Persistent Homology: Filtration building techniques

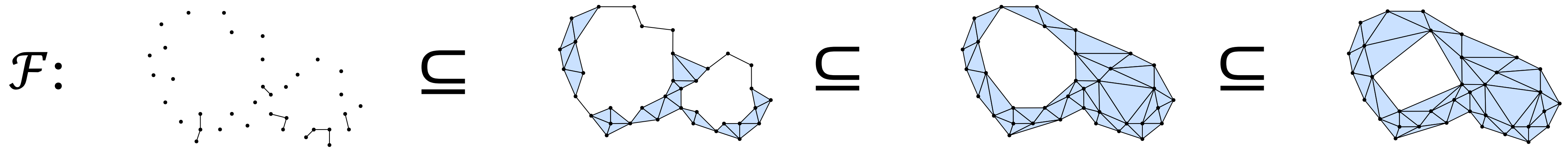
Tao Hou, University of Oregon

Outline for studying persistent homology

1. Intro to persistent homology
   - Build intuitions of persistent homology: what it does, what it produces
2. Formalizing persistent homology
   - Introduce its input (filtration) and study an algorithm for computation
3. Different ways for building filtrations
   - Vietoris-Rips filtration, sub-levelset filtration
   - Cubical complexes (for images)
4. Interpretation and stability of persistence diagram

PD for continuous filtration

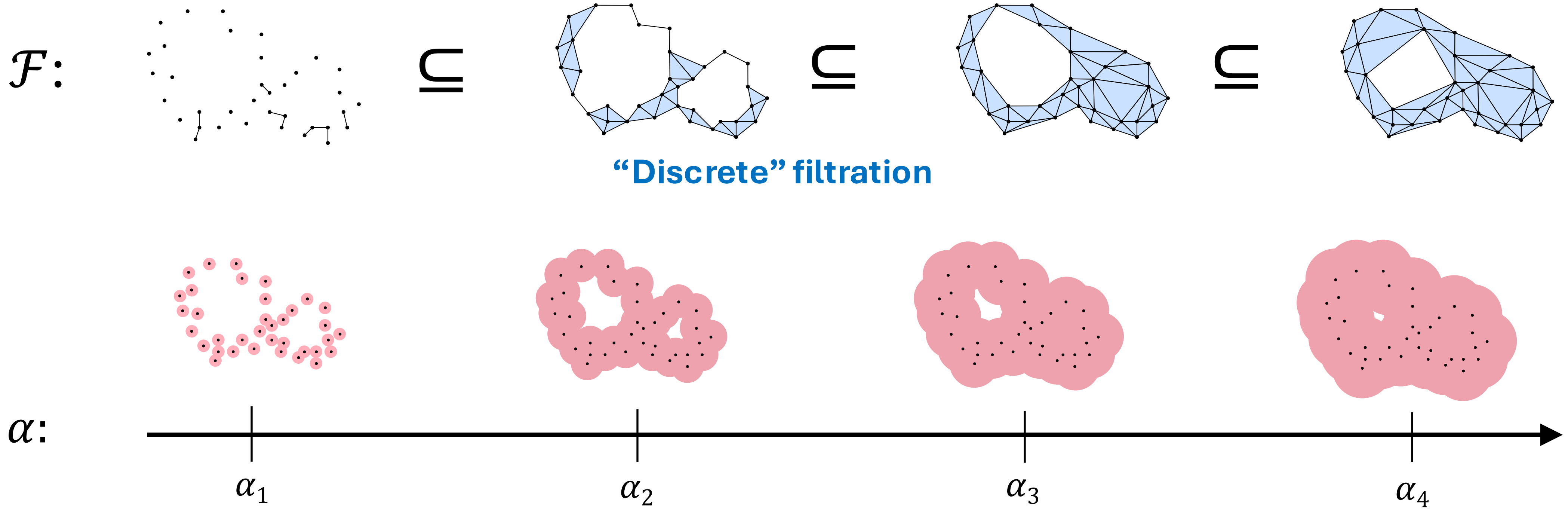
- So far, we have formally defined the PD for a (discrete) filtration, which is a finite sequence of simplicial complexes that are also nested



“Discrete” filtration

PD for continuous filtration

- So far, we have formally defined the PD for a (discrete) filtration, which is a finite sequence of simplicial complexes that are also nested
- But we haven't formally defined PD for a continuous filtration, where we have a space varying over $\alpha \in [0, \infty)$ (technically, there're infinitely many of them)



"Discrete" filtration

"Continuous" filtration

Why do we care about building filtrations?

- Simply put: the most powerful tool to infer the shape (holes) of the data by far is persistent homology, and persistent homology takes a filtration as input

Why do we care about building filtrations?

- Simply put: the most powerful tool to infer the shape (holes) of the data by far is persistent homology, and persistent homology takes a filtration as input
- Typically, your "raw data" (the data you want to process, e.g., point cloud, image, or 3D volume set) does not come directly as a filtration (a growing sequence of simplicial complexes)

Why do we care about building filtrations?

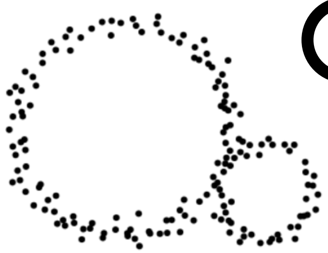
- Simply put: the most powerful tool to infer the shape (holes) of the data by far is persistent homology, and persistent homology takes a filtration as input
- Typically, your “raw data” (the data you want to process, e.g., point cloud, image, or 3D volume set) does not come directly as a filtration (a growing sequence of simplicial complexes)
- So to harness the power of persistence, you have to do this

Why do we care about building filtrations?

- Simply put: the most powerful tool to infer the shape (holes) of the data by far is persistent homology, and persistent homology takes a filtration as input
- Typically, your "raw data" (the data you want to process, e.g., point cloud, image, or 3D volume set) does not come directly as a filtration (a growing sequence of simplicial complexes)
- So to harness the power of persistence, you have to do this
- So we shall not only formally define PD on input data (which are typically continuous at least theoretically), but also learn ways to preprocess the data into filtrations to feed into the persistent homology pipeline

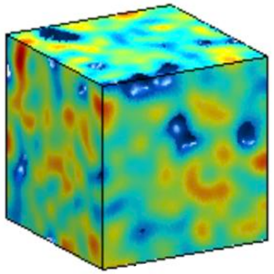
Persistent homology pipeline

1



Point cloud

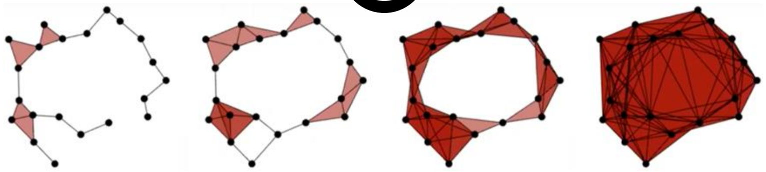
Image



3D volume data

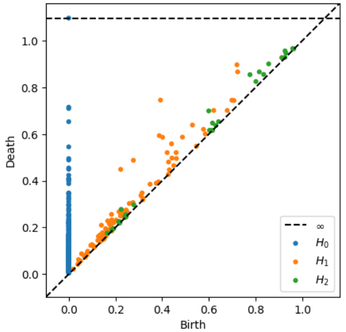
Build Filtration

2



Filtration

Persistent Homology

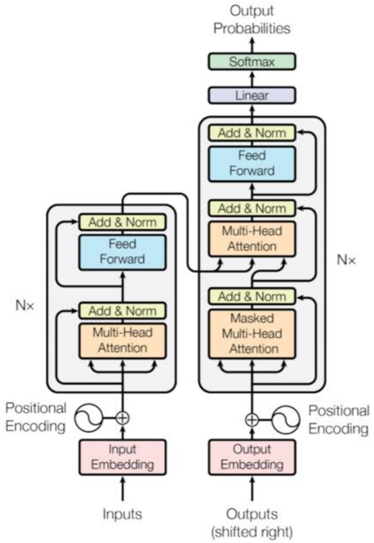


3

PD / barcode

Infer the shape / machine learning / visualization

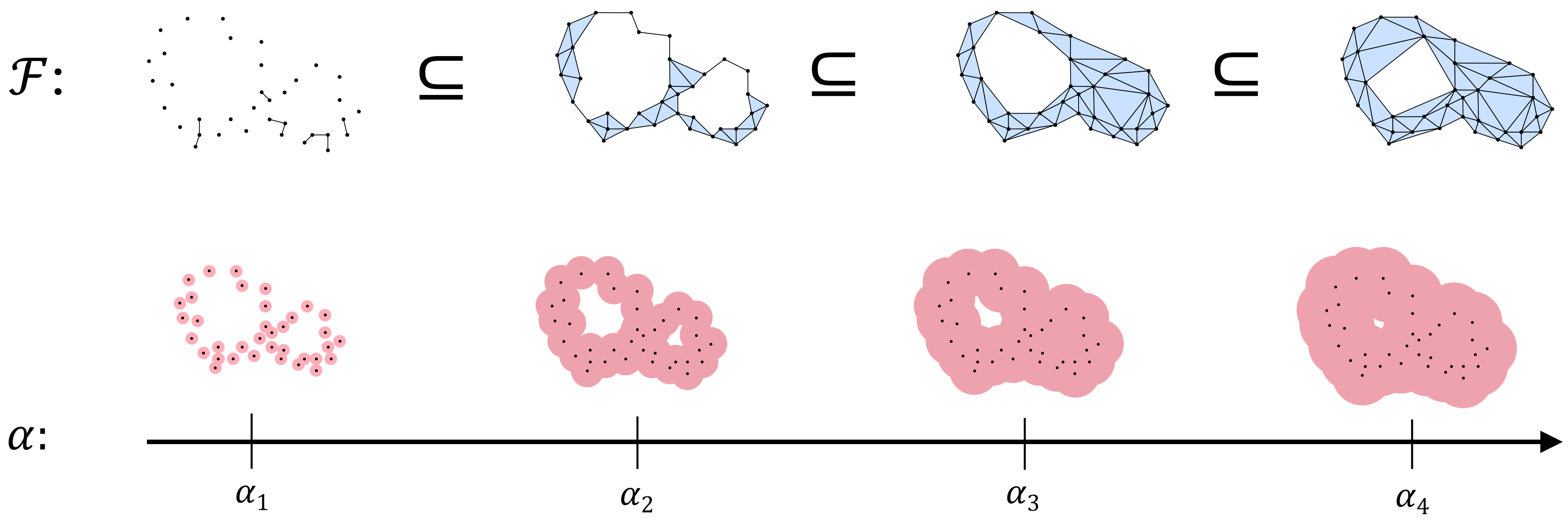
4



Some img from: AATRN; https://quantdare.com/understanding-the-shape-of-data-ii/; https://pixabay.com/photos/new-year-background-tree-sunset-736885/; Adler et al. Persistent homology for random fields and complexes.; https://builtin.com/artificial-intelligence/transformer-neural-network

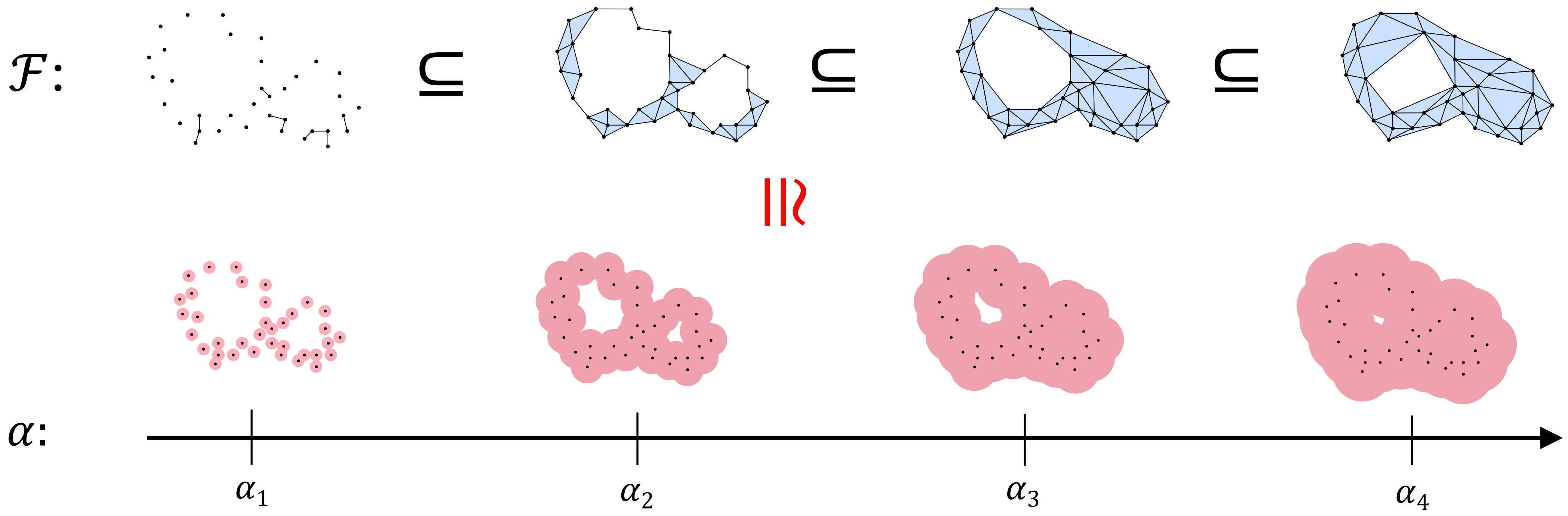
PD for continuous filtration

- To formally define the PD for a continuous filtration, the idea is to "emulate" the continuous filtration using the discrete one, with no or minimal data loss



PD for continuous filtration

- To formally define the PD for a continuous filtration, the idea is to "emulate" the continuous filtration using the discrete one, with no or minimal data loss
- We shall eventually show that continuous filtration is in some sense "equivalent" to the discrete one



Growing balls for point clouds

- For the time being, let’s focus on the “growing-balls” filtration around a set of points (point cloud), as point cloud is a very common type of data in real life.

Growing balls for point clouds

- For the time being, let’s focus on the “growing-balls” filtration around a set of points (point cloud), as point cloud is a very common type of data in real life.
- Examples:
  - Data that unsupervised learning (a type of machine learning) deals with is in some sense point clouds (e.g., finding clusters using k-means)

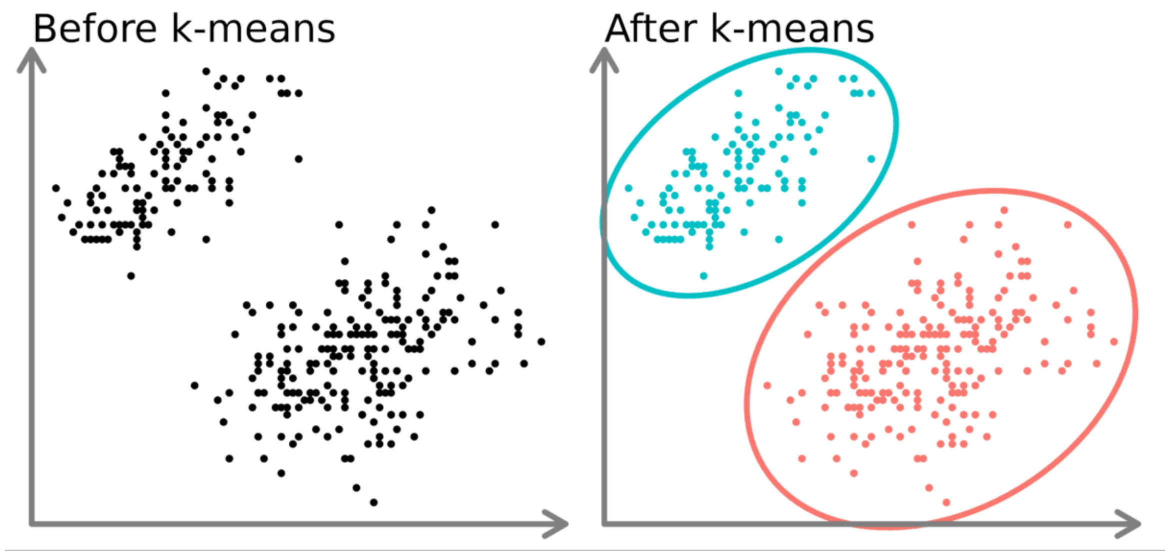


Image source: https://www.datacamp.com/tutorial/k-means-clustering-python

Growing balls for point clouds

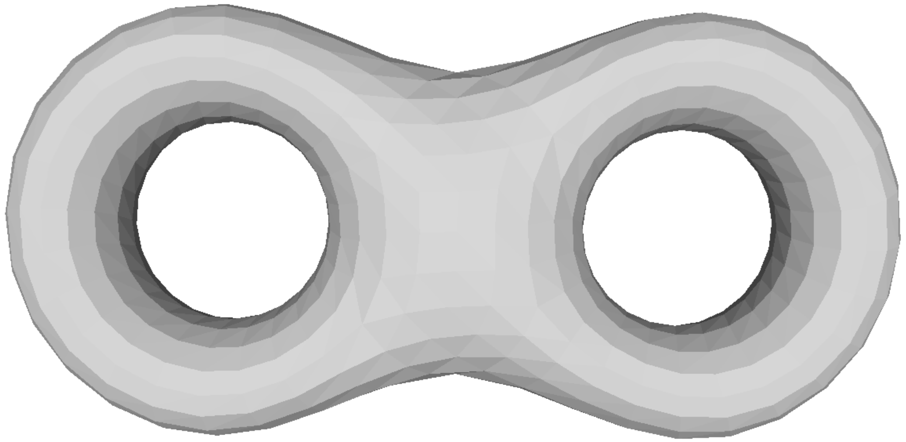
- For the time being, let's focus on the "growing-balls" filtration around a set of points (point cloud), as point cloud is a very common type of data in real life.
- Examples:
  - Data that unsupervised learning (a type of machine learning) deals with is in some sense point clouds (e.g., finding clusters using k-means)
  - Even for supervised learning (another type of more popular? machine learning), if you ignore the "labels" for the data, then the data become point clouds

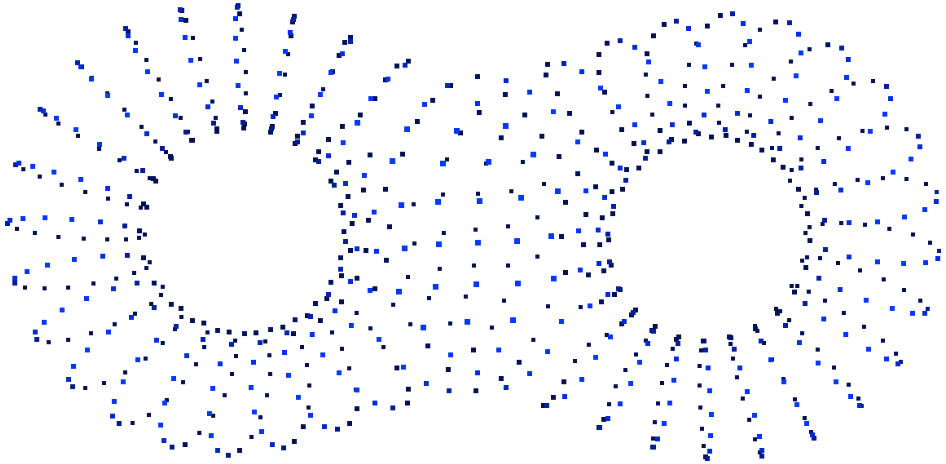
Growing balls for point clouds

- For the time being, let’s focus on the “growing-balls” filtration around a set of points (point cloud), as point cloud is a very common type of data in real life.
- Examples:
  - Data that unsupervised learning (a type of machine learning) deals with is in some sense point clouds (e.g., finding clusters using k-means)
  - Even for supervised learning (another type of more popular? machine learning), if you ignore the “labels” for the data, then the data become point clouds
  - After all, each element in your data is in some sense a “point”

Growing balls for point clouds

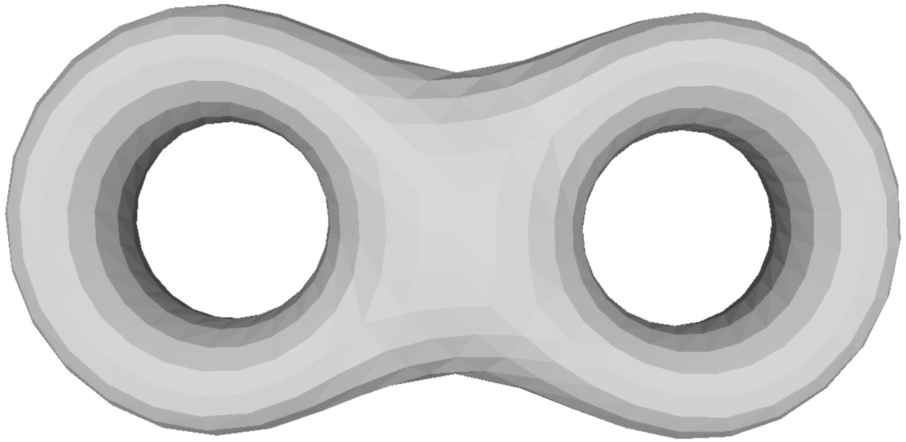
- Furthermore, for geometric models (which topological methods are good with), e.g., triangular meshes, if you ignore the triangles and edges in the data and only focus on the vertices, then this is a point cloud

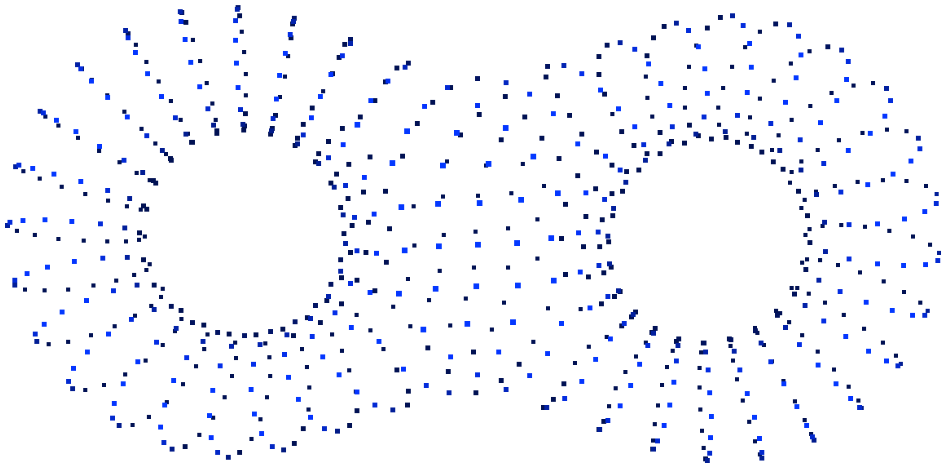




Growing balls for point clouds

- Furthermore, for geometric models (which topological methods are good with), e.g., triangular meshes, if you ignore the triangles and edges in the data and only focus on the vertices, then this is a point cloud
- Trying to infer the shape of point cloud is indeed a major motivation for topological data analysis





Build discrete filtration for “growing balls”

- Recall that each space in the “growing of balls” filtration is to take a ball of the same radius α centering in each point, and then take the union of the α-radius balls of all points
- To get the (continuous) filtration, we then let the radius α increase from 0 to ∞, and let the union of balls grow with it (see: https://gjkoplik.github.io/pers-hom-examples/0d_pers_2d_data_widget.html)

Build discrete filtration for “growing balls”

- Recall that each space in the “growing of balls” filtration is to take a ball of the same radius α centering in each point, and then take the union of the α-radius balls of all points
- To get the (continuous) filtration, we then let the radius α increase from 0 to ∞, and let the union of balls grow with it (see: https://gjkoplik.github.io/pers-hom-examples/0d_pers_2d_data_widget.html)
- Also recall our goal is to use a discrete filtration to “emulate” this continuous filtration

Build discrete filtration for “growing balls”

- Recall that each space in the “growing of balls” filtration is to take a ball of the same radius α centering in each point, and then take the union of the α-radius balls of all points
- To get the (continuous) filtration, we then let the radius α increase from 0 to ∞, and let the union of balls grow with it (see: https://gjkoplik.github.io/pers-hom-examples/0d_pers_2d_data_widget.html)
- Also recall our goal is to use a discrete filtration to “emulate” this continuous filtration
- Since a discrete filtration consists of simplicial complexes, what we really need to do is to find a way to use a simplicial complex to “emulate” the union of balls

Build discrete filtration for “growing balls”

- Recall that each space in the “growing of balls” filtration is to take a ball of the same radius α centering in each point, and then take the union of the α-radius balls of all points
- To get the (continuous) filtration, we then let the radius α increase from 0 to ∞, and let the union of balls grow with it (see: https://gjkoplik.github.io/pers-hom-examples/0d_pers_2d_data_widget.html)
- Also recall our goal is to use a discrete filtration to “emulate” this continuous filtration
- Since a discrete filtration consists of simplicial complexes, what we really need to do is to find a way to use a simplicial complex to “emulate” the union of balls
- To construct a simplicial complex for the union of balls, we first let the point in the point cloud be all the vertices in the simplicial complex

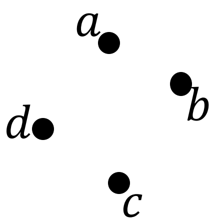
Build discrete filtration for "growing balls"

- Recall that each space in the "growing of balls" filtration is to take a ball of the same radius α centering in each point, and then take the union of the α-radius balls of all points
- To get the (continuous) filtration, we then let the radius α increase from 0 to ∞, and let the union of balls grow with it (see: https://gjkoplik.github.io/pers-hom-examples/0d_pers_2d_data_widget.html)
- Also recall our goal is to use a discrete filtration to "emulate" this continuous filtration
- Since a discrete filtration consists of simplicial complexes, what we really need to do is to find a way to use a simplicial complex to "emulate" the union of balls
- To construct a simplicial complex for the union of balls, we first let the point in the point cloud be all the vertices in the simplicial complex
- The remaining task is to build high-dimensional simplices out of the points (vertices)

Build discrete filtration for “growing balls”

Simplices:

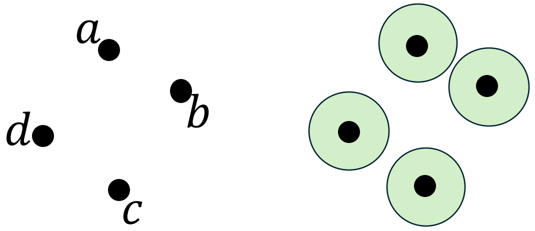
- a
- b
- c
- d



Build discrete filtration for “growing balls”

Simplices:

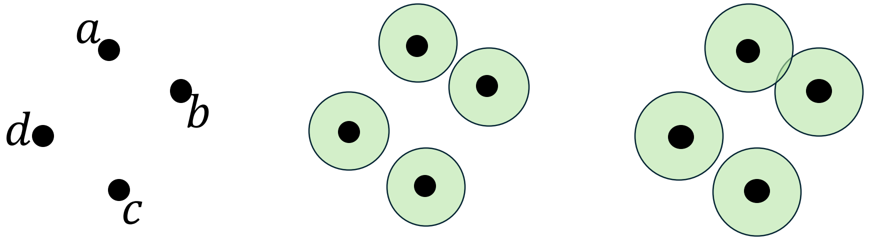
- a
- b
- c
- d



Build discrete filtration for “growing balls”

Simplices:

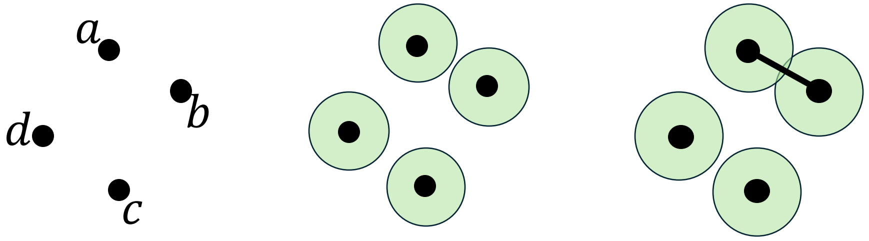
- a
- b
- c
- d



Build discrete filtration for "growing balls"

Simplices:

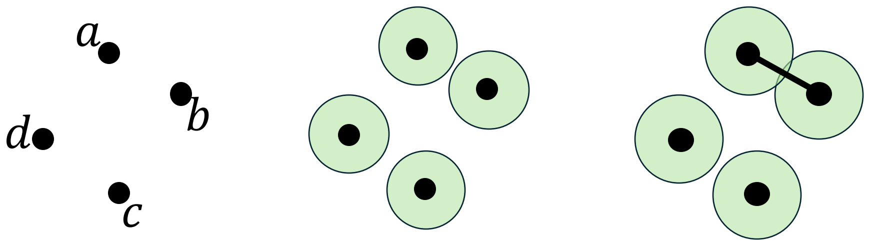
- a
- b
- c
- d



Build discrete filtration for “growing balls”

Simplices:

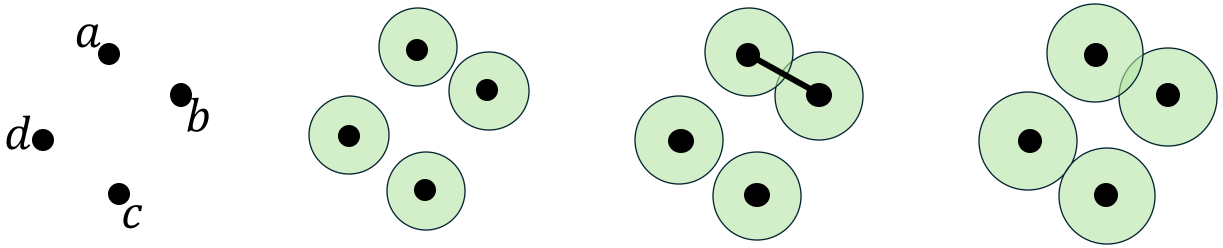
- a
- b
- c
- d
- ab



Build discrete filtration for “growing balls”

Simplices:

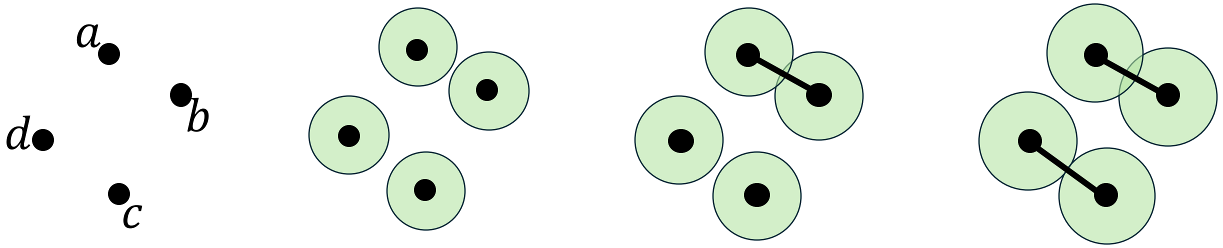
- a
- b
- c
- d
- ab



Build discrete filtration for “growing balls”

Simplices:

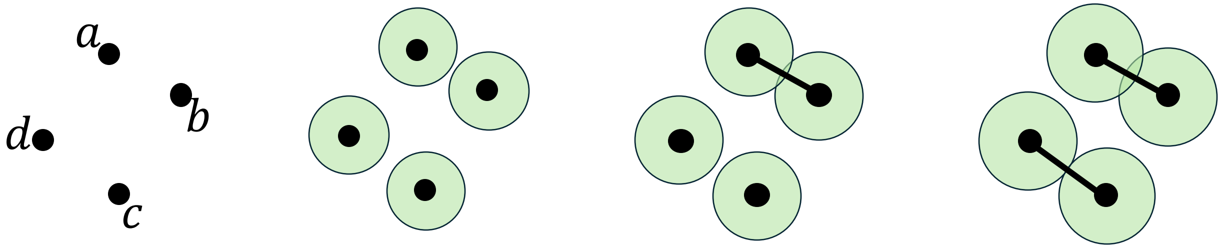
- a
- b
- c
- d
- ab



Build discrete filtration for “growing balls”

Simplices:

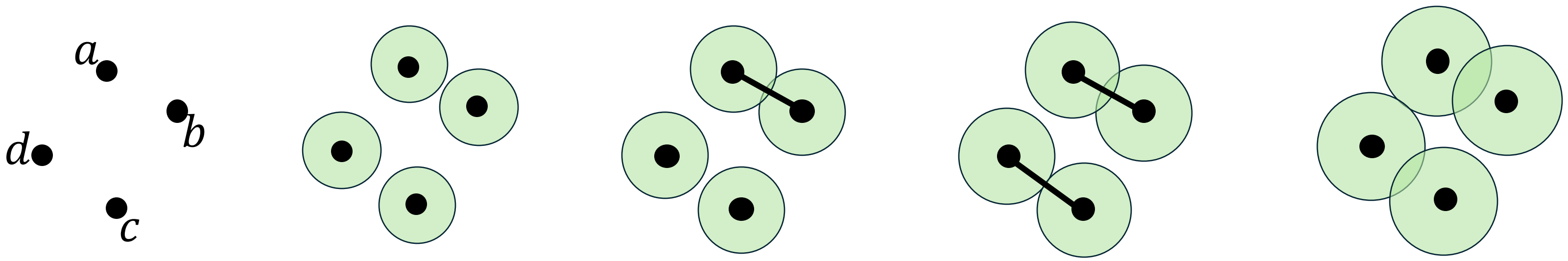
- a
- b
- c
- d
- ab
- cd



Build discrete filtration for "growing balls"

Simplices:

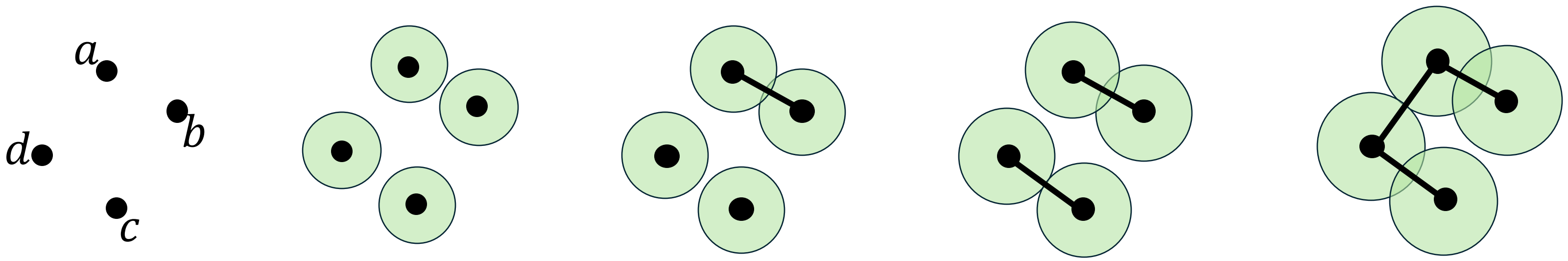
- a
- b
- c
- d
- ab
- cd



Build discrete filtration for “growing balls”

Simplices:

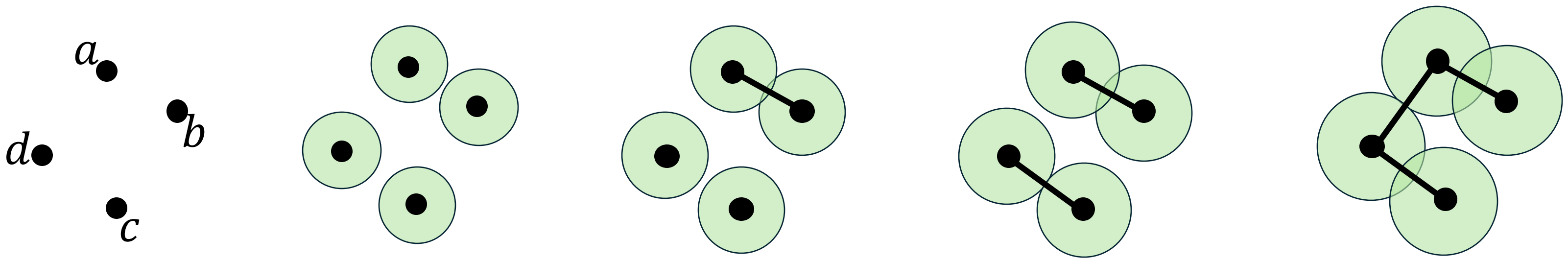
- a
- b
- c
- d
- ab
- cd



Build discrete filtration for “growing balls”

Simplices:

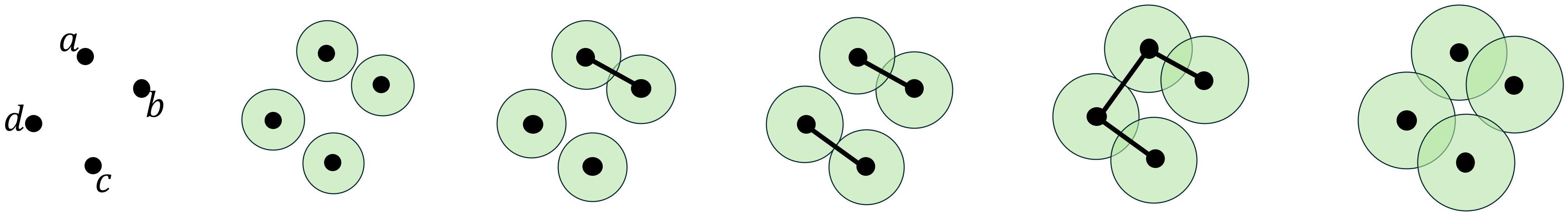
- a
- b
- c
- d
- ab
- cd
- ad



Build discrete filtration for “growing balls”

Simplices:

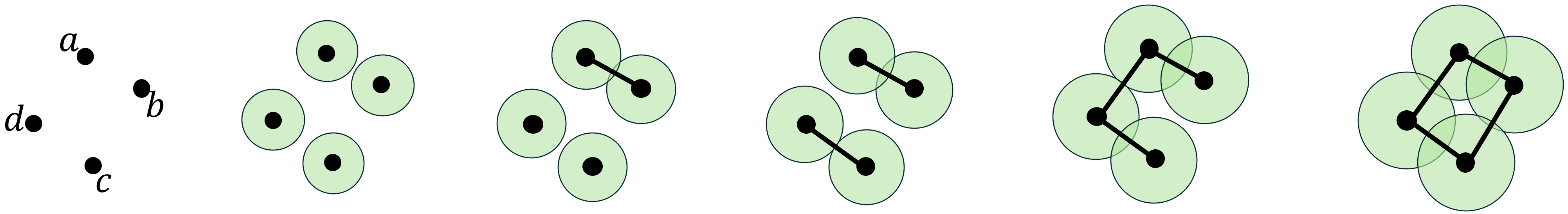
- a
- b
- c
- d
- ab
- cd
- ad



Build discrete filtration for "growing balls"

Simplices:

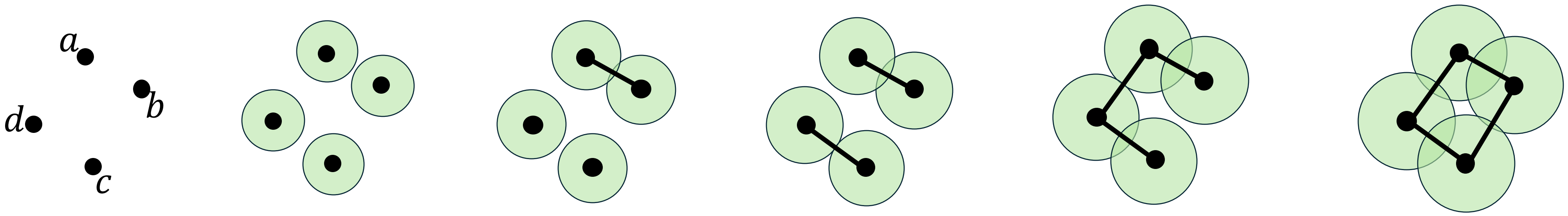
- a
- b
- c
- d
- ab
- cd
- ad



Build discrete filtration for “growing balls”

Simplices:

- a
- b
- c
- d
- ab
- cd
- ad
- bc

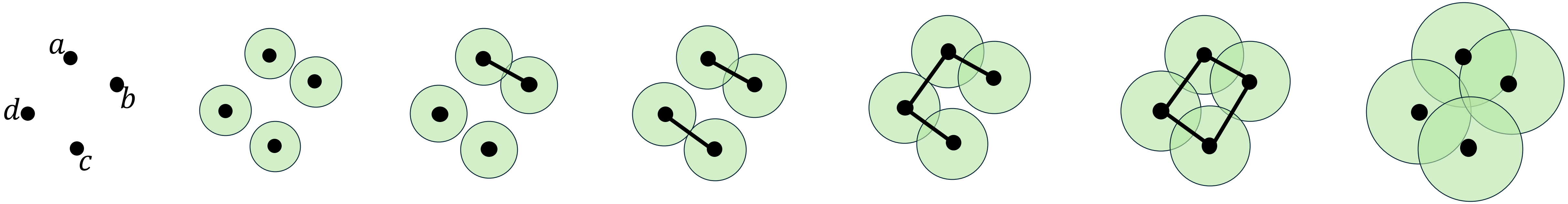


Build discrete filtration for "growing balls"

Simplices:

- a
- b
- c
- d

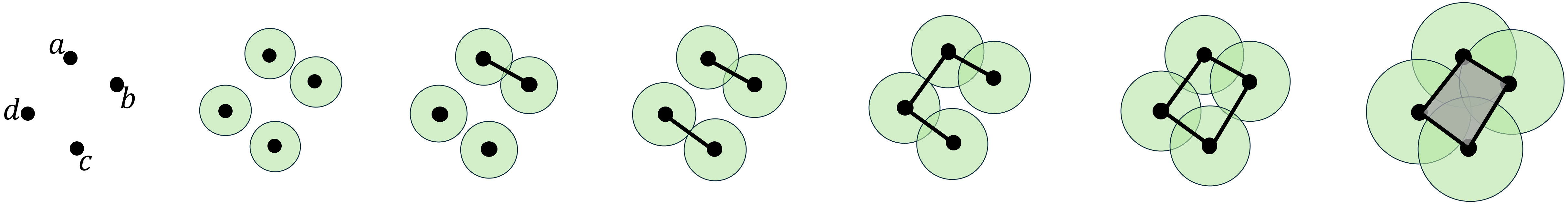
- ab
- cd
- ad
- bc



Build discrete filtration for "growing balls"

Simplices:

- a
- b
- c
- d
- ab
- cd
- ad
- bc



Čech Complex

- Inspired by the previous example, to construct the simplicial complex out of the points, we should let a set of points form a simplex whenever balls of these points "touch" each other

Čech Complex

- Inspired by the previous example, to construct the simplicial complex out of the points, we should let a set of points form a simplex whenever balls of these points "touch" each other
- **Definition**: Given a point set P and a radius α, the **Čech complex** of P corresponding to the radius α, denoted $\mathbb{C}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that:

Čech Complex

- Inspired by the previous example, to construct the simplicial complex out of the points, we should let a set of points form a simplex whenever balls of these points “touch” each other
- **Definition**: Given a point set P and a radius α, the **Čech complex** of P corresponding to the radius α, denoted $\mathbb{C}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that:
    - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if the intersection of all the balls around these points is not empty, i.e.,

$$B_{\alpha}(p_0) \cap B_{\alpha}(p_1) \cap \cdots B_{\alpha}(p_d) \neq \emptyset$$

Čech Complex

- Inspired by the previous example, to construct the simplicial complex out of the points, we should let a set of points form a simplex whenever balls of these points “touch” each other
- **Definition**: Given a point set P and a radius α, the **Čech complex** of P corresponding to the radius α, denoted $\mathbb{C}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that:
    - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if the intersection of all the balls around these points is not empty, i.e.,

$$B_{\alpha}(p_0) \cap B_{\alpha}(p_1) \cap \cdots B_{\alpha}(p_d) \neq \emptyset$$

- To understand this definition, a brute force (inefficient) algorithm for computing $\mathbb{C}^{\alpha}(P)$ out of a point cloud P: enumerate each subset $p_0, p_1, \dots, p_d$ of P, and check whether intersection of balls of the points is non-empty

Čech Complex

- Inspired by the previous example, to construct the simplicial complex out of the points, we should let a set of points form a simplex whenever balls of these points “touch” each other
- **Definition**: Given a point set P and a radius α, the **Čech complex** of P corresponding to the radius α, denoted $\mathbb{C}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that:
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if the intersection of all the balls around these points is not empty, i.e.,

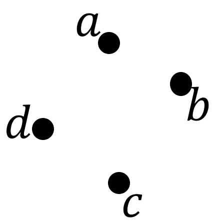
$$B_{\alpha}(p_0) \cap B_{\alpha}(p_1) \cap \cdots B_{\alpha}(p_d) \neq \emptyset$$

- To understand this definition, a brute force (inefficient) algorithm for computing $\mathbb{C}^{\alpha}(P)$ out of a point cloud P: enumerate each subset $p_0, p_1, \ldots, p_d$ of P, and check whether intersection of balls of the points is non-empty
- Of course, in practice, the algorithms that people use to compute Čech Complexes are much more efficient ones

Čech Complex

Simplices:

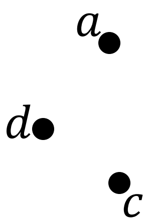
- a
- b
- c
- d

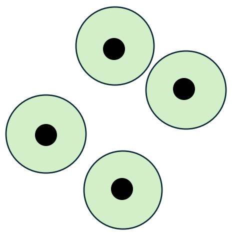


Čech Complex

Simplices:

- a
- b
- c
- d



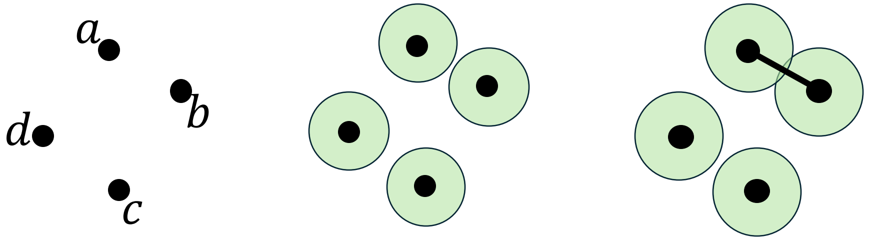


Čech Complex

Simplices:

- a
- b
- c
- d

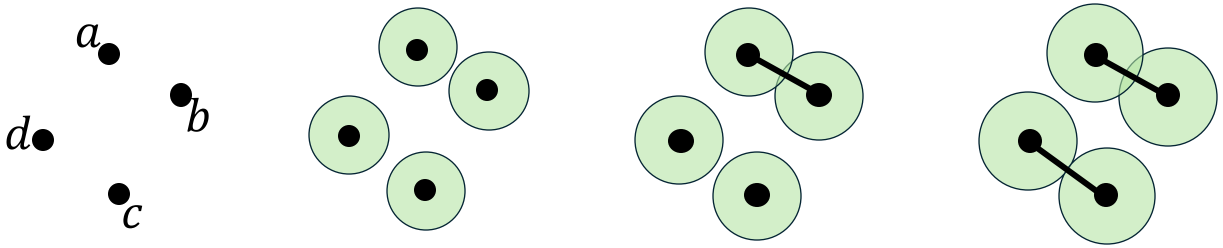
- ab



Čech Complex

Simplices:

- a
- b
- c
- d
- ab
- cd

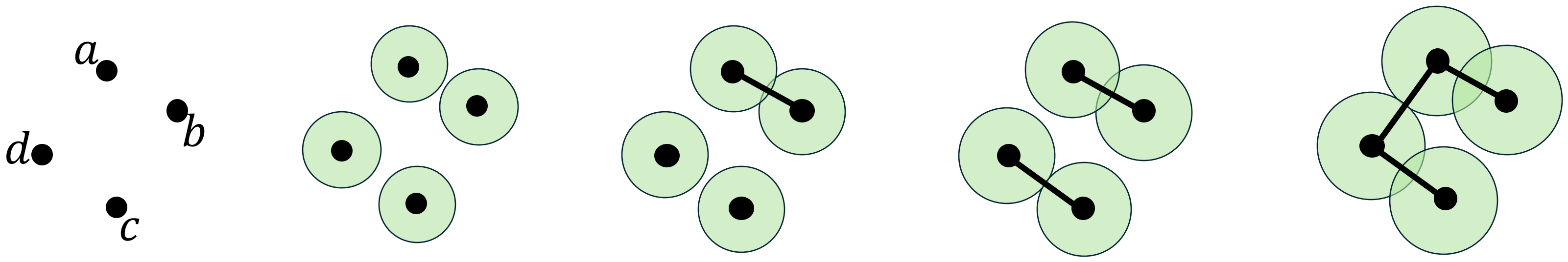


Čech Complex

Simplices:

- a
- b
- c
- d

- ab
- cd
- ad

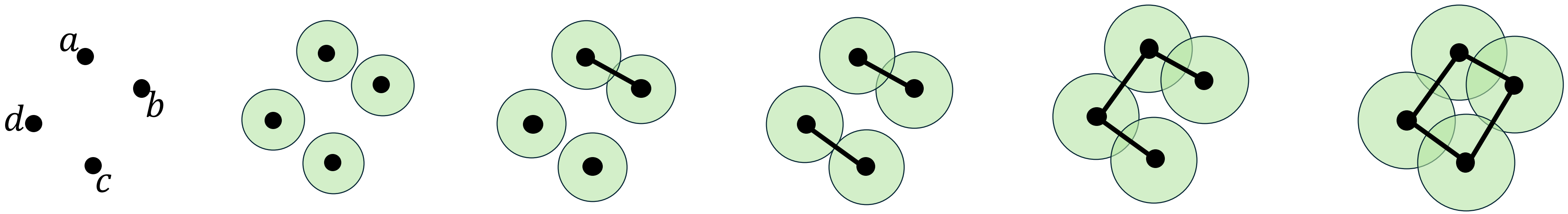


Čech Complex

Simplices:

- a
- b
- c
- d

- ab
- cd
- ad
- bc

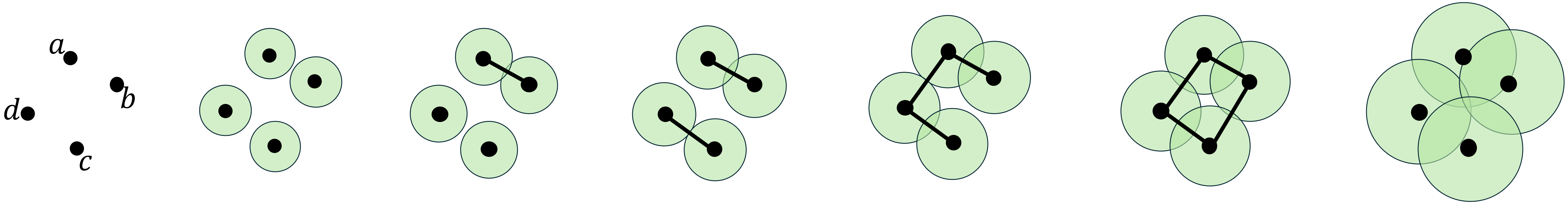


Čech Complex

Simplices:

- a
- b
- c
- d

- ab
- cd
- ad
- bc



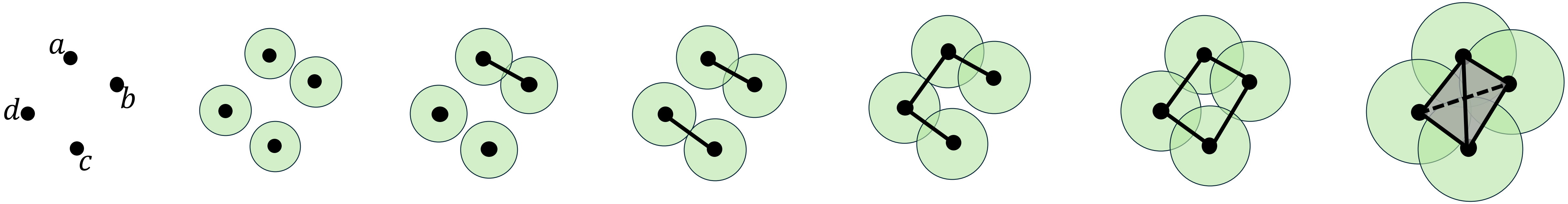
Čech Complex

Simplices:

- a
- b
- c
- d

- ab
- cd
- ad
- bc

- $abcd$ with all its faces (abc, abd, acd, and bcd).



Čech Complex

- In the previous definition, technically, we only defined a **set of simplices** $\mathbb{C}^{\alpha}(P)$.
- To make it well-defined, we need to show that $\mathbb{C}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex.

Čech Complex

- In the previous definition, technically, we only defined a **set of simplices** $\mathbb{C}^{\alpha}(P)$.
- To make it well-defined, we need to show that $\mathbb{C}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex.
- Doing this involves checking that the faces of each simplex in the set $\mathbb{C}^{\alpha}(P)$ are also in $\mathbb{C}^{\alpha}(P)$.

Čech Complex

- In the previous definition, technically, we only defined a **set of simplices** $\mathbb{C}^{\alpha}(P)$.
- To make it well-defined, we need to show that $\mathbb{C}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex.
- Doing this involves checking that the faces of each simplex in the set $\mathbb{C}^{\alpha}(P)$ are also in $\mathbb{C}^{\alpha}(P)$.
- **Verification**: Take any $\sigma \in \mathbb{C}^{\alpha}(P)$, we have a face τ of σ is nothing but a subset of σ, i.e.,$\tau \subseteq \sigma$.

Čech Complex

- In the previous definition, technically, we only defined a **set of simplices** $\mathbb{C}^{\alpha}(P)$.
- To make it well-defined, we need to show that $\mathbb{C}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex.
- Doing this involves checking that the faces of each simplex in the set $\mathbb{C}^{\alpha}(P)$ are also in $\mathbb{C}^{\alpha}(P)$.
- **Verification**: Take any $\sigma \in \mathbb{C}^{\alpha}(P)$, we have a face τ of σ is nothing but a subset of σ, i.e.,$\tau \subseteq \sigma$.
- Since all the α-balls for the points in σ have non-empty intersection, all the α-balls for the points in τ also have non-empty intersection (because τ is a subset), so $\tau \in \mathbb{C}^{\alpha}(P)$

Čech Complex

- In the previous definition, technically, we only defined a **set of simplices** $\mathbb{C}^{\alpha}(P)$.
- To make it well-defined, we need to show that $\mathbb{C}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex.
- Doing this involves checking that the faces of each simplex in the set $\mathbb{C}^{\alpha}(P)$ are also in $\mathbb{C}^{\alpha}(P)$.
- **Verification**: Take any $\sigma \in \mathbb{C}^{\alpha}(P)$, we have a face τ of σ is nothing but a subset of σ, i.e.,$\tau \subseteq \sigma$.
- Since all the α-balls for the points in σ have non-empty intersection, all the α-balls for the points in τ also have non-empty intersection (because τ is a subset), so $\tau \in \mathbb{C}^{\alpha}(P)$
- E.g., if three balls intersect, then any two balls also intersect. So the edges of the corresponding triangle are also in the Cech complex.

Čech Filtration

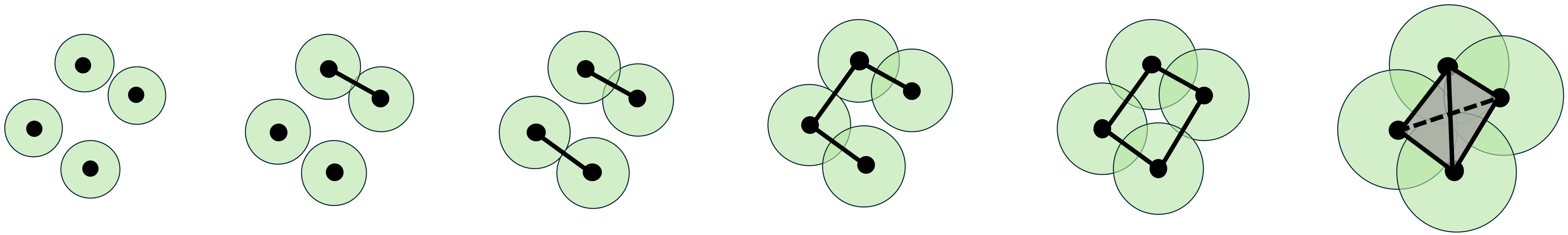
- **Definition**: All the Čech complexes over all radius α as α increases from 0 to ∞ form the **Čech filtration** of the point set P, denoted $\mathcal{C}(P)$

Čech Filtration

- **Definition**: All the Čech complexes over all radius α as α increases from 0 to ∞ form the **Čech filtration** of the point set P, denoted $\mathcal{C}(P)$
- Notice that there are only finitely many α where the Čech complexes change, so the Čech filtration is a discrete finite filtration

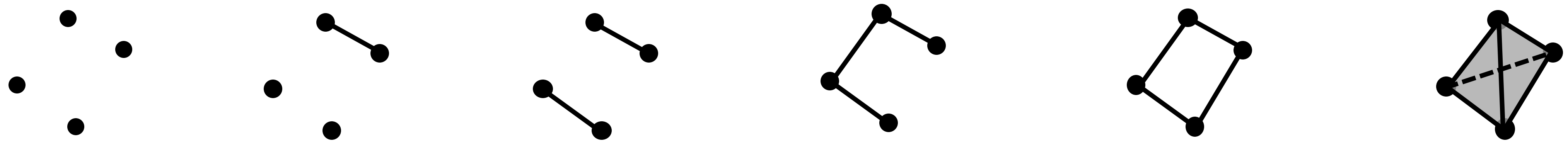
Čech Filtration

- **Definition**: All the Čech complexes over all radius α as α increases from 0 to ∞ form the **Čech filtration** of the point set P, denoted $\mathcal{C}(P)$
- Notice that there are only finitely many α where the Čech complexes change, so the Čech filtration is a discrete finite filtration



Čech Filtration

- **Definition**: All the Čech complexes over all radius α as α increases from 0 to ∞ form the **Čech filtration** of the point set P, denoted $\mathcal{C}(P)$
- Notice that there are only finitely many α where the Čech complexes change, so the Čech filtration is a discrete finite filtration

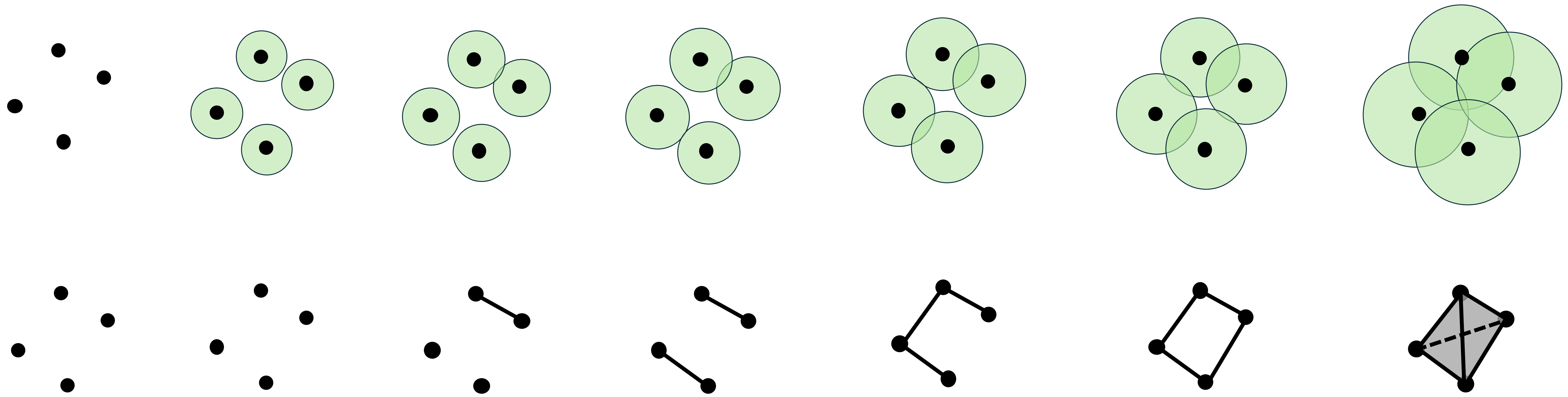


Čech Filtration

- **Nerve Theorem**: For any point set P and any radius α, the Čech complex $\mathbb{C}^{\alpha}(P)$ has the same homology as $\bigcup_{p\in P} B_{\alpha}(p)$, which is the union of α-balls of all points in P

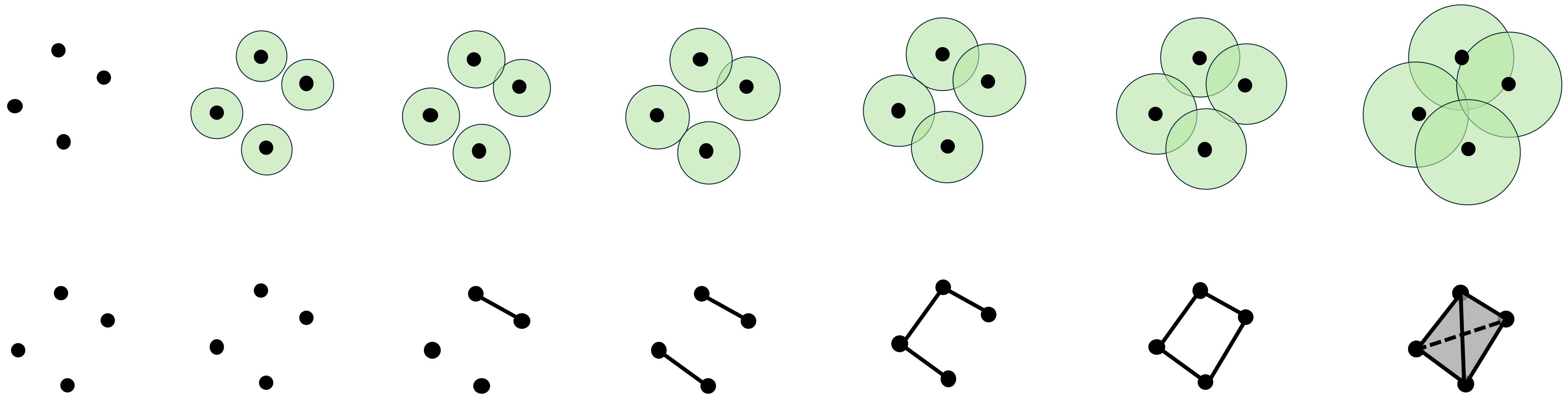
Čech Filtration

- **Nerve Theorem**: For any point set P and any radius α, the Čech complex $\mathbb{C}^{\alpha}(P)$ has the same homology as $\bigcup_{p \in P} B_{\alpha}(p)$, which is the union of α-balls of all points in P



Čech Filtration

- **Nerve Theorem**: For any point set P and any radius α, the Čech complex $\mathbb{C}^{\alpha}(P)$ has the same homology as $\bigcup_{p \in P} B_{\alpha}(p)$, which is the union of α-balls of all points in P
- The above "equivalence" is called the "homotopy equivalence" in algebraic topology, whose definition is beyond the scope



PD for Continuous Filtration

- We now define PD for a continuous filtration

PD for Continuous Filtration

- We now define PD for a continuous filtration
- Just like what we did previously by "inducing" the PD of a general (discrete) filtration from a simplex-wise filtration,
- here, we formally define the PD for a continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Čech filtration $\mathcal{C}(P)$:

PD for Continuous Filtration

- We now define PD for a continuous filtration
- Just like what we did previously by "inducing" the PD of a general (discrete) filtration from a simplex-wise filtration,
- here, we formally define the PD for a continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Čech filtration $\mathcal{C}(P)$:
    - First, we have that each complex in the discrete filtration corresponds to a range of α-values (and a range of spaces) in the continuous filtration

PD for Continuous Filtration

- We now define PD for a continuous filtration
- Just like what we did previously by "inducing" the PD of a general (discrete) filtration from a simplex-wise filtration,
- here, we formally define the PD for a continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Čech filtration $\mathcal{C}(P)$:
  - First, we have that each complex in the discrete filtration corresponds to a range of α-values (and a range of spaces) in the continuous filtration
  - For each interval $[b, d) \in PD(\mathcal{C}(P))$, consider the corresponding complexes in $\mathcal{C}(P)$, which are $K_b, K_{b+1}, \ldots, K_{d-1}$.

