Computation Topology - Shape Detection

Eske Bockelmann, Elliot Morin

December 17, 2025

Algorithm 1 rips_complex(S,r)

```
Require: Point cloud S, radius r
Ensure: Rips complex RC
  n ← length(S)
  RC ← [(i, ) for i ∈ 1 : n]
  for i ∈ 1 : n do
      for j ∈ (i + 1) : n do
          if ‖S[i] − S[j]‖ ≤ r then
              RC ← RC ∪ {(i, j)}
          end if
      end for
  end for
  dim ← 2
  L ← All simplices in RC of length == dim
  while L ≠ ∅ do
      dim ← dim + 1
      new_simplices ← all possible dim combinations of simplices inL
      for sx in new_simplices do
          sx ← {vertices of sx}
      end for
      new_simplices ← Only keep those simplices which have exactly dim different vertices
      RC ← RC ∪ new_simplices
      L ← All simplices in RC of length == dim
  end while
      return RC
```

Algorithm 2 rips_filtration(S,R)

```
Require: Point cloud S, list of different radii R
Ensure: Rips filtration RF
  for r ∈ R do
      RF[r] => rips_complex(S, r)
  end for
      return RF
```

Algorithm 3 minimal_enclosing_ball(S) = MEB(S)

Require: Point cloud S
Ensure: radius and origin of minimal enclosing ball r, origin
  $n \leftarrow length(S)$
  if n == 2 **then**
    $r \leftarrow \|S[1] - S[2]\|/2$
    $\text{origin} = S[1] + \frac{1}{2} * (S[2] - S[1])$
    return r, origin
  else
    for $i \in 1 : n$ **do**
      $r, \text{origin} \leftarrow minimal\_enclosing\_ball(S \setminus S[i])$
      if $\|S[i] - \text{origin}\| \leq r$ **then** ▷ S[i] in the MEB($S \setminus S[i]$)
        return r, origin
      end if
    end for
    ▷ Otherwise the MEB is the ball through all vertices
    $d \leftarrow length(S[1])$

$$A = 2 * \begin{pmatrix} (S[2] - S[1])^T \\ \vdots \\ (S[n] - S[1])^T \end{pmatrix} \in \mathbb{R}^{n-1,d}$$

$$b = \begin{pmatrix} \|S[2]\|^2 - \|S[1]\|^2 \\ \vdots \\ \|S[n]\|^2 - \|S[1]\|^2 \end{pmatrix} \in \mathbb{R}^{n-1}$$

    $\text{origin} = A \setminus b$
    $r = \|\text{origin } - S[1]\|$
    return r, origin
  end if

Algorithm 4 cech_complex(S,r)

Require: Point cloud S, radius r
Ensure: Cech Complex CC

```
n ← length(S)
CC ← [(i, ) for i ∈ 1 : n]
for i ∈ 1 : n do
    for j ∈ (i + 1) : n do
        if ‖S[i] − S[j]‖ ≤ r then
            CC ← CC ∪ {(i, j)}
        end if
    end for
end for
dim ← 2
L ← All simplices in RC of length == dim
while L ≠ ∅ do
    dim ← dim + 1
    new_simplices ← all possible dim combinations of simplices inL
    for sx in new_simplices do
        sx ← {vertices of sx}
    end for
    new_simplices ← Only keep those simplices which have exactly dim different vertices
    new_simplices ← Only keep those simplices where the radius of MEB(vertices)  ≤ r
    CC ← CC ∪ new_simplices
    L ← All simplices in RC of length == dim
end while
return CC
```

Algorithm 5 cech_filtration(S,R)

Require: Point cloud S, list of different radii R
Ensure: Cech filtration CF

```
for r ∈ R do
    CF[r] => cech_complex(S, r)
end for
    return CF
```

```julia
include("graphcomponents.jl")
# example 1 from excercise
V = [1,2,3,4,5,6,7,8]
E = [(1,2),(2,3),(1,3),(4,5),(5,6),(5,7),(6,7),(7,8)]
# example to write some julia code in latex with nice highlighting
```