Vertex models on a mesh

Yann-Edwin Keta
(Dated: 22/02/2024, 14:26)

I. DEFINITION OF THE "CELL" TISSUE

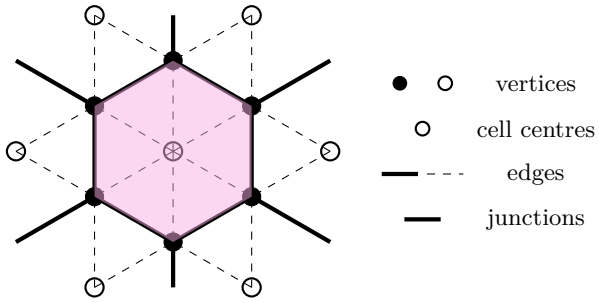

FIG. 1. Schematic of a cell (highlighted in pink) in the vertex model.

A *cell centre* is enclosed by *cell corners* (or *vertices*). These are linked between themselves by *junctions*. We will assume (i) that cells are always convex, (ii) that the mesh remains planar, and (iii) that no edge joins two cell centres. The ensemble of the vertices and the edges that link them constitutes the *geometric mesh*. The specification of the cell centres and the junctions between non-cell-centres defines the *physical mesh*.

II. HALF-EDGE CONSTRUCTION

The edges of the mesh (Fig. 1) divide the entire system in adjacent and non-overlapping *triangles* (or *faces*). We endow all of these triangles with three arrows. These are oriented such that, for an arbitrary vertex (*e.g.*, μ in Fig. 2) and an arbitrary triangle (*e.g.*, (μ, ν, i)), the cross product of the half-edges pointing toward this vertex in this triangle (*i.e.*, $i \to \mu$) on the one hand and the half-edge pointing out of this vertex in this triangle (*i.e.*, $\mu \to \nu$) on the other hand, has a positive scalar product with $\hat{\boldsymbol{e}}_z$. We introduce three relations between half-edges, which we exemplify in Fig. 2 using an arbitrary reference half-egde $\mu \to \nu$.

1. The *next* half-edge departs from the vertex pointed at by the reference half-edge (*i.e.*, ν here) inside the same triangle. "Next" is an order-3 bijection, meaning that the next half-edge of the next half-edge of any half-edge

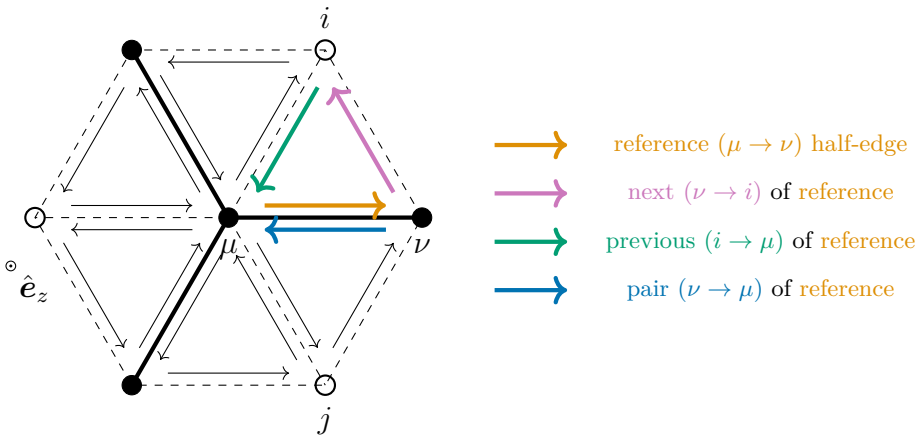

FIG. 2. Half-edge construction in 6 adjacent triangles, highlighting an arbitrary reference half-edge and its associated half-edges.

is this same half-edge.

2. The *previous* half-edge (or *before*) points towards the vertex from which departs the reference half-edge (*i.e.*, μ here) inside the same triangle. "Previous" is also an order-3 bijection.

3. The *pair* half-edge has inverse depart and arrival vertices with respect to the reference half-edge (*i.e.*, ν and μ here) in the adjacent triangle sharing the same edge the reference half-edge belongs to. "Pair" is an order-2 bijecton, meaning that the pair half-edge of any pair half-edge is this same half-edge.