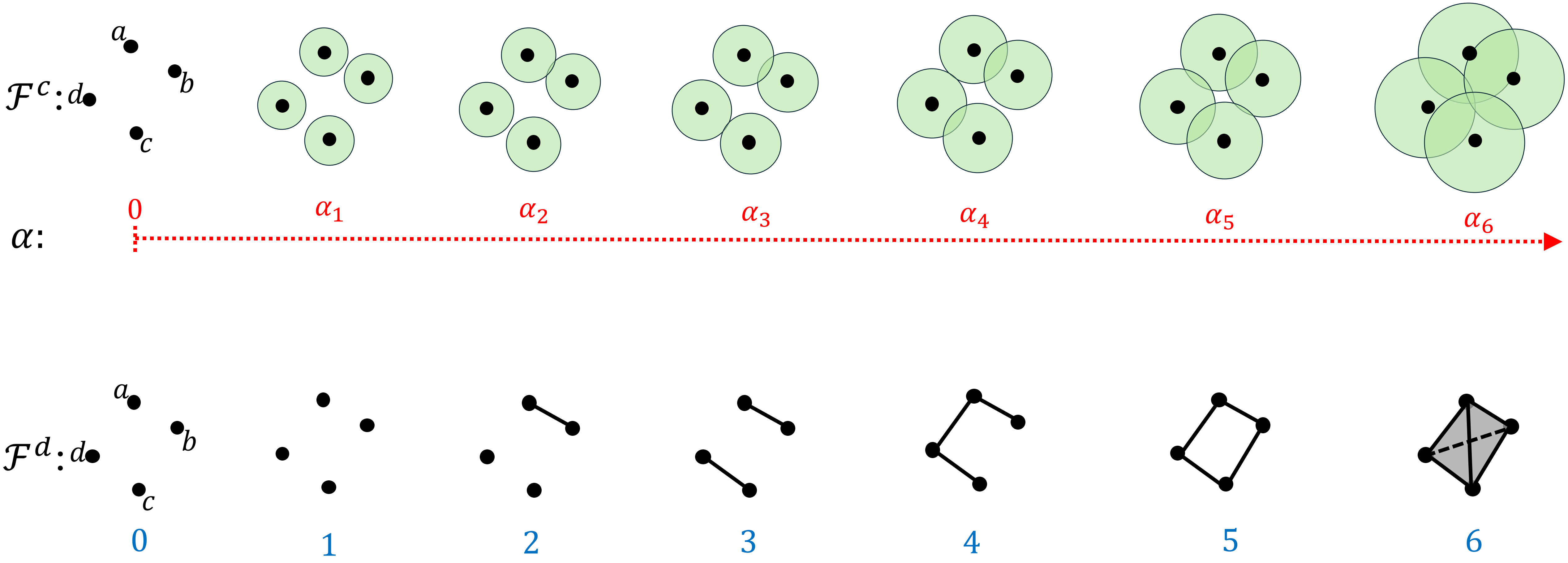
PD for Continuous Filtration

- We now define PD for a continuous filtration
- Just like what we did previously by "inducing" the PD of a general (discrete) filtration from a simplex-wise filtration,
- here, we formally define the PD for a continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Čech filtration $\mathcal{C}(P)$:
  - First, we have that each complex in the discrete filtration corresponds to a range of α-values (and a range of spaces) in the continuous filtration
  - For each interval $[b, d) \in PD(\mathcal{C}(P))$, consider the corresponding complexes in $\mathcal{C}(P)$, which are $K_b, K_{b+1}, \dots, K_{d-1}$.
  - Then, the corresponding interval of $PD(\mathcal{F}^c)$ is just the union of all the α-values corresponding to $K_b, K_{b+1}, \dots, K_{d-1}$.

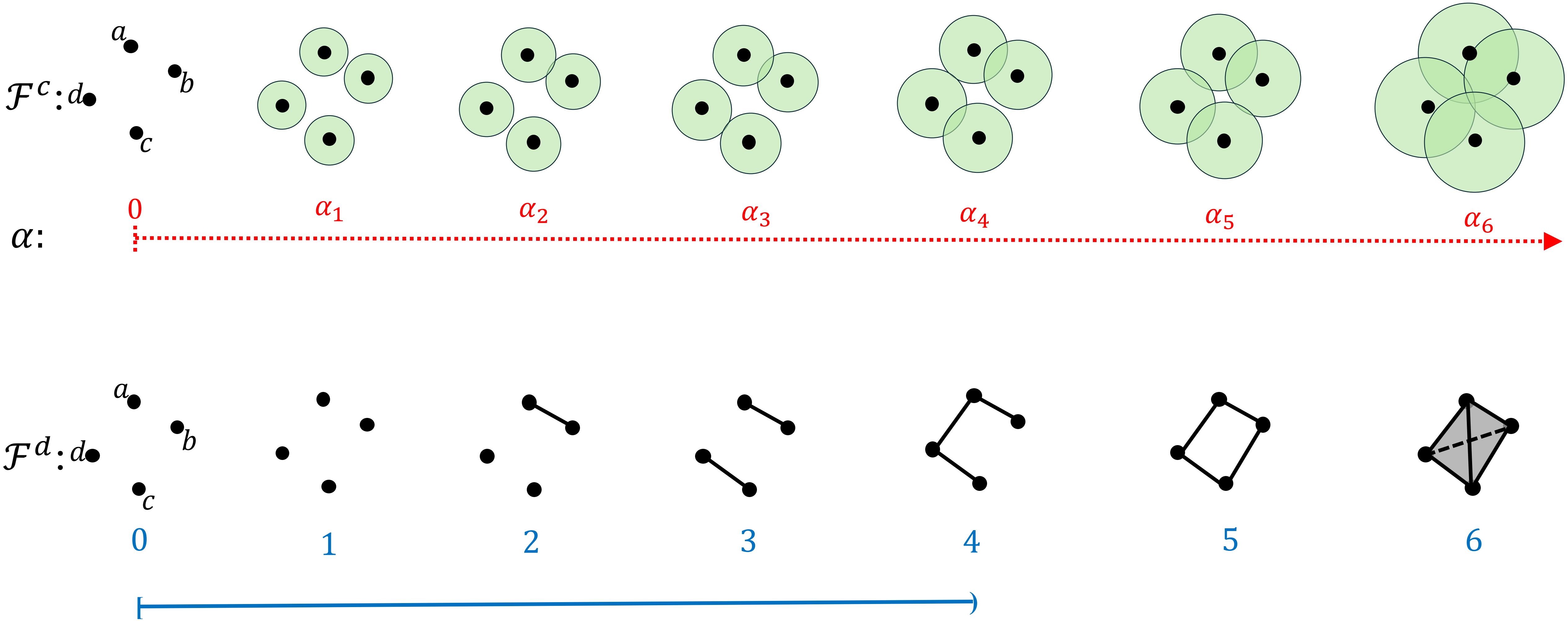
PD for Continuous Filtration

- We now define PD for a continuous filtration
- Just like what we did previously by "inducing" the PD of a general (discrete) filtration from a simplex-wise filtration,
- here, we formally define the PD for a continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Čech filtration $\mathcal{C}(P)$:
  - First, we have that each complex in the discrete filtration corresponds to a range of α-values (and a range of spaces) in the continuous filtration
  - For each interval $[b, d) \in PD(\mathcal{C}(P))$, consider the corresponding complexes in $\mathcal{C}(P)$, which are $K_b, K_{b+1}, \dots, K_{d-1}$.
  - Then, the corresponding interval of $PD(\mathcal{F}^c)$ is just the union of all the α-values corresponding to $K_b, K_{b+1}, \dots, K_{d-1}$.
  - Notice that an interval of $PD(\mathcal{F}^c)$ is actually a continuous interval over the real line $\mathbb{R}$.

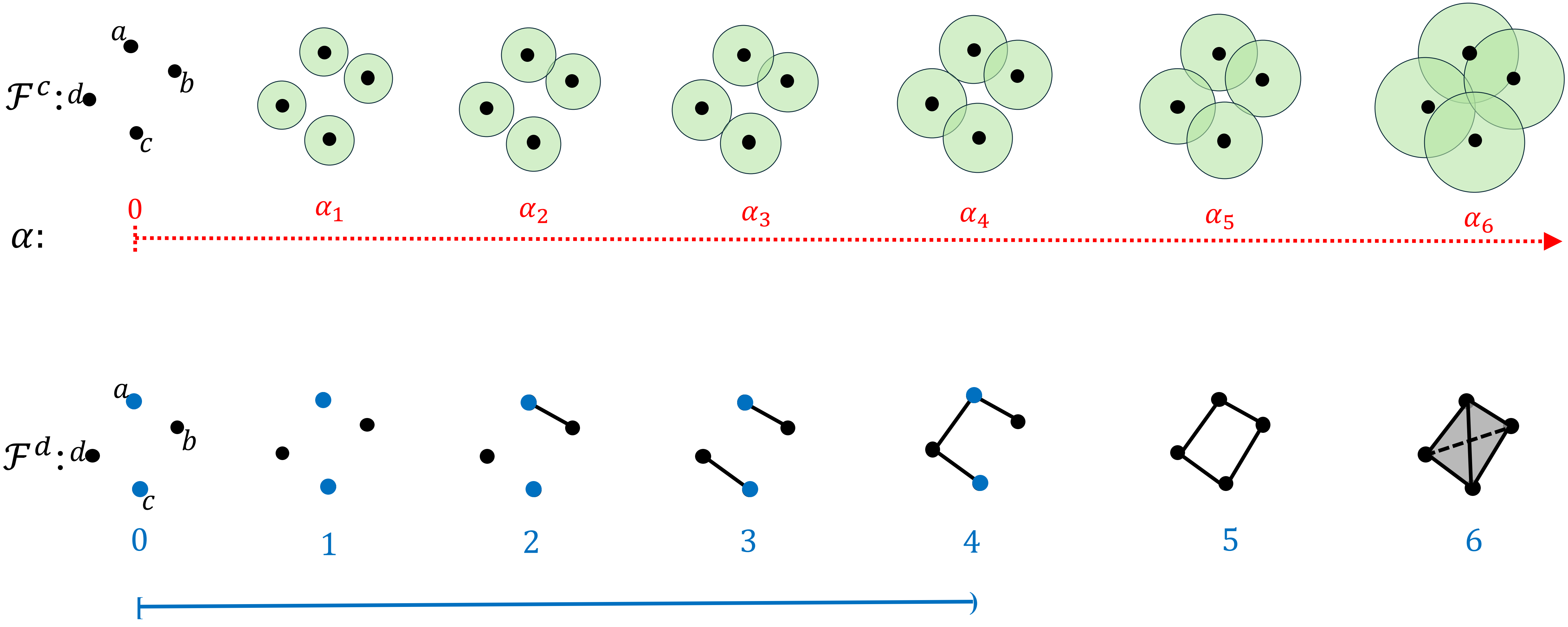
PD for Continuous Filtration



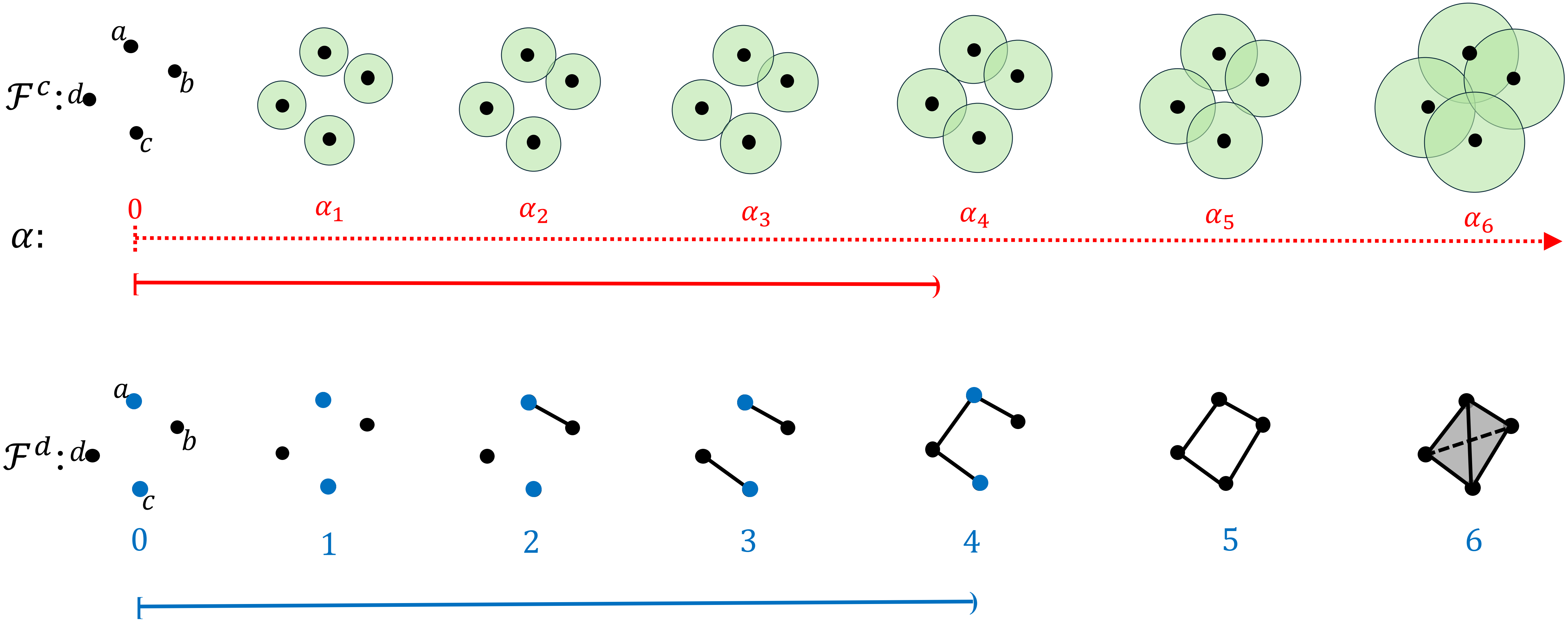
PD for Continuous Filtration



PD for Continuous Filtration

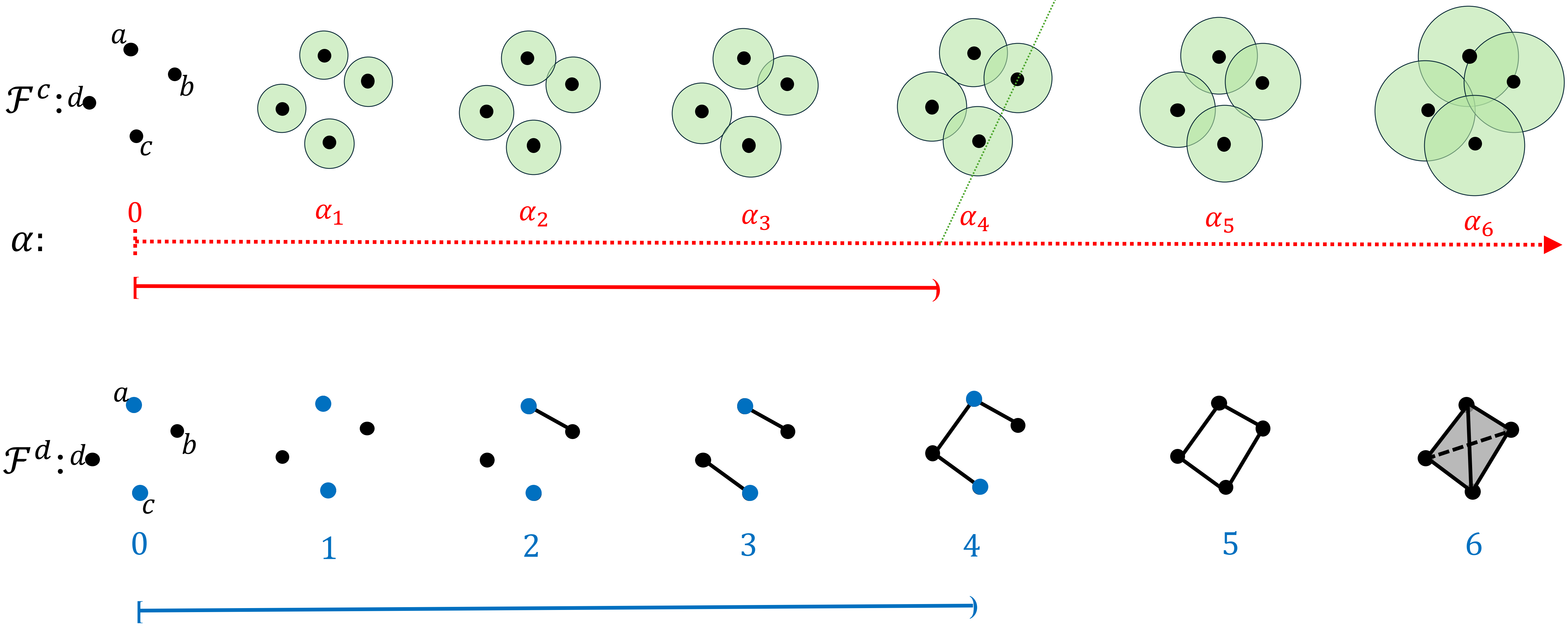


PD for Continuous Filtration

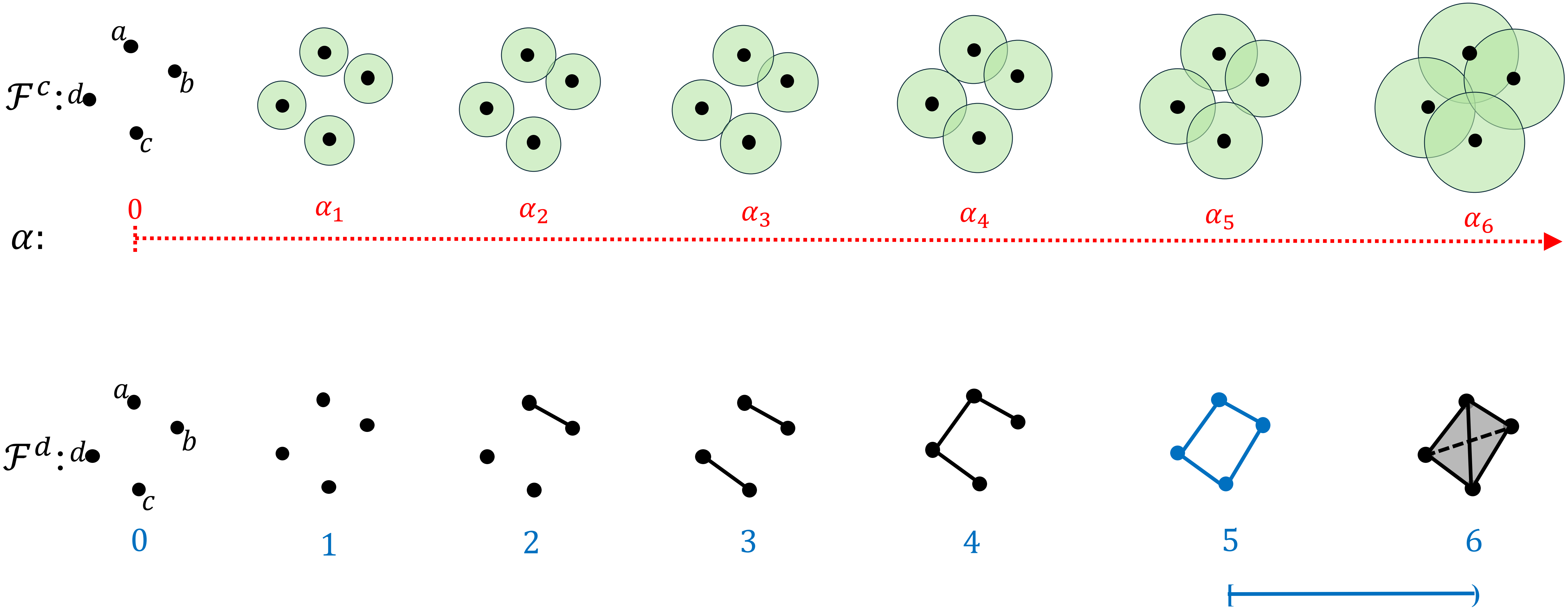


PD for Continuous Filtration

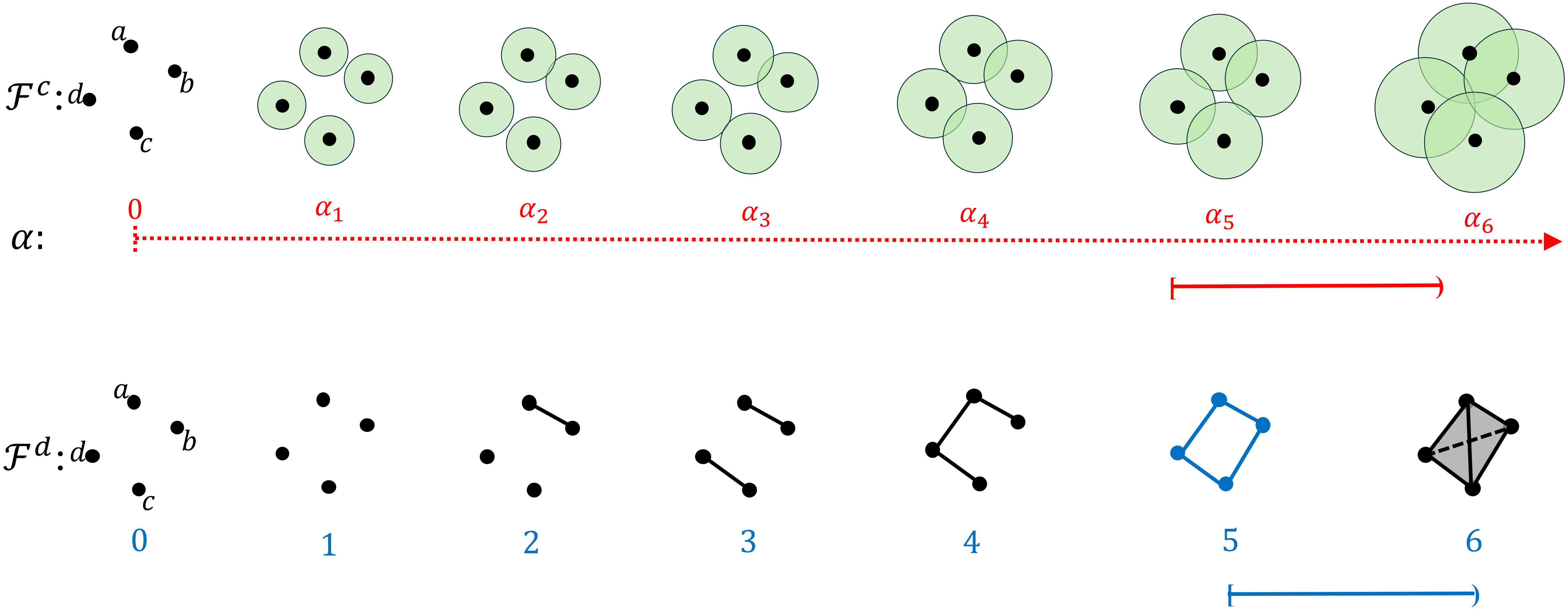
Between α_3 and α_4, balls of a and d touch so the gap between a and d filled



PD for Continuous Filtration

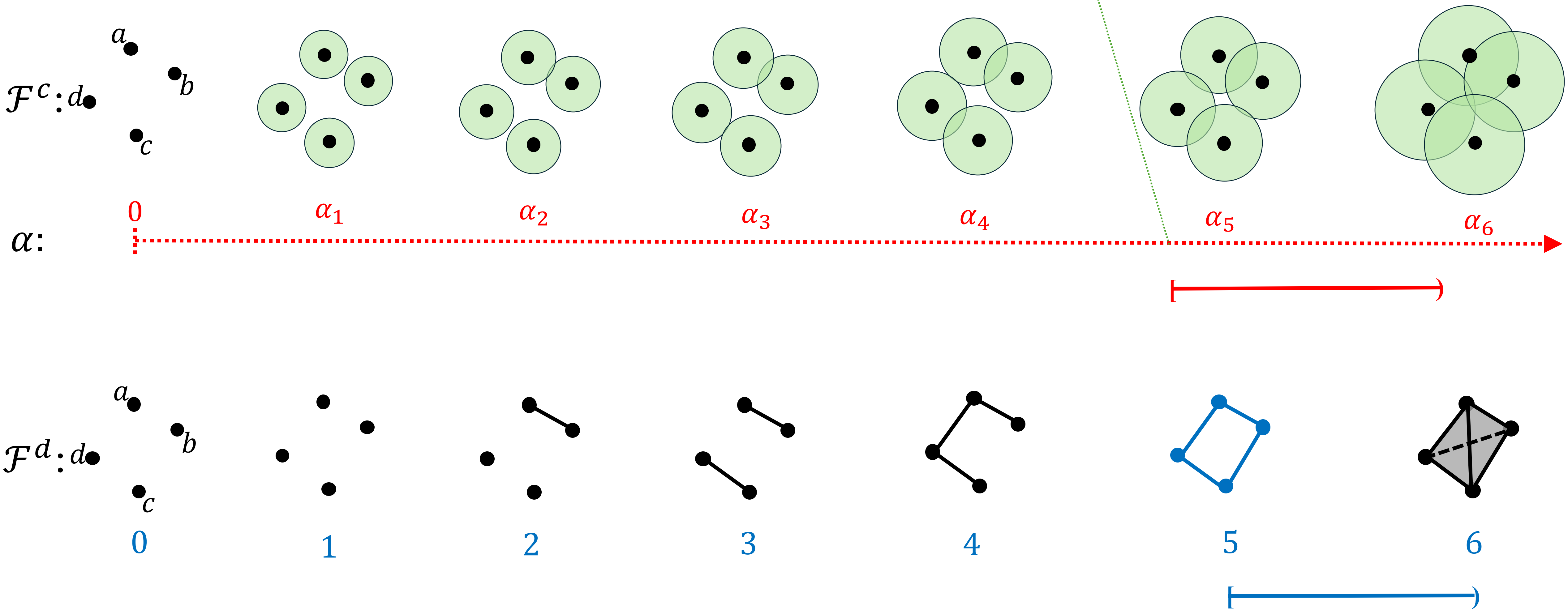


PD for Continuous Filtration

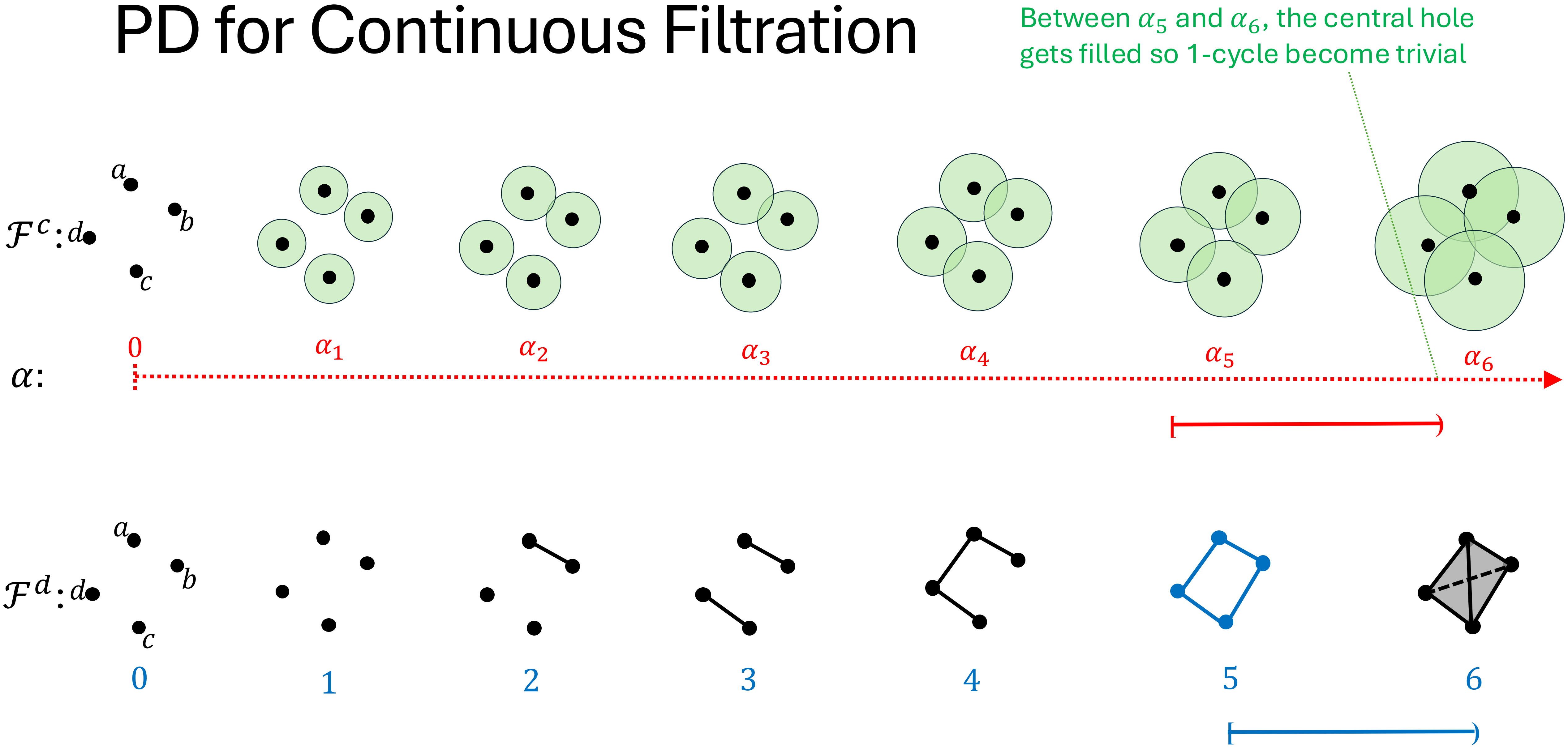


PD for Continuous Filtration

Between α_4 and α_5, balls of b and c touch so the blue 1-cycle forms



PD for Continuous Filtration



Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration

Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration
- If all you care about is the mathematics (theorical model), then things are perfect and we can stop here because the discrete Čech filtration is exactly the same as the original continuously filtration
- Aka. there is no data loss if we use Čech filtration for out point cloud data

Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration
- If all you care about is the mathematics (theorical model), then things are perfect and we can stop here because the discrete Čech filtration is exactly the same as the original continuously filtration
- Aka. there is no data loss if we use Čech filtration for out point cloud data
- But this seems to good to be true: One problem with building Čech filtration is that Čech complexes are very costly to compute

Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration
- If all you care about is the mathematics (theorical model), then things are perfect and we can stop here because the discrete Čech filtration is exactly the same as the original continuously filtration
- Aka. there is no data loss if we use Čech filtration for out point cloud data
- But this seems to good to be true: One problem with building Čech filtration is that Čech complexes are very costly to compute
- You need to check the intersection of arbitrary sets of balls and this is typically inefficient to do

Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration
- If all you care about is the mathematics (theorical model), then things are perfect and we can stop here because the discrete Čech filtration is exactly the same as the original continuously filtration
- Aka. there is no data loss if we use Čech filtration for out point cloud data
- But this seems to good to be true: One problem with building Čech filtration is that Čech complexes are very costly to compute
- You need to check the intersection of arbitrary sets of balls and this is typically inefficient to do
- We then introduce another type of discrete filtration emulating the “union of balls” called the “Vietoris-Rips” filtration
- It approximates Čech filtration (so we have slight data loss now) but it is much more efficient to compute

Is Čech Filtration Perfect?

- We constructed Čech filtration out of Čech complexes and also defined PD for the continous filtration of union of balls using the discrete Čech filtration
- If all you care about is the mathematics (theorical model), then things are perfect and we can stop here because the discrete Čech filtration is exactly the same as the original continuously filtration
- Aka. there is no data loss if we use Čech filtration for out point cloud data
- But this seems to good to be true: One problem with building Čech filtration is that Čech complexes are very costly to compute
- You need to check the intersection of arbitrary sets of balls and this is typically inefficient to do
- We then introduce another type of discrete filtration emulating the “union of balls” called the “Vietoris-Rips” filtration
- It approximates Čech filtration (so we have slight data loss now) but it is much more efficient to compute
- After all, trade-offs were made everywhere in computer science between efficiency and quality

Vietoris-Rips Filtration

- **Definition**: Given a point set P and a radius α, the **Vietoris-Rips complex** (or **Rips complex** for short) of P corresponding to the radius α, denoted $\mathbb{VR}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that

Vietoris-Rips Filtration

- **Definition**: Given a point set P and a radius α, the **Vietoris-Rips complex** (or **Rips complex** for short) of P corresponding to the radius α, denoted $\mathbb{VR}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if for each pair of p_i, p_j points in the subset, their α-balls touch each other, i.e.,

$$B_\alpha(p_i) \cap B_\alpha(p_j) \neq \emptyset$$

Vietoris-Rips Filtration

- **Definition**: Given a point set P and a radius α, the **Vietoris-Rips complex** (or **Rips complex** for short) of P corresponding to the radius α, denoted $\mathbb{VR}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if for each pair of p_i, p_j points in the subset, their α-balls touch each other, i.e.,

$$B_{\alpha}(p_i) \cap B_{\alpha}(p_j) \neq \emptyset$$

- A brute force (inefficient) algorithm for computing $\mathbb{VR}^{\alpha}(P)$ out of a point cloud P: enumerate each subset $p_0, p_1, \dots, p_d$ of P, and then enumerate each pair in the subset to check their α-balls intersect

Vietoris-Rips Filtration

- **Definition**: Given a point set P and a radius α, the **Vietoris-Rips complex** (or **Rips complex** for short) of P corresponding to the radius α, denoted $\mathbb{VR}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if for each pair of p_i, p_j points in the subset, their α-balls touch each other, i.e.,

$$B_{\alpha}(p_i) \cap B_{\alpha}(p_j) \neq \emptyset$$

- A brute force (inefficient) algorithm for computing $\mathbb{VR}^{\alpha}(P)$ out of a point cloud P: enumerate each subset $p_0, p_1, \ldots, p_d$ of P, and then enumerate each pair in the subset to check their α-balls intersect
- **Definition**: All the Vietoris-Rips complexes over all radius α as α increases from 0 to ∞ form the **Vietoris-Rips filtration** (or **Rips filtration** for short) of the point set P. We denote the filtration as $\mathcal{VR}$(P).

