Chapter 4

Separating the Topology and Fine-Scale Geometry of the Medial Scaffold

* Overview of chapter: representing the $\mathcal{MS}$ topology, geometry, and other attributes.

This chapter addresses the *representation* of the medial scaffold ($\mathcal{MS}$) with respect to its *topology*, *geometry*, *dynamics* (of the shock flow), and other attributes. A key idea is to separate the topology (which is pertinent to the qualitative structure) from its fine-scale geometry and other attributes and organize the $\mathcal{MS}$ as a *dual-scale* structure, Figure 4.1: *(i)* The *coarse-scale* structure is a (topological) **hypergraph** describing the global structural inter-connectivity between the medial sheets, curves, and nodes, which essentially implements the topological medial scaffold ($\mathcal{MS}^{\mathcal{T}}$) in § 3.2. *(ii)* The *fine-scale* structure is a **polygonal mesh** describing the local metric attributes such as the geometry and dynamics of the $\mathcal{MS}$. In the dual-scale $\mathcal{MS}$ representation, the coarse-scale sheets/curves explicitly comprise its fine-scale mesh face/edge elements (detailed in § 4.1). We develop a **novel *Extended Half-Edge* ($\mathcal{EHE}$)** data structure by adopting the popular *half-edge* [114, 113, 69] to represent both the coarse-scale hypergraph and the fine-scale mesh. We emphasize its capability to handle *non-manifold* junctions, which is essential in describing the ubiquitous intersections of the medial sheets at the medial curves.

* Main difficulty in representing the $\mathcal{MS}$: represent the medial sheet topology.

The main difficulty in representing the qualitative structure of the $\mathcal{MS}$ lies on representing its topology, *i.e.*, the **global inter-connectivity** of a medial sheet with respect to other sheets. [1] More precisely, the goal is to understand all topological configurations a medial sheet could possibly contain. We classify *three* such cases and represent the topological incidence between the medial sheets and curves using several *chains* of half-edges in the $\mathcal{EHE}$ representation, detailed in § 4.4. A recent related work is James Damon's study [58] on the "irreducible" medial sheet component that is topologically equivalent to a 2D disk.

Upon our solution described above, the second difficulty is on the topological *degeneracies* of a medial sheet, which cause more than one valid forms of a sheet in the $\mathcal{EHE}$ representation. Thus a *canonicalization* of the $\mathcal{EHE}$ representation is required. We analysis these cases and define a **canonical form** of the medial sheet topology, which will be useful in editing the $\mathcal{MS}$ topology in applying the $\mathcal{MS}$ transforms in Chapter 5.

* Organization of chapter.

---

[1] Possibly a medial sheet could intersect itself, see below.