We refer to the ensemble of half-edges and vertices as the *half-edge construction*. We propose in Alg. 3 a way to check its consistency. Compared to the sole definition of vertices and (unoriented) edges, this construction allows for faster and in some cases easier computations. (Such information redundancy comes at the cost of memory usage.) We show with Alg. 4 how to identify all neighbours of any vertex given that each vertex contains the information of (at least) one half-edge leaving the vertex.

Computationally, we define two elementary classes VERTEX and HALFEDGE which are collected in a class MESH (see MESH.HPP). Physical properties are described through an attribute TYPE of these elementary classes. This attribute does not interfere with basic operations on the mesh, in this way we separate the geometrical construction from the physical construction.

III. T1 TRANSITION

T1 transitions correspond to changes of neighbours of 4 cells resulting in a modification of the mesh topology, see Fig. 3. First (1) two vertices are merged (*i.e.*, a junction is deleted) creating a *four-vertex*. In principle, if one of the vertices of the deleted junction had initially more than 3 neighbouring cells, there would be more than 4 cells at the merged vertex. While we expect this not to be frequent, the algorithm designed to perform this merge has to take this into account. Then (2) a junction is created (*i.e.*, a vertex is created).

It is noteworthy that states (A) and (C) (Fig. 3) are described within the half-edge construction with the same number of half-edges and vertices. Therefore it is possible to implement a direct change from (A) to (C) through a clever relabelling of objects. We propose here to implement the two steps (1) and (2) separately in order to enable also the stabilisation of higher order vertices.

Merging two vertices amounts, in the half-edge construction, to deleting 2 triangles (*i.e.*, 2 sets of 3 half-edges) among which one edge as well as one vertex. We represent this operation in Fig. 4, where the edge to delete is $[\mu, \nu]$ (μ is merged into ν) and the triangles to delete are shaded. Relations between half-edges, as well as the knowledge for each vertex of a half-edge going out of itself, imply that we need to relabel and reaffect half-edges and vertices before deleting anything in order for the construction to remain consistent at the end of the operation. We describe the procedure in Alg. 1. Note that this procedure does not rely on physical properties of the edges, and is thus purely geometrical in the mesh sense.

Creating an edge involves the same steps as deleting an edge but in reverse order, *i.e.* first create a vertex and 6 half-edges, then relabelling and affecting half-edges and vertices in the system. To identify unequivocally the topology around the new edge in the half-edge construction, it is necessary to specify two distinct half-edges going out of a same vertex. We represent this operation in Fig. 5, where vertex ξ is created from vertex ν and half-edges $\nu \to k$ and $\nu \to l$.

We can now use these algorithms (to delete an edge and create a new one) in order to perform the T1 transition of Fig. 3 (see Alg. 2). We discard the stabilisation of four-fold vertices [**?**] then, apart from the junction to delete,

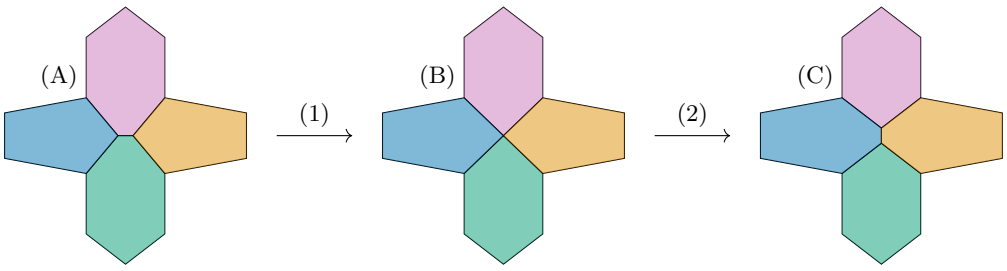

FIG. 3. Four cells undergoing a T1 transition. (A) Cells at the top and at the bottom are in contact while cells on the left and on the right are disjointed. (B) All four cells are touching, at the centre is a four-vertex. (C) Cells at the top and at the bottom are disjointed while cells on the left and on the right are touching.