Vietoris-Rips Filtration

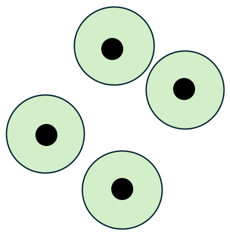
- **Definition**: Given a point set P and a radius α, the **Vietoris-Rips complex** (or **Rips complex** for short) of P corresponding to the radius α, denoted $\mathbb{VR}^{\alpha}(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if for each pair of p_i, p_j points in the subset, their α-balls touch each other, i.e.,

$$B_{\alpha}(p_i) \cap B_{\alpha}(p_j) \neq \emptyset$$

- A brute force (inefficient) algorithm for computing $\mathbb{VR}^{\alpha}(P)$ out of a point cloud P: enumerate each subset $p_0, p_1, \ldots, p_d$ of P, and then enumerate each pair in the subset to check their α-balls intersect
- **Definition**: All the Vietoris-Rips complexes over all radius α as α increases from 0 to ∞ form the **Vietoris-Rips filtration** (or **Rips filtration** for short) of the point set P. We denote the filtration as $\mathcal{VR}$(P).
- There are again finitely many α where the Rips complexes change, so the Rips filtration is a discrete finite filtration

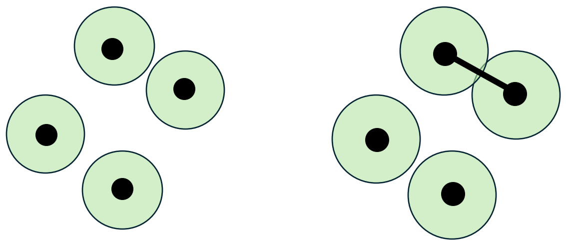
Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



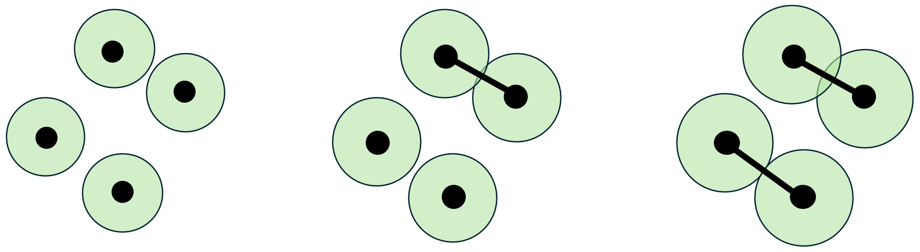
Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



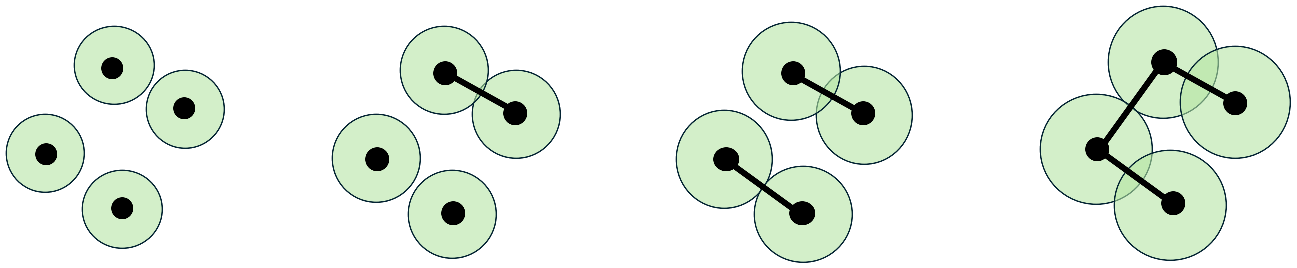
Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



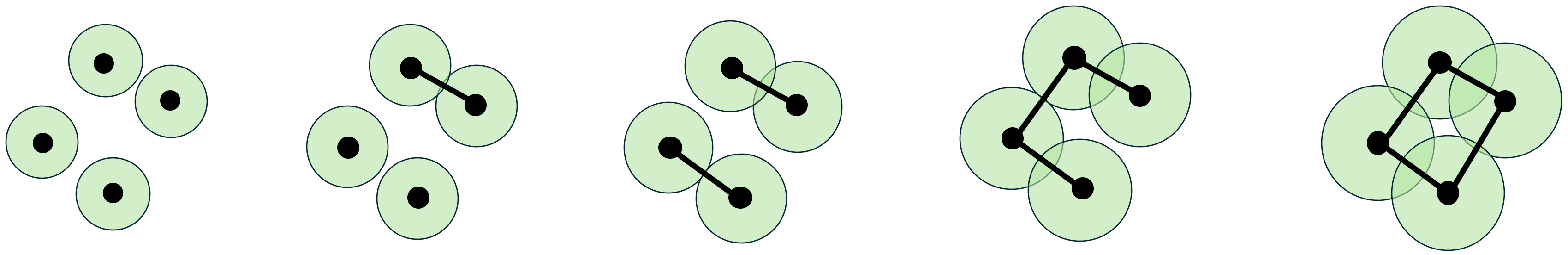
Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



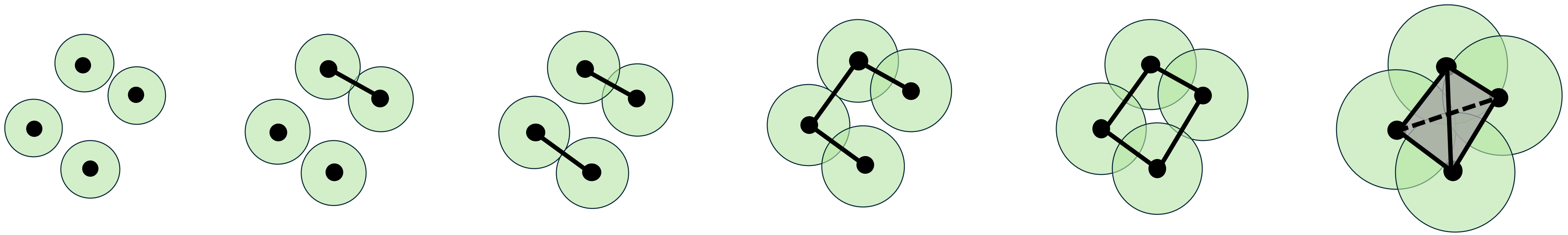
Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



Vietoris-Rips Filtration

- For the previous point cloud, the Rips filtration is the same as the Čech Filtration



Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different

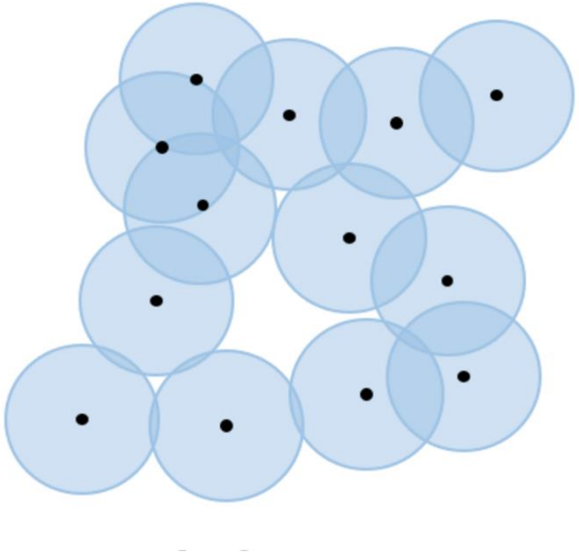


Image source: Reani, Bobrowski. Cycle Registration in Persistent Homology with Applications in Topological Bootstrap

Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different

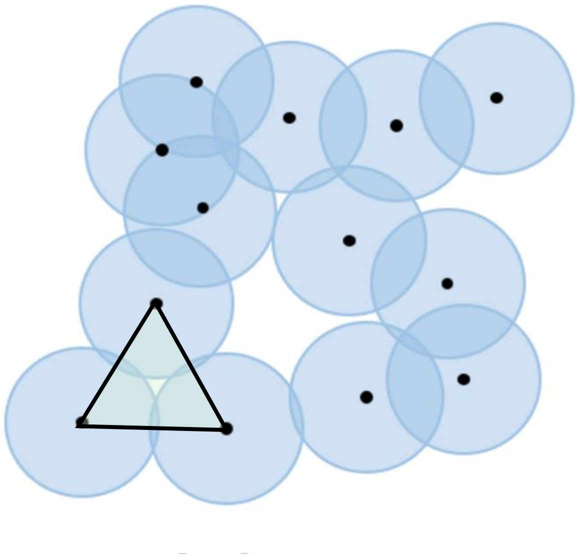


Image source: Reani, Bobrowski. Cycle Registration in
Persistent Homology with Applications in Topological Bootstrap

Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different

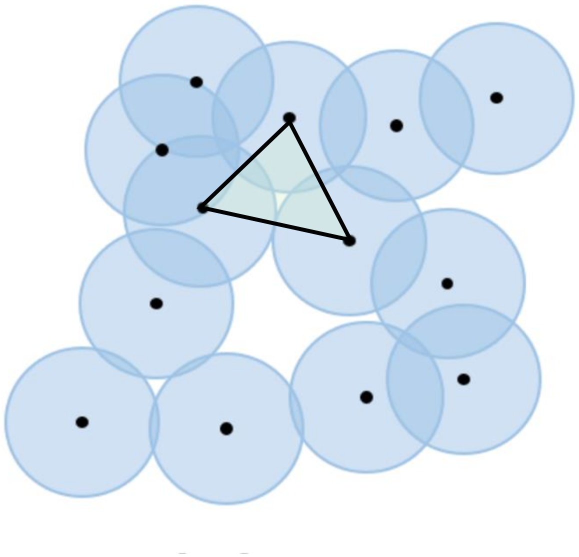
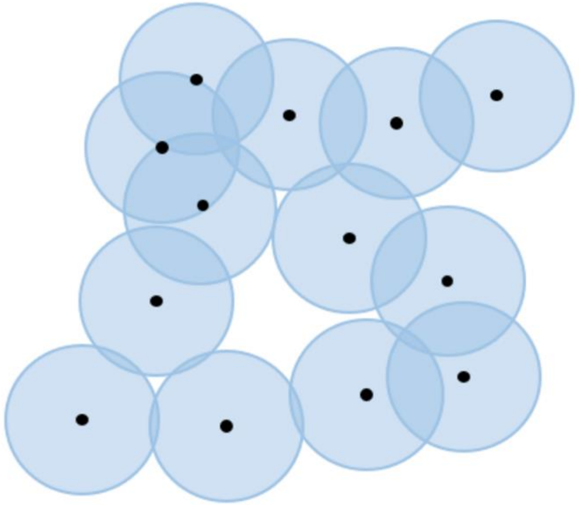
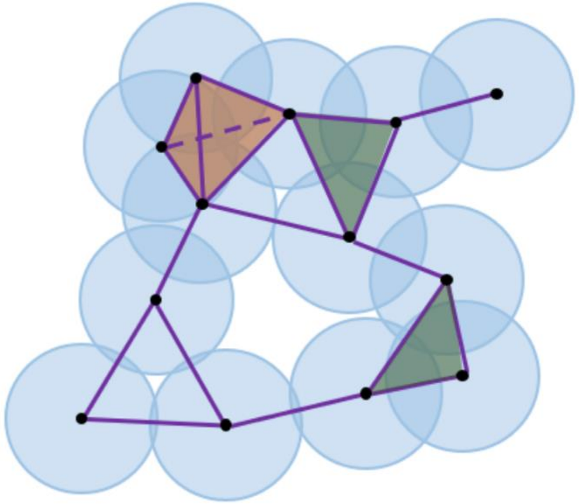


Image source: Reani, Bobrowski. Cycle Registration in Persistent Homology with Applications in Topological Bootstrap

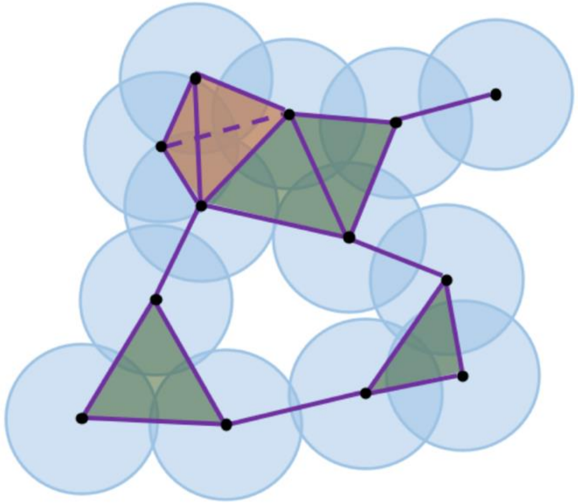
Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different





Čech



Rips

Image source: Reani, Bobrowski. Cycle Registration in Persistent Homology with Applications in Topological Bootstrap

Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different
- But notice that other than the two triangles the remaining simplices in the two complexes are the same

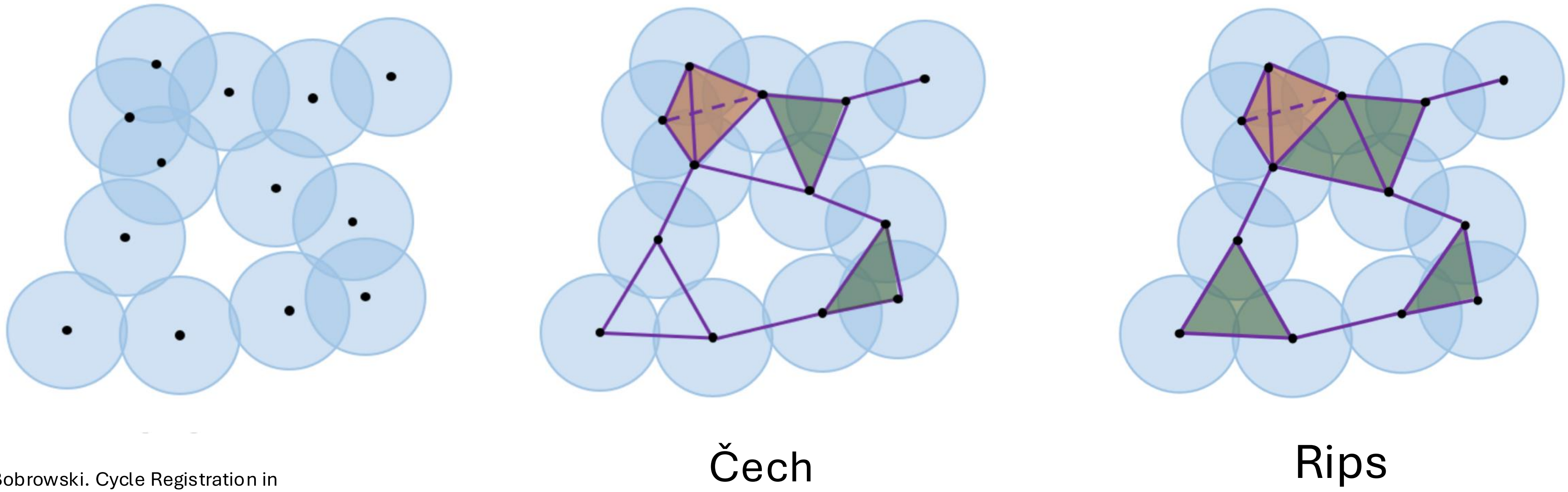
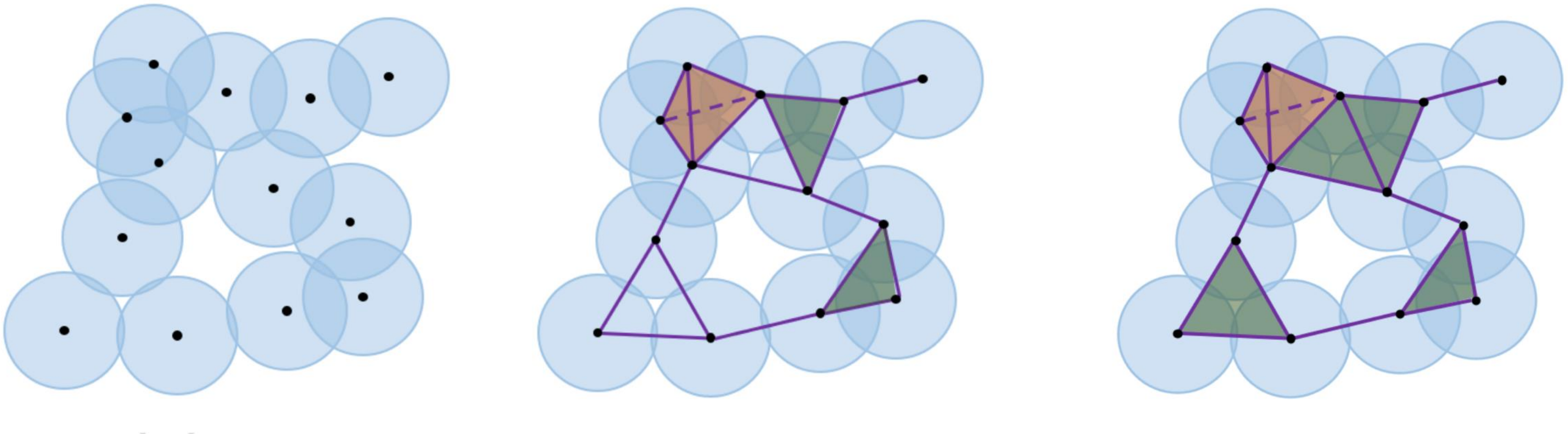


Image source: Reani, Bobrowski. Cycle Registration in Persistent Homology with Applications in Topological Bootstrap

Vietoris-Rips Filtration

- Following is an example where the Čech and Rips complexes are different
- But notice that other than the two triangles the remaining simplices in the two complexes are the same
- Furthermore, if we increase the radius for Čech complex, the two missing two triangles will come into picture ---- in some sense, the sequences of Čech and Rips complexes are "interleaved with each other"



Čech　　　　Rips

Image source: Reani, Bobrowski. Cycle Registration in Persistent Homology with Applications in Topological Bootstrap

Vietoris-Rips Filtration

- As for $\mathbb{C}^{\alpha}(P)$, we also need to show that $\mathbb{VR}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex, which is checking that the faces of each simplex in the set $\mathbb{VR}^{\alpha}(P)$ are also in $\mathbb{VR}^{\alpha}(P)$.

Vietoris-Rips Filtration

- As for $\mathbb{C}^{\alpha}(P)$, we also need to show that $\mathbb{VR}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex, which is checking that the faces of each simplex in the set $\mathbb{VR}^{\alpha}(P)$ are also in $\mathbb{VR}^{\alpha}(P)$.
- **Verification**: Take any $\sigma \in \mathbb{VR}^{\alpha}(P)$ and a face τ of σ (a subset of σ, i.e.,$\tau \subseteq \sigma$).

Vietoris-Rips Filtration

- As for $\mathbb{C}^{\alpha}(P)$, we also need to show that $\mathbb{VR}^{\alpha}(P)$ is not only a set of simplices but also a simplicial complex, which is checking that the faces of each simplex in the set $\mathbb{VR}^{\alpha}(P)$ are also in $\mathbb{VR}^{\alpha}(P)$.
- **Verification**: Take any $\sigma \in \mathbb{VR}^{\alpha}(P)$ and a face τ of σ (a subset of σ, i.e.,$\tau \subseteq \sigma$).
- We have any pairs of the α-balls for the points in σ intersect. So any pairs of the α-balls for the points in τ intersect (because τ is a subset). So $\tau \in \mathbb{C}^{\alpha}(P)$

Remark

- For both the Čech and Vietoris-Rips filtration, all the vertices are always assumed to be present at *any* radius α starting from 0
- So at a certain radius α, you only need to figure out what are the edges and higher-dimensional simplices in the corresponding complex

PD for Vietoris-Rips Filtration

- We now also define PD for a continuous filtration using the discrete Rips filtration
- Again, define the PD for the continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Rips filtration $\mathcal{VR}(P)$:
  - Each complex in the discrete $\mathcal{VR}(P)$ corresponds to a range of α-values (and a range of spaces) in the continuous filtration
  - For each interval $[b, d) \in PD(\mathcal{VR}(P))$, consider the corresponding complexes in $\mathcal{VR}(P)$, which are $K_b, K_{b+1}, \dots, K_{d-1}$.
  - Then, the corresponding interval of $PD(\mathcal{F}^c)$ is the union of all the α-values corresponding to $K_b, K_{b+1}, \dots, K_{d-1}$.

PD for Vietoris-Rips Filtration

- We now also define PD for a continuous filtration using the discrete Rips filtration
- Again, define the PD for the continuous filtration $\mathcal{F}^c$ by "inducing" from the PD of the discrete Rips filtration $\mathcal{VR}(P)$:
  - Each complex in the discrete $\mathcal{VR}(P)$ corresponds to a range of α-values (and a range of spaces) in the continuous filtration
  - For each interval $[b, d) \in PD(\mathcal{VR}(P))$, consider the corresponding complexes in $\mathcal{VR}(P)$, which are $K_b, K_{b+1}, \dots, K_{d-1}$.
  - Then, the corresponding interval of $PD(\mathcal{F}^c)$ is the union of all the α-values corresponding to $K_b, K_{b+1}, \dots, K_{d-1}$.
- Notice now there is **data loss** introduced because $\mathcal{VR}(P)$ is not exactly the same as $\mathcal{C}(P)$

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration** : For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration** : For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the first inclusion, consider any points $p_0, p_1, \ldots, p_d$ of P forming a simplex in $\mathbb{C}^{\alpha}(P)$.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration** : For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the first inclusion, consider any points $p_0, p_1, \dots, p_d$ of P forming a simplex in $\mathbb{C}^{\alpha}(P)$.
- Then, the all their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_1), \dots, B_{\alpha}(p_d)$ intersect.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration** : For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the first inclusion, consider any points $p_0, p_1, \dots, p_d$ of P forming a simplex in $\mathbb{C}^{\alpha}(P)$.
- Then, the all their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_1), \dots, B_{\alpha}(p_d)$ intersect.
- Surely, each pair of such balls $B_{\alpha}(p_i), B_{\alpha}(p_j)$ intersect.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration** : For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the first inclusion, consider any points $p_0, p_1, \dots, p_d$ of P forming a simplex in $\mathbb{C}^{\alpha}(P)$.
- Then, the all their α-balls $B_\alpha(p_0), B_\alpha(p_1), \dots, B_\alpha(p_d)$ intersect.
- Surely, each pair of such balls $B_\alpha(p_i), B_\alpha(p_j)$ intersect.
- So the points $p_0, p_1, \dots, p_d$ of P also form a simplex in $\mathbb{VR}^{\alpha}(P)$.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \ldots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \ldots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$
- *Fix* a point in the set, $p_0, p_1, \ldots, p_d$, say, p_0, and consider another arbitrary point p_i

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \dots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$
- *Fix* a point in the set, $p_0, p_1, \dots, p_d$, say, p_0, and consider another arbitrary point p_i
- Since their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_i)$ intersect,

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \ldots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$
- *Fix* a point in the set, $p_0, p_1, \ldots, p_d$, say, p_0, and consider another arbitrary point p_i
- Since their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_i)$ intersect,
- we have that p_0 is the 2α-ball of p_i, $B_{2\alpha}(p_i)$

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \ldots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$
- *Fix* a point in the set, $p_0, p_1, \ldots, p_d$, say, p_0, and consider another arbitrary point p_i
- Since their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_i)$ intersect,
- we have that p_0 is the 2α-ball of p_i, $B_{2\alpha}(p_i)$
- Since p_i is arbitrary, we have p_0 is in the intersection of 2α-balls for all these points, $B_{2\alpha}(p_0), B_{2\alpha}(p_1), \ldots, B_{2\alpha}(p_d)$

"Similarity" of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Proof**: For the second inclusion, consider any points $p_0, p_1, \dots, p_d$ of P forming a simplex in $\mathbb{VR}^{\alpha}(P)$
- *Fix* a point in the set, $p_0, p_1, \dots, p_d$, say, p_0, and consider another arbitrary point p_i
- Since their α-balls $B_{\alpha}(p_0), B_{\alpha}(p_i)$ intersect,
- we have that p_0 is the 2α-ball of p_i, $B_{2\alpha}(p_i)$
- Since p_i is arbitrary, we have p_0 is in the intersection of 2α-balls for all these points, $B_{2\alpha}(p_0), B_{2\alpha}(p_1), \dots, B_{2\alpha}(p_d)$
- So the points $p_0, p_1, \dots, p_d$ form a simplex in $\mathbb{C}^{2\alpha}(P)$

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Claim**: Due to the previous interleaving of the two sequences, we have that $\mathrm{PD}(\mathcal{VR}(\mathrm{P}))$ “approximates” $\mathrm{PD}(\mathcal{C}(P))$ well.
- Remark: This “approximation well” thing will be made more formal later.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Claim**: Due to the previous interleaving of the two sequences, we have that $\mathrm{PD}(\mathcal{VR}(\mathrm{P}))$ “approximates” $\mathrm{PD}(\mathcal{C}(P))$ well.
- Remark: This “approximation well” thing will be made more formal later.
- **Another observation**: The edges of the two Čech and Vietoris-Rips complexes for the same point set P over the same radius α are the same.

“Similarity” of Čech and Vietoris-Rips

- **Interleaving Theorem of Čech and Rips Filtration**: For any point set P and any radius α,

$$\mathbb{C}^{\alpha}(P) \subseteq \mathbb{VR}^{\alpha}(P) \subseteq \mathbb{C}^{2\alpha}(P)$$

- **Claim**: Due to the previous interleaving of the two sequences, we have that $\mathrm{PD}(\mathcal{VR}(\mathrm{P}))$ “approximates” $\mathrm{PD}(\mathcal{C}(P))$ well.
- Remark: This “approximation well” thing will be made more formal later.
- **Another observation**: The edges of the two Čech and Vietoris-Rips complexes for the same point set P over the same radius α are the same.
- Reason: Edges are formed by two points. If you check the definition of Čech and Vietoris-Rips (“all balls for a set of points intersect” and “each pair of balls for a set of points intersect”), when we only have two point, the two criteria become the same.

Vietoris-Rips Filtration: Alternative Definition

- **Definition**: Given a point set P and a distance r, the Vietoris-Rips complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if for each pair of points p_i, p_j in the subset, their distance is no more than r, i.e.,

$$d(p_i, p_j) \leq r$$

Vietoris-Rips Filtration: Alternative Definition

- **Definition**: Given a point set P and a distance r, the Vietoris-Rips complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if for each pair of points p_i, p_j in the subset, their distance is no more than r, i.e.,

$$d(p_i, p_j) \leq r$$

- The Rips complex in the above definition is the same as the previous Rips complex by taking $r/2$ as radius (if two points p_i, p_j have distance no more than r, then their $r/2$-balls intersect)

Vietoris-Rips Filtration: Alternative Definition

- **Definition**: Given a point set P and a distance r, the Vietoris-Rips complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if for each pair of points p_i, p_j in the subset, their distance is no more than r, i.e.,

$$d(p_i, p_j) \leq r$$

- The Rips complex in the above definition is the same as the previous Rips complex by taking $r/2$ as radius (if two points p_i, p_j have distance no more than r, then their $r/2$-balls intersect)
- The benefit of the above alternative definition is that we can completely eliminate balls and define Rips complexes / filtration **by only considering the pair-wise distances between points**

Vietoris-Rips Filtration: Alternative Definition

- E.g., there could be data where we have objects which are not naturally coming from a Euclidean space $\mathbb{R}^n$, i.e., without positions (like points from a triangular mesh).

Vietoris-Rips Filtration: Alternative Definition

- E.g., there could be data where we have objects which are not naturally coming from a Euclidean space $\mathbb{R}^n$, i.e., without positions (like points from a triangular mesh).
- They are abstract objects but we have some pair-wise “distances” between these objects.

Vietoris-Rips Filtration: Alternative Definition

- In this example, the data points are "regions of the brain", and we can calculate their "similarity" by measuring the correlation of their blood oxygen fluctuation over time (a time series data).

