MidSurfer: Efficient Mid-surface Abstraction from Variable Thin-walled Models

Li Ye[a], Xinhang Zhou[a], Peng Fan[a], Ruofeng Tong[a], Hailong Li[c], Peng Du[a], Min Tang[a,b,*]

[a]*College of Computer Science and Technology, Zhejiang University, Hangzhou, 310007, China*
[b]*Zhejiang Sci-Tech University, Hangzhou, 310018, China*
[c]*Shenzhen Poisson Software Co., Ltd., Shenzhen, 518129, China*

Abstract

This paper addresses the challenge of efficiently abstracting mid-surfaces from complex variable thin-walled models, a critical task in computer-aided design (CAD) and finite element analysis (FEA) for simplifying thin-walled structures. Traditional methods often require manual specification of pairing faces, which can be time-consuming and error-prone. Alternatively, automatic face pairing methods fail to meet the actual needs of variable thin-walled models, resulting in the accumulation of topological errors. Additionally, existing algorithms struggle to extract mid-surfaces from models with varying wall thickness or produce mid-surfaces with poor accuracy, leading to geometric errors. Furthermore, the computational efficiency of these methods is often inadequate for large-scale models. To overcome these challenges, we propose an automated face-pairing mechanism that eliminates the need for manual intervention, enhancing the algorithm's robustness and enabling it to handle cases that the commercial CAD modeling engine, Parasolid, cannot process. Our approach accurately processes variable thin-walled models, with results closely aligning with the ground truth, as demonstrated by the provided error distribution tables. Moreover, our algorithm achieves a $4-12$ times improvement in efficiency than previous methods over the geometry extraction stage and supports multi-threaded acceleration, significantly reducing computation time. Experimental results demonstrate that our algorithm surpasses existing methods in both accuracy and efficiency, offering a promising solution for mid-surface extraction in complex, variable thin-walled models.

Keywords: Mid-surface Abstraction, Variable Thin-walled Models, Automatic Face Pairing

1. Introduction

Mid-surface abstraction has been a classic engineering challenge for many years, playing a pivotal role in the simulation and analysis of thin-walled structures across various industries, including aerospace, automotive, and mechanical engineering. Within CAD systems, for a solid model, mid-surface abstraction is defined as the process of obtaining an intermediate surface for thin-walled structures (such as sheet metal and plastic parts), in which the size of one dimension (thickness) is much smaller than the other two (width and length). This surface ensures that the shortest distances from any point on it to the two faces are equal, thereby simulating the shape. This simplified representation is widely used in CAE operations, such as engineering analysis (finite element analysis, FEA) [1, 2], feature recognition [3], and model reduction [4, 5]. These applications help reduce the need for expensive physical prototypes, thereby shortening product development time and lowering costs. However, for engineering analysis, solid models often face difficulties in topological identification due to numerous detailed features. Additionally, when free-form surfaces are involved, the process of extracting mid-surface becomes time-consuming, and the resulting geometric errors cannot be ignored. Therefore, there is a need to abstract the mid-surface in a rapid way while the results remain reliable.

Current research on mid-surface abstraction employs diverse methodologies, primarily based on medial axis transform (MAT) [6, 7], chordal axis transform (CAT) [8, 9], and face pairing (FP). Compared to MAT and CAT methods, face pairing-based methods eliminate branching structures. This characteristic better reflects the topology of a solid model and has proven to be useful in a variety of applications. Due to this advantage, numerous studies [10, 11, 12] and commercial software (e.g., Parasolid [13]) have adopted this method. A typical face pairing-based workflow comprises three stages: face pairing, mid-surface extraction, and topological repair. Although existing algorithms perform adequately for simple constant-thickness models, they encounter critical limitations when handling complex scenarios such as continuous transitional features (fillets/chamfers) and complex variable-thickness models. These challenges manifest as mid-surface defects, including missing surfaces, gaps, overlaps, positional deviations, and shape inaccuracies. Rectifying such errors typically requires manual intervention, which is often tedious and time-consuming. Consequently, despite extensive research efforts, two primary issues persist:

1. **Complex Topological Structures:** For components with intricate topological structures, the automated face pairing mechanism often fails to yield accurate results. Existing algorithms struggle to handle transitional scenarios and variable thin-walled models, leading to inaccuracies that can compromise the integrity of subsequent analyses.
2. **Variable Thin-Wall Models:** Most algorithms are inadequate in efficiently and accurately processing variable thin-wall models. The variability in wall thickness introduces additional challenges, affecting both the geometric accuracy and computational efficiency of mid-surface extraction.

To address these challenges, this paper introduces a suite of innovative algorithms:

*Corresponding author. This work was funded in part by "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province (No. 2025C01086).

Email addresses: li-ye@zju.edu.cn (Li Ye), 2xh@zju.edu.cn (Xinhang Zhou), fanpeng0103@zju.edu.cn (Peng Fan), trf@zju.edu.cn (Ruofeng Tong), lihailong@poissonsoft.com (Hailong Li), dp@zju.edu.cn (Peng Du), tang_m@zju.edu.cn (Min Tang)

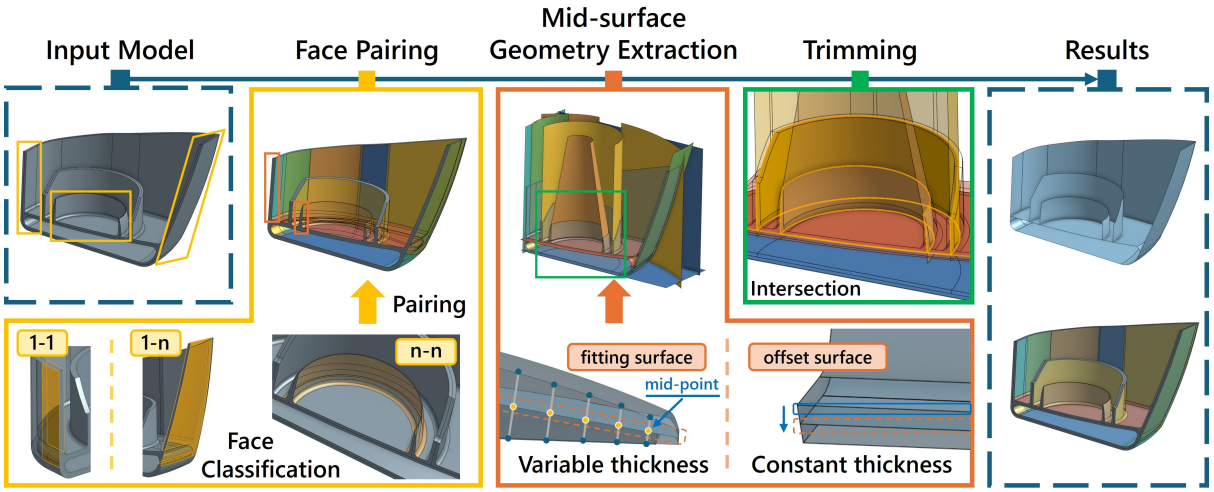

Figure 1: The algorithm overview of *MidSurfer*. The input thin-walled model's faces are first classified into different types and further organized into distinct face group pairs (FGPs). Subsequently, the mid-surface geometry extraction algorithm extracts the mid-surface for each face group based on the wall thickness. Finally, the model undergoes trimming operations, including intersection and imprinting, to determine the boundaries of the mid-surface, thereby yielding the final mid-surface.