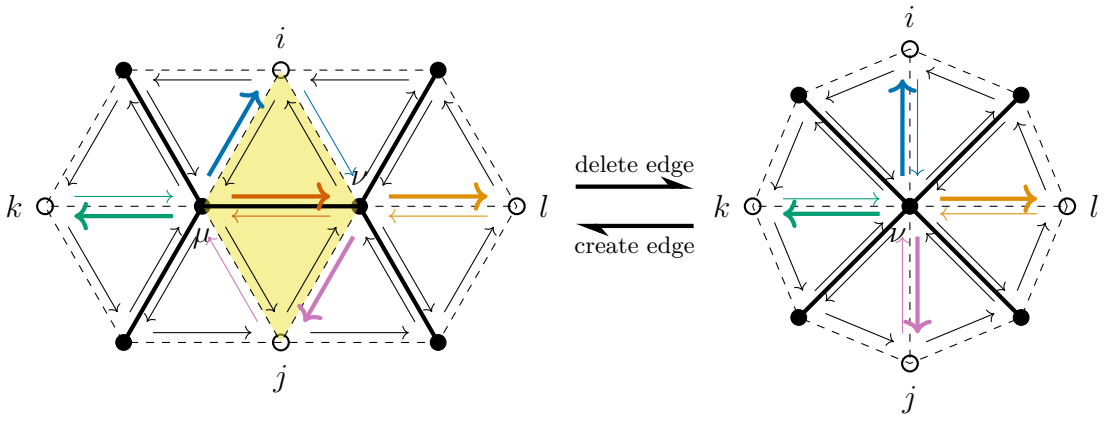

FIG. 4. Representation of edge deletion and creation in the half-edge construction. (See Fig. 5 for representation in the orthogonal direction.)

ALG. 1. Delete an edge (see MESH::DELETEEDGE in MESH.CPP).

Input: Half-edge $\boldsymbol{h}$ belonging to the edge to delete. ▷ *Thick red half-edge in Fig. 4.*
Output: Deleted vertex and half-edges.

Set μ, ν = origin and destination vertices of $\boldsymbol{h}$.
// Relabel half-edge origins and destinations for all half-edges connected to μ.
Set $\boldsymbol{b}$ = previous half-edge of $\boldsymbol{h}$.
Set $\boldsymbol{b}^0$ = pair half-edge of $\boldsymbol{h}$.
while TRUE **do** ▷ *See break condition on line 9.*
  Set destination vertex of $\boldsymbol{b} = \nu$ (instead of μ).
  Set $\boldsymbol{p}$ = pair half-edge of $\boldsymbol{b}$.
  Set origin vertex of $\boldsymbol{p} = \nu$ (instead of μ).
  if $\boldsymbol{b} = \boldsymbol{b}^0$ **then** ▷ *All half-edges going out of μ were found, we should also then have $\boldsymbol{p} = \boldsymbol{h}$.*
    break
  end if
  Set $\boldsymbol{b}$ = previous half-edge of $\boldsymbol{p}$.
end while
// Reassign half-edges associated to vertices belonging to the deleted triangles.
Set $\boldsymbol{p}$ = pair half-edge of previous half-edge of $\boldsymbol{h}$. ▷ *Thick blue half-edge in Fig. 4.*
Associate $\boldsymbol{p}$ to ν as half-edge going out of it.
Set $\boldsymbol{p}$ = pair half-edge of next half-edge of $\boldsymbol{h}$. ▷ *Thin blue half-edge in Fig. 4.*
Set i = origin vertex of $\boldsymbol{p}$.
Associate $\boldsymbol{p}$ to i as half-edge goigng out of it.
Set $\boldsymbol{p}$ = pair half-edge of $\boldsymbol{h}$. ▷ *Thin red half-edge in Fig. 4.*
Set $\boldsymbol{p}$ = pair half-edge of next half-edge of $\boldsymbol{p}$. ▷ *Thin purple half-edge in Fig. 4.*
Set j = origin vertex of $\boldsymbol{p}$.
Associate $\boldsymbol{p}$ to j as half-edge goigng out of it.
// Relabel half-edge pairs.
Set $\boldsymbol{f}$ = pair half-edge of previous half-edge of $\boldsymbol{h}$. ▷ *Thick blue half-edge in Fig. 4.*
Set $\boldsymbol{t}$ = pair half-edge of next half-edge of $\boldsymbol{h}$. ▷ *Thin blue half-edge in Fig. 4.*
Associate $\boldsymbol{f}$ as pair half-edge of $\boldsymbol{t}$ and vice-versa.
Set $\boldsymbol{p}$ = pair half-edge of $\boldsymbol{h}$. ▷ *Thin red half-edge in Fig. 4.*
Set $\boldsymbol{f}$ = pair half-edge of previous half-edge of $\boldsymbol{p}$. ▷ *Thick purple half-edge in Fig. 4.*
Set $\boldsymbol{t}$ = pair half-edge of next half-edge of $\boldsymbol{p}$. ▷ *Thin purple half-edge in Fig. 4.*
Associate $\boldsymbol{f}$ as pair half-edge of $\boldsymbol{t}$ and vice-versa.
// At this point the vertex μ and the half-edges in the triangles adjacent at $\boldsymbol{h}$ are separated from the rest of the mesh.
Delete 6 half-edges in the 2 triangles adjacent at $\boldsymbol{h}$ and the vertex μ, and **return** their indices.

