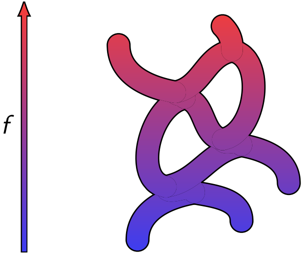
Sequential algorithm

Informal description

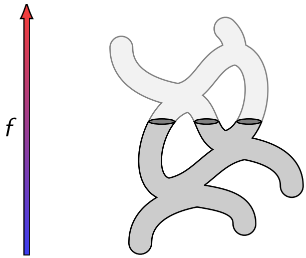
- Process the vertices of the mesh by **increasing** value of f.



Sequential algorithm

Informal description

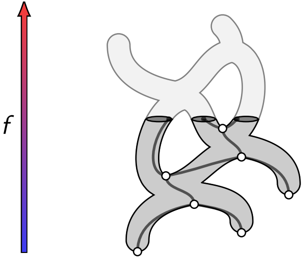
- Process the vertices of the mesh by **increasing** value of f.
- Construct the Reeb graph $\mathcal{R}(f)$ incrementally.



Sequential algorithm

Informal description

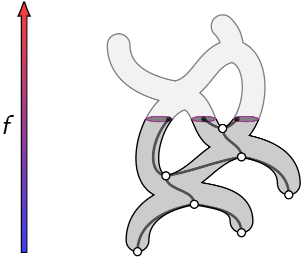
- Process the vertices of the mesh by **increasing** value of f.
- Construct the Reeb graph $\mathcal{R}(f)$ incrementally.
- While sweeping upwards, keep:
  - the **partial Reeb graph** constructed so far;



Sequential algorithm

Informal description

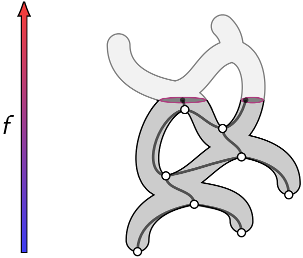
- Process the vertices of the mesh by **increasing** value of f.
- Construct the Reeb graph $\mathcal{R}(f)$ incrementally.
- While sweeping upwards, keep:
  - the **partial Reeb graph** constructed so far;
  - the current **level set** $f^{-1}(r)$.



Sequential algorithm

Informal description

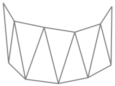
- Process the vertices of the mesh by **increasing** value of f.
- Construct the Reeb graph $\mathcal{R}(f)$ incrementally.
- While sweeping upwards, keep:
  - the **partial Reeb graph** constructed so far;
  - the current **level set** $f^{-1}(r)$.
- When processing a vertex, **update** the level set and the Reeb graph accordingly.



Sequential algorithm

The preimage graph

The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

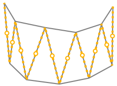


Sequential algorithm

The preimage graph

The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;

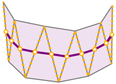


Sequential algorithm

The preimage graph

The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



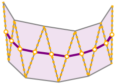
a triangle connects its two sides intersecting $f^{-1}(r)$

Sequential algorithm

The preimage graph

The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



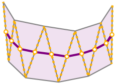
Updating G_r

Sequential algorithm

The preimage graph

The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.

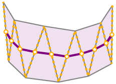
  from $r = f(v) - \epsilon$ to $r = f(v) + \epsilon$

Sequential algorithm

The preimage graph

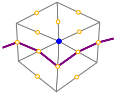
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of $\mathrm{Star}(v)$ separately.

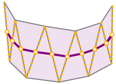


Sequential algorithm

The preimage graph

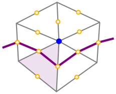
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
    1. v is the lower vertex of $\mathcal{T}$.

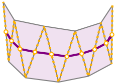


Sequential algorithm

The preimage graph

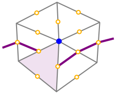
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.

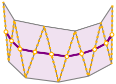


Sequential algorithm

The preimage graph

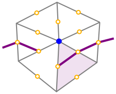
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.