- **Automated Face Pairing Mechanism:** We propose a novel automated face pairing mechanism that enhances the algorithm's robustness by expanding the face types and complementing the results based on face adjacencies. This mechanism can handle complex variable wall-thickness scenarios that are beyond the capabilities of conventional methods.
- **Efficient Variable Thin-Wall Processing:** We have designed an efficient algorithm for generating mid-surfaces for variable thin-wall models. By using the precise mid-point query strategy and the multi-threaded parallel mechanism, the algorithm not only improves computational efficiency but also maintains high geometric accuracy, addressing the limitations of existing methods.

Fig. 1 illustrates an overview of our proposed method. The initial thin-walled model first enters the face-pairing stage, where different face types are classified and face adjacencies are used to obtain pairing results. Then, in the geometry extraction stage, the wall thicknesses of the paired faces are determined. For constant-thickness models, offset surfaces are directly generated. For variable-thickness models, a two-step mid-point extraction method is applied to obtain precise mid-points, which are then used to fit surfaces. Finally, through intersection and imprinting operations, the surfaces are trimmed or stitched to produce the final mid-surface.

To validate our approach, we employed a series of complex engineering benchmarks. Our results demonstrate that we can generate mid-surfaces with high geometric accuracy for various variable wall thickness scenarios while significantly improving efficiency through parallel computing, with a $2-4$ fold increase compared to single-threaded implementations, and a $4-12$ fold increase compared to previous methods. Furthermore, by testing our manually created dataset of thin-walled models, comprising 213 models, our method outperforms existing techniques in terms of usability and generalizability. By addressing these long-standing challenges, our work contributes to more reliable and efficient mid-surface abstraction processes, paving the way for enhanced simulation and analysis of thin-walled structures.

2. Related Works

In this section, we briefly review recent advances in the most relevant areas: mid-surface abstraction methods, face pairing techniques, and mid-surface geometry extraction.

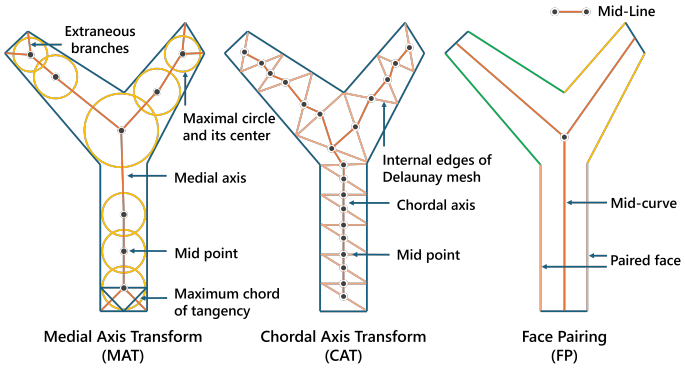

Figure 2: Differences between MAT, CAT, and FP in a 2D perspective.

2.1. Mid-surface Abstraction Method

Mid-surface abstraction techniques are primarily categorized based on their construction processes into three main methods, as shown in Fig. 2:

MAT, introduced by Blum [14], is an early dimensionality reduction method that generates mid-surfaces by computing the set of centers of the maximum inscribed circles within a geometric body. Although it is widespread in various analyses such as model simplification, shape analysis, and mesh generation [15, 6], its topological complexity leads to many small branches that need to be manually pruned.

To address these issues, Lee et al. [16] proposed a 3D MAT algorithm to trim small branches, but it struggled with degenerate models. Donaghy et al. [6] introduced a mixed-dimensional modeling approach using aspect ratio thresholds. Ramanathan et al. [7] combined medial axis and surface pairing methods to automatically generate mid-surfaces. However, these methods generally face challenges in topological correction and geometric accuracy due to their reliance on the model's medial axis.

CAT was initially proposed by Prasad [17] for skeletal representation in biological morphology. It defines a skeletal shape formed by connecting mid-points of Delaunay triangulated edges within a 2D shape. Quadros et al. [18] extended this concept to 3D, converting thin-walled models into single-layer tetrahedral meshes to generate gridded mid-surfaces. However, the initial tetrahedral mesh generation was overly simplistic, and mesh

classification was incomplete, failing to handle certain tetrahedral elements.

Kwon et al. [19] improved mesh quality using the advancing front method, and Quadros [8] introduced Delaunay meshing to avoid internal node insertion. Nolan et al. [9] proposed a hybrid mesh method that decomposes structures into beams, shells, and complex regions. Despite these advancements, generating accurate tetrahedral meshes for complex models remains challenging, and the geometric quality of the resulting mid-surface is often suboptimal.

Face pairing is currently the mainstream method for mid-surface abstraction. The core idea is to identify face pairs within a model, process the mid-surface geometry, and repair the model's topology. Rezayat [10] pioneered this approach by creating a Face Adjacency Graph (FAG) framework to match face pairs based on distance and normal. While foundational, this method was limited to simple models.

Lee et al. [20] enhanced the algorithm by introducing boundary extension rules, improving topological integrity by using the model's topological boundaries to extend mid-surfaces. Sheen et al. [11] proposed a comprehensive method by suppressing the features of the model and pairing the faces in the simplified model. However, these methods struggle with discontinuous surface connections and branching, leading to errors in regions with significant topological changes.

Other methods, such as the feature-based shrink-wrapping technique proposed by Sheen et al. [21], simulate a "deflation" process but rely on user-defined feature information. Commercial systems, such as Parasolid [13], integrate mid-surface generation tools but lack robustness for complex models and require manual face-pair specification.

Given the limitations of existing methods, developing an effective mid-surface abstraction process that ensures topological stability while supporting variable wall thicknesses and complex free-form surfaces is crucial. This paper builds upon the Face Pair method, generalizing the process into face pairing, geometry extraction, and trimming. Our newly designed algorithms not only ensure topological correctness but also effectively support variable wall thickness models.

2.2. Face Pairing Technique

While face-pair-based methods have shown promise in handling thin-walled models, their core challenge lies in robustly identifying face pairs in complex topologies. Traditional heuristic rules [10, 11, 20] struggled with both computational efficiency and accuracy in high-complexity models, prompting recent efforts to use model segmentation to reduce the search space.

Early segmentation methods focused on geometry-driven decomposition. Chong et al. [22] introduced a concave edge loop-based method, extracting boundary features and generating mid-surfaces within sub-modules. While pioneering mixed-thickness compatibility, it suffered from poor free-form adaptability due to its single concave edge criterion. Woo [12] advanced this with a divide-and-conquer strategy, decomposing models into basic units, constructing redundant connections, and hierarchically stitching to form a global mid-surface. Despite success with simple geometries, its reliance on extensive Boolean operations reduced stability for free-form surfaces and rib-like structures. Zhu et al. [23] improved this by decomposing models based on rib features, leveraging hierarchical semantics, and using offset operations to represent face pairs in sub-regions. However, it still struggled with face pairing in variable-thickness models.

To mitigate topological discontinuities caused by segmentation, hybrid-dimensional modeling and connectivity constraints have emerged. Robinson et al. [24] proposed a hybrid mid-surface paradigm, preserving 3D structures in connection regions and maintaining continuity via mixed 2D-3D representations. Kulkarni et al. [25, 26] decomposed thin-walled models into mid-surface generation nodes and interaction parsing nodes based on connection types, constructing mid-surfaces via offsetting and sweeping. This enhanced topological effectiveness but required feature-rich CAD inputs and was restricted to sheet metal parts, lacking adaptability for models with fillets or chamfers.