---

additional information needed to do this transition are the direction of new junction as well as its length. A possible choice is to create the junction orthogonal to the original junction.

ALG. 2. Perform T1 (see System::doT1 in system.cpp).

Input: Half-edge $\boldsymbol{h}$ belonging to the junction to delete, length ℓ of the new junction to create.

Set μ, ν = origin and destination vertices of $\boldsymbol{h}$.
// Identify half-edge to split to create new junction.
Set $\mathcal{N}_\mu, \mathcal{N}_\nu$ = neighbours of μ and ν. ▷ *Use Alg. 4.*
Set ensemble of half-edges $\Omega = \emptyset$.
for $\xi \in \mathcal{N}_\mu$ **do**
  if ξ is a cell centre and $\xi \notin \mathcal{N}_\nu$ **then**
    Add half-edge $\mu \to \xi$ to Ω.
  end if
end for
assert $\Omega \neq \emptyset$
Set $\boldsymbol{h}'$ = randomly picked element of Ω.
Set ensemble of half-edges $\Omega = \emptyset$.
for $\xi \in \mathcal{N}_\nu$ **do**
  if ξ is a cell centre and $\xi \notin \mathcal{N}_\mu$ **then**
    Add half-edge $\nu \to \xi$ to Ω.
  end if
end for
assert $\Omega \neq \emptyset$
Set $\boldsymbol{h}''$ = randomly picked element of Ω.
// Determine angle of new junction.
Set $\alpha = \arg(\boldsymbol{h}) + \pi/2$.
// Perform T1.
Delete edge to which $\boldsymbol{h}$ belongs. ▷ *Use Alg. 1. At this point $\boldsymbol{h}'$ and $\boldsymbol{h}''$ go out of the same vertex ν.*
Create an edge from $\boldsymbol{h}'$ and $\boldsymbol{h}''$ with additional vertex ξ. ▷ *Use Alg. 5. Index of vertex ξ is returned by the algorithm.*
Move vertices ν and ξ a distance ℓ apart in the direction given by α.

IV. OPEN BOUNDARIES

Open boundaries are treated as special vertices for which the orientation condition (Alg. 3, l. 13) is not checked. It is important that the half-edge construction remains well defined so that it is possible to move along the boundary. Finally, when performing T1s (Alg. 2), half-edges towards an open boundary can be considered as half-edges to be split ($\boldsymbol{h}'$ and $\boldsymbol{h}''$) even though the boundary is a shared neighbour of two vertices, the only important condition is that at least one of these two half-edges has to not go to the boundary.

Appendix A: Half-edge construction algorithms

ALG. 3. Check half-edge construction (see MESH::CHECKMESH in MESH.CPP).