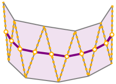


Sequential algorithm

The preimage graph

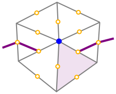
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

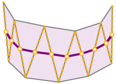
- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.



Sequential algorithm

The preimage graph

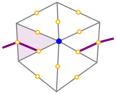
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:



- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.

Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.

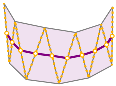


Sequential algorithm

The preimage graph

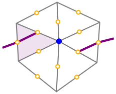
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.

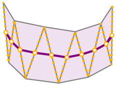


Sequential algorithm

The preimage graph

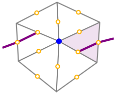
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.

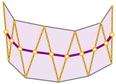


Sequential algorithm

The preimage graph

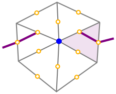
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.

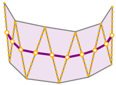


Sequential algorithm

The preimage graph

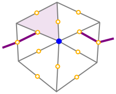
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

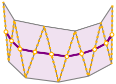


Sequential algorithm

The preimage graph

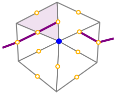
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

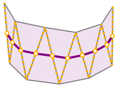


Sequential algorithm

The preimage graph

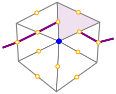
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

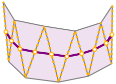


Sequential algorithm

The preimage graph

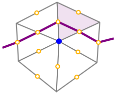
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

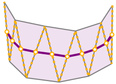
- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.



Sequential algorithm

The preimage graph

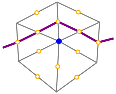
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:



- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.

Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.



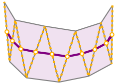
- **Data structure**: the following operations are required;

Sequential algorithm

The preimage graph

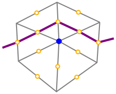
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

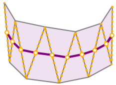


- **Data structure**: the following operations are required;
  - `find` the connected component of a node e;

Sequential algorithm

The preimage graph

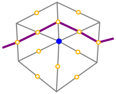
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:



- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.

Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.



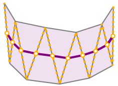
- **Data structure**: the following operations are required;
  - `find` the connected component of a node e;
  - `insert` a new edge between nodes e_1, e_2;

Sequential algorithm

The preimage graph

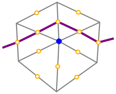
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

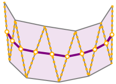


- **Data structure**: the following operations are required;
  - `find` the connected component of a node e;
  - `insert` a new edge between nodes e_1, e_2;
  - `delete` the edge between nodes e_1, e_2;

Sequential algorithm

The preimage graph

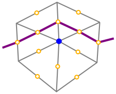
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:



- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.

Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.

  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.

- **Data structure**: the following operations are required;
  - `find` the connected component of a node e;
  - `insert` a new edge between nodes e_1, e_2;
  - `delete` the edge between nodes e_1, e_2;

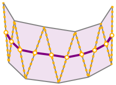
$\rightsquigarrow$ offline dynamic connectivity problem

Sequential algorithm

The preimage graph

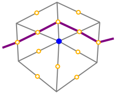
The level set $f^{-1}(r)$ can be represented by an abstract **graph** G_r:

- **nodes** $\rightsquigarrow$ edges of the mesh $\mathcal{M}$;
- **edges** $\rightsquigarrow$ triangles of $\mathcal{M}$ intersecting $f^{-1}(r)$.



Updating G_r

- **Trigger**: update when processing a vertex v.
- **Action**: process each triangle $\mathcal{T}$ of Star(v) separately.
  1. v is the lower vertex of $\mathcal{T}$.
  2. v is the middle vertex of $\mathcal{T}$.
  3. v is the upper vertex of $\mathcal{T}$.