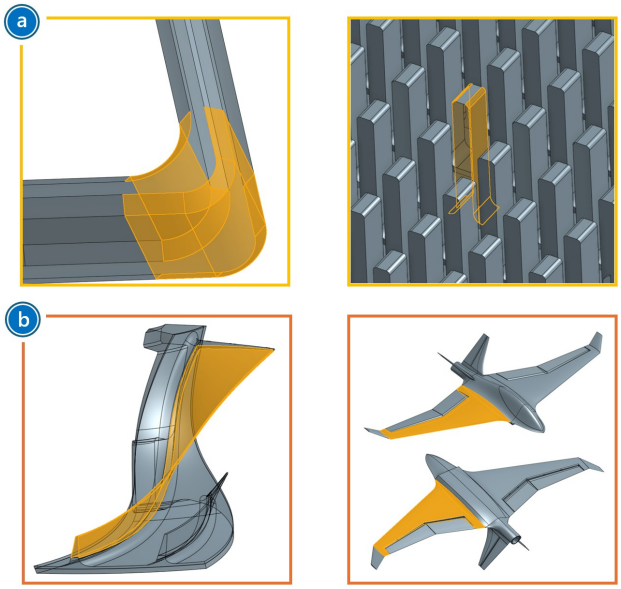

Figure 3: The cases where previous methods failed to address during the face pairing (a) and geometry extraction (b) stages.

Given these challenges, current research faces two core challenges (Fig. 3-a): (1) Existing face pairing methods (heuristic rules, virtual segmentation) lack geometric compatibility with variable-thickness free-form surfaces, causing pairing failures and hindering mid-surface geometry generation. (2) Current pairing criteria (feature suppression, 1-1 pairing) fail to support transitional scenarios (fillets, chamfers, 1-n/n-n pairing), compromising topological correctness in complex connections. Notably, thin-walled models in aerospace and automotive industries often share these features, demanding robust face pairing methods. This paper optimizes topological judgment for variable-thickness models by extending face pair types and versatile rules, thus enhancing mid-surface abstraction applicability.

2.3. Mid-surface Geometry Extraction

The creation of mid-surface geometry typically involves generating the mid-surface and trimming these surfaces. Most existing work [10, 11, 12] focuses on constant-thickness models, where mid-surfaces are efficiently derived from offset surfaces, producing high-quality results. Recent efforts, however, target variable-thickness models, where the core idea is to obtain mid-points on the mid-surface and interpolate them to form the final surface.

Several classic methods exist for obtaining mid-points. Shewchuk et al. [27, 28] used Delaunay triangulation to generate meshes for face pairs, treating vertices as feature points to find corresponding points on opposite faces and compute mid-points. Zhu et al. [23, 29] extended this by using projection and intersection: projecting points from the upper surface to the lower surface and constructing lines from original points and their normals to intersect with the lower surface, then the mid-point was interpolated by both points. However, the method is limited to nearly

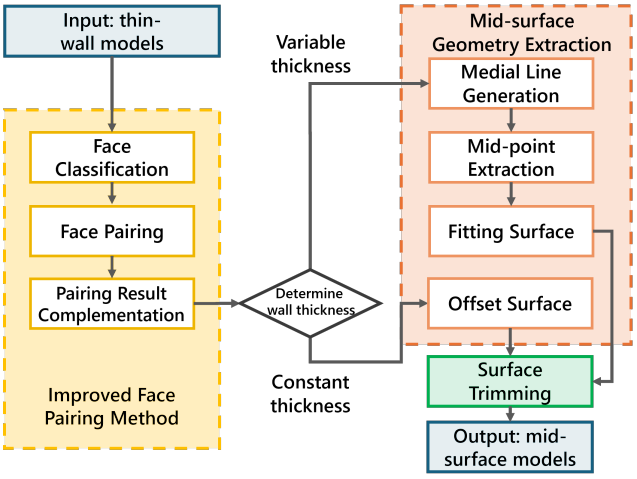

Figure 4: The algorithm pipeline of *MidSurfer* consists of two stages to obtain the correct face pairing results, the precise mid-surface fitting points, and trim or stitch the surface boundaries to abstract the correct mid-surface.

parallel face pairs, with significant errors occurring when surface angles exceed 15, highlighting the need for high-precision and efficient computation for variable-thickness models.

During the trimming or stitching stage, Lee et al. [20] created a Face Adjacency Graph (FAG) to represent relationships between connected faces, identifying face pairs with distances. These pairs were then used to extend and stitch mid-surfaces. Woo et al.[12] created mid-surfaces by trimming, extending, and stitching them together. The extension of mid-surfaces is complex and often relies on various methods (e.g., FAGs and MATs). However, these methods are sensitive to input and can result in incomplete or invalid mid-surfaces for complex solid models due to topological changes that occur when converting faces to mid-surfaces.

Given these challenges, current research faces two core issues (Fig. 3-b): (1) Existing mid-surface methods primarily target constant-thickness cases, while industrial models often feature variable thickness, with current support limited to specific surface types, yielding inaccurate results. (2) Variable-thickness methods heavily depend on geometric operations, leading to inefficiency, especially for complex models. This paper proposes a novel variable-thickness mid-surface extraction method, employing a heuristic algorithm on discretized face pairs to compute mid-points without geometric operations, enhancing efficiency and accuracy.

3. Algorithm Pipeline

The proposed mid-surface abstraction method follows a two-stage pipeline: (1) face pairing and (2) geometry generation (Fig. 4).

In the face-pairing stage, the input model is processed to extract constant and variable wall thickness face pairs using an improved face-pairing algorithm. This involves classifying faces into planar and transitional sets, grouping them based on distance and overlap rules, and refining the results by correcting misgroupings using face adjacency relationships.

Once the face groups are established, the geometry generation stage employs a two-step mid-point extraction algorithm. First, medial lines are generated by computing shortest distances between sampled face points using a Bounding Volume Hierarchy (BVH) and connecting them to form candidate medial

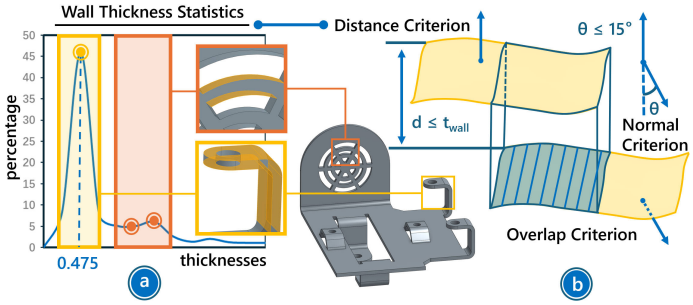

Figure 5: The illustration of distance, normal, and overlap criteria in face pairing.

lines. Next, precise mid-points are identified through a dual-index table-based iterative search, ensuring accurate placement.

Finally, the extracted mid-points are interpolated to construct NURBS surfaces, which are then trimmed based on face group connectivity and model boundaries. This refinement process ensures that the final mid-surface accurately represents the input model while maintaining structural consistency.

3.1. Improved Face Pairing Method

To address complex variable-thickness models, the first step is to accurately identify face group pairs (FGPs). Traditional face pairing methods (heuristic rules, virtual segmentation) [20, 11, 12, 26] lack geometric compatibility with variable-thickness free-form surfaces. These approaches typically require model feature suppression prior to pairing, while lacking robust support for transitional geometries (fillets and chamfers, etc.), thereby compromising topological integrity in complex junction regions. Consequently, we propose an improved face pairing algorithm specifically optimized for variable-thickness models.

