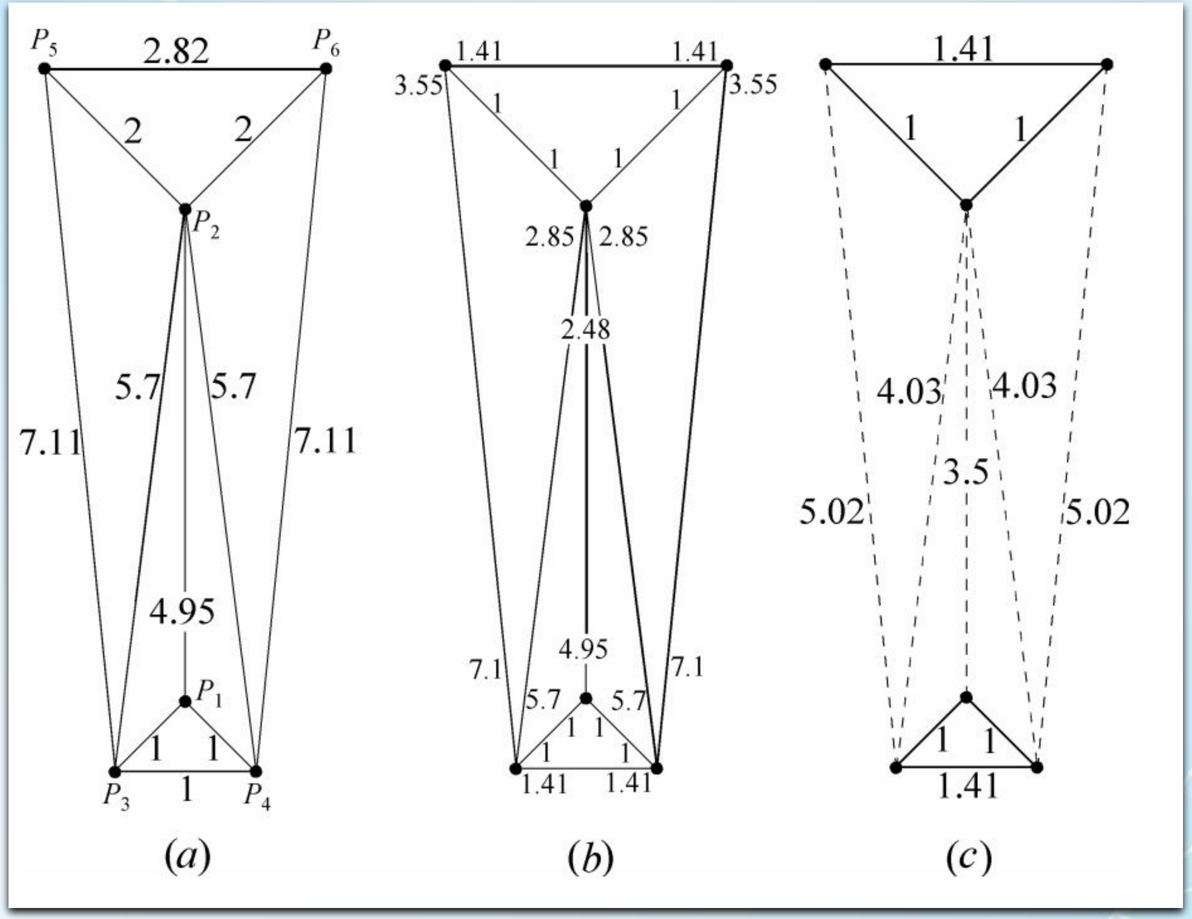
In doing so, two ratios $r_1(e) = \xi(p, q)$ *and* $r_2(e) = \xi(q, p)$ are assigned to each edge in the graph. Then, reduce them to one quantity, their **geometric average**:

$$r(e) = \sqrt{r_1(e) \cdot r_2(e)}$$

And remove every edge from the graph whose *r(e)* is greater than a certain **threshold** r_T:

$$V_{RDG} = V_{DG}; \qquad E_{RDG} = \{ e \in E_{DG} \mid r(e) \le r_T \}$$

Reduced Delaunay Graph



RDG: Algorithm

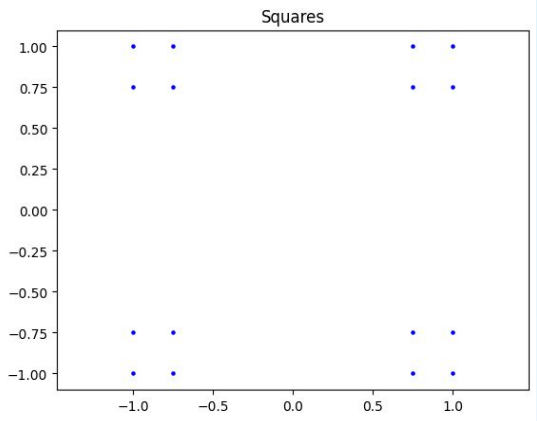
Input: an array of points in the plane characterized by a pair of coordinates *[x, y]*.