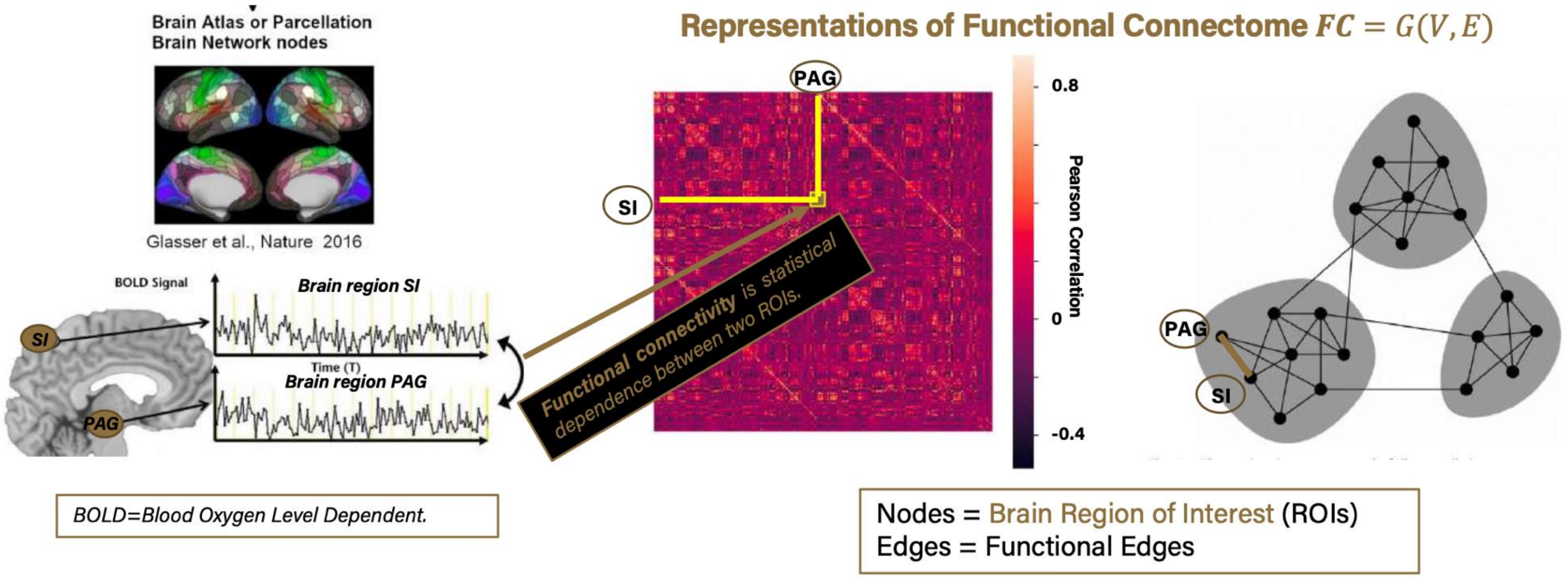


Figure courtesy of: Duy Duong-Tran

Vietoris-Rips Filtration: Alternative Definition

- In this example, the data points are "regions of the brain", and we can calculate their "similarity" by measuring the correlation of their blood oxygen fluctuation over time (a time series data).
- We then measure distances of two brain regions by taking inverse of similarity

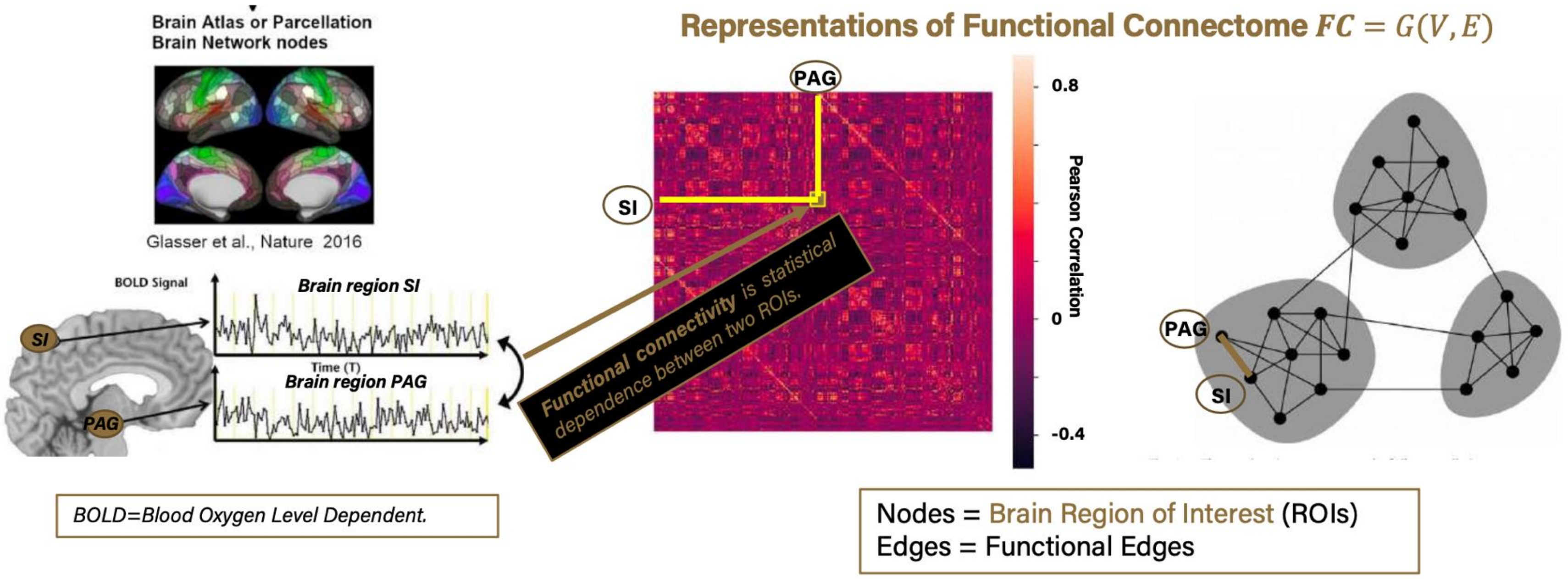


Figure courtesy of: Duy Duong-Tran

Vietoris-Rips Filtration: Alternative Definition

- In this example, the data points are “regions of the brain”, and we can calculate their “similarity” by measuring the correlation of their blood oxygen fluctuation over time (a time series data).
- We then measure distances of two brain regions by taking inverse of similarity
- These regions are not really technically having a position (each region is represented by a blood oxygen level function over time), but we have distances between the regions

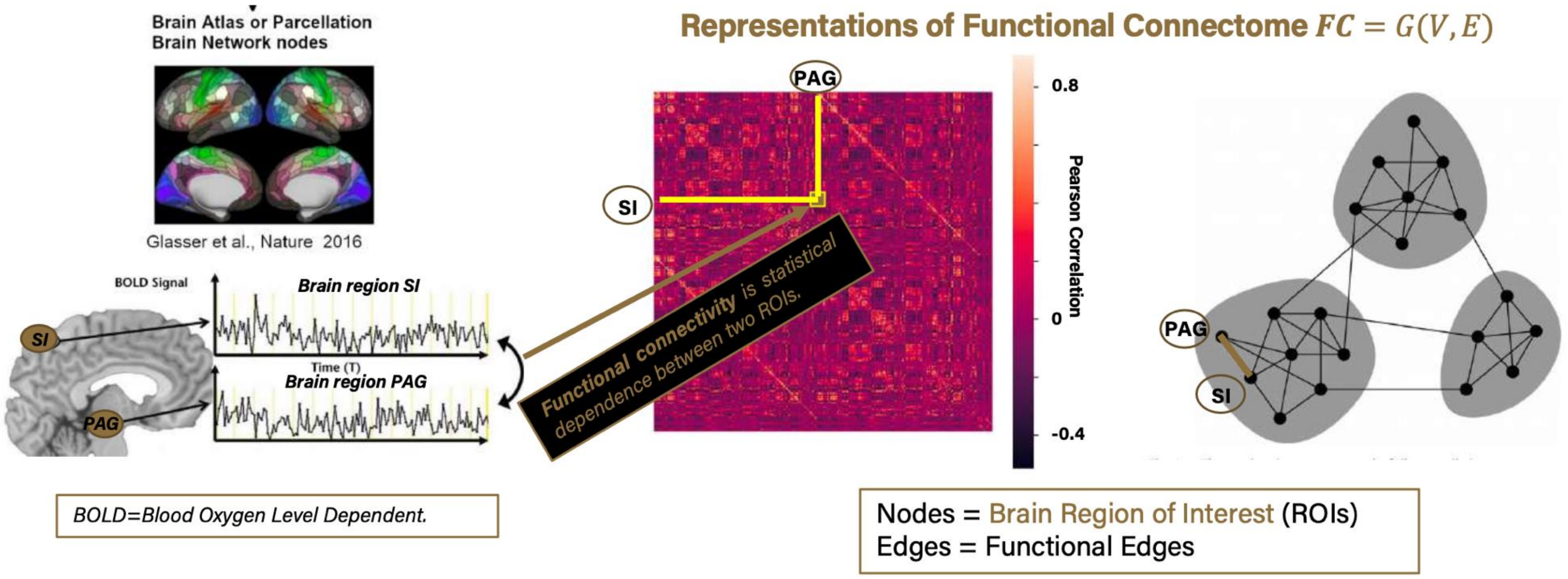


Figure courtesy of: Duy Duong-Tran

Vietoris-Rips Filtration: Alternative Definition

- For this data, we still can build Rips filtration on these brain regions

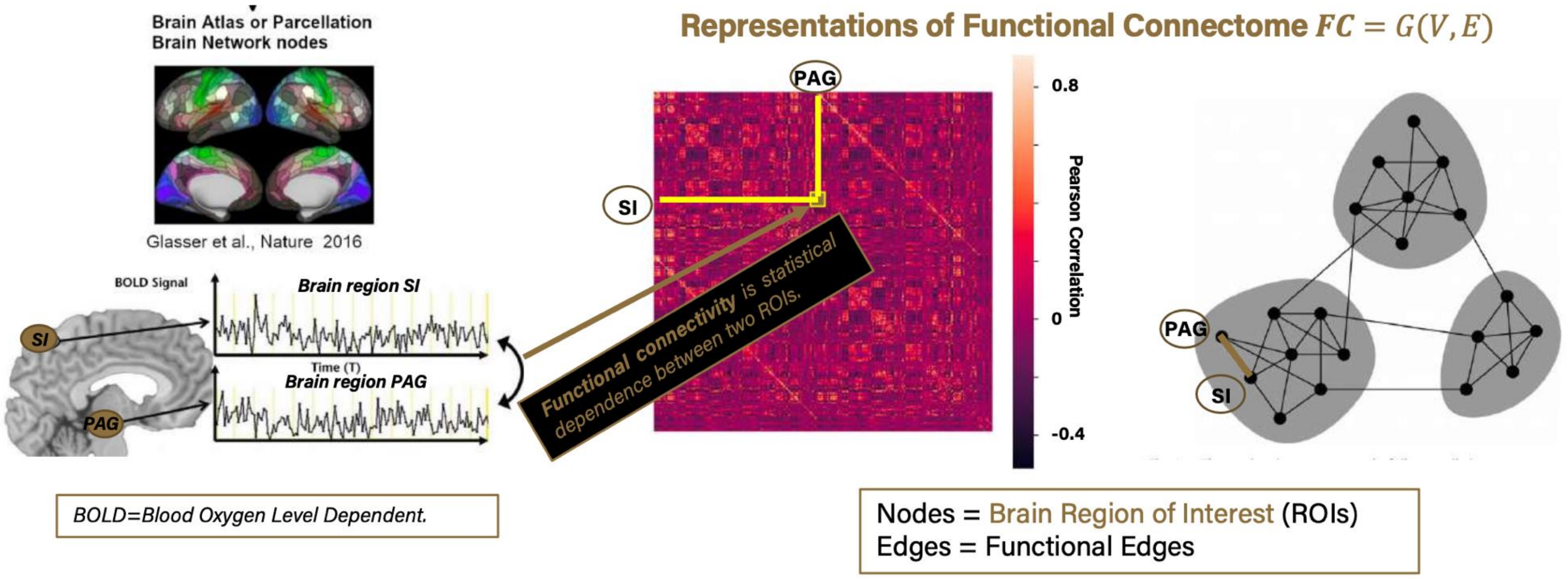


Figure courtesy of: Duy Duong-Tran

Vietoris-Rips Filtration: Computation

- We shall briefly look at some facts concerning computing Rips Filtration.
- Recall:
- **Definition**: Given a point set P and a distance r, the **Vietoris-Rips** complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if *for each pair of points p_i, p_j* in the subset, *their distance is no more than r*.

Vietoris-Rips Filtration: Computation

- We shall briefly look at some facts concerning computing Rips Filtration.
- Recall:
- **Definition**: Given a point set P and a distance r, the **Vietoris-Rips** complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \ldots, p_d$ of P form a d-simplex if and only if *for each pair of points* p_i, p_j in the subset, *their distance is no more than* r.
- This means that a Rips complex over a certain distance (radius) r is completely determined by the distances of each pair of points in P

Vietoris-Rips Filtration: Computation

- We shall briefly look at some facts concerning computing Rips Filtration.
- Recall:
- **Definition**: Given a point set P and a distance r, the **Vietoris-Rips** complex of P corresponding to the distance r, denoted $\mathbb{VR}^r(P)$, is a simplicial complex whose vertices are points in P such that
  - A subset of points $p_0, p_1, \dots, p_d$ of P form a d-simplex if and only if *for each pair of points p_i, p_j* in the subset, *their distance is no more than r*.
- This means that a Rips complex over a certain distance (radius) r is completely determined by the distances of each pair of points in P
- Since a pair of points form an edge, this also means that a $\mathbb{VR}^r(P)$ can be completely determined once we have figured out the edges (1-simplices) for $\mathbb{VR}^r(P)$

Vietoris-Rips Filtration: Computation

- So, for a certain r, to compute $\mathbb{VR}^r(P)$, our first thing to do: Enumerate each pair of points in P and check whether their distance is no more than r.
  - If this is true, we let the pair form an edge in $\mathbb{VR}^r(P)$
- By doing this, we have all the edges in $\mathbb{VR}^r(P)$.
- This can be done in $O(n^2)$ time where n is the number of points in P.

Vietoris-Rips Filtration: Computation

- So, for a certain r, to compute $\mathbb{VR}^r(P)$, our first thing to do: Enumerate each pair of points in P and check whether their distance is no more than r.
  - If this is true, we let the pair form an edge in $\mathbb{VR}^r(P)$
- By doing this, we have all the edges in $\mathbb{VR}^r(P)$.
- This can be done in $O(n^2)$ time where n is the number of points in P.
- All the edges in $\mathbb{VR}^r(P)$ form a graph denoted as $\mathbb{G}^r(P)$.

Vietoris-Rips Filtration: Computation

- So, for a certain r, to compute $\mathbb{VR}^r(P)$, our first thing to do: Enumerate each pair of points in P and check whether their distance is no more than r.
  - If this is true, we let the pair form an edge in $\mathbb{VR}^r(P)$
- By doing this, we have all the edges in $\mathbb{VR}^r(P)$.
- This can be done in $O(n^2)$ time where n is the number of points in P.
- All the edges in $\mathbb{VR}^r(P)$ form a graph denoted as $\mathbb{G}^r(P)$.
- $\mathbb{VR}^r(P)$ is then the **Clique complex** of the graph $\mathbb{G}^r(P)$.

Clique

- **Definition**: A **clique** of a graph $G = (V(G), E(G))$ is a subset S of $V(G)$ such that each pair of vertices of S form an edge in G.
- A clique of G is also sometimes termed a complete subgraph of G.

Clique

- **Definition**: A **clique** of a graph $G = (V(G), E(G))$ is a subset S of $V(G)$ such that each pair of vertices of S form an edge in G.
- A clique of G is also sometimes termed a complete subgraph of G.

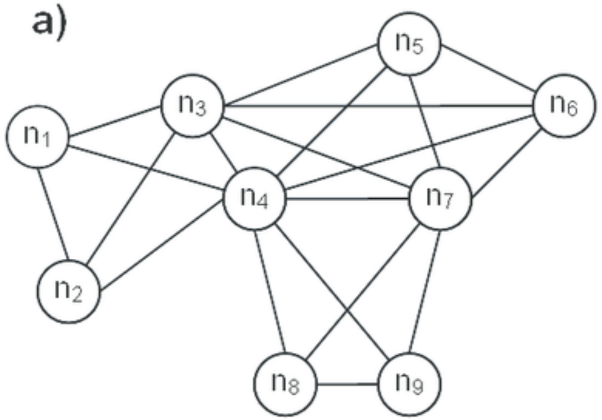


Image source: Santamaría, Therón. Overlapping Clustered Graphs: Co-authorship Networks Visualization

Clique

- **Definition**: A **clique** of a graph $G = (V(G), E(G))$ is a subset S of $V(G)$ such that each pair of vertices of S form an edge in G.
- A clique of G is also sometimes termed a complete subgraph of G.
- E.g, the following graph has three maximal cliques (a clique not contained in another clique)

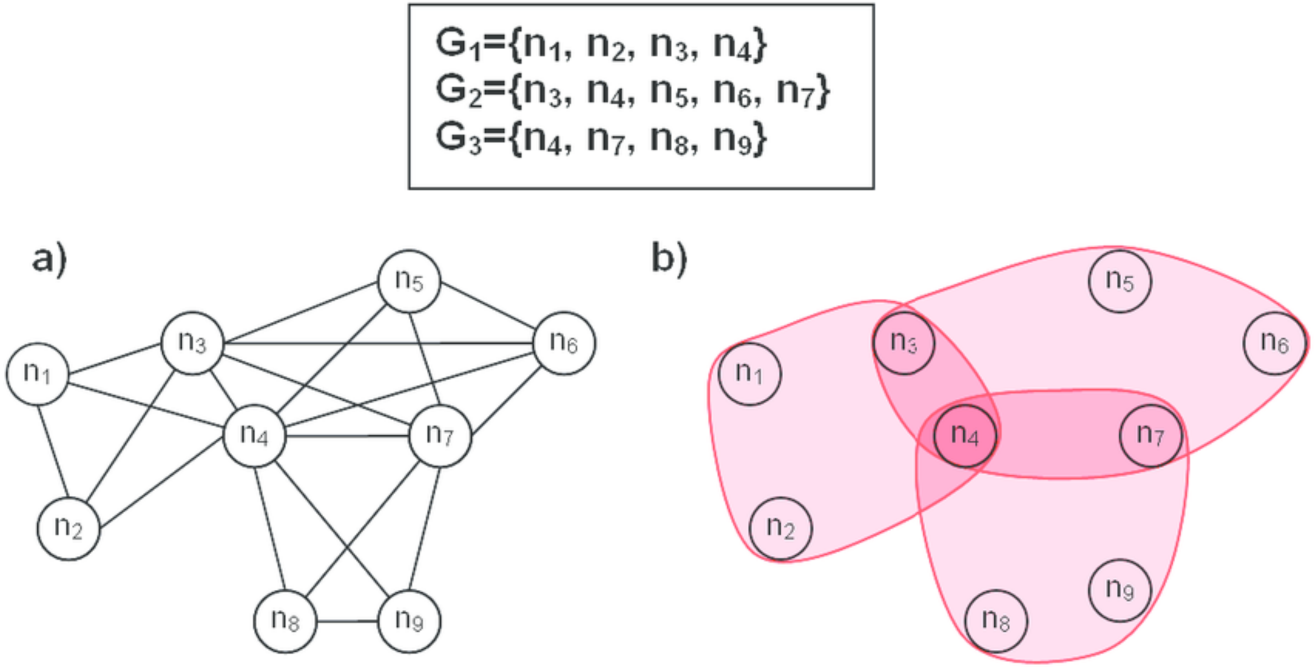


Image source: Santamaría, Therón. Overlapping Clustered Graphs: Co-authorship Networks Visualization

Clique Complex

- **Definition**: A **clique complex** $K(G)$ of a graph G is define as: there is a simplex σ in $K(G)$ if and only if the set of vertices of σ form a clique in G.

Clique Complex

- **Definition**: A **clique complex** $K(G)$ of a graph G is define as: there is a simplex σ in $K(G)$ if and only if the set of vertices of σ form a clique in G.
- E.g, the clique complex of below graph has three maximal simplices G_1, G_2, and G_3 (a simplex not being a face of another simplex) and all their faces

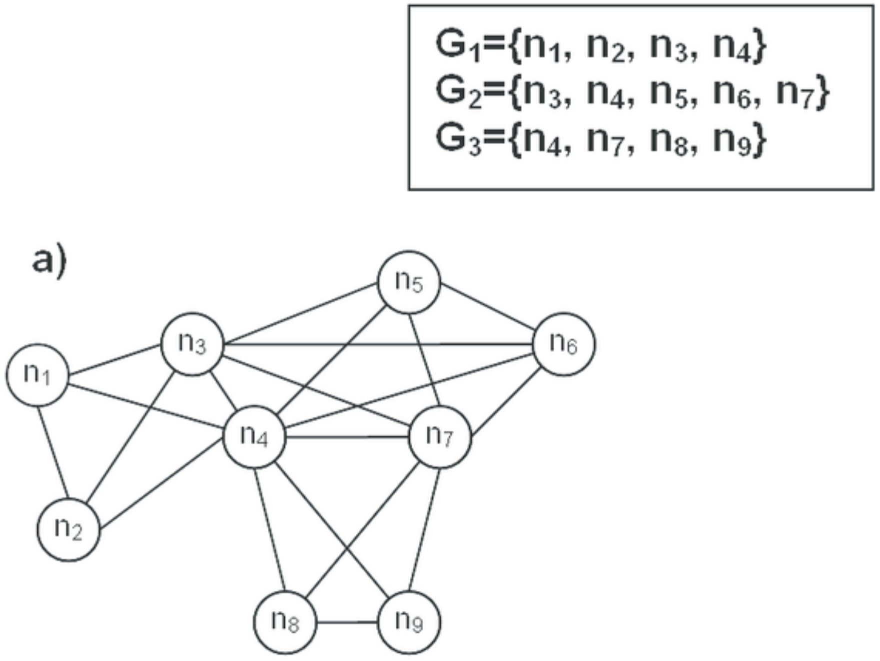
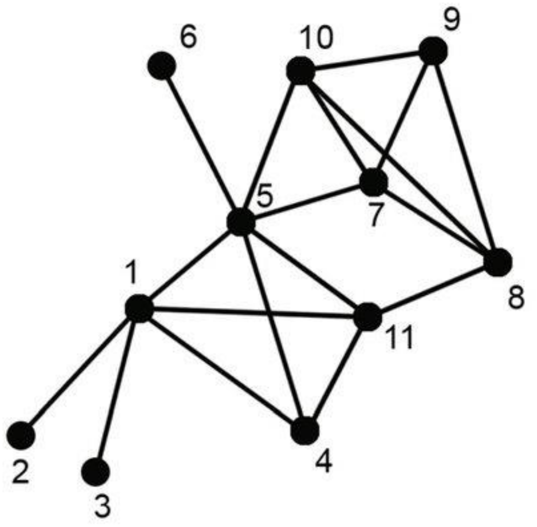


Image source: Santamaría, Therón. Overlapping Clustered Graphs: Co-authorship Networks Visualization

Clique Complex

- **Definition**: A **clique complex** $K(G)$ of a graph G is define as: there is a simplex σ in $K(G)$ if and only if the set of vertices of σ form a clique in G.



{1 4, 1 5, 1 11, 4 5, 4 11, 5 11} → [1, 4, 5, 11]
{7 8, 7 9, 7 10, 8 9, 8 10, 9 10} → [7, 8, 9, 10]
{5 7, 5 10, 7 10} → [5, 7, 10]
{1 2} → [1, 2]
{1 3} → [1, 3]
{5 6} → [5, 6]
{8 11} → [8, 11]

Image source: Horak, Maletić, Rajkovic. Persistent Homology of Complex Networks

Clique Complex

- **Definition**: A **clique complex** $K(G)$ of a graph G is define as: there is a simplex σ in $K(G)$ if and only if the set of vertices of σ form a clique in G.

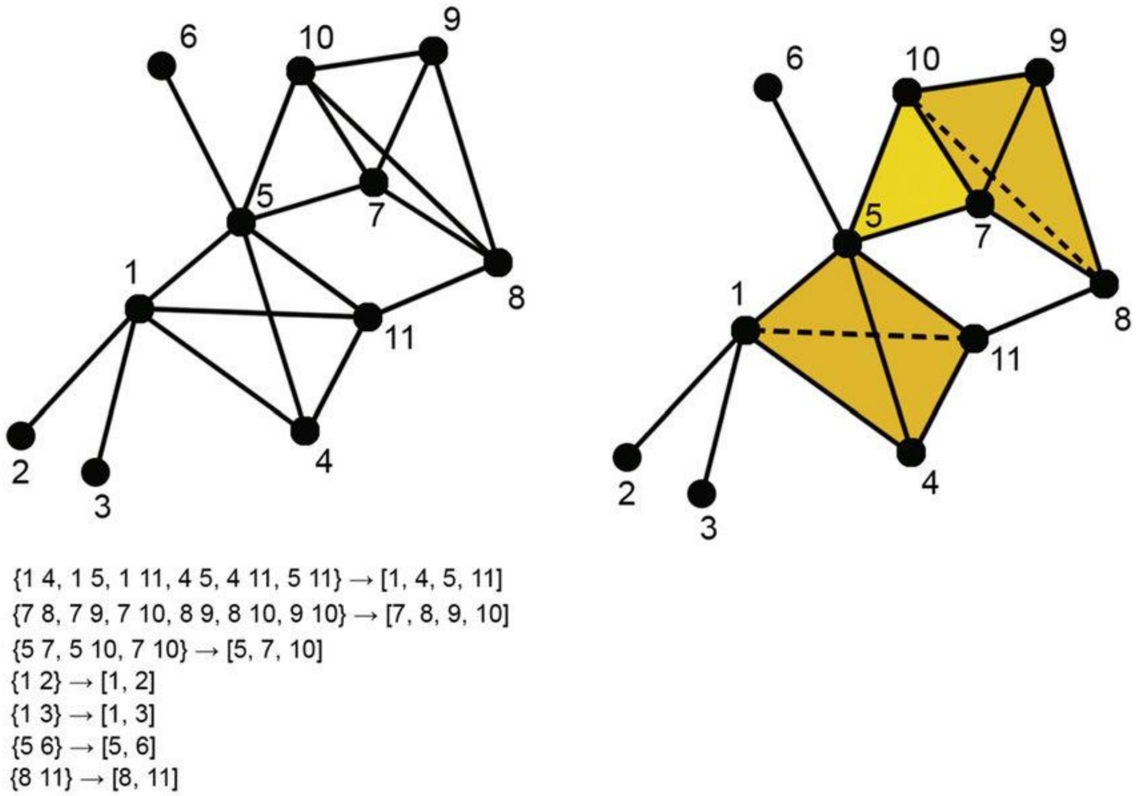


Image source: Horak, Maletić, Rajkovic. Persistent Homology of Complex Networks

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
  1. Build $\mathbb{G}^r(P)$

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
  1. Build $\mathbb{G}^r(P)$

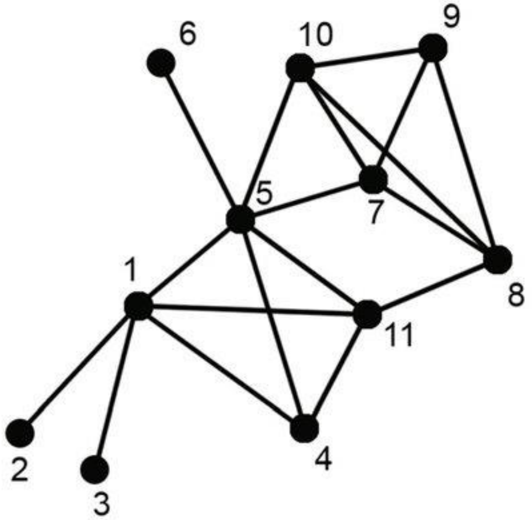


Image source: Horak, Maletić, Rajkovic. Persistent Homology of Complex Networks

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
    1. Build $\mathbb{G}^r(P)$
    2. Based on $\mathbb{G}^r(P)$, build $\mathbb{VR}^r(P)$ by taking the cliques

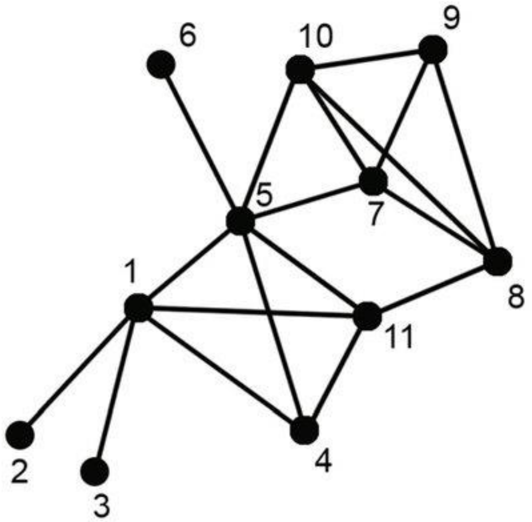


Image source: Horak, Maletić, Rajkovic. Persistent Homology of Complex Networks

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
  1. Build $\mathbb{G}^r(P)$
  2. Based on $\mathbb{G}^r(P)$, build $\mathbb{VR}^r(P)$ by taking the cliques

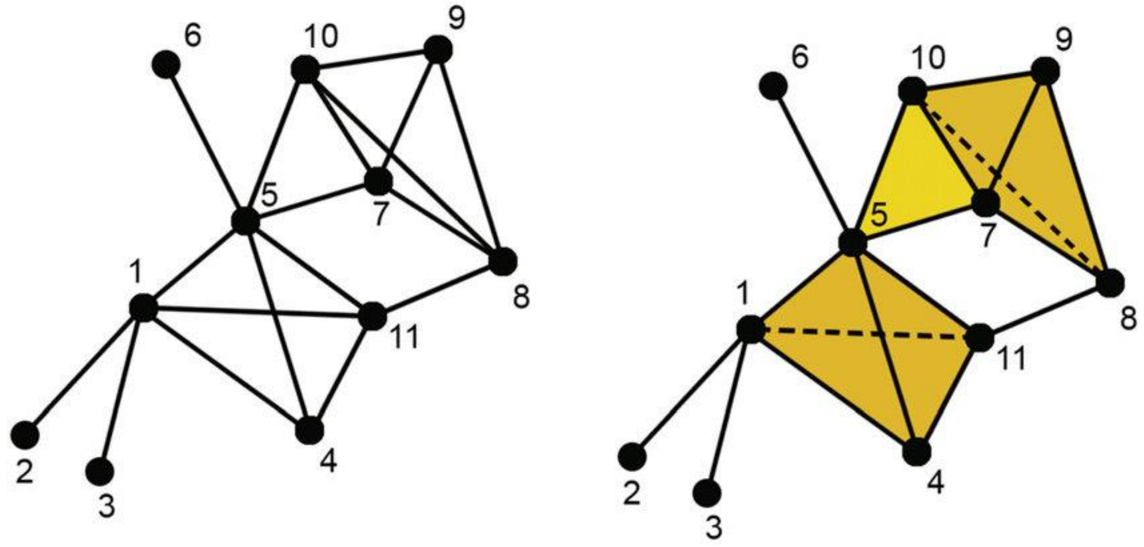


Image source: Horak, Maletić, Rajkovic. Persistent Homology of Complex Networks

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
  1. Build $\mathbb{G}^r(P)$
  2. Based on $\mathbb{G}^r(P)$, build $\mathbb{VR}^r(P)$ by taking the cliques
- The remaining thing to do:
  - Find the values r where $\mathbb{VR}^r(P)$ changes

Vietoris-Rips Filtration: Computation

- Now, given a value r, we know how to build $\mathbb{VR}^r(P)$:
    1. Build $\mathbb{G}^r(P)$
    2. Based on $\mathbb{G}^r(P)$, build $\mathbb{VR}^r(P)$ by taking the cliques
- The remaining thing to do:
    - Find the values r where $\mathbb{VR}^r(P)$ changes
    - This is equivalent to finding the values r where $\mathbb{G}^r(P)$ changes

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$
  - We consider all pairs of vertices of P (no matter the distance)

$$e_1, e_2, \dots, e_m;$$

  they are all possible edges of $\mathbb{G}^r(P)$.

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$
  - We consider all pairs of vertices of P (no matter the distance)

    $$e_1, e_2, \dots , e_m;$$

    they are all possible edges of $\mathbb{G}^r(P)$.
  - Furthermore, we let their distances be sorted:

$$d(e_1) < d(e_2) < \cdots < d(e_m)$$

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$
  - We consider all pairs of vertices of P (no matter the distance)

$$e_1, e_2, \dots, e_m;$$

    they are all possible edges of $\mathbb{G}^r(P)$.
  - Furthermore, we let their distances be sorted:

$$d(e_1) < d(e_2) < \cdots < d(e_m)$$

  - If we draw these pairs and their distances on the real line, then for a certain r, edges in $\mathbb{G}^r(P)$ are those one to the left of r

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$
  - We consider all pairs of vertices of P (no matter the distance)

    $$e_1, e_2, \dots, e_m;$$

    they are all possible edges of $\mathbb{G}^r(P)$.
  - Furthermore, we let their distances be sorted:

    $$d(e_1) < d(e_2) < \cdots < d(e_m)$$
  - If we draw these pairs and their distances on the real line, then for a certain r, edges in $\mathbb{G}^r(P)$ are those one to the left of r
  - This means that only crossing the values $d(e_1) < d(e_2) < \cdots < d(e_m)$, $\mathbb{G}^r(P)$ changes

Finding the values r where $\mathbb{G}^r(P)$ changes

- Thoughts:
  - Edges of $\mathbb{G}^r(P)$ contains those pairs of vertices of P whose distance $\leq r$
  - We consider all pairs of vertices of P (no matter the distance)

    $$e_1, e_2, \dots, e_m;$$

    they are all possible edges of $\mathbb{G}^r(P)$.
  - Furthermore, we let their distances be sorted:

    $$d(e_1) < d(e_2) < \cdots < d(e_m)$$

  - If we draw these pairs and their distances on the real line, then for a certain r, edges in $\mathbb{G}^r(P)$ are those one to the left of r
  - This means that only crossing the values $d(e_1) < d(e_2) < \cdots < d(e_m)$, $\mathbb{G}^r(P)$ changes
  - Finding these values takes $O(n^2 \log n)$ time dominated by the sorting, where n is the number of points in P

Another type of filtration

- **Delauney** complexes / filtrations: growing the balls around points, construct a simplex whenever their set of balls intersect (the same as Cech complexes)
- **Difference**: the part of a ball stops growing when touching another ball.