Basic Concept. Three key criteria are widely used in the current face pairing method, as illustrated in Fig. 5:

1. **Distance Criterion**: the distance d between the two candidate faces should not exceed a threshold t_{wall}:

$$\text{Dist}(f_{\text{left}}, f_{\text{right}}) \leq t_{\text{wall}}, \tag{1}$$

   where t_{wall} defaults to the maximum thickness of the thin-walled model and can be manually specified.
2. **Normal Criterion**: The outward normals N_1 and N_2 of the paired faces must be approximately antiparallel, satisfying:

$$\left|180^\circ - \angle(\mathbf{N}_{\text{left}}, \mathbf{N}_{\text{right}})\right| \leq \theta, \tag{2}$$

   where θ denotes the angular tolerance threshold, typically set $\theta \leq 15^\circ$ in industrial applications.
3. **Overlap Criterion**: When projecting either face (the projection face) along its normal direction onto the other face (the target face), the projected area ratio should satisfy:

$$\text{Project}(f_{\text{left}}, f_{\text{right}}) \geq R \wedge \text{Project}(f_{\text{right}}, f_{\text{left}}) \geq R, \tag{3}$$

   where R defaults to 50% and can be manually specified.

Face Classification. We first classify each face in the thin-walled model into two distinct categories: face pairs and lateral faces.

1. Face pair: refers to a pair of faces in a thin-walled model that satisfies the aforementioned three criteria.
2. Lateral face: an elongated intermediate surface connecting the two constituent faces of a face pair in thin-walled structures.

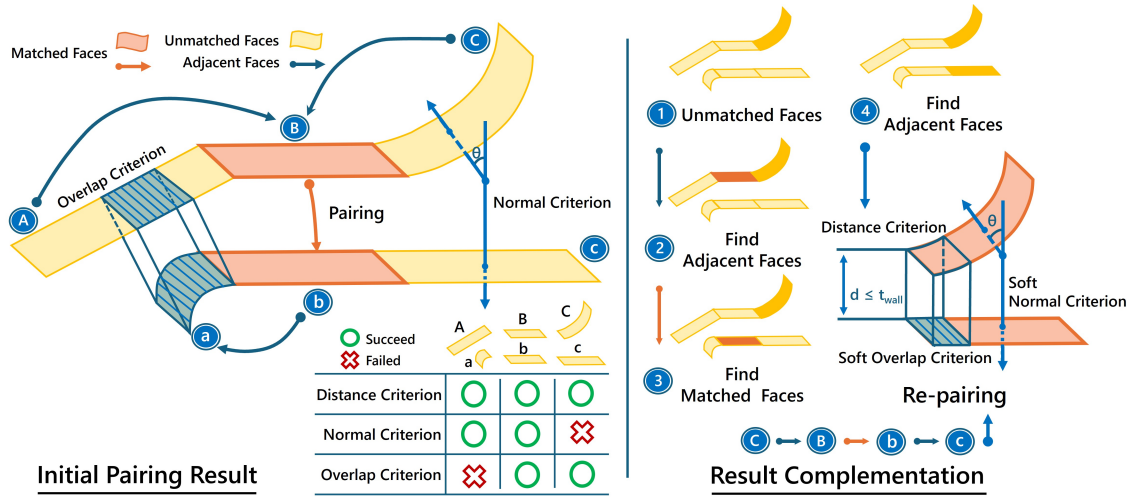

Figure 6: The face pairing algorithm consists of two steps. The left side illustrates the first step, where face pairs are filtered using three criteria. The table records whether each pair satisfies these criteria. The right side demonstrates the second step with a challenging face pair example, showing how its matched faces are identified by incorporating additional local adjacency information and relaxing thresholds for criteria 2 and 3.

Algorithm 1: Face Pairing Algorithm

Input : Face set F
Output: Face Group Pair FGP, candidate faces $F_{\text{candidate}}$

1. **Initialize:**
   $F_{\text{remain}} \leftarrow F$, $F_{\text{candidate}} \leftarrow \emptyset$
2. **while** $F_{remain} \neq \emptyset$ **do**
   3. $f_{\text{left}} \leftarrow \text{SelectSeed}(F_{\text{remain}})$
   4. $GroupRight \leftarrow \emptyset$
   5. **foreach** $f_{right} \in F_{remain}$ **do**
      6. **if** $Equation(1), (2), (3)$ **then**
         7. $GroupRight \leftarrow GroupRight \cup f_{\text{right}}$
         8 .$F_{\text{remain}} \leftarrow F_{\text{remain}} \setminus f_{\text{right}}$
      end
   end
   9. **if** $GroupRight = \emptyset$ **then**
      10. $F_{\text{candidate}} \leftarrow F_{\text{candidate}} \cup f_{\text{left}}$
      continue
   end
   11. $FGP \leftarrow GroupRight \cup \{f_{\text{left}}, GroupRight\}$
end
return $FGP, F_{candidate}$

Considering the real topology of the model, face pairs are categorized into 1-1 face pairs and 1-n/n-n face pairs.

1. 1-1 Face Pair: A face pair comprising exclusively paired left and right faces, where these two faces alone sufficiently encapsulate all geometric information required for mid-surface generation.
2. 1-n/n-n Face Pair: A face pairing formed by one or multiple left faces corresponding to multiple right faces. This type of pairing requires combinations of multiple left and right faces to fully describe the model features of thin-walled models at that location, which primarily represent transitional scenarios in the model structure, such as fillets, chamfers, etc, as shown in Fig. 1-Face Pairing.

Face Pairing. During the face pairing stage, three key criteria (*Distance*, *Normal* and *Overlap*) are precomputed for all distinct face combinations ($face_i$ and $face_j$, $i \neq j$) and recorded in a corresponding table. The *Overlap* is calculated by projecting sampled points from one face to another face and statistically determining the hit rate. The computation of *Normal* involves taking sample point within $face_i$'s overlap region and projecting point along $face_i$'s local normal onto $face_j$ to obtain corresponding points, calculating the angle θ_{ij} between the normal at these points, symmetrically performing the identical projection and angle calculation θ_{ji} from sample points within $face_j$'s overlap region onto $face_i$, and finally taking the arithmetic mean of all θ_{ij} and θ_{ji} values from these sampled points as the normal between the two faces, with the angle defined 0° if no overlap exists. After acquiring the criteria between different combinations, the combinations satisfying both $Overlap \geq 80\%$ and $|180° - \angle(Normal)| \leq 10°$ are filtered (such combinations exhibit higher probabilities of forming a face pair in thin-walled regions). The distance of these filtered combinations is then partitioned into intervals to construct a wall thickness statistics (as illustrated in Fig. 5-a). Finally, the maximum distance within the peak frequency interval is defined as the *Distance* threshold of the model.

The core framework of the face pairing algorithm is presented in Algorithm 1 and illustrated by Fig. 6. Select a seed face f_{left} and choose another face f_{right} from the remaining faces (line 3). Upon the precomputed table, f_{left} and f_{right} satisfy criterion(1, 2, 3) (line 6), add f_{right} to f_{left}'s matched pair group $GroupRight$, then continue iterating through the remaining faces (lines 7-8). If no qualifying face f_{right} is found, f_{left} is removed from the current face set (lines 9-10). A new seed face is then selected from the remaining faces, and the pairing operation is repeated. This iterative process continues until all faces in the model have been traversed, identifying partial 1-1 face pairs and 1-n face pairs (line 11). Any faces that remain unmatched are designated as candidate face sets $F_{candidate}$ for subsequent processing.