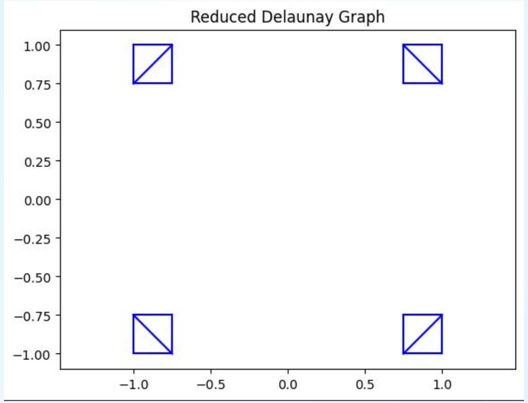
Output: a reduced array of points which form the RDG when connected with edges.

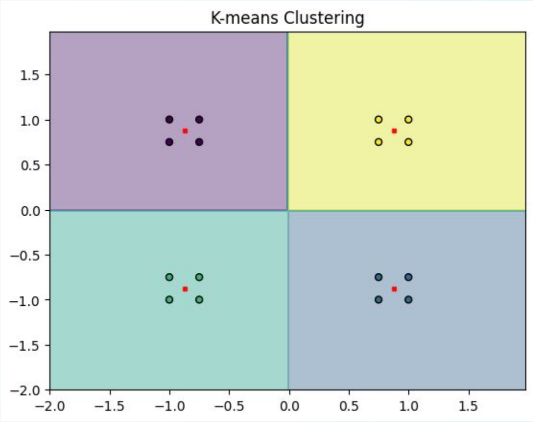
1. Compute the **Delaunay Triangulation** on the points given as input
2. For each point, compute the distances to its **neighbours**
3. **Normalize** the distances with the minimum of the distances to the neighbours
4. Compute the **geometric average** between the two ratios found for each edge and assign the result to it
5. **Thresholding**: remove the edges whose value is greater than a certain threshold