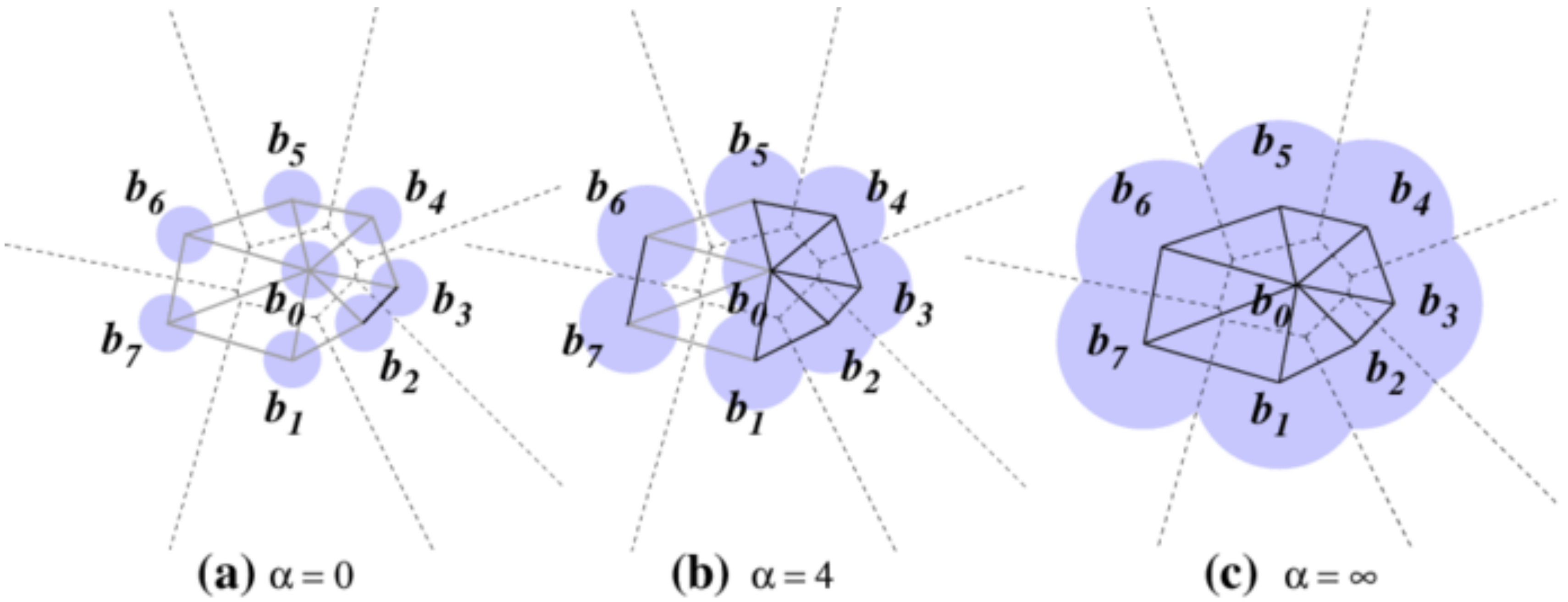


Image source: Li & Liang. Knowledge-Based Energy Functions for Computational Studies of Proteins

Delauney complexes / filtrations

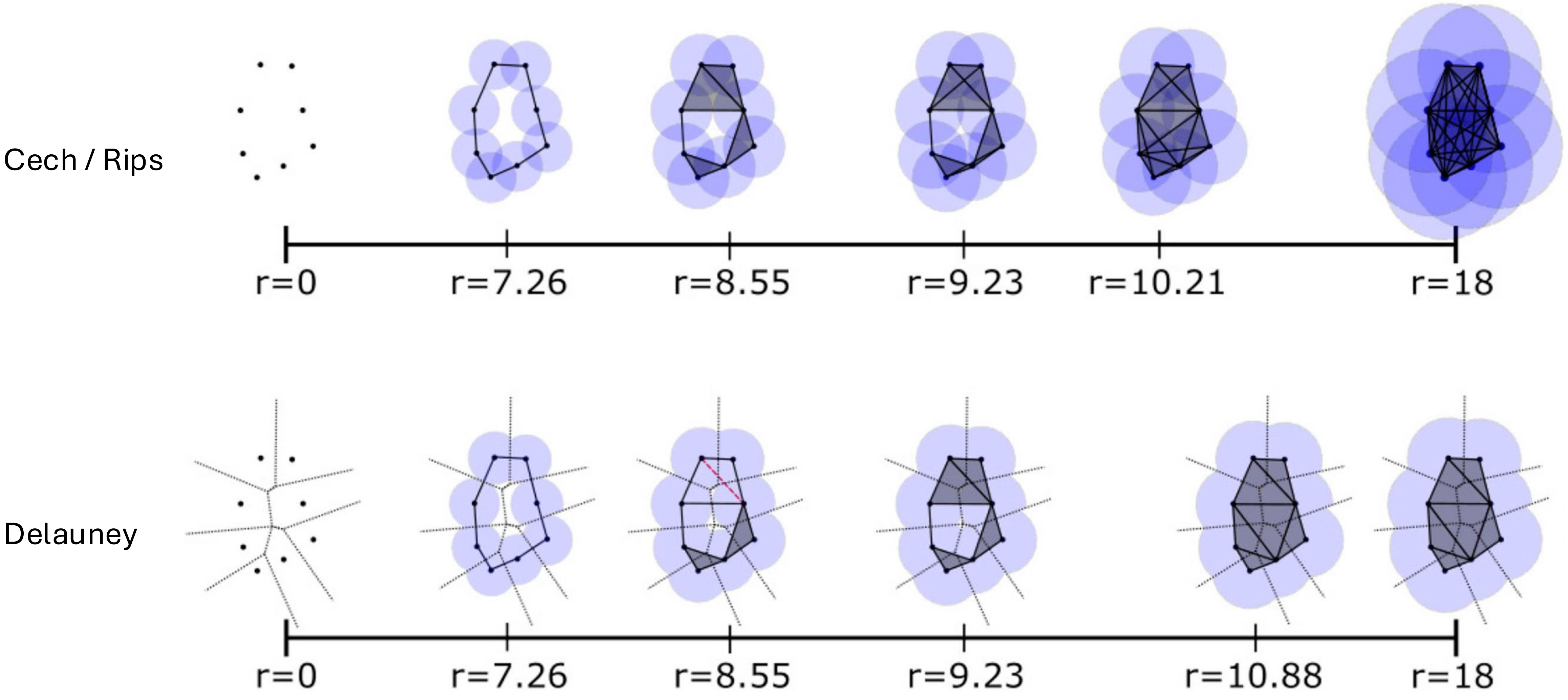


Image source: Mishra, Motta. Stability and machine learning applications of persistent homology using the Delaunay-Rips complex

Delauney complexes / filtrations

- An advantage of Delauney complexes is that it is always embedded in the space containing the point cloud (aka. simplices in the built complexes do not intersect with each other) ---- so having nicer geometric properties

Delauney complexes / filtrations

- An advantage of Delauney complexes is that it is always embedded in the space containing the point cloud (aka. simplices in the built complexes do not intersect with each other) ---- so having nicer geometric properties
- As a result, Delauney complexes are typically of lower dimensions than Rips / Cech complexes, and is of smaller size

Delauney complexes / filtrations

- An advantage of Delauney complexes is that it is always embedded in the space containing the point cloud (aka. simplices in the built complexes do not intersect with each other) ---- so having nicer geometric properties
- As a result, Delauney complexes are typically of lower dimensions than Rips / Cech complexes, and is of smaller size
- What's more, Delauney filtrations also have no information loss: a Delauney complex is "equivalent" to the union of balls as a Cech complex (so better than Rips)

Delauney complexes / filtrations

- An advantage of Delauney complexes is that it is always embedded in the space containing the point cloud (aka. simplices in the built complexes do not intersect with each other) ---- so having nicer geometric properties
- As a result, Delauney complexes are typically of lower dimensions than Rips / Cech complexes, and is of smaller size
- What's more, Delauney filtrations also have no information loss: a Delauney complex is "equivalent" to the union of balls as a Cech complex (so better than Rips)
- Disadvantage of Delauney complexes: costly to compute, especially when the dimension of the points in the point cloud is high

Delauney complexes / filtrations

- An advantage of Delauney complexes is that it is always embedded in the space containing the point cloud (aka. simplices in the built complexes do not intersect with each other) ---- so having nicer geometric properties
- As a result, Delauney complexes are typically of lower dimensions than Rips / Cech complexes, and is of smaller size
- What's more, Delauney filtrations also have no information loss: a Delauney complex is "equivalent" to the union of balls as a Cech complex (so better than Rips)
- Disadvantage of Delauney complexes: costly to compute, especially when the dimension of the points in the point cloud is high
- The go-to filtration for point cloud is **Rips filtration** because of (1) its computational efficiency and (2) the fact that it still faithfully recover the shape of the data (despite data loss)

Other types of complexes

- There are other types of complexes:
    - Witness complex
    - Graph-induced complex
    - Tangential complex
    - …
- Will not cover them at least for the time being

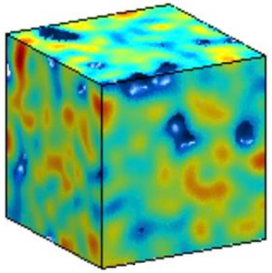
Data as a function

- So far we have covered how to build complexes / filtrations for point cloud
- The remaining part of this section will be devoted to the other major type of data: a **function**

Data as a function

- So far we have covered how to build complexes / filtrations for point cloud
- The remaining part of this section will be devoted to the other major type of data: a **function**
- Why function matters? Well, 2D images, or 3D volume data (aka. 3D images) are basically functions (and there are other types of functions occurring naturally in practice)

Image



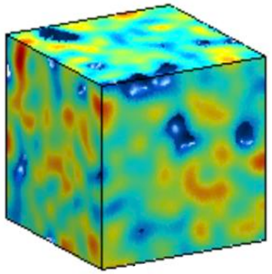
3D volume data

Img source: https://pixabay.com/photos/new-year-background-tree-sunset-736885/; Adler et al. Persistent homology for random fields and complexes

Data as a function

- So far we have covered how to build complexes / filtrations for point cloud
- The remaining part of this section will be devoted to the other major type of data: a **function**
- Why function matters? Well, 2D images, or 3D volume data (aka. 3D images) are basically functions (and there are other types of functions occurring naturally in practice)

Image



3D volume data

- E.g., all pixels in an image form the domain of a function and the color value on each pixel is basically the function value on a point of the domain

Img source: https://pixabay.com/photos/new-year-background-tree-sunset-736885/; Adler et al. Persistent homology for random fields and complexes

Setting

- We shall consider a general real-valued function:

$$f : X \to \mathbb{R}$$

where X is a certain domain (such as a rectangle, a cube, or even a simplicial complex).

Setting

- We shall consider a general real-valued function:

$$f: X \to \mathbb{R}$$

where X is a certain domain (such as a rectangle, a cube, or even a simplicial complex).

- Why only real-valued function?
    - Sometimes this is enough: e.g., for a colored image, we can convert it into a gray-scale image (0-255), which is basically a real-valued function

Setting

- We shall consider a general real-valued function:

$$f : X \to \mathbb{R}$$

where X is a certain domain (such as a rectangle, a cube, or even a simplicial complex).

- Why only real-valued function?
  - Sometimes this is enough: e.g., for a colored image, we can convert it into a gray-scale image (0-255), which is basically a real-valued function
  - Even if the range of the function is more than a single real value, say again, a colored image, we can take each channel (RGB), this will give you three individual real-valued functions. We can analyze each individually using persistence

Setting

- For simplicity of illustration, we first show how to construct everything from a gray-scale 2d image
- Doing this for 3d images or arbitrary simplicial complexes can be generalized

Setting

- For simplicity of illustration, we first show how to construct everything from a gray-scale 2d image
- Doing this for 3d images or arbitrary simplicial complexes can be generalized

Image

➡ $f: X \to \mathbb{R}$ ➡ Filtration ➡ PD

Img source: https://pixabay.com/photos/new-year-background-tree-sunset-736885/

Image function

- We visualize the domain X of a 2d image as a regular grid, where pixels are grid points (below an example of 4x4 image)

Image function

- We visualize the domain X of a 2d image as a regular grid, where pixels are grid points (below an example of 4x4 image)

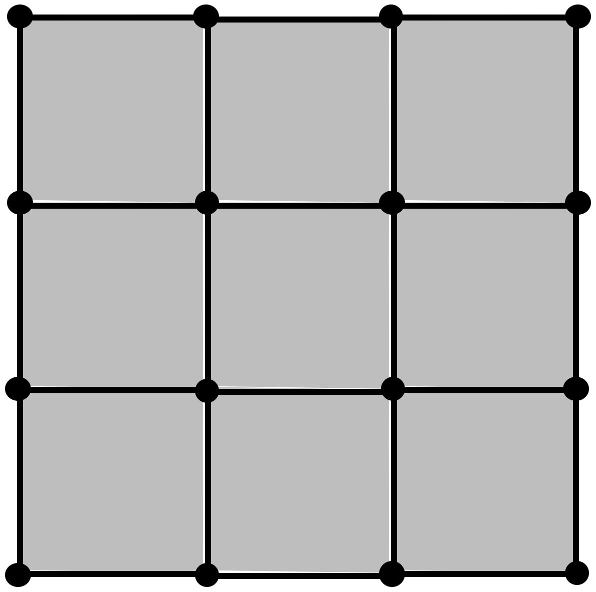


Image function

- We visualize the domain X of a 2d image as a regular grid, where pixels are grid points (below an example of 4x4 image)
- Then a 2d image is basically having a gray-scale value for each grade point

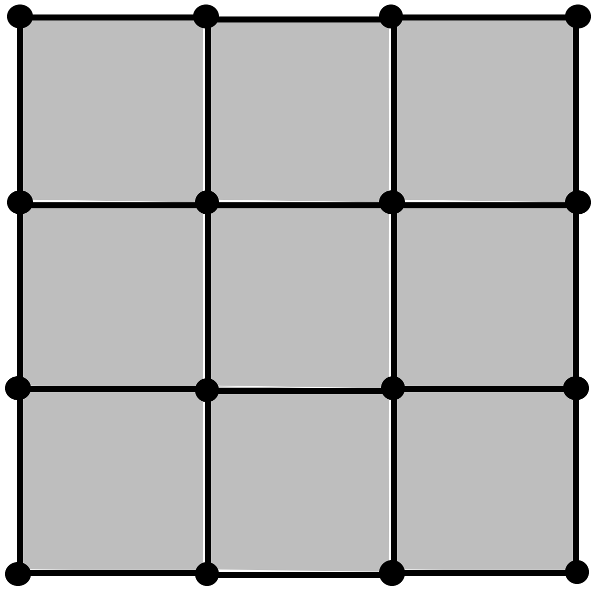


Image function

- We visualize the domain X of a 2d image as a regular grid, where pixels are grid points (below an example of 4x4 image)
- Then a 2d image is basically having a gray-scale value for each grade point

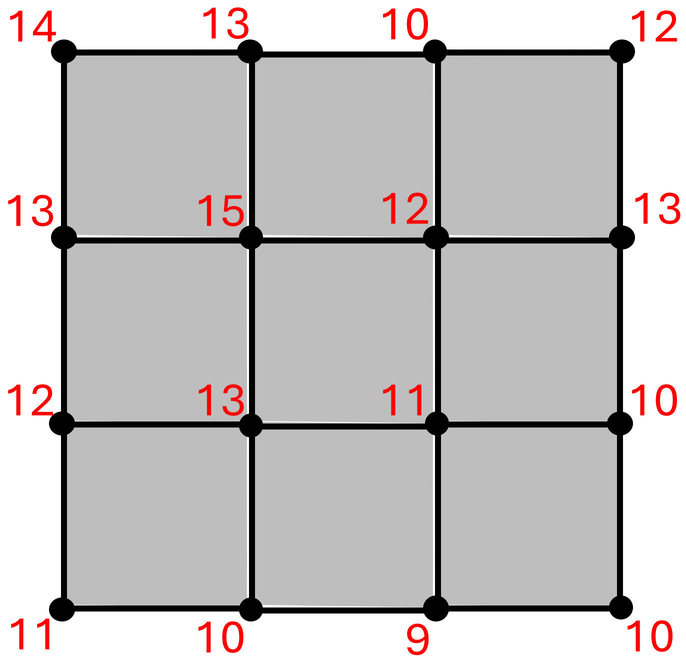


Image function

- Notice: the below regular grid does not form a simplicial complex (because of the square)

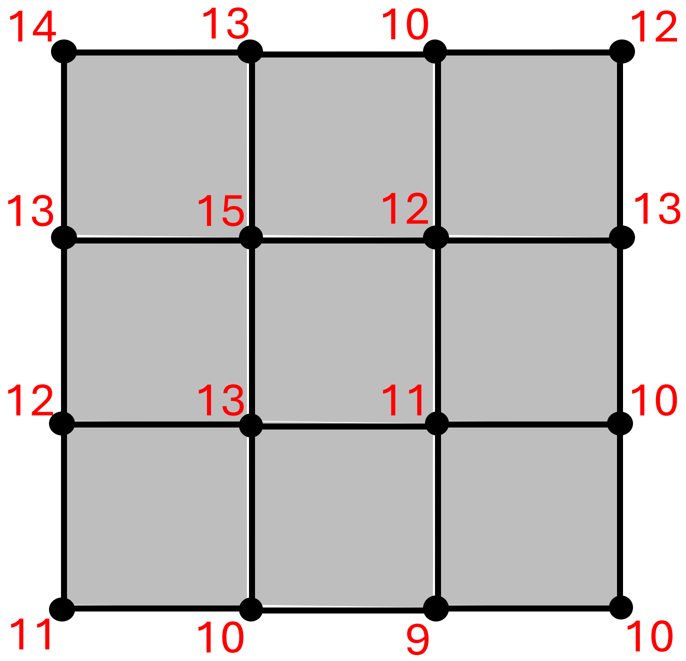


Image function

- Notice: the below regular grid does not form a simplicial complex (because of the square)
- So we subdivide the grid to be consisting of triangles, so X becomes a simplicial complex

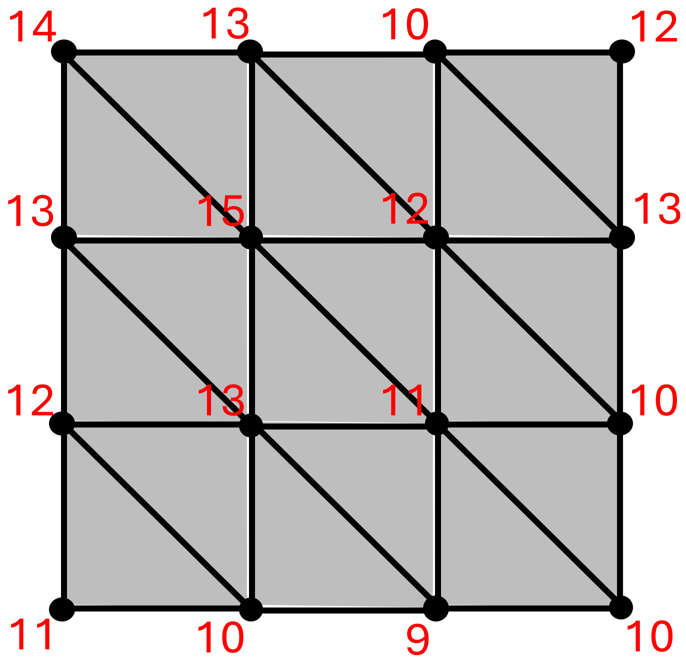


Image function

- Another problem: the function values are only given on the vertices (which are gray-scale values on the pixels from the given image)
- We need function values on the edges and triangles: for this we take the "maximum" value of the vertices that an edge or triangle contains

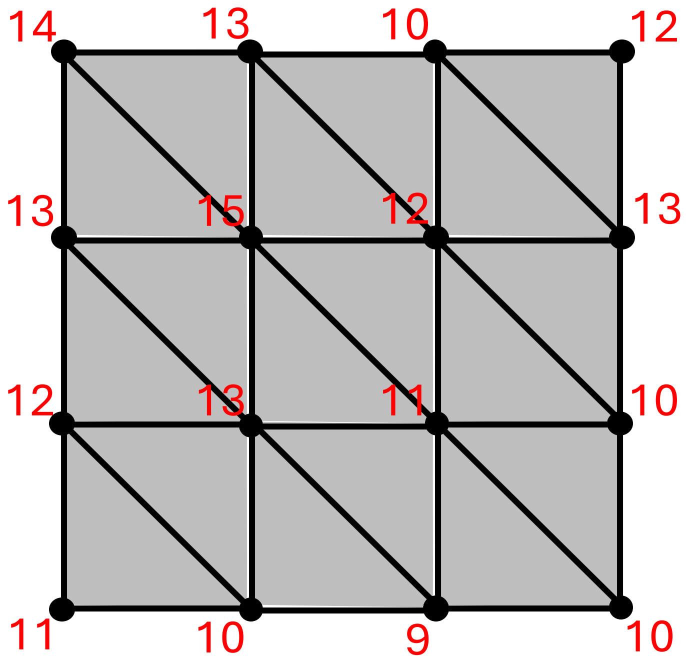


Image function

- Another problem: the function values are only given on the vertices (which are gray-scale values on the pixels from the given image)
- We need function values on the edges and triangles: for this we take the "maximum" value of the vertices that an edge or triangle contains

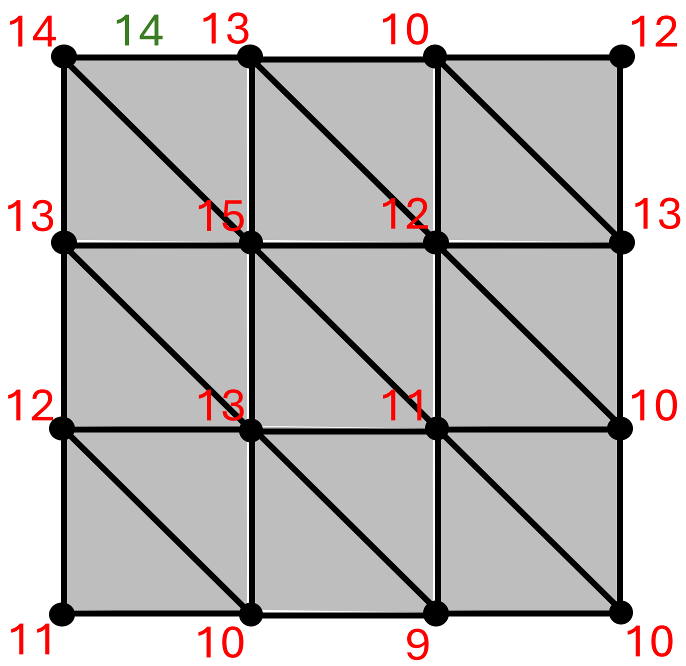


Image function

- Another problem: the function values are only given on the vertices (which are gray-scale values on the pixels from the given image)
- We need function values on the edges and triangles: for this we take the "maximum" value of the vertices that an edge or triangle contains

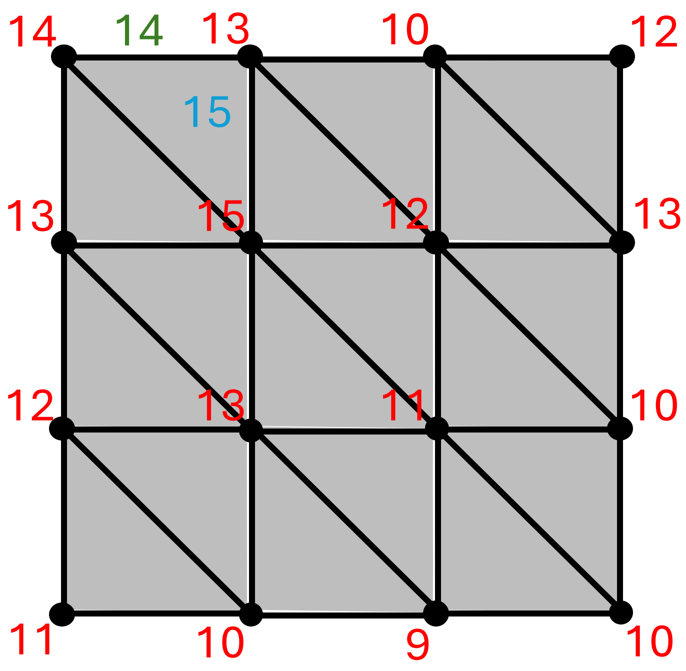
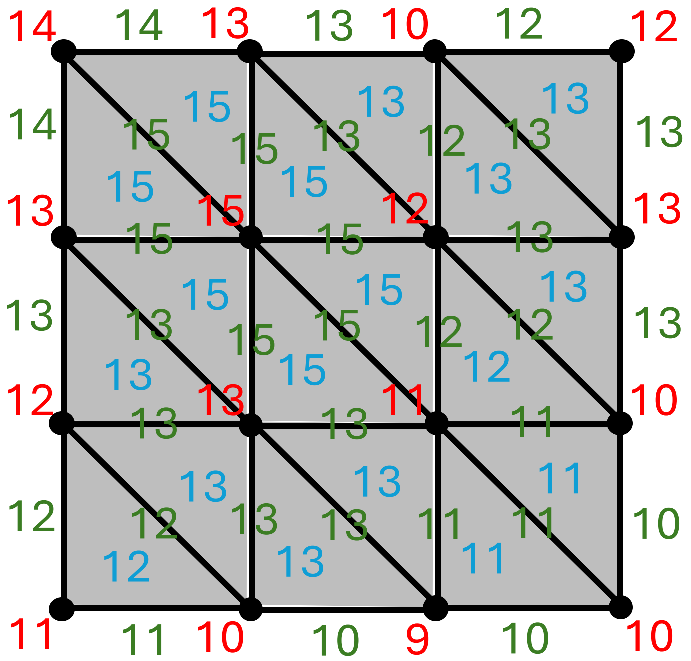


Image function

- Another problem: the function values are only given on the vertices (which are gray-scale values on the pixels from the given image)
- We need function values on the edges and triangles: for this we take the "maximum" value of the vertices that an edge or triangle contains



Filtration for image function

- We build the "**sublevelset filtration**" for the function $f: X \to \mathbb{R}$

Filtration for image function

- We build the “**sublevelset filtration**” for the function $f: X \to \mathbb{R}$
- A sublevelset of is the subset of X whose function values are less than or equal to a value α, and we denote it as $f^{-1}(-\infty, \alpha]$

Filtration for image function

- We build the "**sublevelset filtration**" for the function $f: X \to \mathbb{R}$
- A sublevelset of is the subset of X whose function values are less than or equal to a value α, and we denote it as $f^{-1}(-\infty, \alpha]$
- We then take all possible functions values (there are finitely many of them) and sort them (i.e, start with the lowest value):

$$\alpha_0 < \alpha_1 < \cdots < \alpha_m$$

Filtration for image function

- We build the "**sublevelset filtration**" for the function $f: X \to \mathbb{R}$
- A sublevelset of is the subset of X whose function values are less than or equal to a value α, and we denote it as $f^{-1}(-\infty, \alpha]$
- We then take all possible functions values (there are finitely many of them) and sort them (i.e, start with the lowest value):

$$\alpha_0 < \alpha_1 < \cdots < \alpha_m$$

- The sublevelset filtration is then the sublevelsets over the above values:

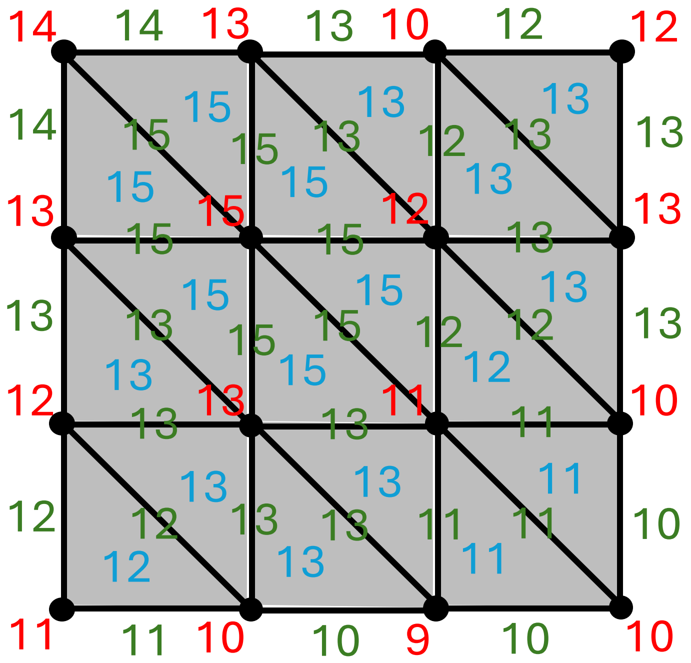
$$f^{-1}(-\infty, \alpha_0] \subseteq f^{-1}(-\infty, \alpha_1] \subseteq \cdots \subseteq f^{-1}(-\infty, \alpha_m]$$

Filtration for image function

- We build the "**sublevelset filtration**" for the function $f: X \to \mathbb{R}$
- A sublevelset of is the subset of X whose function values are less than or equal to a value α, and we denote it as $f^{-1}(-\infty, \alpha]$
- We then take all possible functions values (there are finitely many of them) and sort them (i.e, start with the lowest value):
$$\alpha_0 < \alpha_1 < \cdots < \alpha_m$$
- The sublevelset filtration is then the sublevelsets over the above values:
$$f^{-1}(-\infty, \alpha_0] \subseteq f^{-1}(-\infty, \alpha_1] \subseteq \cdots \subseteq f^{-1}(-\infty, \alpha_m]$$
- It should be esay to verify that $f^{-1}(-\infty, \alpha_i] \subseteq f^{-1}(-\infty, \alpha_{i+1}]$ for any i

Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11] \subseteq f^{-1}(12] \subseteq f^{-1}(13] \subseteq f^{-1}(14] \subseteq f^{-1}(15]$

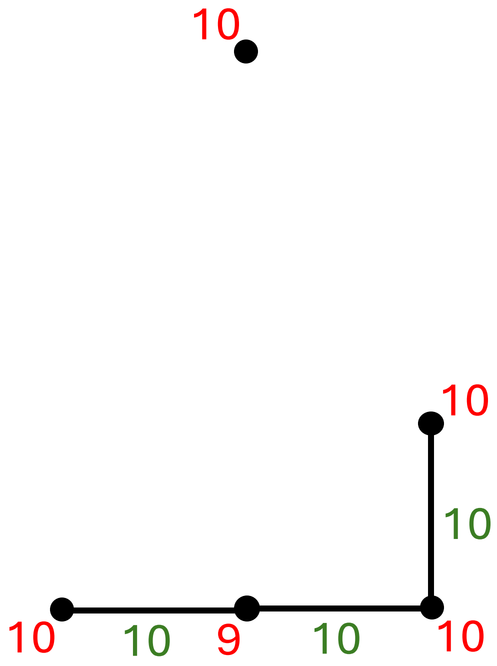


Sublevelset filtration

- $f^{-1}(9]$

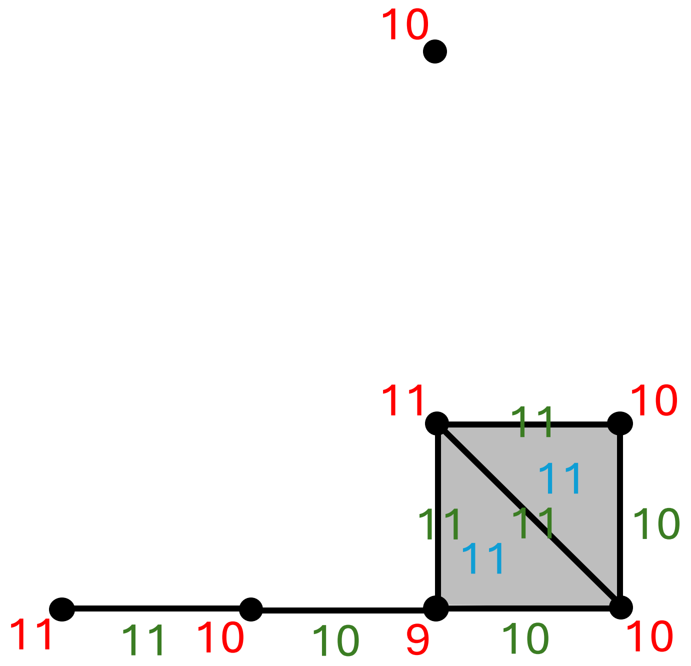
Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10]$



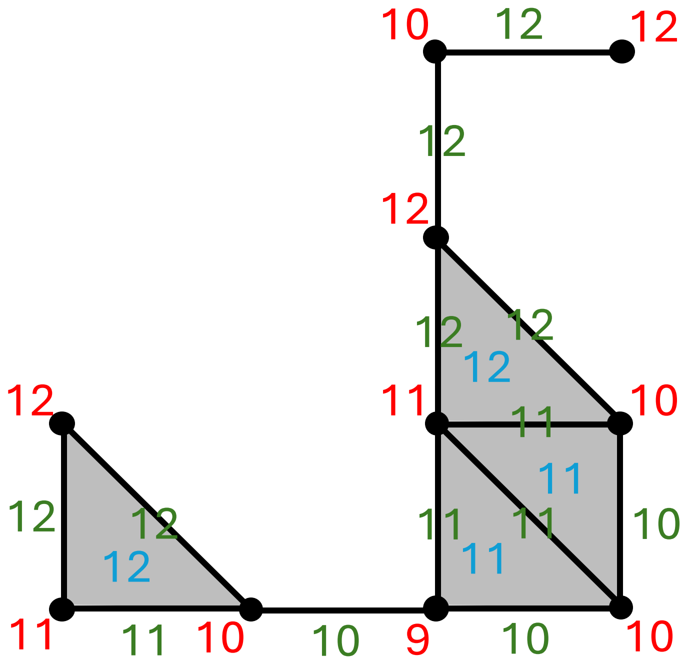
Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11]$