Pairing Result Complementation. In the preliminary face pairing stage, the aforementioned three criteria have successfully matched a subset of face pairs. However, the candidate face set $F_{candidate}$ still contains some faces (e.g., faces with significant normal variations or fragmented faces) that have potential matches but are difficult to pair using criteria 2 and 3. To address this, a complementary face pairing algorithm is proposed. This algorithm leverages local adjacency information from pre-matched face pairs to provide additional topological context, while relaxing the threshold in criteria 2 and 3, thereby repairing the unmatched faces to obtain all face pairs.

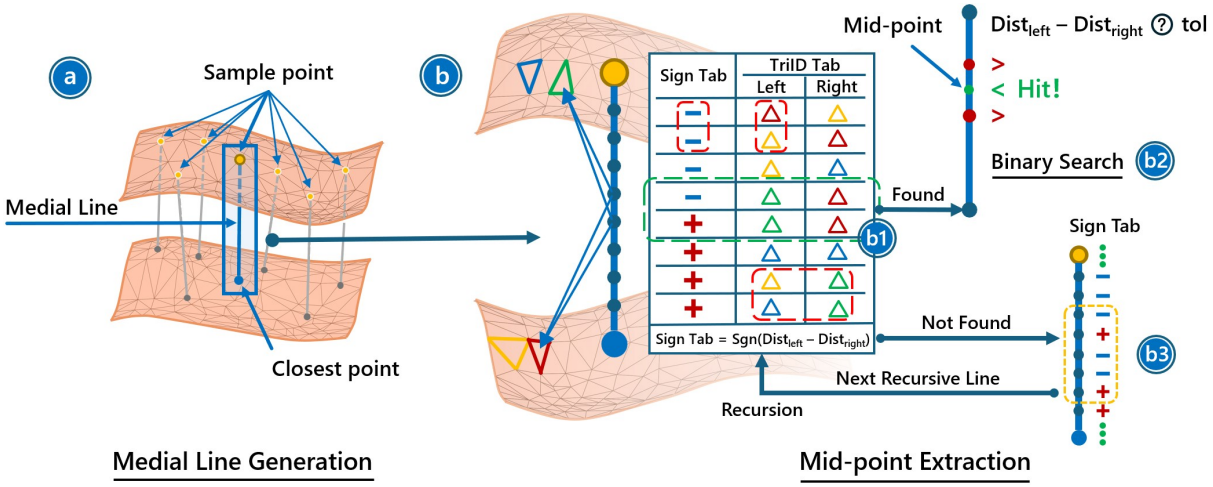

Figure 7: The two-step mid-point extraction method, including medial line generation and mid-point extraction, where the blue box represents the medial line to be extracted, the green box represents the recursive stopping condition, and the red box represents the non-stop cases, in which case the yellow box is the medial line of the new iteration.

Algorithm 2: Complementary Face Pairing Algorithm

Input : Candidate Face Set $F_{\text{candidate}}$, t_{wall}
Output: Face Group Pair FGP

1. **Initialize:**
   $F_{\text{remain}} \leftarrow F_{\text{candidate}}$
2. **while** $F_{remain} \neq \emptyset$ **do**
   3. $f_{\text{left}} \leftarrow \text{SelectSeed}(F_{\text{remain}})$
   4. $F_{\text{adj}} \leftarrow \text{GetAdjacentFaces}(f_{\text{left}})$
   5. $GroupRight \leftarrow \emptyset$
   6. **foreach** $f_{adj} \in F_{adj}$ **do**
      7. **if** $\exists f_{adj}^{matched} \leftarrow GetPreMatchedPairs(f_{adj})$ **then**
         8. $F_{\text{adj}}^{matched} \leftarrow \text{GetAdjacentFaces}(f_{\text{adj}}^{matched})$
         9. **foreach** $f_{right} \in F_{adj}^{matched}$ **do**
            $SoftOverlap(f_{left}, f_{right})$
            10. **if** $and\ SoftNormal(f_{left}, f_{right})$ **then**
            $and\ \neg AreAdjacent(f_{left}, f_{right})$
            11. $GroupRight \leftarrow GroupRight \cup f_{\text{right}}$
            end
         end
      end
   end
   12. $FGP \leftarrow GroupRight \cup \{f_{\text{left}}, GroupRight\}$
   13. $F_{\text{remain}} \leftarrow F_{\text{remain}} \setminus f_{\text{left}}$
end
return FGP

Algorithm 2 and Fig. 6 illustrate the main architecture of the proposed method, Within the candidate face set $F_{candidate}$, select a seed face f_{left} and retrieve its adjacent face set F_{adj}. For each face $f_{adj} \in F_{adj}$: if f_{adj} has a pre-matched face $f_{adj}^{matched}$, obtain $f_{adj}^{matched}$'s adjacent set $F_{adj}^{matched}$ (lines 3-8, Fig. 6-Result Complementation). For each $f_{\text{right}} \in F_{adj}^{matched}$. If f_{left} and f_{right} satisfy criterion 2 under a soft angle threshold (default: 0°-30°), criterion 3 under a soft overlap threshold (default: 30%), and are non-adjacent (no direct topological connection), add f_{right} to f_{left}'s matched pair group $GroupRight$ (line 10). Repeat this process by reselecting seed faces from the remaining candidates until all $F_{candidate}$ are exhausted. The final results include all 1-1 face pairs, 1-n face pairs, and the lateral face set excluded from mid-surface generation.

After obtaining all face pairs, abstract each face as a graph node. Connect corresponding left-right face pairs to form an undirected graph. A maximal connected subgraph in the undirected graph constitutes an FGP. For each maximal connected subgraph, perform graph coloring: initiate traversal from an arbitrary node, color this node red, and assign blue to all adjacent nodes and vice versa during propagation. Upon completing subgraph traversal, the faces corresponding to the red nodes are defined as left faces, and the faces corresponding to the blue nodes are defined as right faces. This process eliminates redundant face pair couplings, providing concise and precise FGPs for subsequent mid-surface generation.

3.2. Abstraction of Mid-surface Geometry

In geometric processing, given a face group pair, $FGP_i = \{FG_{\text{left}}, FG_{\text{right}}\}$, which consists of two face groups FG_{left} and FG_{right}, where $FG = (F_1, F_2, \ldots)$. For a mid-surface MS_i, it is represented as:

$$\forall P \in MS_i, \quad \text{Dist}(P, FG_{\text{left}}) = \text{Dist}(P, FG_{\text{right}}), \tag{4}$$

where the distance function $\text{Dist}(P, FG)$ is defined as the minimum Euclidean distance from point P to all faces in the face group FG.

After face pairing, we first determine the wall thickness of each FGP. We determine the type of wall thickness by Parasolid's range function, and if the difference between its maximum and minimum distances is less than a pre-specified tolerance (default: 1E-3), then it is judged to be constant wall thickness. For constant thickness, precise geometry can be directly achieved by creating an offset surface. For variable thickness, the resulting mid-surface can be generated by fitting precise mid-points to create a Non-Uniform Rational B-Spline (NURBS) surface. However, obtaining accurate mid-points is challenging. Previous methods used projection or intersection to obtain mid-points, but these approaches often resulted in low-precision mid-points, leading to cumulative errors in the fitted surface.