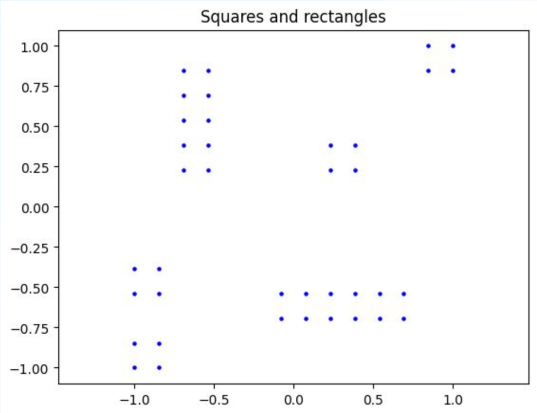
4. Comparison: K-means vs RDG

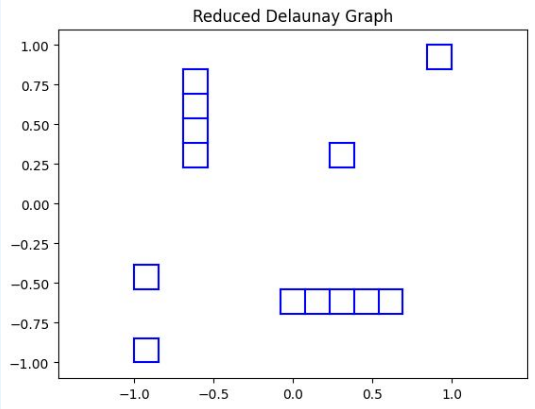
K-means vs RDG

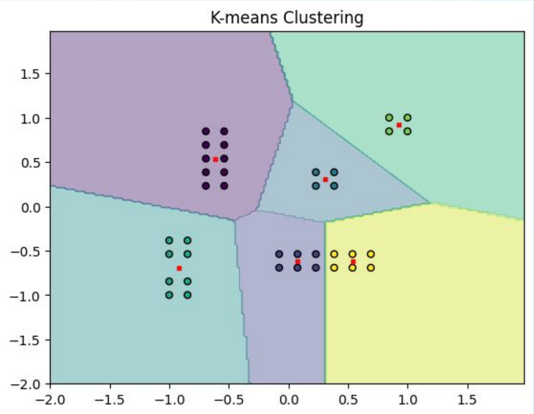




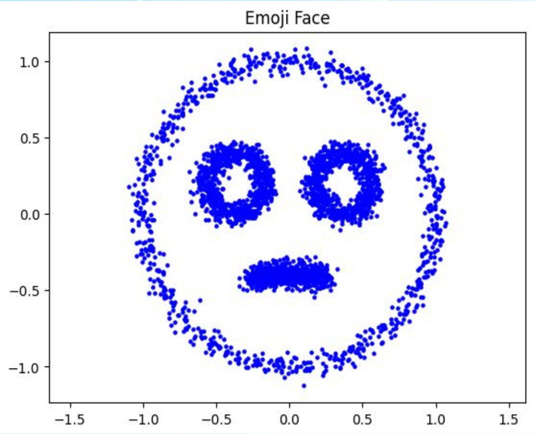


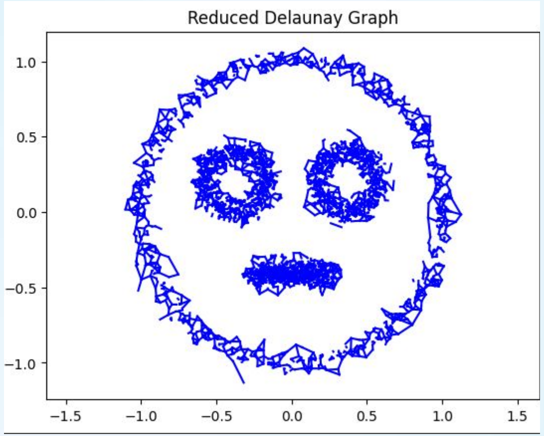


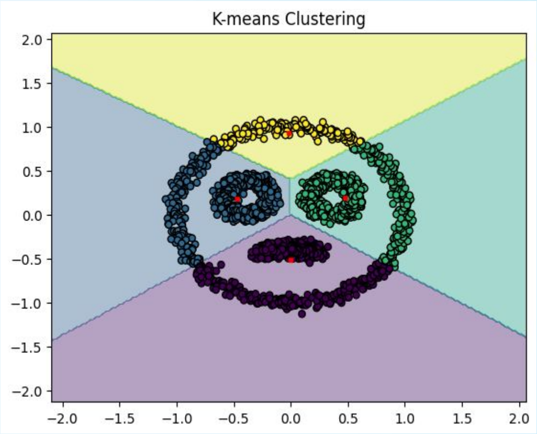


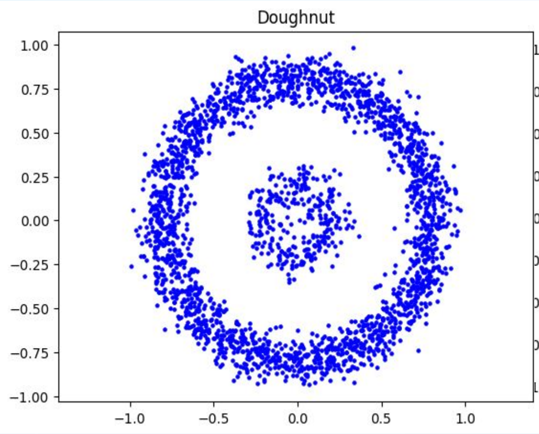


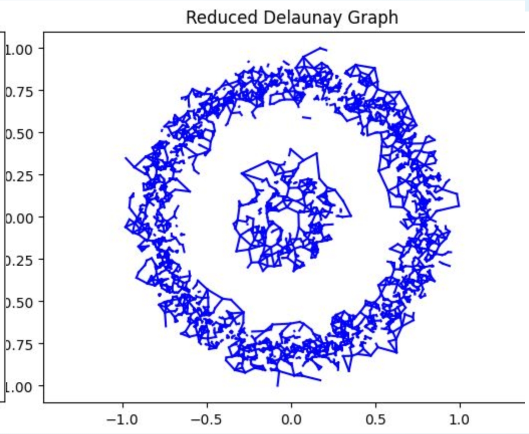
K-means vs RDG

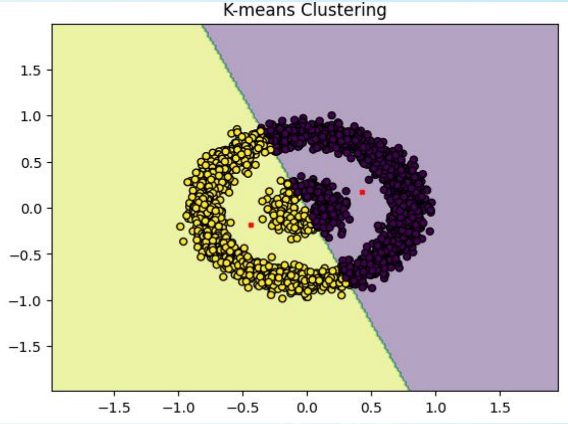




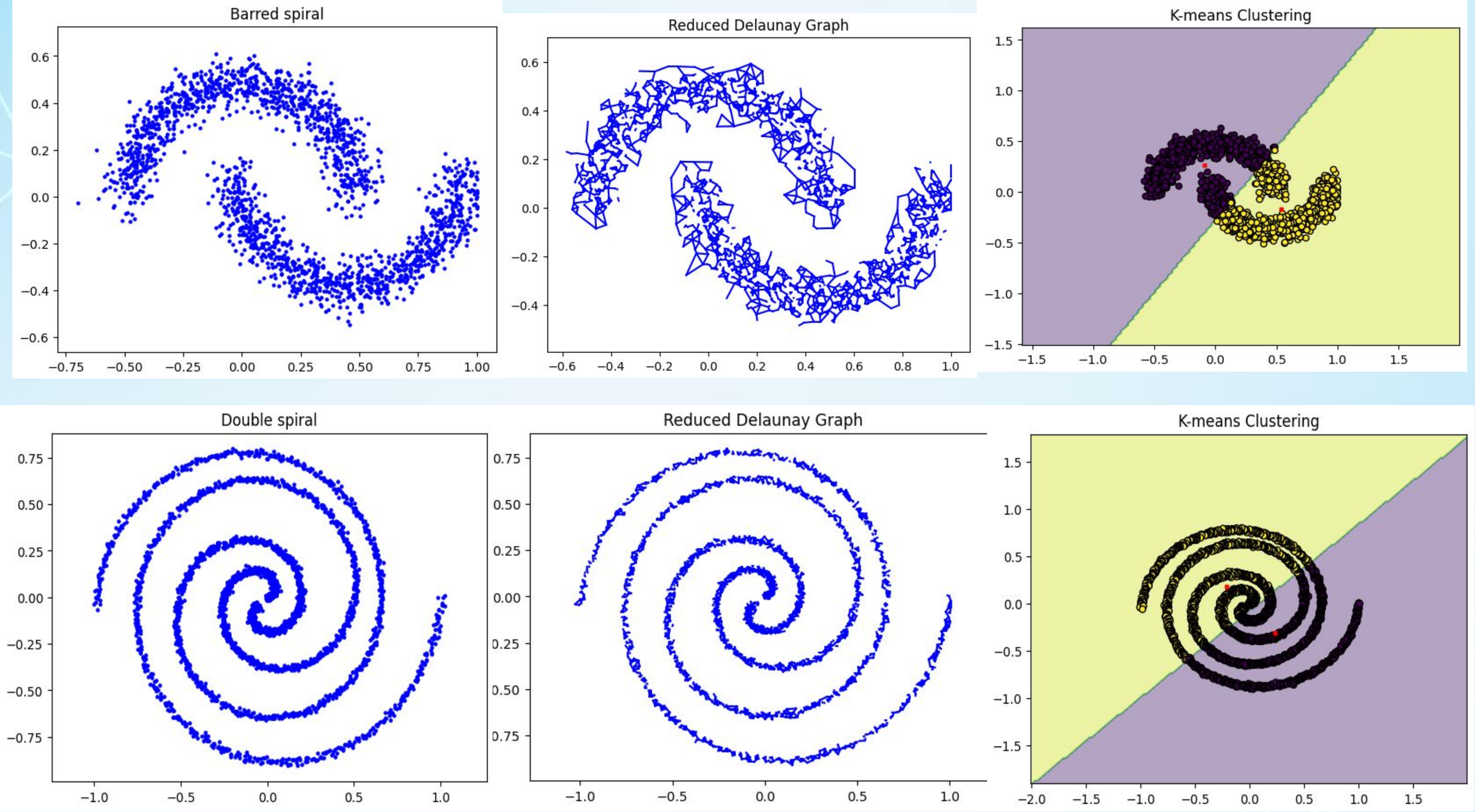








K-means vs RDG



K-means vs RDG

Similarities:
- Both are **non-supervised** algorithms
- The **tessellation** of the plane can be part of their operations or output
- Both work well with separated **point clouds**

Differences:
- RDG uses a **graph** to consider the distances between points
- RDG technically doesn’t select representatives for the computed groups
- K-means can have elements of **randomness**

K-means vs RDG

K-means pros:
- **Simple** to implement
- Computationally (pretty) **efficient**
- **Always converges** to a **local minimum** for each cluster

K-means cons:
- Setting *k* is difficult
- Sensitive to **random** initial centroids and **noise**
- Suited mainly to **spherical clusters**
- Doesn’t model human perception that well

RDG pros:
- Models human perception well
- Can find groups even in **complex shapes**
- Robust to **far away outliers**