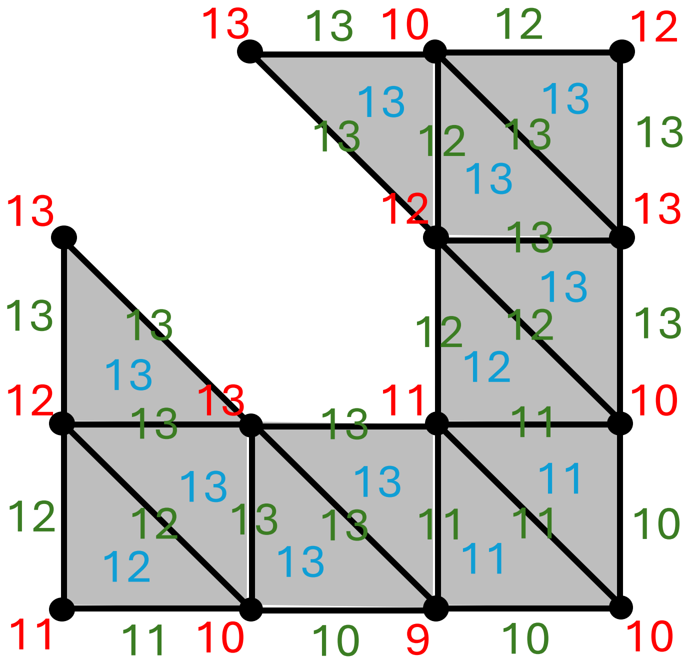
Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11] \subseteq f^{-1}(12]$



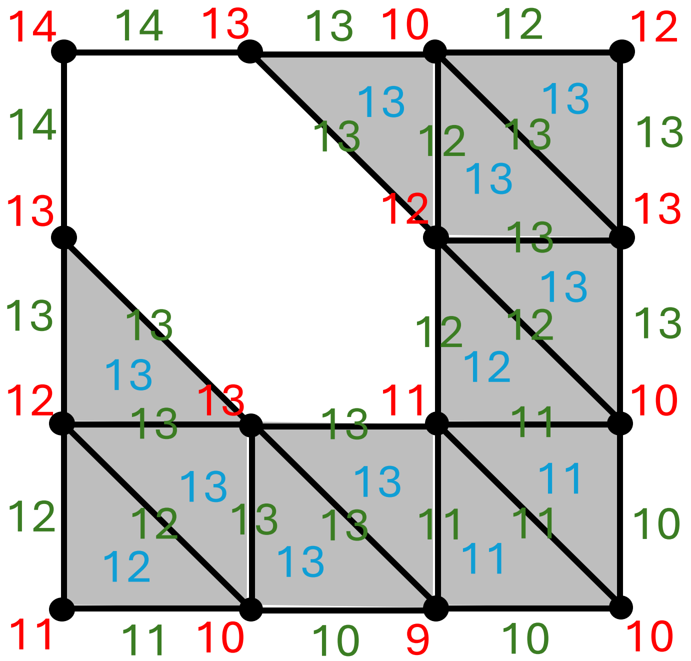
Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11] \subseteq f^{-1}(12] \subseteq f^{-1}(13]$



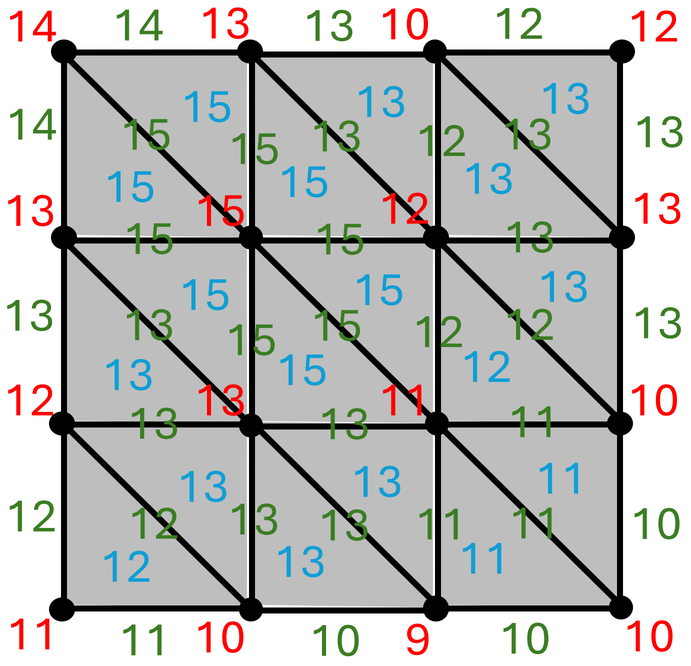
Sublevelset filtration

- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11] \subseteq f^{-1}(12] \subseteq f^{-1}(13] \subseteq f^{-1}(14]$



Sublevelset filtration

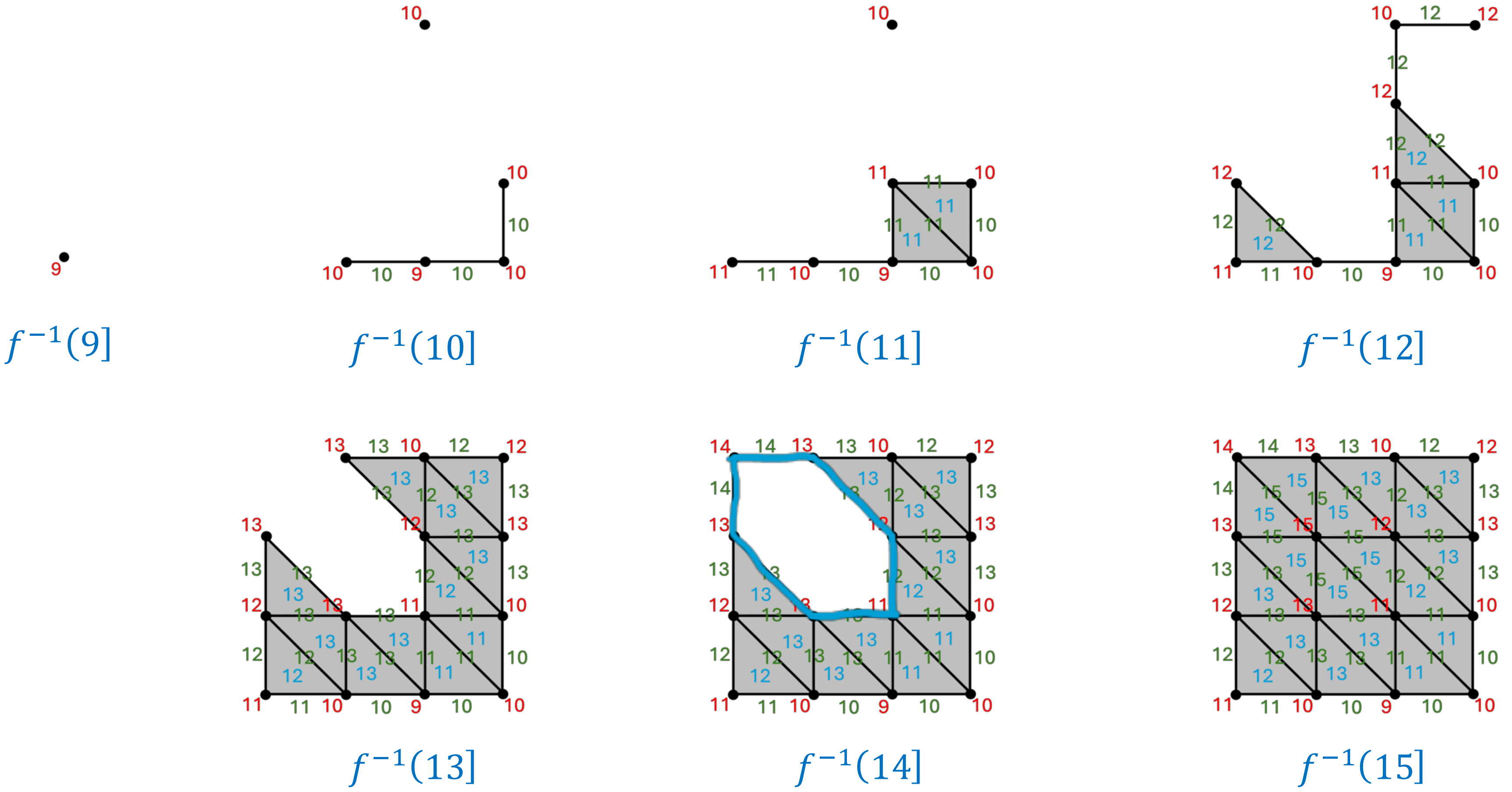
- $f^{-1}(9] \subseteq f^{-1}(10] \subseteq f^{-1}(11] \subseteq f^{-1}(12] \subseteq f^{-1}(13] \subseteq f^{-1}(14] \subseteq f^{-1}(15]$



PD for the sublevelset filtration

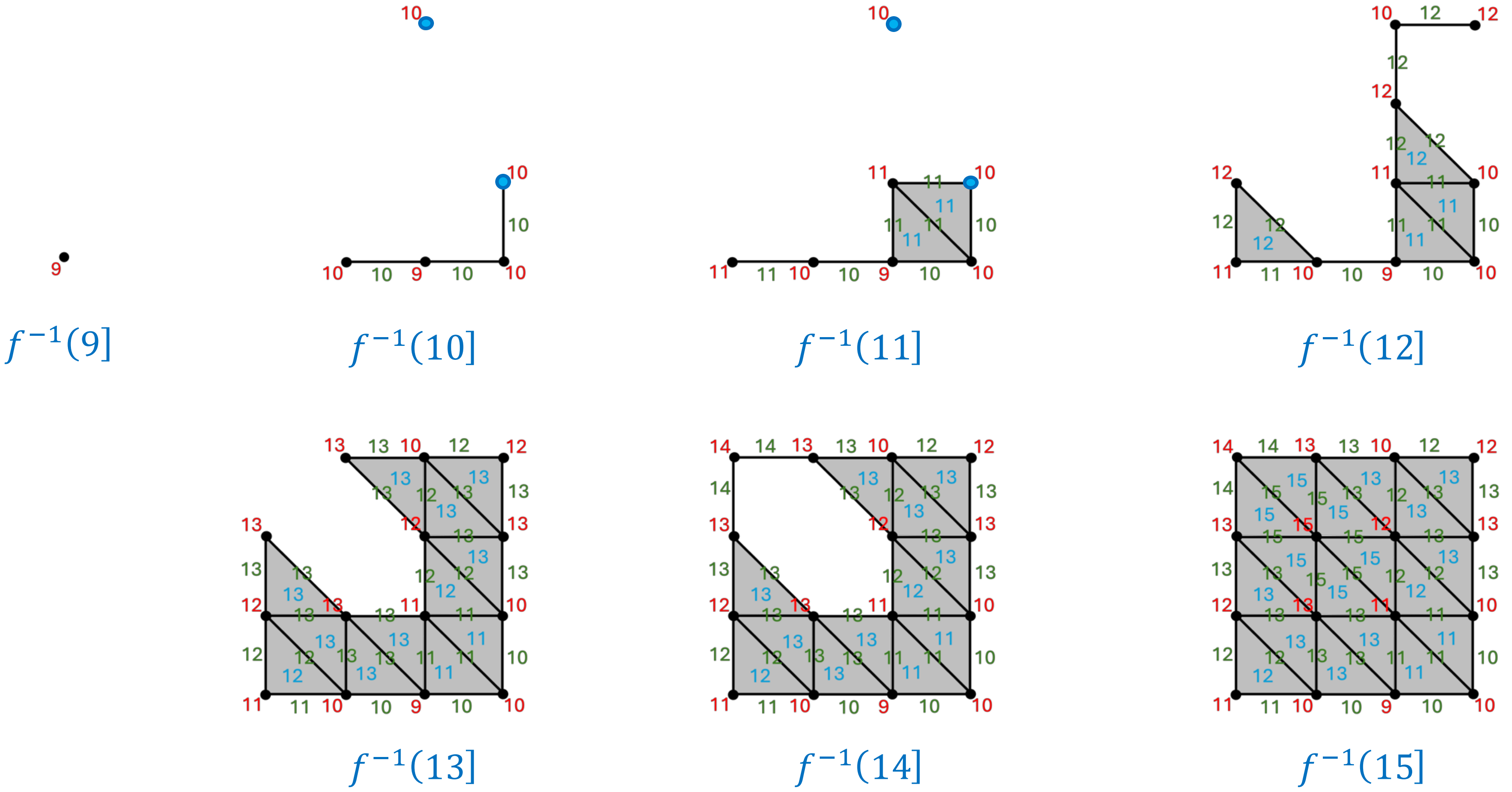
PD for the sublevelset filtration

- There is a 1-dimensional bar $[14,15)$ in the PD



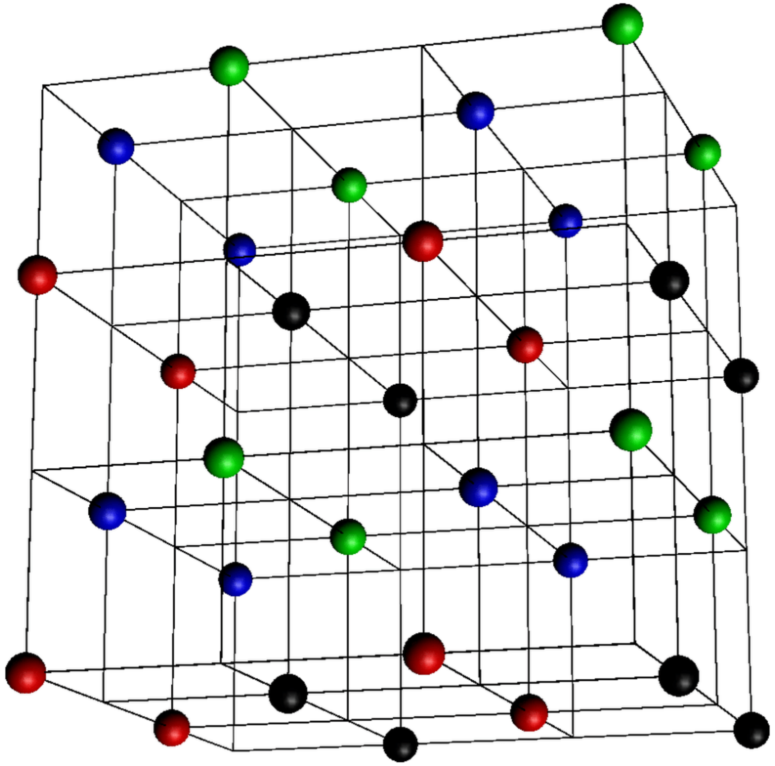
PD for the sublevelset filtration

- There is a 0-dimensional bar [10,12) in the PD



3D image

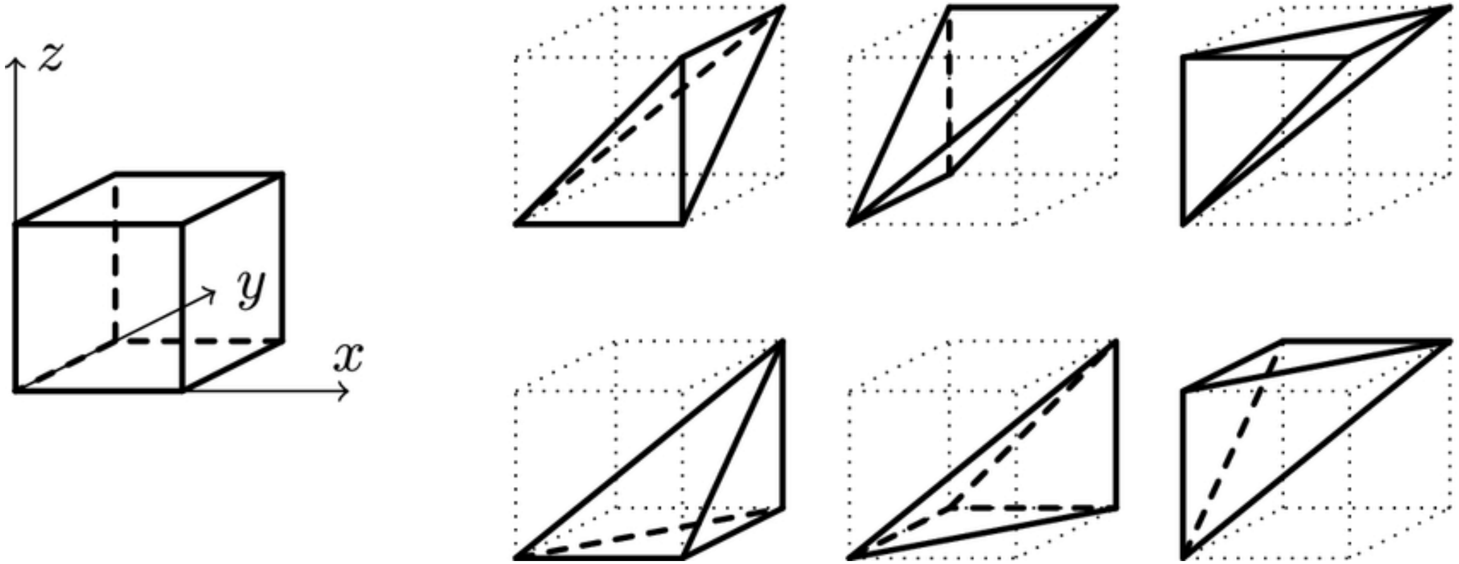
- We view the domain X for a 3D image as a 3D grid, and we have a function value on each grid point



Img source: Hamilton&Webb. Room Acoustics Modelling Using GPU-Accelerated Finite Difference and Finite Volume Methods on a Face-Centered Cubic Grid

3D image

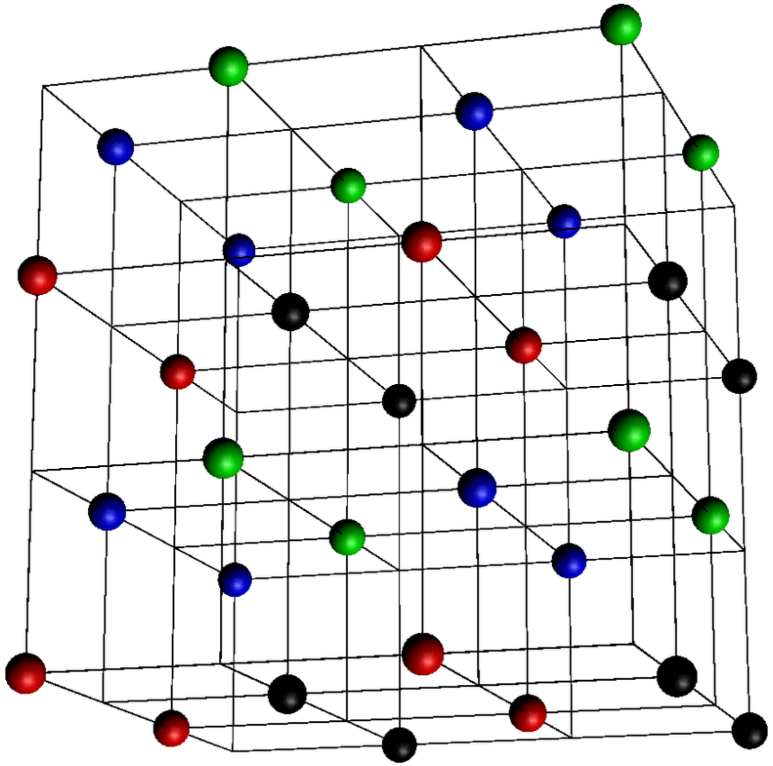
- We also need to subdivide the cube into (six) tetrahedra to make the domain a simplicial complex



de Crouy-Chanel, Simon. Random Knots in 3-Dimensional 3-Colour Percolation: Numerical Results and Conjectures

3D image

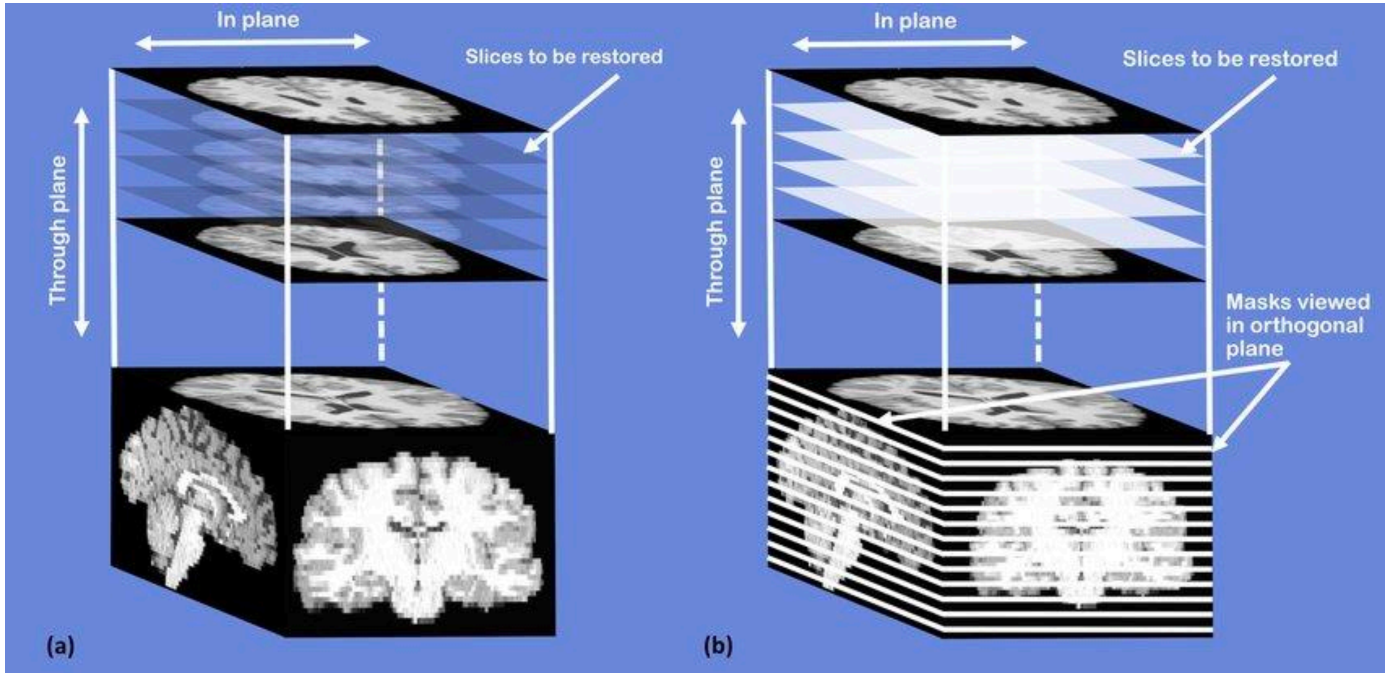
- And then we only need to assign value to each edge, triangle, tetrahedron based on the maximum values of their vertices
- The sublevelset filtration can then be defined similarly



Img source: Hamilton&Webb. Room Acoustics Modelling Using GPU-Accelerated Finite Difference and Finite Volume Methods on a Face-Centered Cubic Grid

More about 3D images

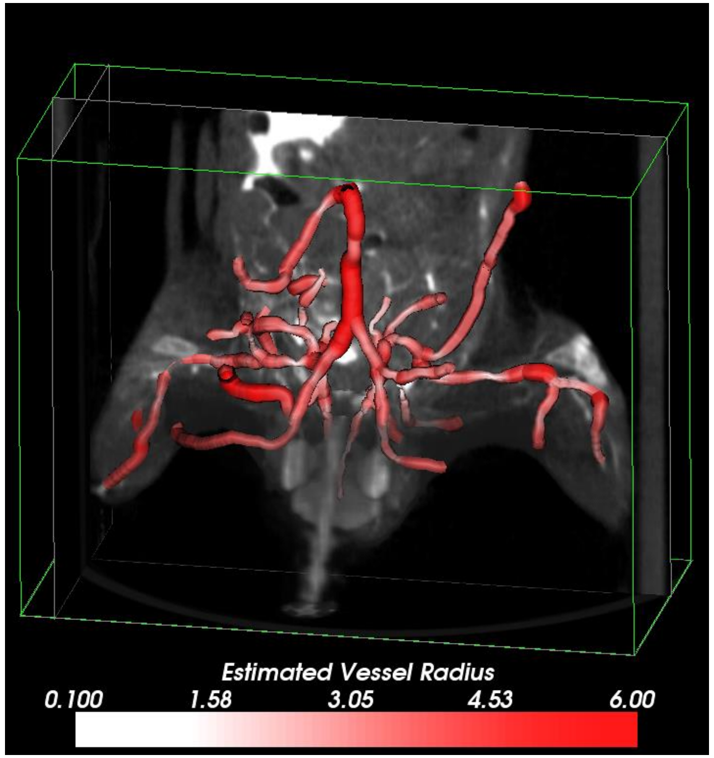
- 3D images can be considered as a stacking of several 2D images, and are commonly used in medical imaging (e.g., CT-scans, MRI)



Img source:. Chai et al. MRI Restoration Using Edge-Guided Adversarial Learning

More about 3D images

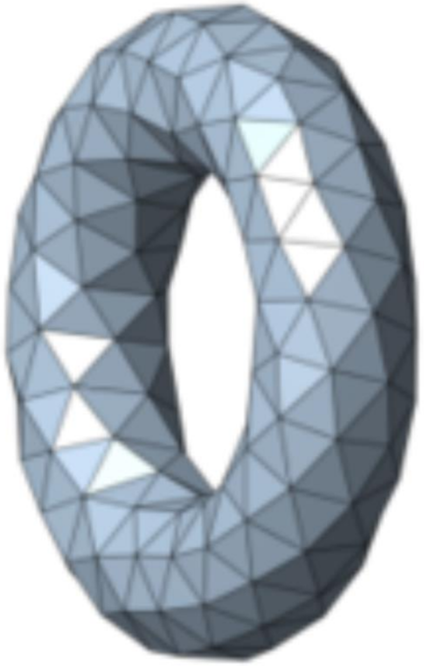
- 3D images can be considered as a stacking of several 2D images, and are commonly used in medical imaging (e.g., CT-scans, MRI)
- Analyzing medical images is a hot and important in image processing



Img source: Jackowski, Papademetris, Dobrucki, Staib. Characterizing Vascular Connectivity from microCT Images

Triangular meshes

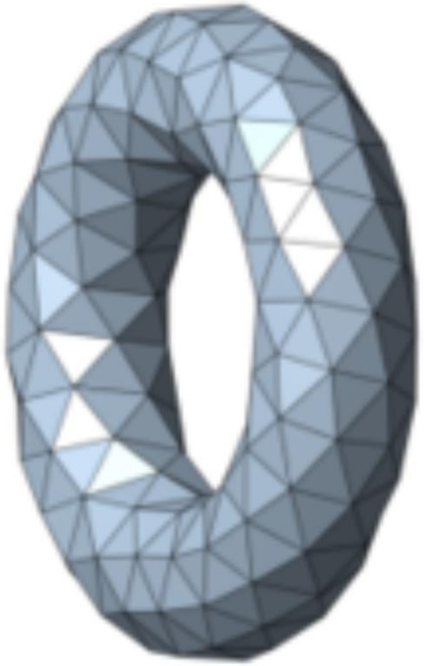
- Naturally, we could also define sublevelset filtrations on triangular meshes by assigning function values to the vertices (edges / triangles are then induced)



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

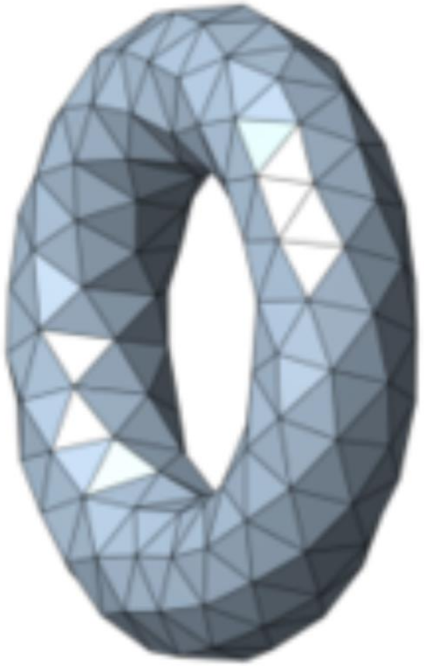
- Naturally, we could also define sublevelset filtrations on triangular meshes by assigning function values to the vertices (edges / triangles are then induced)
- There is a natural way to assign values to the vertices which is to use the "height function"



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

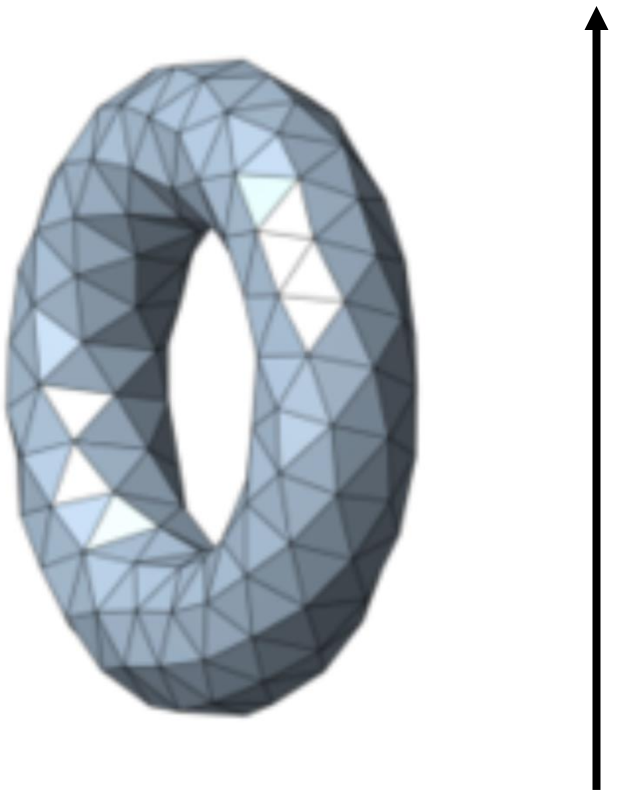
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

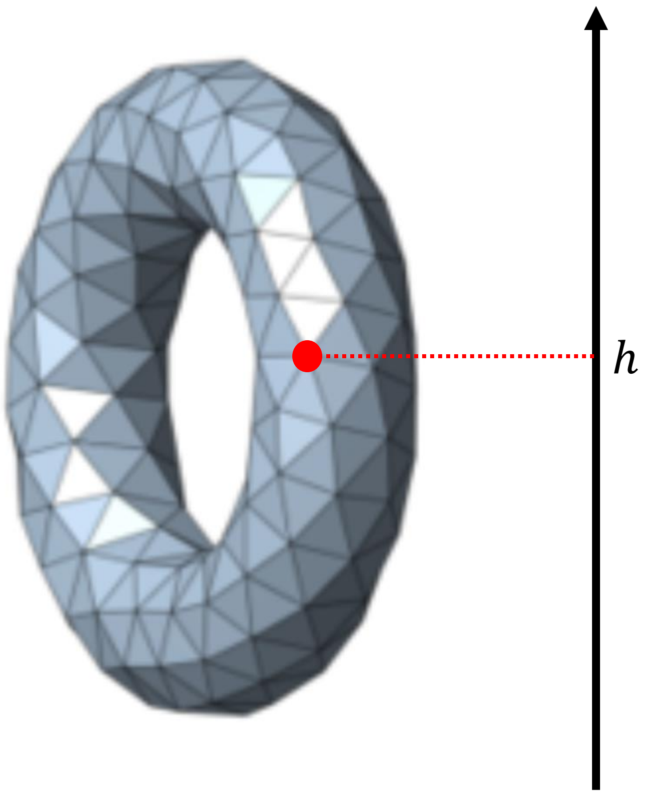
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

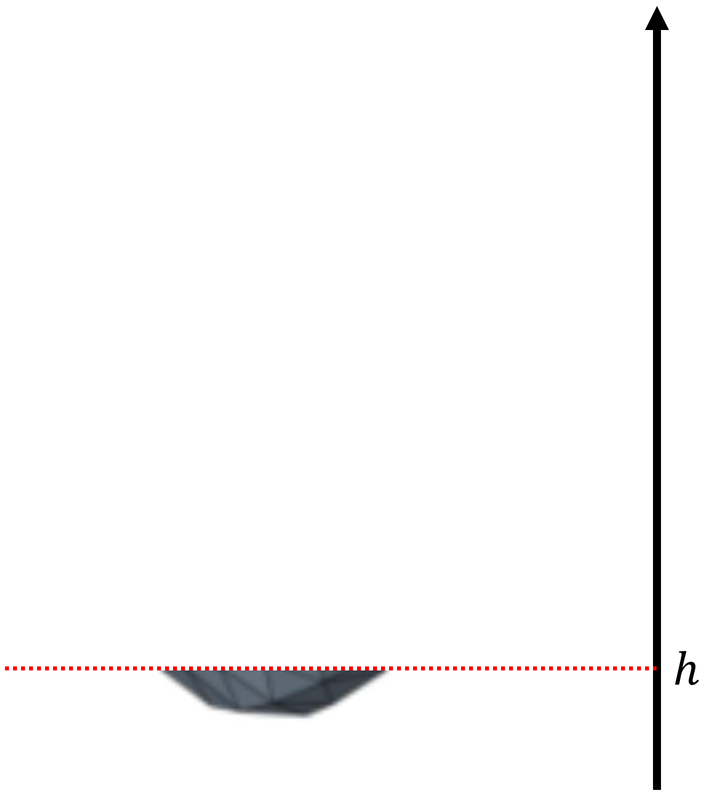
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