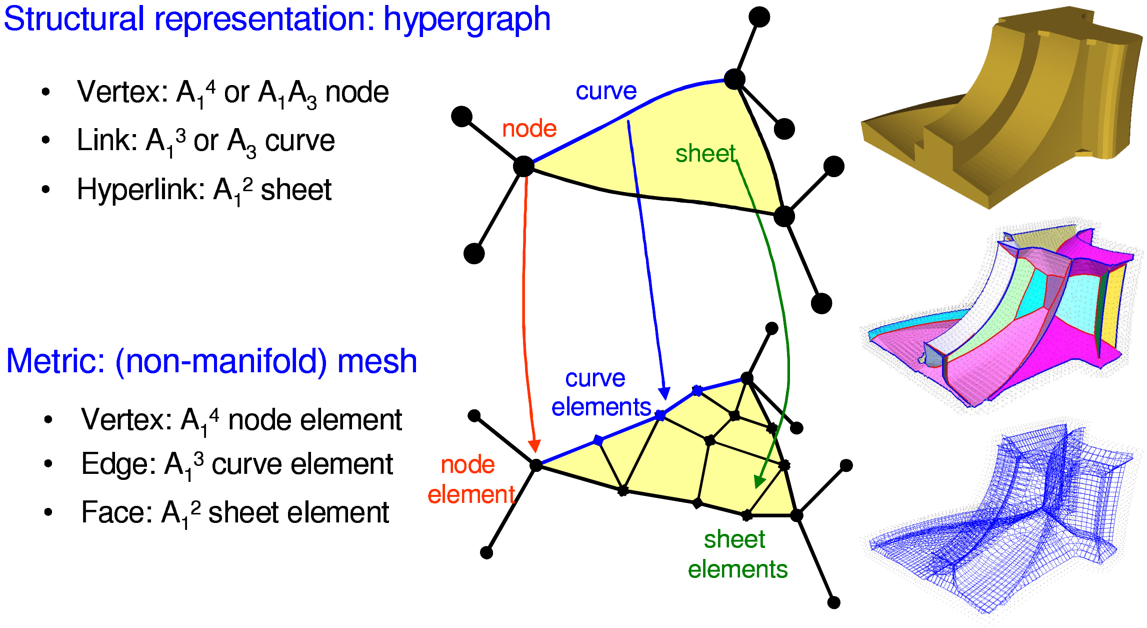


Figure 4.1: The *dual-scale* $\mathcal{MS}$ structure [43]: The coarse-scale is a *hypergraph* capturing the global *topology* and the fine-scale is a (typically non-manifold) *mesh* capturing the metric information such as the *geometry* and other attributes. The figures on the right depicts an industrial part (fandisk) and its $\mathcal{MS}$ in both scales.

This chapter is organized as follows. Section 4.1 discusses how the $\mathcal{MS}$ representation can be separated into two parts: a coarse-scale structure representing its global topology and a fine-scale structure capturing its detailed metric properties. The remaining of this chapter describes how the coarse-scale $\mathcal{MS}$ topology can be represented by extending the half-edge ($\mathcal{HE}$) data structure (originally designed to represent *2-manifold* meshes) to efficiently handle *non-manifold* meshes. Specifically, Section 4.2 reviews the $\mathcal{HE}$ data structure to represent a polyhedral surface. Section 4.3 extends the $\mathcal{HE}$ data structure to represent a *non-manifold* polygonal mesh. Section 4.4 further extends the above approach to represent the topology of a *geometric hypergraph*, which can be viewed as a generalization of the mesh to a hypergraph consisting of sheets and curves by deforming the mesh faces and edges.

4.1 The Dual-Scale Representation of the $\mathcal{MS}$

* Motivation: separating the $\mathcal{MS}$ topology from its fine-scale attributes.

The following advantages motivate the separation of the $\mathcal{MS}$ topology from its fine-scale metrics:

- *Enable an effective representation of the $\mathcal{MS}$ attributes.* Among the important attributes of the $\mathcal{MS}$ such as the topology, geometry, dynamics, *etc.*, the topology is a global attribute describing the structure, while the others are local properties. [2] We explicit construct a coarse-scale structure to handle the global topology of the $\mathcal{MS}$, namely the coarse-scale *hypergraph* composing of medial sheets/curves/nodes as hyperlinks/links/vertices, respectively. All other attributes is stored in a fine-scale structure, namely the fine-scale polygonal *mesh* composing

---

[2] The division of the shock flow dynamics interior to each shock sheet into *districts* of monotonic flows results in a *surface network*, which is another global attribute. This belongs to the future work of the shock scaffold ($\mathcal{SC}$) in § 3.4.

of face/edge/vertex elements. The coarse-scale components explicitly compose of their fine-scale elements (*i.e.*, contain pointers pointing to the mesh elements), as shown in Figure 4.1.

- *Enable an explicit implementation of the $\mathcal{MS}$ hierarchy.* The hierarchy of $\mathcal{MS}$ representations from the most complete $\mathcal{MS}^{\mathcal{H}}$ to the reduced ones of $\mathcal{MS}^{\mathcal{H}-}$, $\mathcal{MS}^{\mathcal{G}}$, $\mathcal{MS}^{\mathcal{G}-}$, and finally the $\mathcal{MS}^{\mathcal{T}}$ in Chapter 3 can be explicitly implemented via the dual-scale structure by dropping pertinent information.
- *Represent the qualitative $\mathcal{MA}$.* The explicit separation of two-level of structures enables representation of the qualitative $\mathcal{MA}$ where both the simplified global topology and detailed metrics are available. This is related to an idea in scientific computation (numerical analysis) that instead of aiming to compute an exact $\mathcal{MA}$ of the input shape, which is virtually no possible. We aim to approximate the exact $\mathcal{MA}$ of a *close-by* (perturbed) shape, whose $\mathcal{MA}$ is succinct and represents the qualitative structure. This can be done by apply a set of transforms operating on the two-level structure of the $\mathcal{MS}$, which will be elaborated in Chapter 5.
- *Enable an explicit simulation of $\mathcal{MS}$ transforms.* The explicit separation of a global structure to handle topology also enables to explicitly "edit" the hypergraph in applying the $\mathcal{MS}$ transforms. Note that the exact re-computation of the $\mathcal{MS}$ is costly. Instead, we *simulate* a $\mathcal{MS}$ transform in the coarse-scale hypergraph, while keeping the fine-scale $\mathcal{MS}$ elements intact (Chapter 5). In addition, the fine-scale geometry of the $\mathcal{MS}$ and its associated boundary generators are useful in estimating the transform costs.