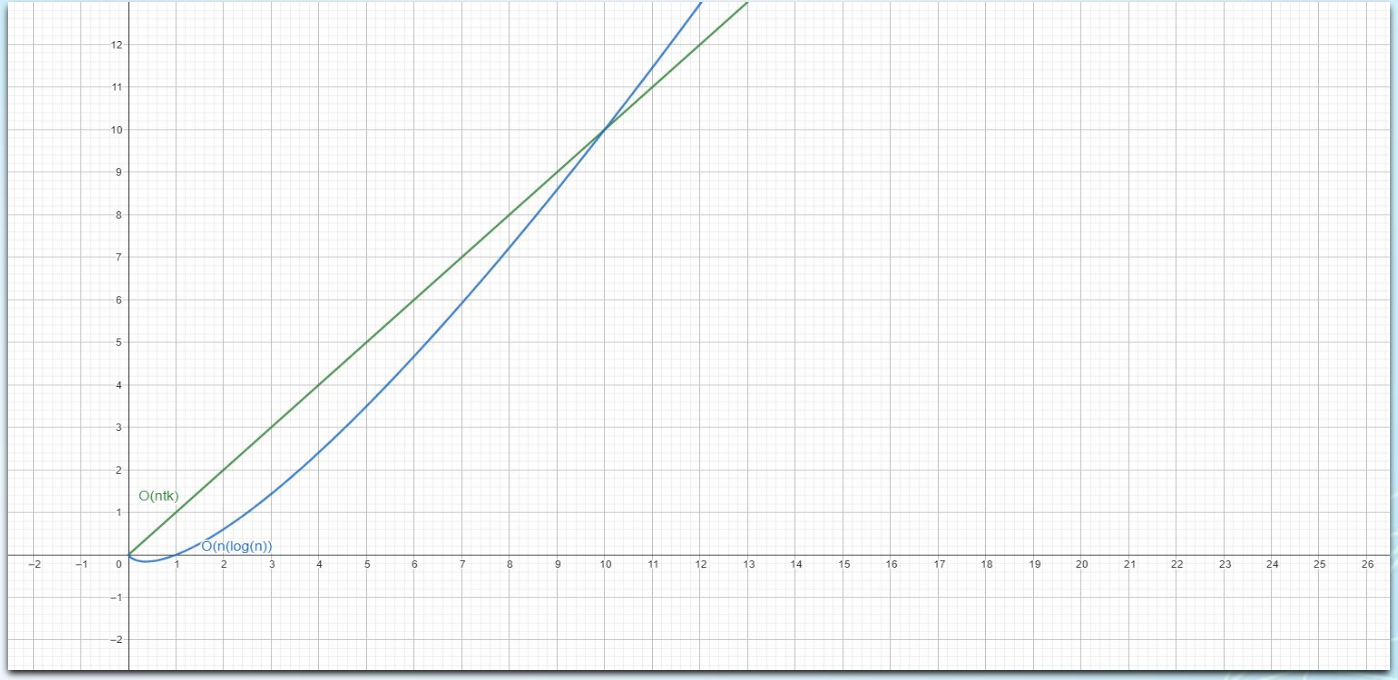
RDG cons:
- Setting the threshold is difficult (especially at different scales)
- Computationally **heavy**

K-means vs RDG: complexity

K-means: *O(ntk)*, with *n* as number of points, *t* as number of iterations for converging to a local minimum, *k* as chosen number of clusters.

RDG: *O(nlog(n))*, with *n* as the number of points.

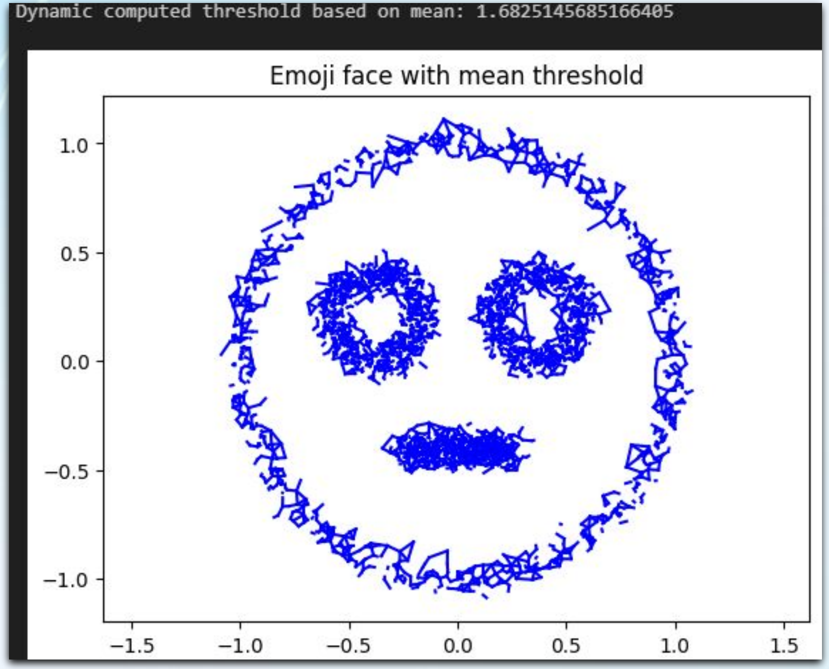
Complexity deducted empirically.



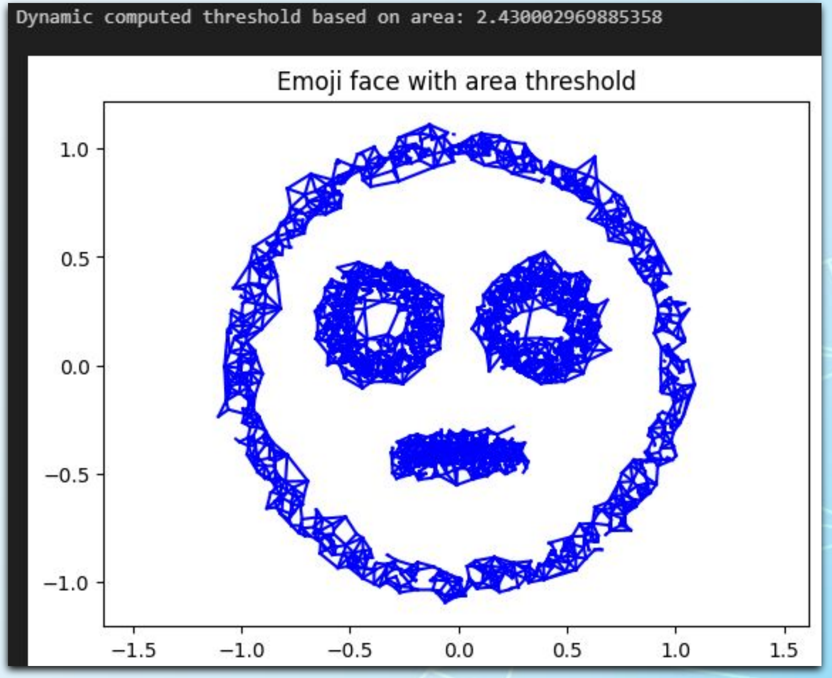
How could it be improved?