To address this issue, we propose a two-step method to compute the mid-points, as shown in Fig. 7. The process consists of medial line generation followed by mid-point extraction. Unlike prior methods, our algorithm directly calculates mid-points with a high degree of accuracy. Finally, the generated mid-surfaces are spliced or removed through trimming operations, guided by the model's topology, to consolidate the final mid-surface.

Algorithm 3: Medial Line Generation

Input : Face group pair $FGP_i = \{FG_{\text{left}}, FG_{\text{right}}\}$
Output: Medial line ML

1. **Initialize:**
   Discretize $FG_{\text{left}}, FG_{\text{right}}$ into triangular meshes $M_{\text{left}}, M_{\text{right}}$
2. $BVH_{\text{left}} \leftarrow \text{BuildBVH}(M_{\text{left}})$
3. $BVH_{\text{right}} \leftarrow \text{BuildBVH}(M_{\text{right}})$
4. $SP_{\text{left}} \leftarrow \text{DenseSample}(M_{\text{left}}, n)$ // n: sample density
5. **foreach** *sample point* $P_i \in SP_{left}$ ***in parallel*** **do**
   6. $Q_i \leftarrow \text{BVH\_ClosestPoint}(BVH_{\text{right}}, P_i)$
   end
7. Connect $\{(P_i, Q_i)\}$ pairs to form medial line ML
return ML

Medial Line Generation. The algorithm's overarching structure is delineated in Algorithm 3 and illustrated by Fig. 7-a. To generate the medial line, we first discretize the face groups (i.e., FG_{left} and FG_{right}, line 1) into triangle meshes M_{left} and M_{right}, respectively, then uniformly sample each triangle mesh M_{left} to obtain sample points (line 4), each triangle mesh retains its vertices and n interior points (default: $n = 1$), denoted as SP_{left}. Simultaneously, we construct Bounding Volume Hierarchies (BVH) for both the M_{left} and M_{right}, resulting in BVH_{left} and BVH_{right} (lines 2-3). After that, we perform closest-point queries for each sample point against BVH_{right} in parallel (lines 5-6). By connecting each sample point to its closest point on the M_{right}, we construct a medial line that passes through both face groups (line 7). This medial line exists at least one mid-point, which can then be further extracted.

Based on Theorem 1 in the supplementary material, we have the conclusion that if a line intersects both the left and right faces of a face pair, there must exist a point on the line equidistant to the left and right faces, which is the mid-point to be extracted.

Mid-point Extraction. The procedure for mid-point extraction Strategy is outlined in Algorithm 4. For each medial line, perform a recursive search (lines 2-16) by initially sampling n (default: 16) points (Fig. 7-b, line 3). Using BVH_{left} and BVH_{right}, calculate the shortest distance from each point to the left and right face groups (i.e., M_{left} and M_{right}, lines 5-6). During this process, for each point(P_i), we store the distance difference as $signTable$ (line 7):

$$Sign_i = \text{sgn}(\text{Dist}(P_i, M_{\text{left}}) - \text{Dist}(P_i, M_{\text{right}})), \tag{5}$$

and the corresponding triangle indices $TriId^{P_i}_{\text{left}}$ and $TriId^{P_i}_{\text{right}}$ in a triangle index table ($triIDTable$, line 8). Iterate through adjacent sample points P_i and P_j (lines 9–13). If adjacent points fulfill:

$$TriId^{P_i}_{\text{left}} = TriId^{P_j}_{\text{left}} \wedge TriId^{P_j}_{\text{right}} = TriId^{P_j}_{\text{right}} \wedge Sign_i \neq Sign_j, \tag{6}$$

indicate that there is a mid-point in P_i and P_j with equal shortest distance triangular indices from M_{left} and M_{right} (Fig. 7-b1, line 10). We then perform a binary search between P_i and P_j until the distance difference between the left and right face groups is less than the tolerance (Fig. 7-b2, lines 11-12). Return the resulting mid-point (line 13).

If no adjacent points meet these criteria, further subdivide the medial line (lines 14–16). Identify the first P_i and last P_j points in the $signTable$ where the sign changes (line 15), and recursively apply the sampling search method to the segment between these points (line 16, Fig. 7-b3).

Algorithm 4: Mid-point Extraction

Input : Medial line ML, BVH_{left}, BVH_{right}
Output: Mid-point MP

1. **Initialize:**
   MP, $d_{\max}$ // Maximum recursion depth
2. Function **RecursiveSearch**($[P_{\text{start}}, P_{\text{end}}]$, depth d):
3. Sample n points $\{P_1, ..., P_n\}$ along $[P_{\text{start}}, P_{\text{end}}]$
4. **foreach** $P_k \in \{P_1, ..., P_n\}$ ***in parallel*** **do**
   5. $d_{\text{left}} \leftarrow \text{BVH\_MinDist}(BVH_{\text{left}}, P_k)$
   6. $d_{\text{right}} \leftarrow \text{BVH\_MinDist}(BVH_{\text{right}}, P_k)$
   7. Store $signTable[k] \leftarrow \text{sgn}(d_{\text{left}} - d_{\text{right}})$
   8. Store $triIDTable[k] \leftarrow (\text{TriID}_{\text{left}}, \text{TriID}_{\text{right}})$
   end
9. **for** $k = 1$ ***to*** $n - 1$ **do**
   10. **if** *Equation(6)* **then**
      11. $P_{\text{mid}} \leftarrow \text{BinarySearch}(P_k, P_{k+1}, \epsilon)$
      12. **if** $\|d_{left}(P_{mid}) - d_{right}(P_{mid})\| < \epsilon$ **then**
         13. $MP \leftarrow P_{\text{mid}}$
      end
   end
end
14. **if** *No MP found* $\wedge$ $d < d_{max}$ **then**
   15. Find first i and last j where $signTable$ changes
   16. RecursiveSearch($[P_i, P_j]$, $d + 1$)
end
17. **Launch:**
   RecursiveSearch(ML, initial depth $d = 0$)
return MP

Based on Theorem 2 in the supplementary material, we have the conclusion that if the indices of the closest triangular meshes to the left and right faces are identical for two sampled points, then all points along the connecting segment will have their closest distances to the same triangular meshes. Combining this with Theorem 1, a mid-point must exist between two sample points with sign changes, allowing for efficient localization of this mid-point via binary search.

The most computation-intensive part of the above stage is the calculation of sampled points in each iteration. Since there is no data dependency between these processes, we accelerated them in parallel with multiple CPU threads using OpenMP [30]. Due to potential variations in the shortest distance calculations for different sampled points, dynamic task scheduling was employed, allowing threads to dynamically claim task blocks and further balance the workload. Additionally, since multiple threads simultaneously write to adjacent memory locations (contiguous elements in $signTable$ and $triIDTable$), we manually allocated independent local storage for each thread, which was merged into the global array, thereby eliminating the effects of false sharing.

Surface Fitting and Trimming. After computing all mid-points, those belonging to a face pair are fitted to a single mid-surface. NURBS surface fitting is a classical problem [31]. In our method, we use default fitting parameters for all FGPs, specifically bicubic with continuous second derivatives. The surface's u and v directions both have a degree of 3, and maintain C^2 continuity to ensure smoothness. Additionally, the surface is kept open in both parametric directions to facilitate trimming. For robustness, we extend the NURBS surface to preserve correct topology between adjacent mid-surfaces.