Input: Ensemble $\mathcal{V}$ of all vertices, ensemble $\mathcal{H}$ of all half-edges.
Make copies $\mathcal{V}'$ and $\mathcal{H}'$ of $\mathcal{V}$ and $\mathcal{H}$.
for $\boldsymbol{h} \in \mathcal{H}$ **do**
  if $\boldsymbol{h} \notin \mathcal{H}'$ **then** ▷ *Half-edge* $\boldsymbol{h}$ *has already been checked.*
    continue
  end if
  Set $\mathcal{T} = \{\boldsymbol{h},$ next of $\boldsymbol{h},$ previous of $\boldsymbol{h}\}$. ▷ *Triangle to which* $\boldsymbol{h}$ *belongs.*
  for $\boldsymbol{h}' \in \mathcal{T}$ **do** ▷ *Loop over the half-edges in the triangle.*
    Set o and d as the origin and destination vertices of $\boldsymbol{h}'$.
    if $o \in \mathcal{V}'$ **then** ▷ *Origin vertex* o *has not been checked.*
      assert o knows one half-edge which *does* leave from o
      Remove o from $\mathcal{V}'$.
    end if
    assert $[\boldsymbol{h}' \times (\boldsymbol{h}' + 1)] \cdot \hat{\boldsymbol{e}}_z > 0$ ▷ *Check orientation,* $\boldsymbol{h}' + 1$ *is meant as next element in order in* $\mathcal{T}$.
    assert pair half-edge of $\boldsymbol{h}'$ has opposite origin and destination vertices ▷ *Check half-edge.*
    assert $\boldsymbol{h}'$ is the pair of half-edge of its pair half-edge
    assert $\boldsymbol{h}'$ is previous half-edge of $\boldsymbol{h}' + 1$. ▷ *Check next half-edge.*
    assert destination vertex of $\boldsymbol{h}'$ is the origin vertex of $\boldsymbol{h}' + 1$
    assert $\boldsymbol{h}'$ is next half-edge of $\boldsymbol{h}' - 1$. ▷ *Check previous half-edge.*
    assert origin vertex of $\boldsymbol{h}'$ is the destination vertex of $\boldsymbol{h}' - 1$
    Remove $\boldsymbol{h}'$ from $\mathcal{H}'$.
  end for
end for
assert $\mathcal{V}' = \emptyset$ and $\mathcal{H}' = \emptyset$

ALG. 4. Find all neighbours of an arbitrary vertex and all half-edges from this vertex to its neighbours. All vertices (cell centre or not) are denoted with latin indices (see MESH::GETNEIGHBOURVERTICES in MESH.CPP).

Input: Arbitrary half-edge $\boldsymbol{h}$ departing from a given vertex v.
Output: Ensemble $\mathcal{D}$ of neighbours of vertex v, ensemble $\mathcal{H}$ of half-edges from vertex v to its neighbours.
Set $d^0 =$ destination vertex of $\boldsymbol{h}$. ▷ *Save first neighbour.*
while TRUE **do** ▷ *See break condition on line 7.*
  Set $\boldsymbol{h} =$ pair half-edge of previous half-edge of $\boldsymbol{h}$.
  Set $d =$ destination vectex of $\boldsymbol{h}$.
  Add d to $\mathcal{D}$, add $\boldsymbol{h}$ to $\mathcal{H}$. ▷ *By convention of the half-edge construction, these are added in anticlockwise order.*
  if $d = d^0$ **then** ▷ *All neighbours have been found.*
    break
  end if
end while
return $\mathcal{D}$, $\mathcal{H}$

Appendix B: Create an egde

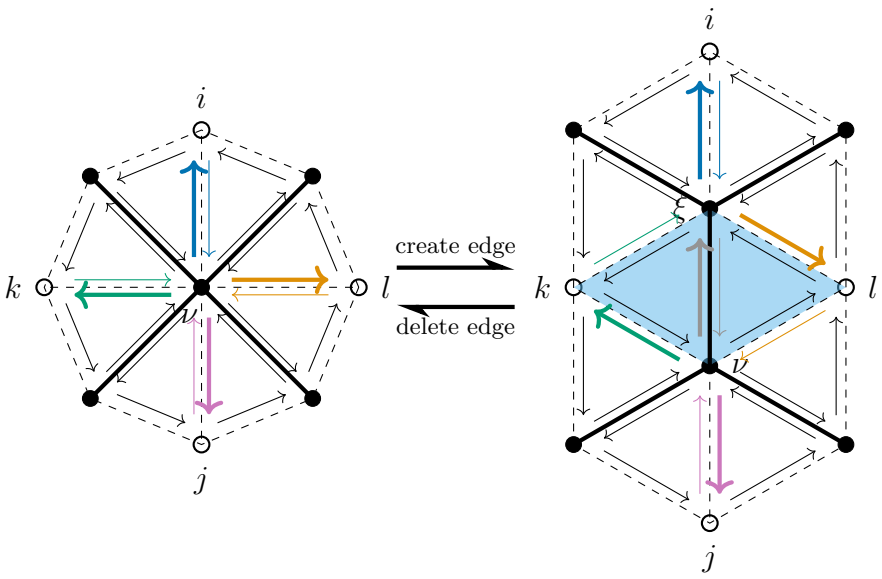

FIG. 5. Representation of edge creation and deletion in the half-edge construction. (See Fig. 4 for representation in the orthogonal direction.)