One thing we thought about was introducing a **dynamic threshold**.

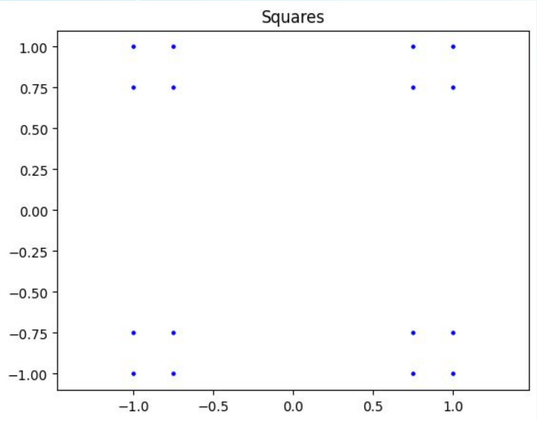
Based on the **average of the values** of each edge.

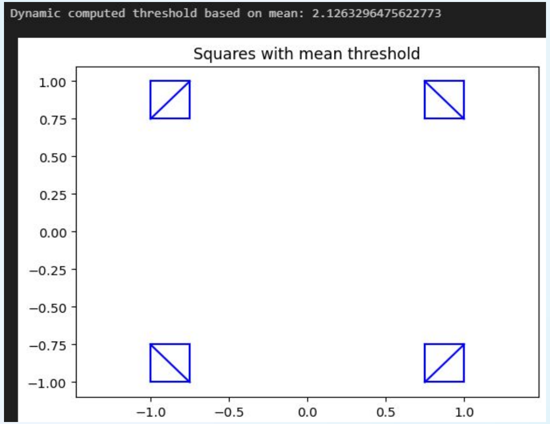


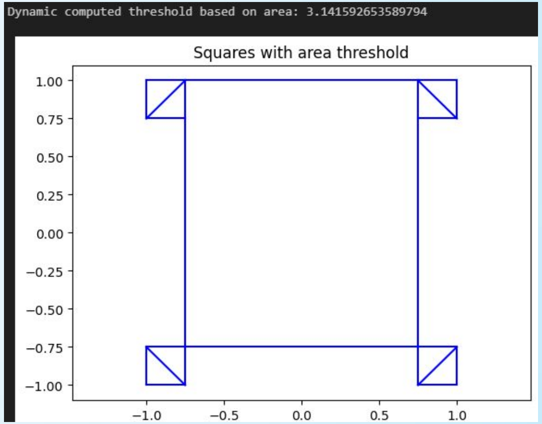
Based on the area of the **minimum bounding circle**.