* Coarse-scale medial sheets/curves/nodes to represent the global structure of the $\mathcal{MS}$.

We explicitly construct the coarse-scale medial sheets/curves/nodes of the hypergraph. They not only represent the global structure of the $\mathcal{MS}$ but also organize their corresponding fine-scale mesh elements. Specifically, the coarse-scale medial sheet composes of a set of fine-scale faces (which can be viewed as also a "meshing" of the faces), the coarse-scale medial curve composes a set of fine-scale mesh edges as a *poly-line*, and the coarse-scale medial node simply contain a fine-scale vertex. In our implementation a coarse-scale component only stores two items to reduce unnecessary redundancy: *(i)* its global (topological) connectivity via the $\mathcal{EHE}$ detailed below and *(ii)* its pointers to all fine-scale elements.

* Fine-scale polygonal mesh elements to handle the local metric attributes of the $\mathcal{MS}$.

We explicit construct a fine-scale polygonal mesh to organize and store local attributes as follows:

- *Geometry:* We store the position $p(x, y, z)$ of each fine-scale vertex v at it. Note that this is the only *external* information we store in the dual-scale structure. The geometry of the $\mathcal{MS}$ such as the length and area, are then available by linear approximation via the polygonal mesh.
- *Associated boundary points:* The boundary sample points (generators) of the $\mathcal{MS}$ are explicitly associated with the fine-scale A_1^2 mesh faces. The generators of an A_1^3 edge and A_1^4 vertex elements are available through the fine-scale mesh connectivity. The issue of maintaining a consistent boundary-$\mathcal{MS}$ association in the $\mathcal{MS}$ transforms will be handled in $\S$ 7.3.
- *Dynamics of shock radius and derivatives:* The **dynamics** of the shock *radius* r and its derivatives (*velocity* $v = \frac{dr}{ds}$, where s is the arc-length along the shock, and *acceleration* $a = \frac{dv}{ds}$) can be computed from the associated generators in the local configuration. We thus do not explicitly store the initial dynamics, but for applications where they are explicitly used such as to reconstruct the shape in $\S$ 8.2, we could store them in the A_1^2 face elements.

Table 4.1: A summary of the coarse-scale hypergraph components and the fine-scale mesh elements.

	Topological representation
fine-scale $\mathcal{MS}$ vertex v	incident $\{e_i \vert i = 1, ..., n_{ve}\}$
fine-scale $\mathcal{MS}$ edge e	(v_s, v_e), $\{he_i \vert i = 1, ..., n_{ef}\}$ circular list
fine-scale $\mathcal{MS}$ face f	$\{he_i \vert i = 1, ..., n_{fe}\}$ circular list
coarse-scale $\mathcal{MS}$ node N	incident $\{C_i\} \vert i = 1, ..., n_{NC}$
coarse-scale $\mathcal{MS}$ curve C	(N_s, N_e), $\{he_i \vert i = 1, ..., n_{CS}\}$ circular list
coarse-scale $\mathcal{MS}$ sheet S	boundary $\{he_i \vert i = 1, ..., n_{SC_b}\}$ circular list, interior $\{he_j \vert j = 1, ..., n_{SC_i}\}$ circular lists

Table 4.1 summarizes the components comprising the dual-scale $\mathcal{MS}$ representation. Observe that the main topological relationship between the components is the **incidence relationship** between a 2D sheet (or face) to an 1D curve (or edge). Such topological incidence can be efficiently described by a "half-edge", a key feature of the $\mathcal{HE}$ data structure. Below we describe a general definition of the polygonal mesh and review the $\mathcal{HE}$ data structure in details.

4.2 A Review of the Half-Edge ($\mathcal{HE}$) Data Structure to Represent Polyhedral Meshes

We first define a general polygonal mesh as follows. Note that our definition is generalized from the popular view of a (surface-like) *2-manifold* mesh commonly used in computer graphics [82]. Specifically, we consider the *non-manifold* cases and the *degeneracies* (*e.g.* of a face degenerates into an edge, *etc.*), which is useful in modeling the 3D $\mathcal{MA}$ and other applications.