ALG. 5. Create an edge (see MESH::CREATEEDGE in MESH.CPP).

Input: Half-edges $\boldsymbol{h}$ and $\boldsymbol{h}'$ going out of the same vertex. ▷ *Thick green and yellow half-edges in Fig. 5.*
Input: Types $t, t^\dagger$ for half-edges at which new triangles are adjacent. ▷ *Thick and thin grey lines in Fig. 5.*
Output: Created vertex and half-edges.

Set ν = origin vertex of $\boldsymbol{h}$. ▷ *ν should also be the origin vertex of $\boldsymbol{h}'$.*
// Create new vertex.
Create vertex ξ associated to half-edge $\boldsymbol{h}'$ going out of it.
Associate $\boldsymbol{h}$ to ν as half-edge going out of it.
// Relabel half-edge origins and destinations for part of the half-edges connected to ν (between $\boldsymbol{h}$ and $\boldsymbol{h}'$).
Set $\boldsymbol{p} = \boldsymbol{h}'$.
while $\boldsymbol{p} \neq \boldsymbol{h}$ **do**
    Set origin vertex of $\boldsymbol{p} = \xi$ (instead of ν).
    Set $\boldsymbol{b}$ = previous half-edge of $\boldsymbol{p}$.
    Set destination vertex of $\boldsymbol{b} = \xi$ (instead of ν).
    Set $\boldsymbol{p}$ = pair half-edge of $\boldsymbol{b}$.
end while
// Create new half-edges and relabel half-edge pairs.
Create half-edge $\boldsymbol{h}^1 = \nu \to \xi$ and its pair half-edge $\boldsymbol{h}^{1\dagger} = \xi \to \nu$. ▷ *Thick and thin grey lines in Fig. 5.*
Associate types t and $t^\dagger$ to half-edges $\boldsymbol{h}^1$ and $\boldsymbol{h}^{1\dagger}$.
Set k = destination vertex of $\boldsymbol{h}$.
Create $\boldsymbol{b} = k \to \nu$ as previous half-edge of $\boldsymbol{h}^1$, and $\boldsymbol{n} = \xi \to k$ as next half-edge of $\boldsymbol{h}^1$. ▷ *$(\boldsymbol{n}, \boldsymbol{b}, \boldsymbol{h}^1)$ form a triangle.*
Set $\boldsymbol{h}^\dagger$ = pair half-edge of $\boldsymbol{h}$. ▷ *At this stage, $\boldsymbol{h}^\dagger$ is the thin green half-edge in Fig. 5.*
Associate $\boldsymbol{b}$ as pair half-edge of $\boldsymbol{h}$ and vice-versa.
Associate type of $\boldsymbol{h}$ to $\boldsymbol{b}$.
Associate $\boldsymbol{n}$ as pair half-edge of $\boldsymbol{h}^\dagger$ and vice-versa.
Associate type of $\boldsymbol{h}^\dagger$ to $\boldsymbol{n}$.
Set l = destination vertex of $\boldsymbol{h}'$.
Create $\boldsymbol{b} = l \to \xi$ as previous half-edge of $\boldsymbol{h}^{1\dagger}$, and $\boldsymbol{n} = \nu \to l$ as next half-edge of $\boldsymbol{h}^{1\dagger}$. ▷ *$(\boldsymbol{n}, \boldsymbol{b}, \boldsymbol{h}^{1\dagger})$ form a triangle.*
Set $\boldsymbol{h}'^\dagger$ = pair half-edge of $\boldsymbol{h}'$. ▷ *At this stage, $\boldsymbol{h}'^\dagger$ is the thin yellow half-edge in Fig. 5.*
Associate $\boldsymbol{b}$ as pair half-edge of $\boldsymbol{h}'$ and vice-versa.
Associate type of $\boldsymbol{h}'$ to $\boldsymbol{b}$.
Associate $\boldsymbol{n}$ as pair half-edge of $\boldsymbol{h}'^\dagger$ and vice-versa.
Associate type of $\boldsymbol{h}'^\dagger$ to $\boldsymbol{n}$.
return indices of vertex ξ and of half-edges within the created triangles.