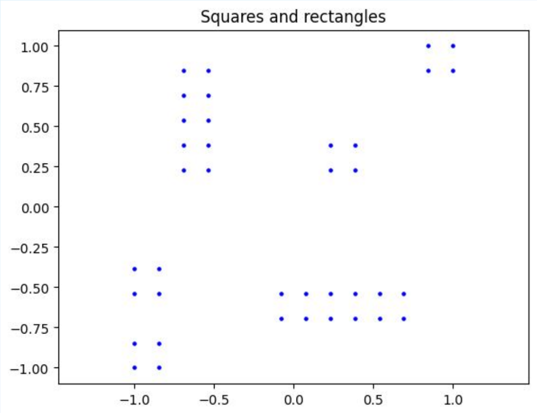


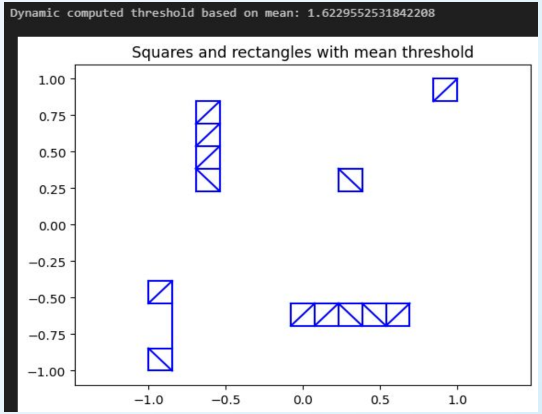
Mean vs Area threshold

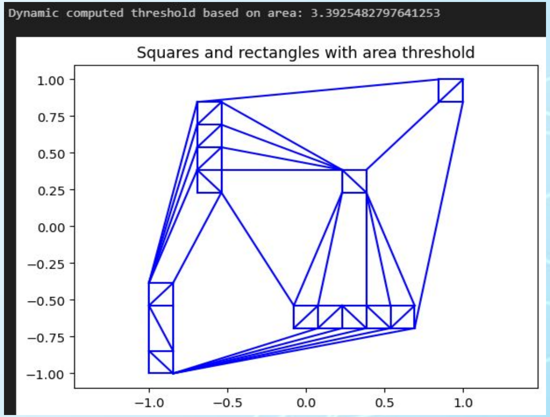




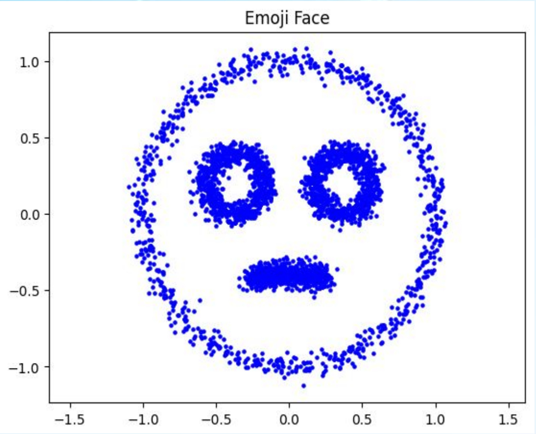


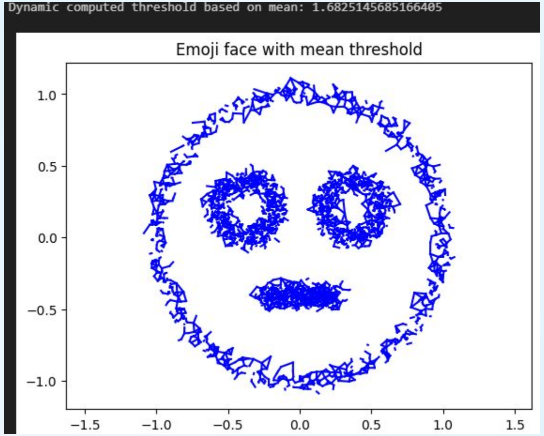


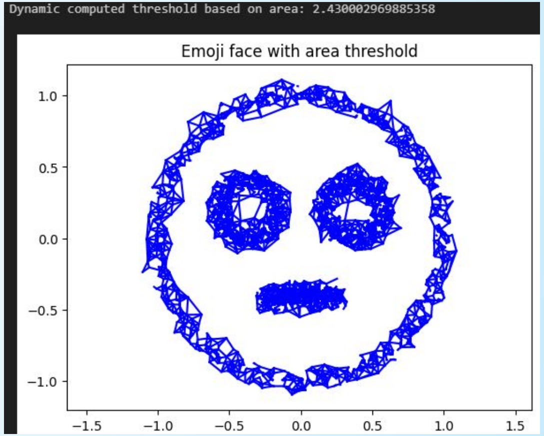


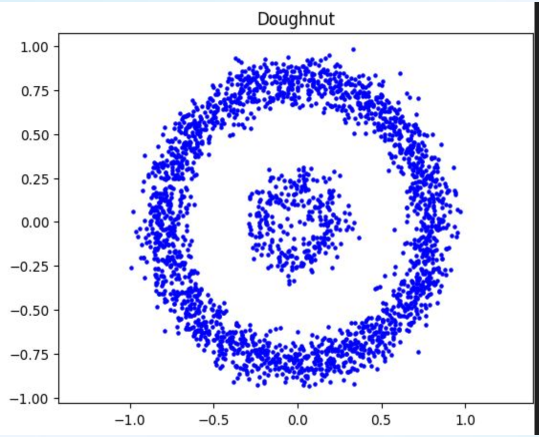


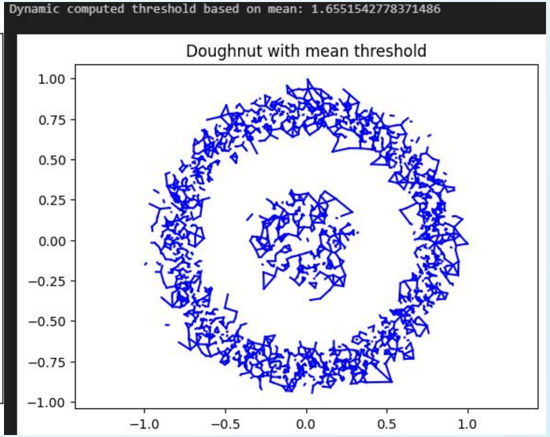
Mean vs Area threshold