Definition 2 *A* **general polygonal mesh** M *is a collection of* vertices $V = \{v_i | v_i = (x_i, y_i, z_i), i = 1, ..., n_v\}$, edges $E = \{e_j | j = 1, ..., n_e\}$, *and* faces $F = \{f_k | k = 1, ..., n_f\}$, *such that each vertex is a point in* $\mathbb{R}^3$*, each edge is a line segment with two ending vertices in* V*, each face is a polygon with* m *edges in* E*, i.e.:*

$$\forall e \in E, \exists v_1, v_2 \in V, e = \{v_1, v_2\}, \tag{4.1}$$

$$\forall f \in F, \exists e_1, e_2, ..., e_m \in E, f = \{e_1, e_2, ..., e_m\}, \tag{4.2}$$

and no two edges intersect except at the ending vertices, no two faces intersect except at the boundary edges.

A **complete** *mesh is a mesh with no isolated edges nor vertices, i.e.,*

$$\forall e \in E, \exists f \in F, e \cap f \neq \emptyset, \quad \forall v \in V, \exists e \in E, v \cap e \neq \emptyset, \tag{4.3}$$

A **2-manifold** *mesh is a* complete *mesh where each edge is shared by at most two faces, where* (i) *the set of all edges with only one incident face is the* boundary *of the manifold, and* (ii) *the neighborhood of any interior point on the mesh (except the boundary) is homeomorphic to a small disk in* $\mathbb{R}^2$. *A* **triangular** *mesh is a* complete *mesh with triangles as faces. A* **regular** *2-manifold mesh does not contain any* degenerate *vertex, which if removed, disconnects the mesh (i.e., a "vertex-only" connectivity).*

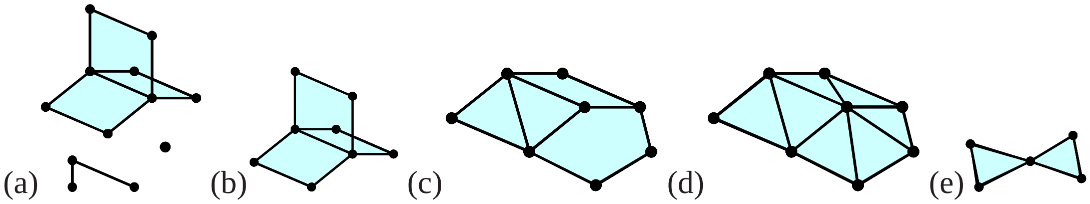


Figure 4.2: This figure illustrates several example meshes defined in the text. (a) A *general polygonal mesh* containing several degeneracies of isolated edges and vertices. (b) A complete *non-manifold* polygonal mesh. (c) A complete *2-manifold* polygonal mesh. (d) A complete 2-manifold *triangular* mesh. (e) An *irregular* triangular mesh with a *"vertex-only"* connectivity.

Figure 4.2 illustrates several example meshes defined above. Again we note that the surface mesh commonly used in the computer graphics community is the *2-manifold* polygonal mesh such as the one in Figure 4.2(c), and the $\mathcal{HE}$ data structure we review below is originally designed for this category of meshes.

* A review of the data structures to represent a polygonal mesh.

The developing of an efficient data structure to store and represent a polygonal mesh has been an important topic in solid modeling. A suitable data structure provides efficient connectivity query between the mesh elements (faces, edges, vertices) and the modification of them; it as well avoid duplications in the representation. We survey a few basic mesh data structure in the footnote. [3]

* A review of the half-edge data structure.

The **Half-Edge** ($\mathcal{HE}$) data structure [114, 113, 69]) is a popular *edge-centered* data structure originally proposed to describe a *2-manifold* mesh. Its basic element is a *half-edge*, which describes the *incidence relationship* between a mesh face and a mesh edge. Since originally only the 2-manifold mesh is considered, such *edge-face* incidence corresponds to only "half" of an edge, in that, an edge can only share at most two faces (two faces at the interior and one face at the boundary), Figure 4.3(b). Specifically, we include a C++ implementation of the $\mathcal{HE}$ data structure in Figure 4.3(c). The "next" pointers of several half-edges form a circular list describing all boundary edges of a mesh face. The "pair" pointer of an half-edge enables to "navigate" across the other half-edge to fetch the two ending vertices of the current edge. The representation is decently optimized for a 2-manifold mesh.