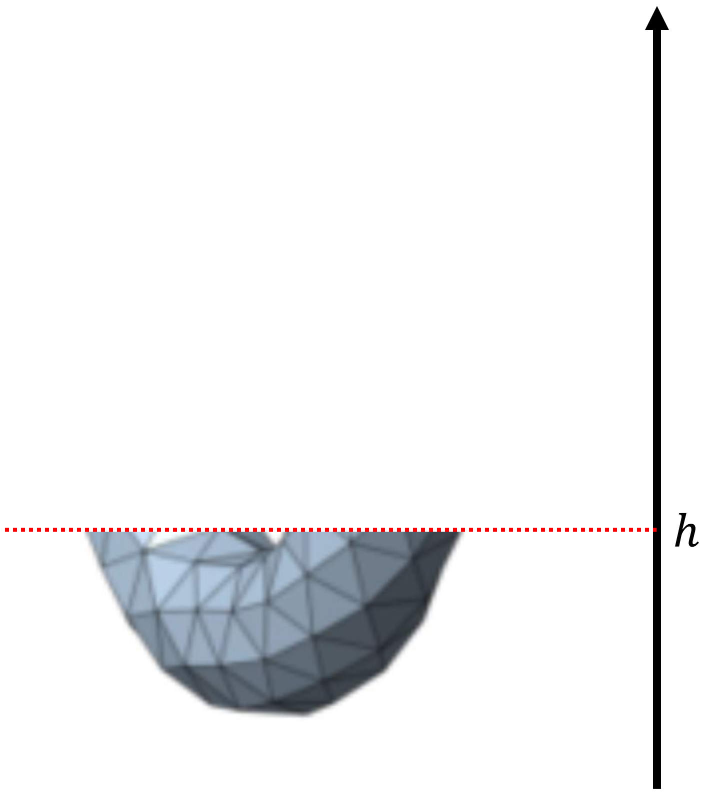
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

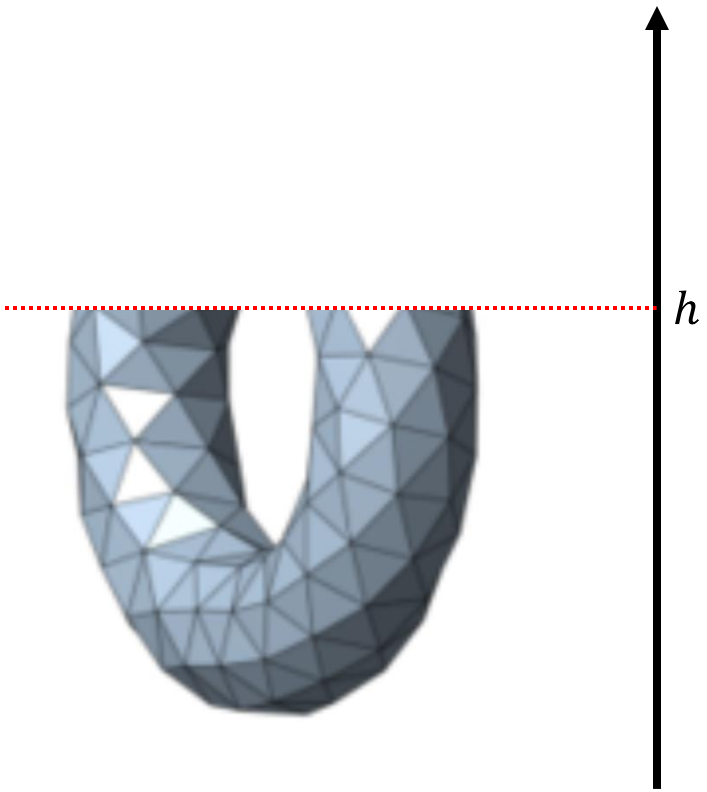
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

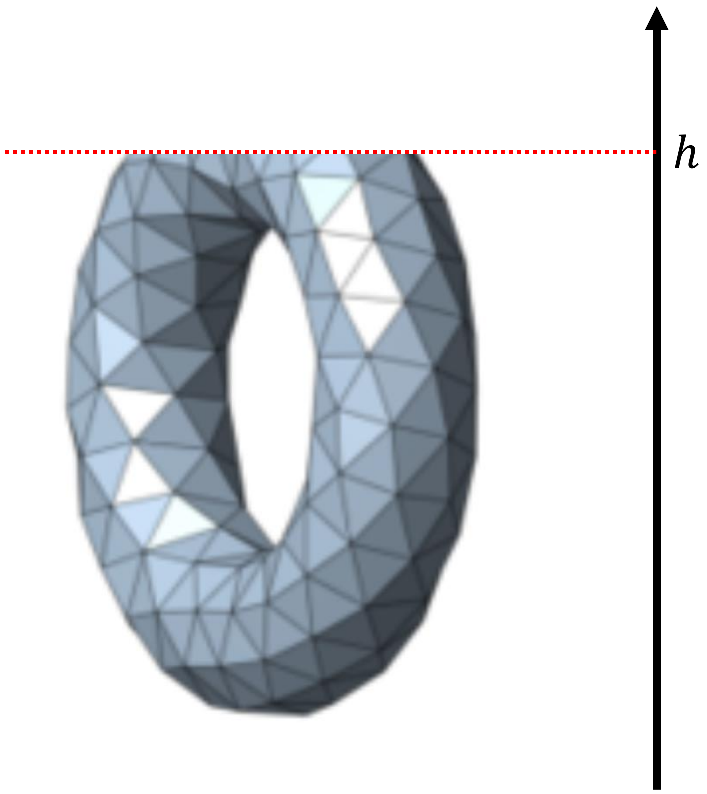
- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Triangular meshes

- For each vertex, we project the vertex to a certain direction and get its height value



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

More sublevelset filtrations

- Indeed we have also seen sublevelset filtrations in previous slides
- An interactive example: https://iuricichf.github.io/ICT/filtration.html

Superlevelset filtration

- There is a counterpart of sublevelset filtration called **super**levelset filtration
- A superlevelset of is the subset of X whose function values are **greater than or equal to** a value α, and we denote it as $f^{-1}[\alpha, \infty)$
- We then take all possible functions values and **descreasingly** sort them (i.e, start with the height value):

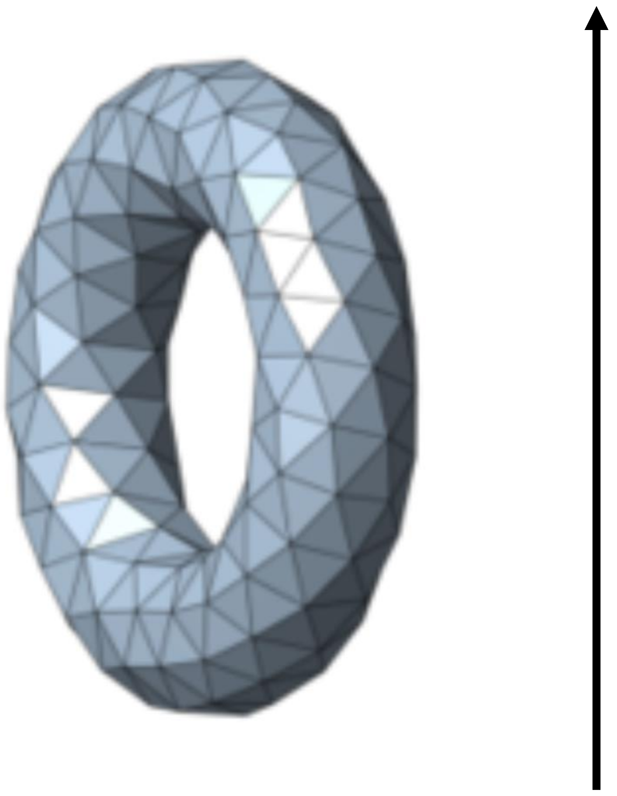
$$\alpha_0 > \alpha_1 > \cdots > \alpha_m$$

- The superlevelset filtration is then the superlevelsets over the above values:

$$f^{-1}[\alpha_0, \infty) \subseteq f^{-1}[\alpha_1, \infty) \subseteq \cdots \subseteq f^{-1}[\alpha_m, \infty)$$

Superlevelset filtration

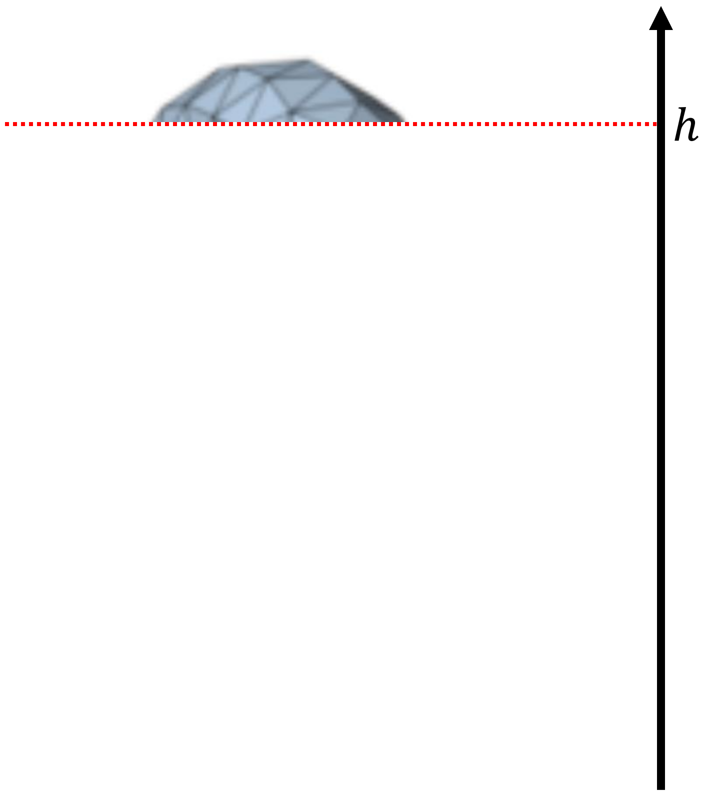
- The superlevelset filtration for the previous height function on torus would be:



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Superlevelset filtration

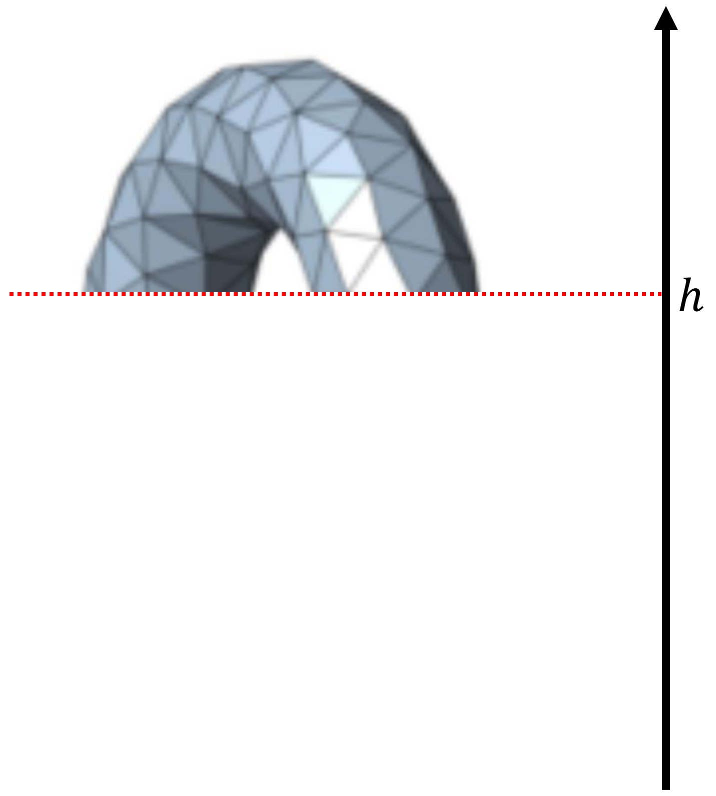
- The superlevelset filtration for the previous height function on torus would be:



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Superlevelset filtration

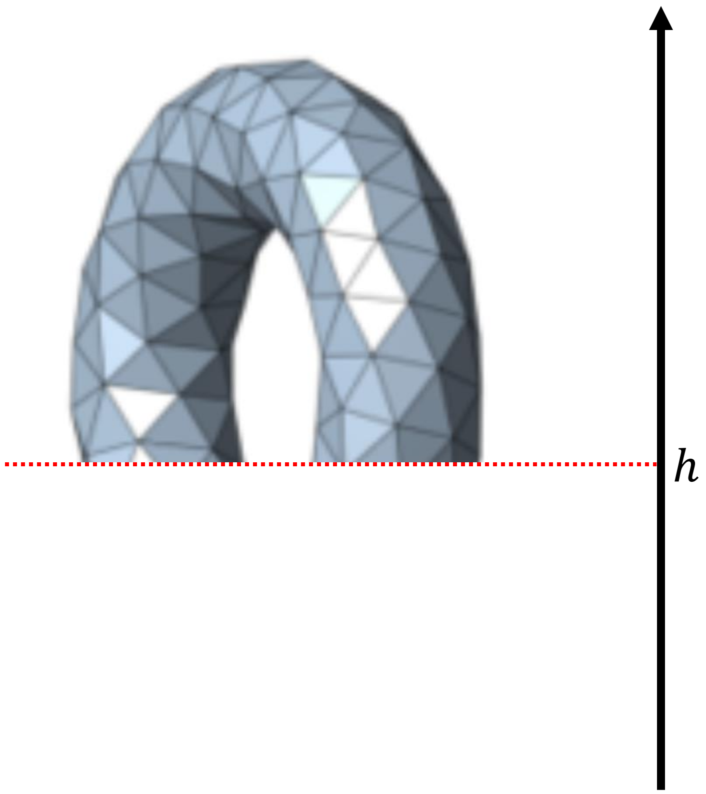
- The superlevelset filtration for the previous height function on torus would be:



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Superlevelset filtration

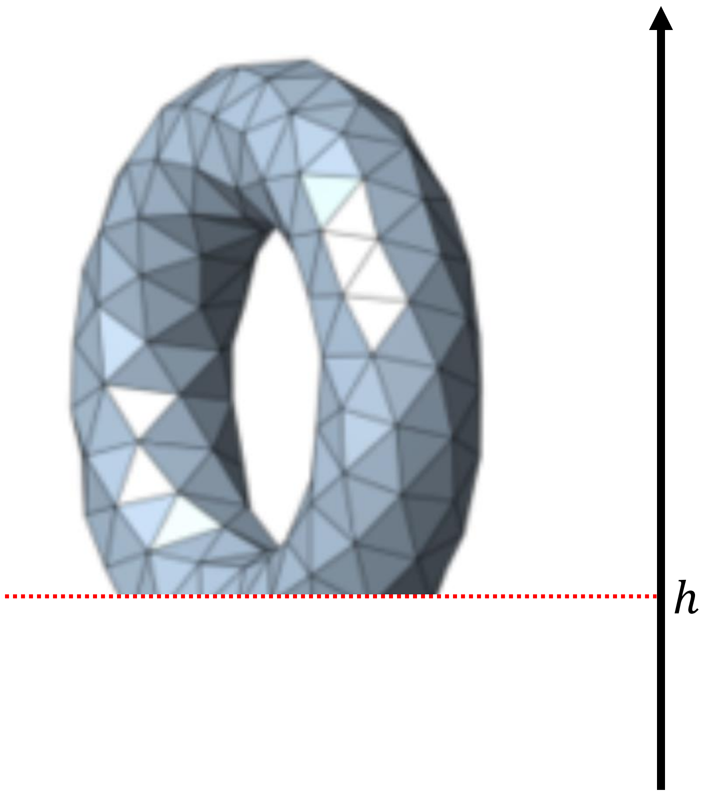
- The superlevelset filtration for the previous height function on torus would be:



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Superlevelset filtration

- The superlevelset filtration for the previous height function on torus would be:



Img source: Lockerby. Integration over discrete closed surfaces using the Method of Fundamental Solutions

Clarification on PD for different filtrations

- Previously when we define PD by computing it from a discrete filtration:

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

intervals in the PD are "integer intervals" (e.g., $[3, 6) = \{3, 4, 5\}$).

Clarification on PD for different filtrations

- Previously when we define PD by computing it from a discrete filtration:

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

intervals in the PD are "integer intervals" (e.g., $[3, 6) = \{3, 4, 5\}$).

- This applies when it is not clear where the discrete filtration is built from

Clarification on PD for different filtrations

- Previously when we define PD by computing it from a discrete filtration:

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

intervals in the PD are "integer intervals" (e.g., $[3,6) = \{3,4,5\}$).

- This applies when it is not clear where the discrete filtration is built from
- In practice, filtrations are built from different types of data. Each complex in the discrete filtration is associate with a real value (or a bunch of them)

Clarification on PD for different filtrations

- Previously when we define PD by computing it from a discrete filtration:

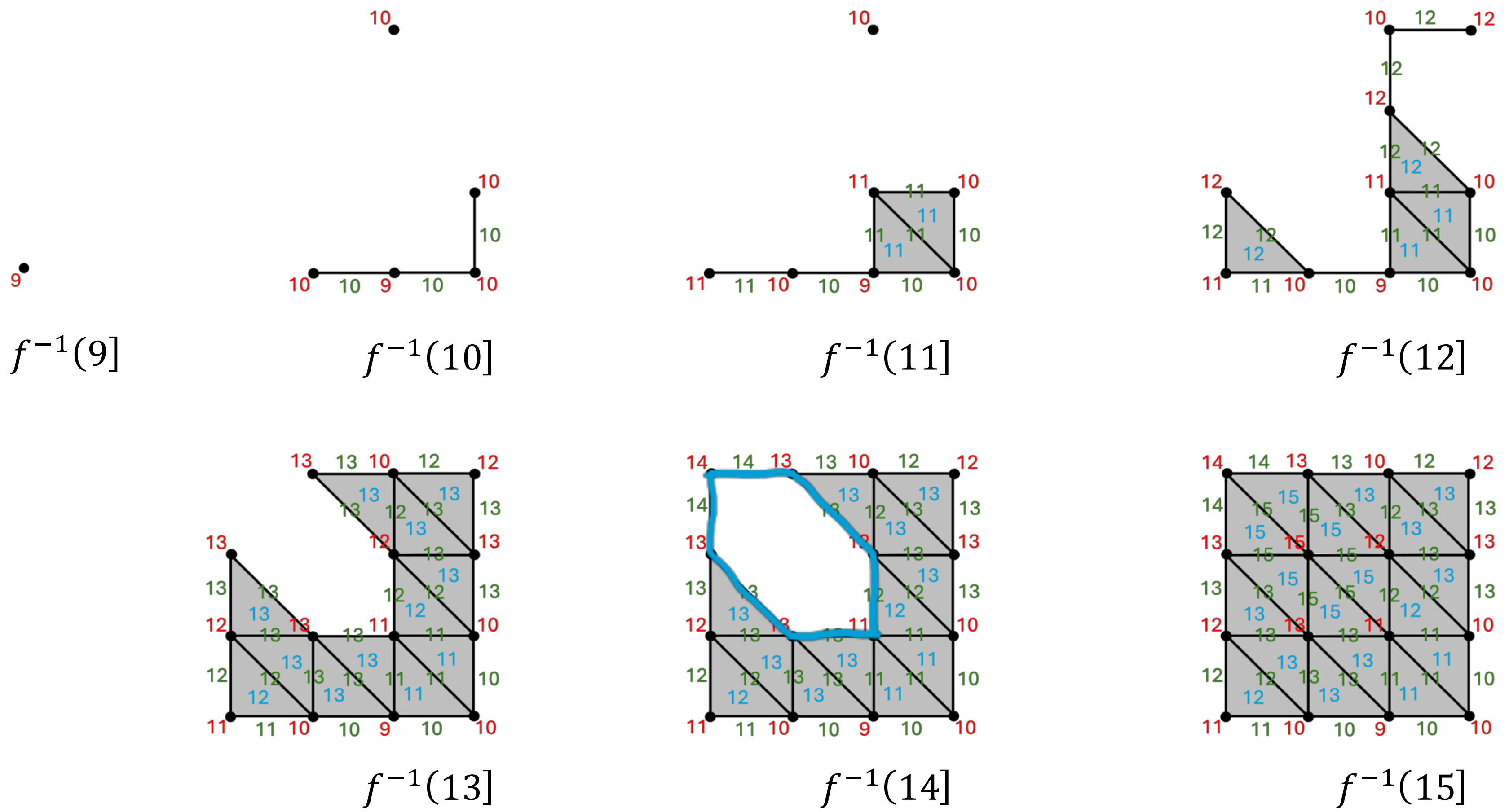
$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

intervals in the PD are "integer intervals" (e.g., $[3,6) = \{3,4,5\}$).

- This applies when it is not clear where the discrete filtration is built from
- In practice, filtrations are built from different types of data. Each complex in the discrete filtration is associate with a real value (or a bunch of them)
- Intervals in the PD for such a filtration (when we know source data) is then continuous intervals of real values (e.g., $[3.52, 6.37)$)

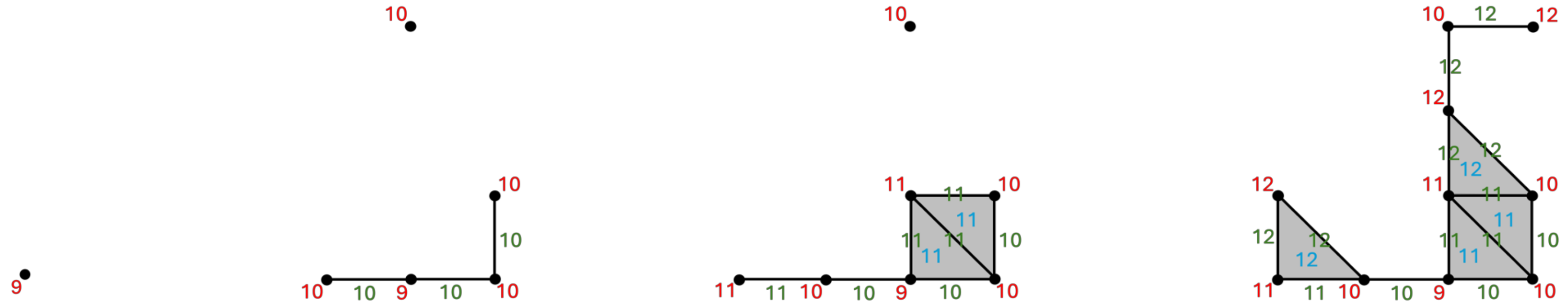
Clarification on PD for different filtrations

- For the previous sublevelset filtration for image, we can also number each complex in the filtration from 0 to 6

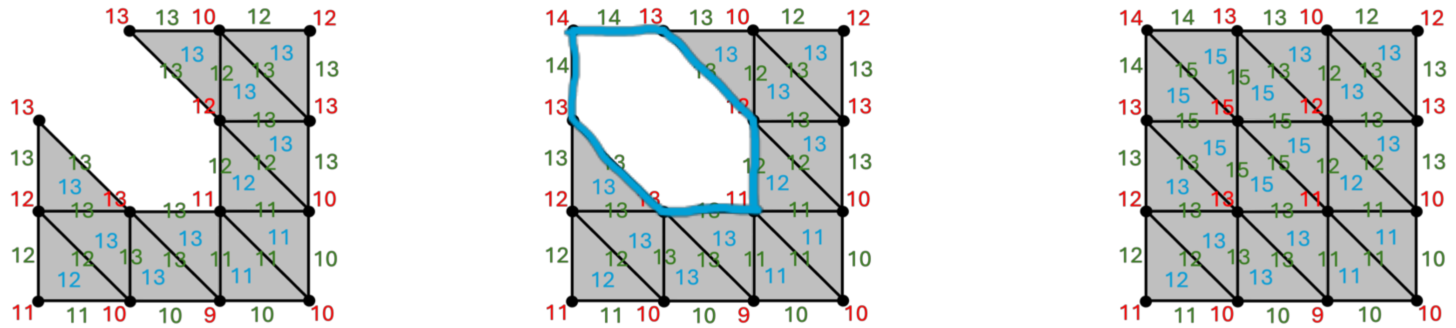


Clarification on PD for different filtrations

- For the previous sublevelset filtration for image, we can also number each complex in the filtration from 0 to 6



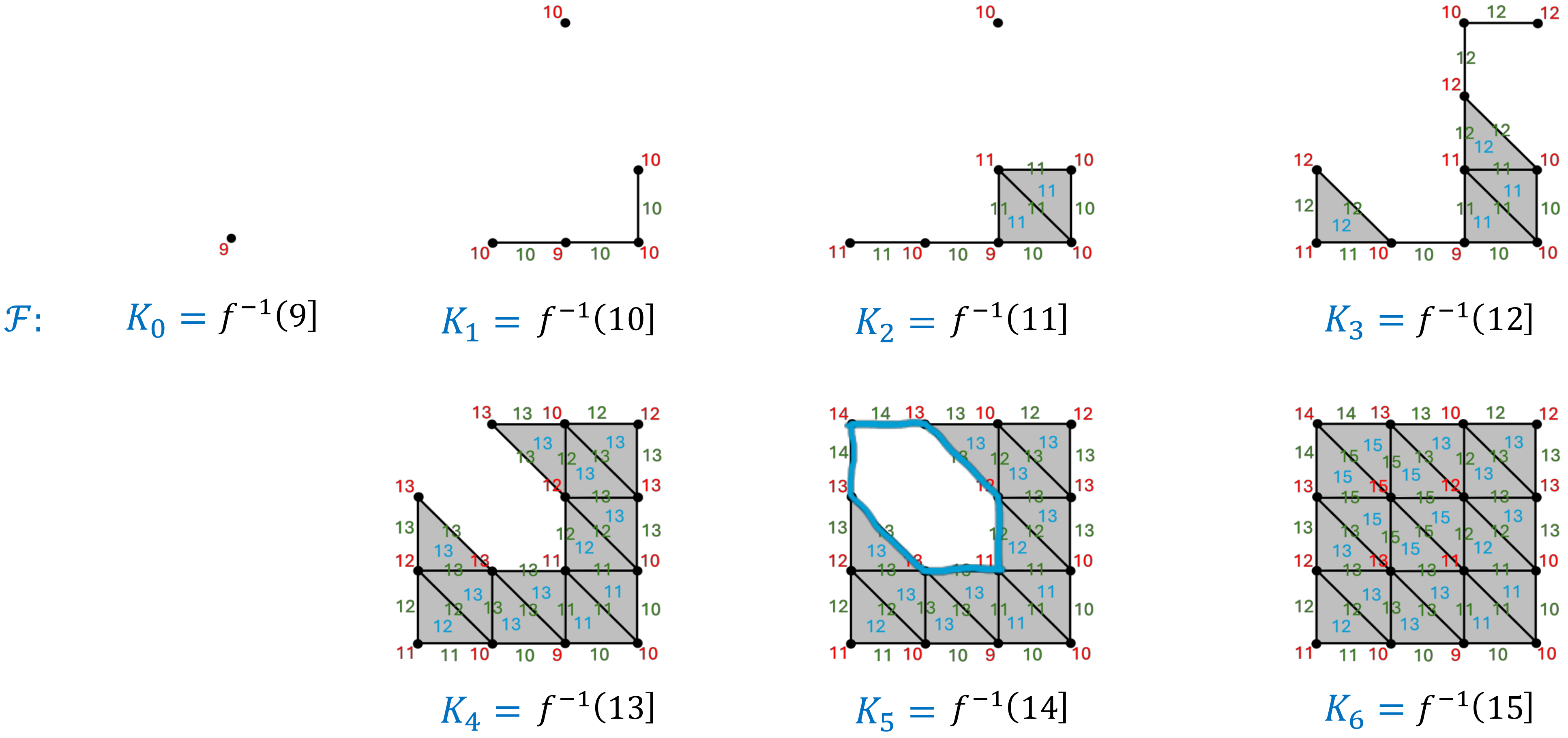
$\mathcal{F}$: $K_0 = f^{-1}(9]$ $K_1 = f^{-1}(10]$ $K_2 = f^{-1}(11]$ $K_3 = f^{-1}(12]$



$K_4 = f^{-1}(13]$ $K_5 = f^{-1}(14]$ $K_6 = f^{-1}(15]$

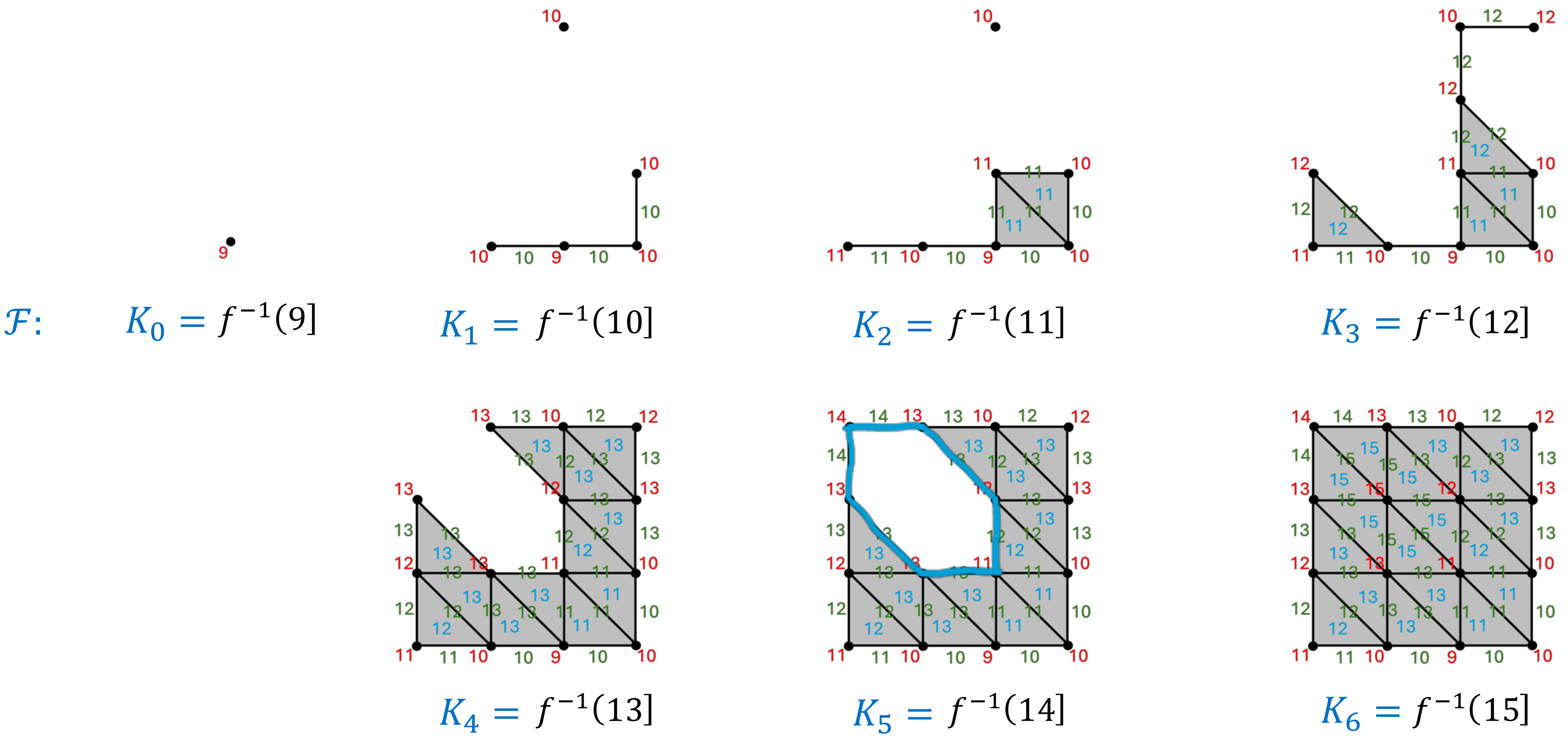
Clarification on PD for different filtrations

- For the previous sublevelset filtration for image, we can also number each complex in the filtration from 0 to 6
- But we use the pixel values (e.g., 9, 10, …) instead of the integer indices (e.g., 0, 1, …) for the PD



Clarification on PD for different filtrations

- E.g., the below 1d interval is $[14,15)$ rather than $[5,6)$



Clarification on PD for different filtrations

- The below 0d interval is $[10,12)$ rather than $[1,3)$