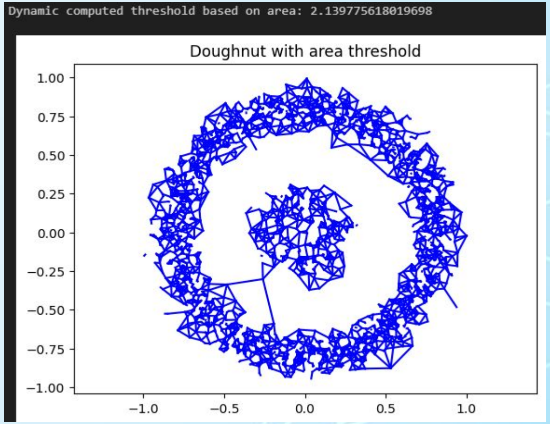




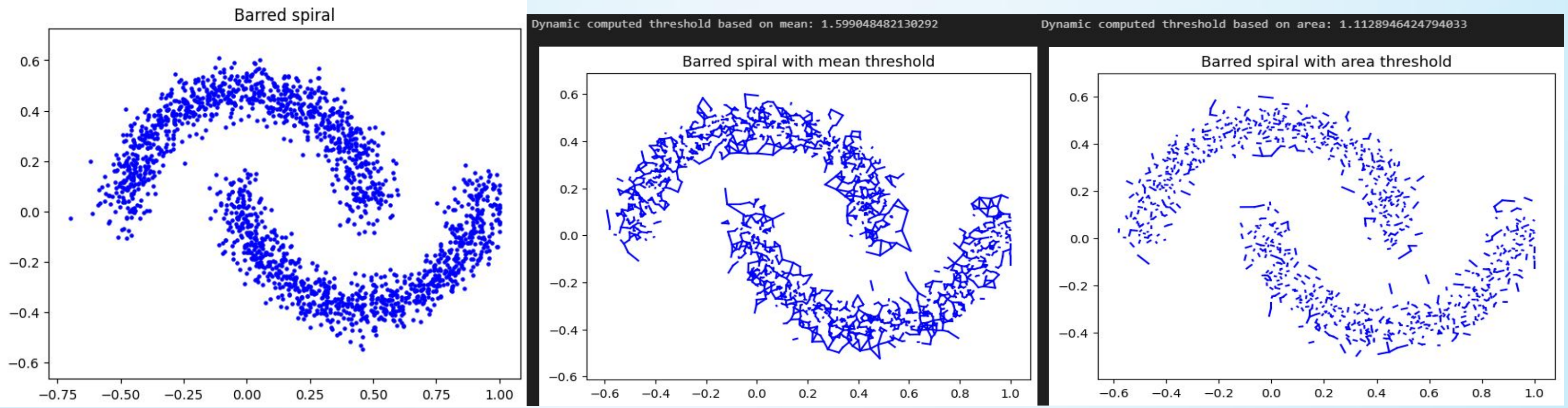


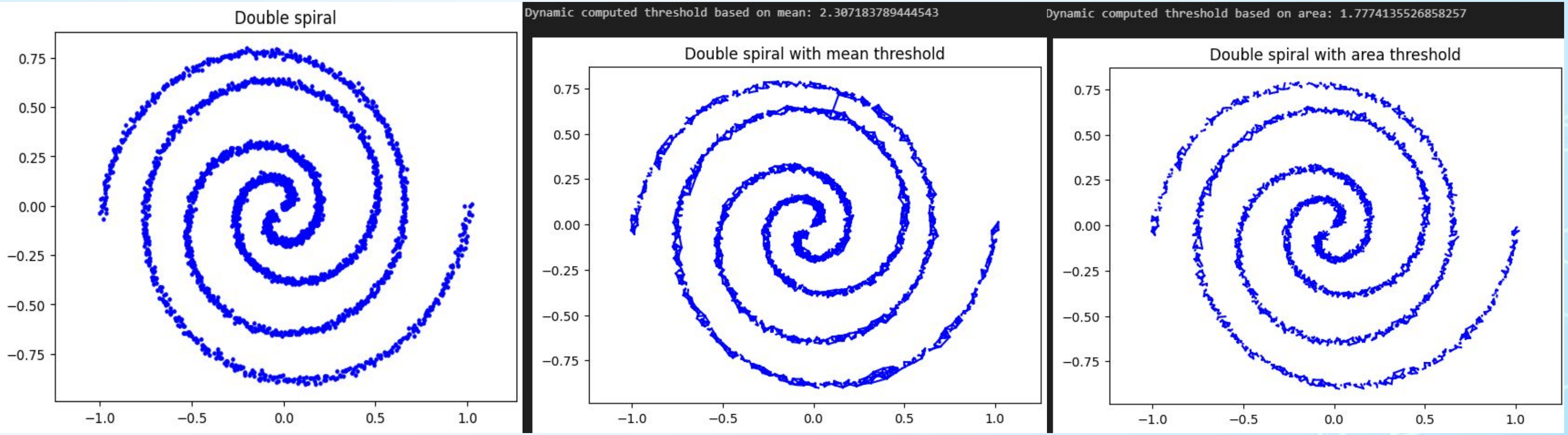






Mean vs Area threshold





External sources

- https://pressbooks.umn.edu/sensationandperception/chapter/columns-and-hypercolumns-in-v1
- https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms
- https://www.cs.rug.nl/~petkov/publications/2005LNCS3704_grouping_dots.pdf