The original $\mathcal{HE}$ only handles a *regular 2-manifold* mesh with several limitations: It can not represent a mesh with *(i)* a *degenerate* vertex as shown in Figure 4.2(e), which if removed, the mesh is disconnected, or *(ii)* a *degenerate* edge which is with no incident face(s).

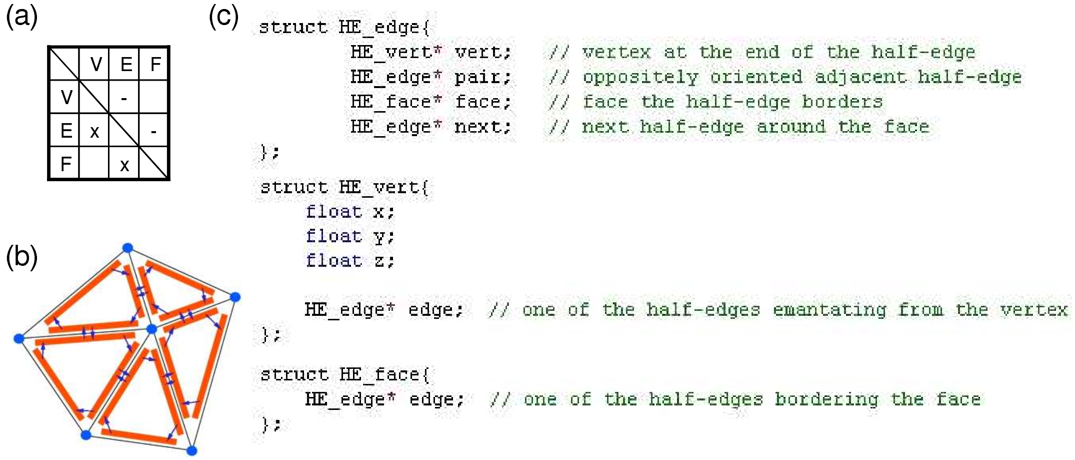
4.3 Extend the Half-Edge Data Structure to Represent General Non-Manifold Polygonal Meshes

We extend the $\mathcal{HE}$ data structure with the following *two* capabilities to represent a *general polygonal mesh* defined in Definition 2: [4]

---

[3] The **Indexed Face Set** (IFS) [82, p.473] is a common data structure for mesh storage and visualization (used in OpenGL and VRML), but it is not suitable for mesh connectivity queries. The **Winged-Edge** ($\mathcal{WE}$) data structure [16] can be viewed as the precursor of the half-edge ($\mathcal{HE}$) data structure. For recent development of the mesh data structures in *non-manifold* object modeling, see *e.g.*, [79, 80, 17].

[4] The author gratefully thanks J. Mundy for helpful discussions in developing this data structure.



	V	E	F
V		-	
E	x		-
F		x	

```
struct HE_edge{
        HE_vert* vert;   // vertex at the end of the half-edge
        HE_edge* pair;   // oppositely oriented adjacent half-edge
        HE_face* face;   // face the half-edge borders
        HE_edge* next;   // next half-edge around the face
};
struct HE_vert{
    float x;
    float y;
    float z;

    HE_edge* edge;  // one of the half-edges emantating from the vertex
};
struct HE_face{
    HE_edge* edge;  // one of the half-edges bordering the face
};
```

Figure 4.3: The *half-edge* ($\mathcal{HE}$) data structure adapted from the *flipcode* web site (http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml). (a) shows a table describing the topological relationship between mesh faces, edges, and vertices. A topologically valid representation allows to access the edges of a face and the ending vertices of an edge, but not *vice versa*. (b) is an example mesh represented in the $\mathcal{HE}$ data structure. (c) shows the C++ code implementing the $\mathcal{HE}$ data structure.

- *Handle non-manifold junctions.* By extending the couple of half-edges in the $\mathcal{HE}$ in representing an edge, we make the "pair" pointer of a half-edge to form an ordered circular loop around a non-manifold junction edge, Figure 4.4(a). [5]
- *Handle degenerate mesh configurations.* We explicitly represent the mesh *edge* in our data structure, Figure 4.4(a), such that a degenerate edge (with no incident faces) can be represented. In addition, we store the *vertex-edge* incidence at each vertex with a dynamic array. [6]

Our new data structure is named the **Extended Half-Edge** ($\mathcal{EHE}$) data structure. It is *effective* in storing only the minimum set of information to represent the general polygonal mesh, including the non-manifold cases and all degenerate configurations. It is also *efficient* in providing instant access to the incidence relationship between all mesh faces, edges, and vertices in traversal.