- **Data structure**: the following operations are required;
  - `find` the connected component of a node e;
  - `insert` a new edge between nodes e_1, e_2;
  - `delete` the edge between nodes e_1, e_2;

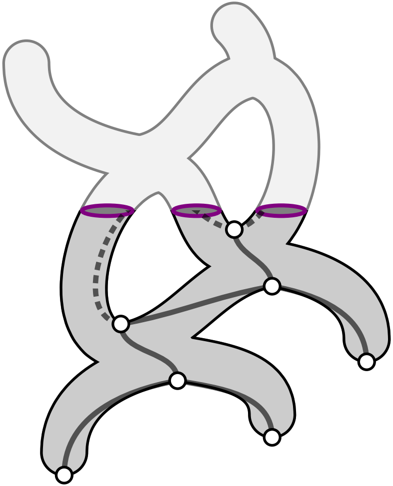
$\rightsquigarrow$ offline dynamic connectivity problem $\rightsquigarrow$ **ST-trees**

support all the operations in $O(\log m)$

Sequential algorithm

The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

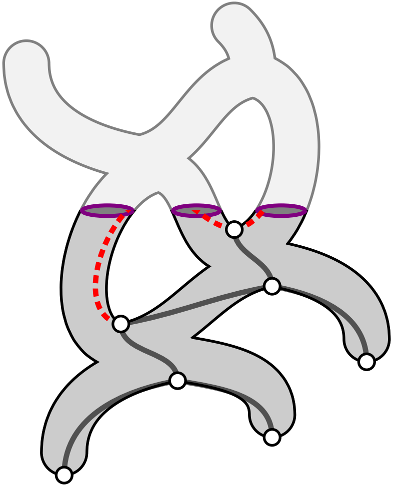


Sequential algorithm

The augmented Reeb graph

Reeb graph constructed so far;
has one open edge for each component of G_r

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

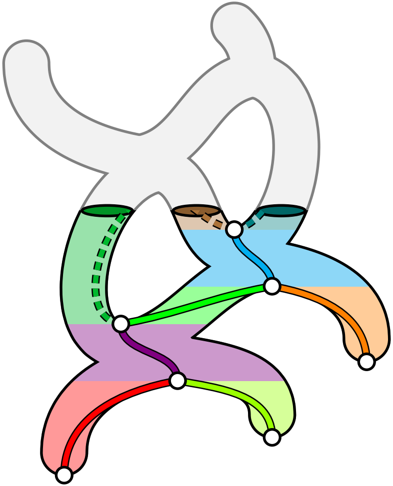


Sequential algorithm

The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

partial segmentation map;
maps each vertex of the mesh to a node or an edge of $\mathcal{R}$



Sequential algorithm

The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

Sequential algorithm

The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** $\mathrm{Lc} = \left\{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^-(v) \right\}$.

lower components

Sequential algorithm

The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^{-}(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^{+}(v) \right\}$.

upper components

Sequential algorithm

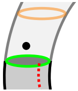
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^{-}(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^{+}(v) \right\}$.
3. **If** $|\mathrm{Lc}| = |\mathrm{Uc}| = 1$ **then**:



Sequential algorithm

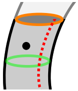
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^{-}(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^{+}(v) \right\}$.
3. **If** $|\mathrm{Lc}| = |\mathrm{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;



Sequential algorithm

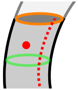
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^-(v) \}$.
2. **Let** Uc $= \{ G_{f(v)+\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^+(v) \}$.
3. **If** $|\text{Lc}| = |\text{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



Sequential algorithm

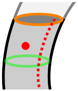
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

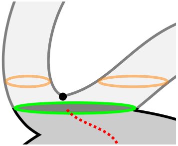
Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathsf{Link}^{-}(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\mathtt{find}([vv']) : v' \in \mathsf{Link}^{+}(v) \right\}$.
3. **If** $|$Lc$| = |$Uc$| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



4. **Otherwise**:



Sequential algorithm

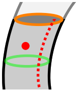
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