The trimming operation primarily involves selecting or discarding surface patches through imprinting and intersecting pro-

cesses. The first step is to obtain the trim curves, which are derived from three operations: projecting the edges of the left and right faces of an FGP, determining the intersections between the mid-surfaces, and finding the intersections between the mid-surface and the lateral faces. These curves are then imprinted onto the original mid-surface, dividing it into a new set of mid-surface patches. Next, perform projection and intersection operations between face pairs to determine which mid-surface patches overlap with the face pairs. Retain these overlapping mid-surface patches and discard the unnecessary ones. Finally, rebuild the topological relationships to generate the final mid-surface.

4. Implementation and Results

The proposed algorithm was implemented as a prototype system in C/C++, utilizing Parasolid V35 [13] and Visual Studio 2022 on a Windows 11 (64-bit) platform. The system was executed on a 3.4GHz Intel Core i7-14700KF CPU with 28 cores and 32GB of RAM.

The system takes solid models as input and requires no user interaction. Algorithm efficacy was validated through topological and geometric accuracy of the extracted mid-surfaces, as well as its computational efficiency.

Extensive testing used diverse solid models from Onshape's public dataset[1], with six iconic models chosen as benchmarks to demonstrate specific algorithmic capabilities. These models highlight challenges in automated mid-surface abstraction, particularly where existing methods fail due to topological or geometric constraints, except for Model 1.

- **Model 1**: A plastic thin-shell model with constant wall thickness, consisting of 37 faces and no multi-face pairing scenarios, used to validate the algorithm's basic capabilities.
- **Model 2**: A ceramic bowl with 17 faces, including 8 variable-thickness face pairs. This model represents a simple case with only 1-1 or 1-n face pairs.
- **Model 3**: A turbo impeller, comprising 91 faces, most of which involve 1-n or n-n pair faces. The freeform surfaces exhibit more complex variations, challenging the algorithm's handling of intricate geometry.
- **Model 4**: The original version of Model 1 without feature suppression, containing both constant and variable wall thickness scenarios with 78 faces. It better represents the true mid-surface of Model 1.
- **Model 5**: A tire model from the automotive industry, featuring 167 faces, including 28 variable-wall thickness face pairs and 27 n-n face pairs. Its complex topology and diverse variable wall thickness scenarios pose significant challenges.
- **Model 6**: An aircraft model from a real-world aerospace scenario, with 286 faces and highly complex topological and geometric structures, used to verify the algorithm's robustness under the most challenging conditions.

[1] https://cad.onshape.com/documents?nodeId=3&resourceType=filter

4.1. Correctness and Accuracy

Correctness of the mid-surface topology. We demonstrate the superiority of our method through benchmarks, with results shown in Fig. 8. Model 1, a simple constant wall thickness model, can be extracted by previous methods, and *MidSurfer* also correctly extracts the mid-surface, demonstrating the algorithm's support for classic cases. Fig. 8-b illustrates a 1-n variable wall thickness scenario, revealing numerous fillets for transitions. Traditional methods often suppress these transition features, even though they significantly impact simulation results, and fail to handle them correctly. *MidSurfer* preserves these transition features to ensure topological accuracy. Fig. 8-c presents a similar situation, but the model features more n-n face pairs. The connection between the fan blade and the base, shown in the figure, includes many fillet transitions, which may lead to incorrect topological boundaries in the generated mid-surface. By using the soft normal and overlap criteria, *MidSurfer*'s n-n pairing effectively extracts a more complete topology, providing smoother transitions, reducing local topological discontinuities, and reflecting a more accurate model topology.

Fig. 8-d to Fig. 8-f showcase more complex topological scenarios, with models containing mixed regions and intricate connections. Model 4 is the pre-feature-suppression version of Model 1, and by comparing the mid-surfaces generated by both, Fig. 8-d better reflects the true topology of the model. Models 5 and 6 are real-world industrial models, where details reveal numerous slanted or tapered surfaces and corresponding small patches. Such mixed features are prone to generating incorrect mid-surfaces. In these cases, initial face pairing often deviates significantly from the true results. However, through iterative searches of adjacent faces, the final face pairs are completed, yielding accurate mid-surfaces, as shown in Fig. 8-f.

Geometrical accuracy analysis. We conducted geometric accuracy tests on the benchmarks, noting that each model includes both constant and variable wall thickness scenarios, with all statistical data focusing on the variable wall thickness cases. Fig. 8 displays the final mid-surface results for all benchmarks, demonstrating that *MidSurfer* consistently produces satisfactory outcomes across various scenarios. To measure the geometric accuracy of the mid-surfaces, this study employs relative error (R-error) [29], defined as follows:

$$R(e) = \frac{|d_{\text{left}} - d_{\text{right}}|}{t}. \tag{7}$$

Here, d_{left} and d_{right} represent the distances from the sample point to the left and right FGPs, respectively, and t denotes the average thickness of an FGP. Given the potentially large number of sample points, statistical results of geometric errors are used. Sample points are obtained by uniformly sampling the mid-surface, and several error intervals are defined. The statistical results of geometric errors for mid-surfaces generated by different benchmarks are listed in Table 1. Most sample points fall within the $[0, 0.5‰]$ interval, indicating the algorithm's high geometric accuracy. However, Models 3 and 6 exhibit the poorest accuracy, with nearly half of the sampled points falling within the $[1, 1\%]$ interval. This is attributed to the significant curvature changes in the left and right faces of each face pair in Model 3 (Fig. 8-c), leading to partial geometric errors in the fitted mid-surface. Due to boundary effects, these errors are mostly concentrated in the interior rather than the edges. Model 6's errors stem from its complexity, featuring 37 variable wall thickness face pairs, and as shown in Fig. 8-f, the wing has a large dorsal area, which also affects the geometric accuracy.