4.4 The Extended Half-Edge ($\mathcal{EHE}$) Data Structure to Represent the $\mathcal{MS}$ Hypergraph

This section describes a further extension of the $\mathcal{EHE}$ to represent the topology of the $\mathcal{MS}$ hypergraph. We first define the notion of two types of hypergraphs and then describe the data structure to representation them as follows.

---

[5] The half-edge in this circular list can be called a *partial-edge* instead of a "half" edge.

[6] This is the only dynamic array necessary in our mesh data structure, which can be efficiently implemented using the C++ vector of the Standard Template Library (STL).

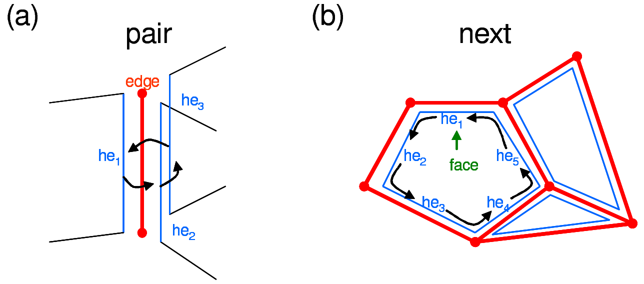


Figure 4.4: The *extended half-edge* ($\mathcal{EHE}$) data structure to represent the *non-manifold* polygonal meshes. (a) The half-edge's pair pointers form a circular loop to represent the *edge-face* incidence around a junction. (b) The half-edge's next pointers form a circular loop to represent all *edge-face* incidence of a face at its boundary.

Definition 3 *An* **ordered topological hypergraph** *is a collection of vertices V, edges E, and hyperedges H such that:*

$$V = \{v_i \mid i = 1, ..., n_v\} \tag{4.4}$$

$$E = \{e_j = \{v_a^j, v_b^j\} \mid v_a^j, v_b^j \in V, j = 1, ..., n_e\} \tag{4.5}$$

$$H = \{h_k = (e_1^k, e_2^k, ..., e_m^k) \mid e_1^k, e_2^k, ..., e_m^k \in E, e_i^k \cap e^k_{(i+1)\ mod\ m} = v_i^k, \quad i = 1, ..., m, m > 2, k = 1, ..., n_h\} \tag{4.6}$$

Definition 4 *A* **geometric hypergraph** *is an* ordered topological hypergraph *where each vertex is a point in $\mathbb{R}^3$, each edge is a curve in $\mathbb{R}^3$ with two ending vertices in V, each hyperedge is a surface in $\mathbb{R}^3$ bounded by the edges in E. No two edges intersect except at the ending vertices, and no two hyperedges intersect except at the* boundary *edges. The* internal boundary *on a hyperedge surface is called an* **internal *anchor curve*** *of it. A hypergraph is* **complete** *if it has no isolated edges nor vertices.*

The geometric hypergraph defined above can be viewed as a generalized case of a polygonal mesh defined in Definition 2, in that a mesh edge is now allowed to deform into a curve, and a mesh face is deformed into a surface.

Observe in Figure 4.1 for the dual-scale representation of the $\mathcal{MS}$, that the *incidence relationships* between the fine-scale mesh face/edge/verex elements and the coarse-scale hypergraph sheets/curves/vertices are **identical** in a local sense. The major extension in the $\mathcal{MS}$ hypergraph is that non-planar variations of the sheets (in comparison to the mesh face as a planar polygon) may cause additional topological variations. We describe the new cases below.

* Three *general* types of topological connectivity of the medial sheets.

A coarse-scale medial sheet S in general may contain the following *three* types of boundary topology, based on Giblin and Kimia's analysis of the generic $\mathcal{MA}$ transitions [88], Figure 4.5:

- (One-incident) *boundary curve* including the A_3 *ribs* and A_1^3 *axials* bordering with other sheets. The boundary curve of any internal *'void'* of the sheet S, which can be either a hole or part of other sheets, also belongs to this category.
- (Two-incident) *anchor curve* internal to S where another medial sheet (such as a *tab*) intersects with this sheet S.