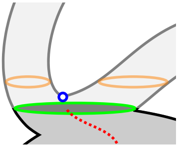
Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^-(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\texttt{find}([vv']) : v' \in \mathrm{Link}^+(v) \right\}$.
3. **If** $|\mathrm{Lc}| = |\mathrm{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



4. **Otherwise**:
   - create a new node w in $\mathcal{R}$;



Sequential algorithm

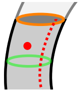
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

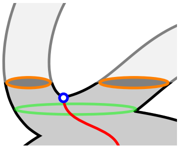
Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** $\mathrm{Lc} = \left\{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^-(v) \right\}$.
2. **Let** $\mathrm{Uc} = \left\{ G_{f(v)+\epsilon}.\mathtt{find}([vv']) : v' \in \mathrm{Link}^+(v) \right\}$.
3. **If** $|\mathrm{Lc}| = |\mathrm{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



4. **Otherwise**:
   - create a new node w in $\mathcal{R}$;
   - all the open edges associated to the lower components end at w;



Sequential algorithm

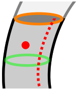
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

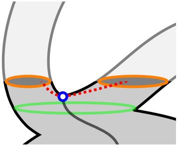
Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** Lc $= \left\{ G_{f(v)-\epsilon}.\mathtt{find}([vv']) : v' \in \mathsf{Link}^-(v) \right\}$.
2. **Let** Uc $= \left\{ G_{f(v)+\epsilon}.\mathtt{find}([vv']) : v' \in \mathsf{Link}^+(v) \right\}$.
3. **If** $|\text{Lc}| = |\text{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



4. **Otherwise**:
   - create a new node w in $\mathcal{R}$;
   - all the open edges associated to the lower components end at w;
   - open a new edge in $\mathcal{R}$ starting at w for each upper component.



Sequential algorithm

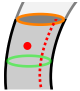
The augmented Reeb graph

The partial augmented Reeb graph is represented by a pair $(\mathcal{R}, \Phi)$.

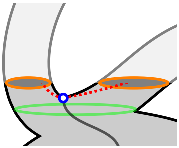
Updating $(\mathcal{R}, \Phi)$

When processing a vertex v:

1. **Let** $\text{Lc} = \left\{ G_{f(v)-\epsilon}.\texttt{find}([vv']) : v' \in \text{Link}^{-}(v) \right\}$.
2. **Let** $\text{Uc} = \left\{ G_{f(v)+\epsilon}.\texttt{find}([vv']) : v' \in \text{Link}^{+}(v) \right\}$.
3. **If** $|\text{Lc}| = |\text{Uc}| = 1$ **then**:
   - $\mathcal{R}$ is unchanged;
   - $\Phi(v) =$ the open edge associated to the lower component.



4. **Otherwise**:
   - create a new node w in $\mathcal{R}$;
   - all the open edges associated to the lower components end at w;
   - open a new edge in $\mathcal{R}$ starting at w for each upper component.
   - $\Phi(v) = w$.



Sequential algorithm

Full implementation

input : a triangulated mesh $\mathcal{M}$
a scalar field f on $\mathcal{M}$
output : the augmented Reeb graph $(\mathcal{R}, \Phi)$

begin
  $\mathcal{R}$, $\Phi \leftarrow \varnothing$ [graph], $\varnothing$ [function]
  $G_r \leftarrow \varnothing$ [ST-tree]
  sort the vertices of $\mathcal{M}$ by increasing value of f
  foreach v *vertex of* $\mathcal{M}$ **do**
    Lc $\leftarrow$ `GetLowerComponents(`v`)`
    `UpdatePreimageGraph()`
    Uc $\leftarrow$ `GetUpperComponents(`v`)`
    if $|\,\mathrm{Lc}\,| = |\,\mathrm{Uc}\,| = 1$ **then** update $\Phi(v)$
    else `UpdateReebGraph(`v, Lc, Uc`)`
  end
  return $(\mathcal{R}, \Phi)$
end