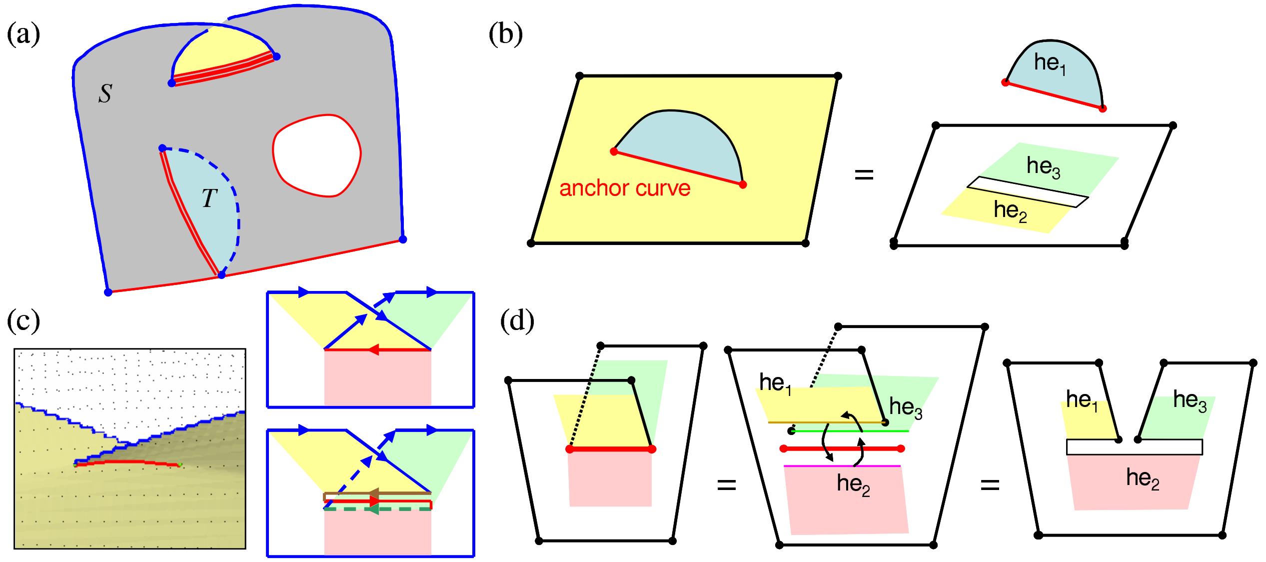


Figure 4.5: The general topology of an A_1^2 shock sheet $\mathcal{S}$ in the $\mathcal{MS}$ hypergraph and its representation using the *extended half-edge* ($\mathcal{EHE}$) data structure [43]. (a) Three possible types of sheet-curve incidence of $\mathcal{S}$: the boundary curve, anchor curve (with a tab $\mathcal{T}$), and swallow-tailed self-intersection. The topology of the 2-incident anchor curve (double red curve in (a)) can be represented by two half-edges in (b). (c) The A_1^3 curve is triply incident to $\mathcal{S}$, where the 3 incidence can be ordered to create a loop in $\mathcal{S}$'s boundary chain. (d) In the $\mathcal{EHE}$ representation, the 3-incident swallow-tailed self-intersection (triple red curve in (a)) can be represented by 3 half-edges both in a *loop* (at the junction) and in a *chain* in (d).

- (Three-incident) *swallow-tailed* self-intersection of S, which occurs near an A_5 transition (detailed in [88] and Chapter 5). This configuration can be viewed as that the A_1^3 curve is triply incident to S in a loop, and the three incidences are ordered as a single boundary chain (bordering the boundary of S), Figure 4.5c.

The three types of medial sheet topology coincide with James Damon's study in the global $\mathcal{MA}$ topology [58, p.2390] (in decomposing the $\mathcal{MA}$ into *irreducible* medial components). The above three types cover all cases we observed in practice; no further case is encountered in all our experiments.

We are now ready to present a proper data structure to describe the $\mathcal{MS}$ hypergraph topology. We have described the topology of the medial *sheets* above, and the topology of medial curves and nodes are relatively simple: *(i)* an A_1^n medial *curve* is the intersection of n medial sheets ($n \geq 3$), and *(ii)* a medial *node* is the intersection of several medial curves and sheets, where only the incidence relationship between the node-curve needs to be explicitly stored.

* Representing the topology of a medial sheet: the *boundary curve chain* and *internal curve chains*.

We further apply the $\mathcal{EHE}$ data structure described above to handle the $\mathcal{MS}$ hypergraph as a *geometric hypergraph.*[7] Specifically, the mesh edge (line segment) is now a medial curve; and the mesh face (polygon) is now the medial sheet. The half-edge now represents each *sheet-curve* incidence, in that each boundary of the medial sheet can be organized into a ***chain* of half-edges** in three cases:

---

[7] We can view the $\mathcal{MS}$ hypergraph is a special case of a *geometric* hypergraph: Any representation to describe a geometric hypergraph are capable to describe the $\mathcal{MS}$ hypergraph.

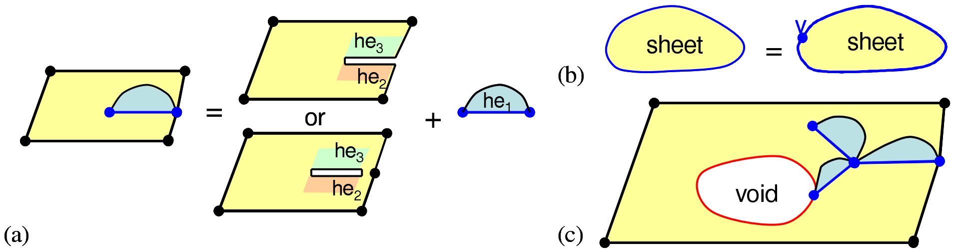


Figure 4.6: Degenerate medial sheet topology. (a) An anchor curve intersecting at the sheet boundary can be viewed as two way in the $\mathcal{EHE}$ representation. *(i)* a sheet with a single boundary chain given from an Euler tour, which contains a (degenerate) overlap, or *(ii)* a boundary chain and a separate internal *anchor curve* chain. (b) A sheet with a closed curve (with no ending node) as boundary can be represented in $\mathcal{EHE}$ by introducing a “dummy” node. (c) An example that the combination of the three types of general sheet topology could create complex configuration this requires a *canonicalization*.

- *A boundary curve chain* to describe the outer boundary of a medial sheet. [8]
- *An internal curve chain* to describe the inner boundary of a medial sheet. [9]
- *An internal curve pairs* to describe the internal *anchor* curves (2-incidence), which can be viewed as a degenerate case of an internal curve chain of two half-edges.

* Handle *degenerate* cases in the medial sheet topology.

While the three general types of $\mathcal{MS}$ sheet topology is explicitly handled above, the combination of them could create *degenerate* configurations which cause complication in handling the $\mathcal{EHE}$ data structure. Specifically, the degenerate configuration could cause more than one valid representations in the $\mathcal{EHE}$. Figure 4.6(a) depicts one example, where two interpretations using the $\mathcal{EHE}$ are both valid. We list more cases observed in practice below:

- Medial sheet with a closed curve (loop with no ending node) as boundary, Figure 4.6(b).
- Medial sheet with internal voids, where the internal boundary has no distinction from the outer boundary.
- Other combinations of the above cases could produce more complication, such as the one in Figure 4.6(c).

* Canonicalization of $\mathcal{MS}$ sheet topology in the $\mathcal{EHE}$ representation.

The above complications of the multiple representations of the degenerate $\mathcal{MS}$ topology can be handled by a *canonicalization* of the $\mathcal{EHE}$ representation. Specifically, we convert the $\mathcal{EHE}$ into a **canonical form**, which effectively remove the non-standard but valid representations. This can be done by comparing different representations for equivalence and standardize their sorting order. This canonicalization of medial sheet topology will be useful in two ways in our implementation: *(i)* in editing the $\mathcal{MS}$ topology in the $\mathcal{MS}$ transforms in Chapter 5 and *(ii)* in constructing of the coarse-scale $\mathcal{MS}$ sheets in Chapter 7.

---

[8] The A_5 swallowtail (self-intersection) is a special case of a boundary curve, Figure 4.5(c), where the half-edges can also be ordered in a sequential chain.

[9] The internal curve chains can be treated identical to a boundary chain, since topologically they are equivalent. We explicit distinguish the (unique) boundary chain for a major reason, that it provides a better compatibility in designing the C++ class hierarchy: a sheet (surface) is a sub-class (an extension) of a mesh face, where only a boundary chain is defined.