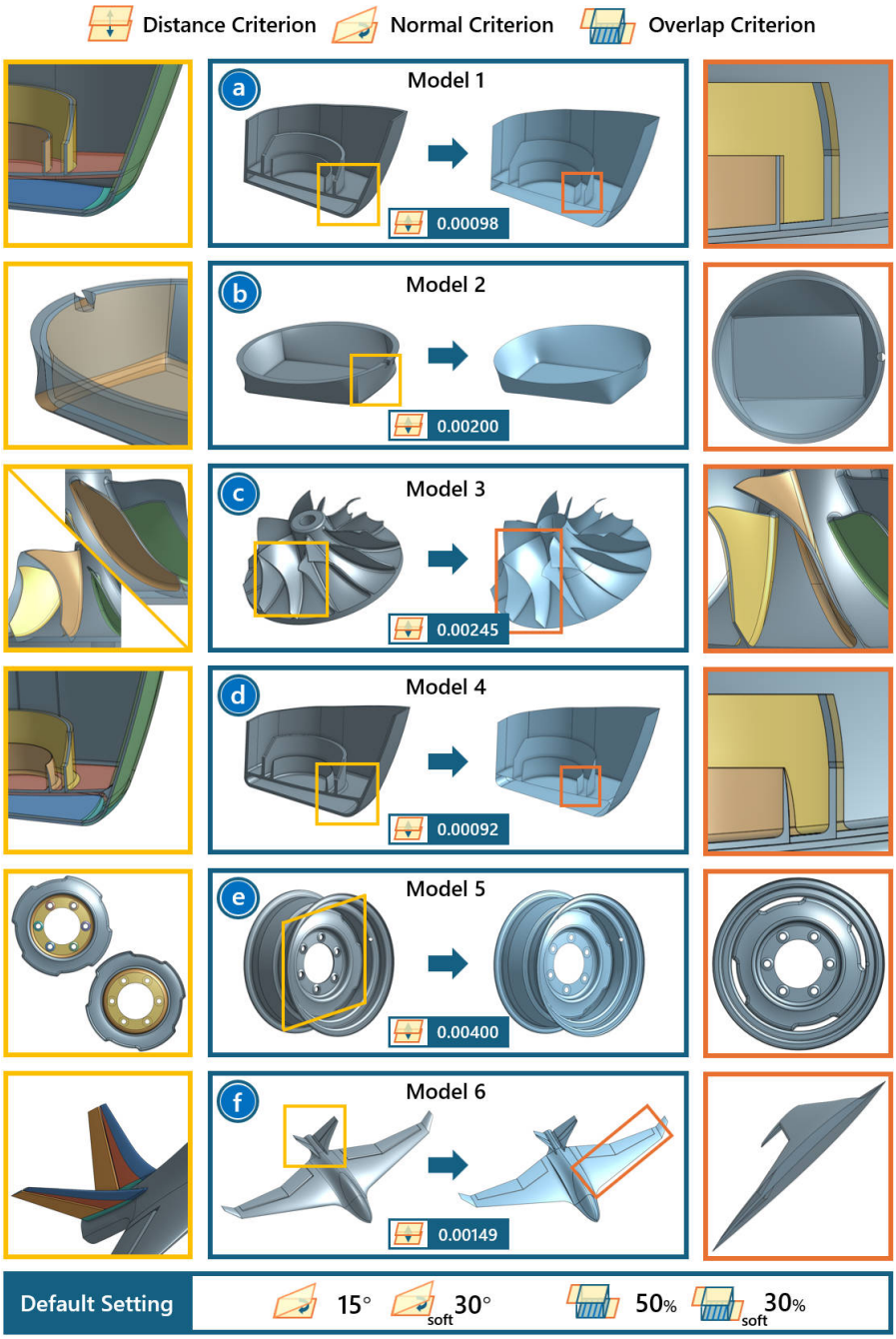

Figure 8: The mid-surface results for each benchmark. The original model on the left and its corresponding mid-surface on the right. Yellow boxes denote topological analysis, while red boxes indicate geometric analysis. Different face pairs are highlighted using varied colors for clarity.

In contrast, *MidSurfer* has better support for these types of models due to the relative simplicity of the model (Fig. 8-b), the clarity of the topology (Fig. 8-d), and the small rate of curvature change in the face of variable wall thickness (Fig. 8-e). Notably, Model 4 achieves exceptionally high precision (94% of points in the $[0, 0.5]$ interval). Since Model 4 represents Model 1 before feature suppression, this demonstrates that our method can achieve high-quality mid-surfaces even without feature suppression. It is acknowledged that, due to the use of fitted surfaces, some points inevitably fall into larger error intervals ($[1, 20\%]$), such as Models 3, 5, and 6. These are inherent fitting errors that are difficult to eliminate but remain small effects, ensuring the overall quality of the final mid-surface results.

4.2. Performance

We further conducted efficiency tests on various stages of the algorithm, with the main results presented in Table 2, including face pairing, the two primary steps of the mid-surface geometric extraction, and the total elapsed time.

Face Pairing. In the face pairing stage, all benchmarks demonstrated strong performance. For models with simple topological structures, such as Models 1, 2, and 4, preliminary pairing sufficed to achieve relatively accurate face pairs. Comparing the pairing times of Models 1 and 4 reveals that the pre-feature-suppression model, due to its variable wall thickness scenarios, was 3.6X slower than the post-suppression model, indicating that repairing variable wall thickness face pairs consumes significant time. Similarly, for the remaining models, the presence of numerous transitional structures or increased complexity necessitated subsequent result complementation to achieve acceptable pairing results. Although Model 3 has fewer faces, each blade involves n-n face pairs (Fig. 8-c), requiring more time for classification and complementation. For Models 5 and 6, their

Table 1: The statistical results for R-errors of different benchmarks (%).

R-error	[0, 0.5]	[0.5, 1]	[1, 1%]	[1%, 10%]	[10%, 20%]	[20%, 1]
Model 2	78.57	14.29	7.14	0	0	0
Model 3	42.15	42.15	6.89	5.76	3.05	0
Model 4	94.21	5.53	0.26	0	0	0
Model 5	67.63	20.20	6.41	5.45	0.32	0
Model 6	44.65	40.07	6.11	5.46	3.71	0

Note: The data in the table represents the percentage of sample points within a specific error interval relative to the total number of sample points; Model 1, being a constant wall thickness model, is not listed in the table.

Table 2: The statistical results of efficiency for different benchmarks (s).

	M1	M2	M3	M4	M5	M6
Face Pairing	0.76	0.34	3.13	2.72	9.13	18.41
Medial Line Gen.	-	<0.01	0.03	0.01	0.09	0.17
Mid-Point Ext.	-	0.03	0.25	0.09	0.34	0.59
Total (s)	1.31	0.61	5.36	4.35	15.73	28.76

inherent complexity further exacerbated the time consumption. Since the face pairing method involves iterative pairing for each face, model complexity significantly impacts the algorithm's efficiency. For instance, Models 5 and 6 have $2.1X$ and $3.7X$ the number of faces compared to Model 4, respectively, but their face pairing times increased by $3.3X$ and $6.8X$, accounting for a substantial portion of the total extraction time. It is acknowledged that the face pairing method, which heavily relies on Parasolid API calls and exhibits data dependencies, is challenging to parallelize. Nevertheless, given the precise face pairing results our method achieves for variable wall thickness models, the time cost is deemed acceptable.

Mid-surface Geometry Extraction. We calculated the time for two steps: Medial Line Generation and Mid-point Extraction. Note that all times represent the total time for all variable wall thickness face pairs in the model. It can be observed that the time for these two steps constitutes a minimal portion of the total time, highlighting the advantages of our algorithm.

Since the core idea of the algorithm is to collect precise mid-points in space, it exhibits excellent parallelism. We employed parallel acceleration in both steps, achieving speedups of $3-4X$ and $2-4X$ in the medial line extraction and mid-point extraction, respectively, compared to single-threaded implementations. All models demonstrated good efficiency (except for Model 1, which has constant wall thickness and thus no reported time). Moreover, the geometric extraction time is proportional to the model's complexity. Models 3, 5, and 6, with 13, 28, and 37 variable thickness face pairs, respectively, required extraction times of $0.25s$, $0.34s$, and $0.59s$, further proving the algorithm's strong parallel performance. Furthermore, we evaluated the computational efficiency based on the number of sampling points and the use of threads, as detailed in the supplementary material.

By examining the total time, it is evident that some unlisted time is consumed, primarily for fitting and offsetting surfaces. Since these operations are performed directly using Parasolid APIs, they are not included in the table. Given that mid-surfaces are primarily used for subsequent analysis tasks, which are often one-time and do not require real-time interaction, the current total time is regarded as reasonable.

5. Comparison and Analysis

To analyze the usability and generalizability of the mid-surfaces generated by *MidSurfer* compared to previous methods, we conducted a series of comparative studies. Since there is currently no dedicated dataset for mid-surface abstraction tasks, we manually created a dataset focused on thin-walled parts. The dataset comprises 213 representative models comprising 7894 faces. Some models from the dataset and their extracted mid-surfaces are shown in the supplementary material.

Based on this dataset, we conducted three comparative experiments to evaluate the geometric accuracy, performance, and generalizability of *MidSurfer* compared to other methods.

Experimental Setup. We implemented the corresponding comparison methods. For face pairing, we replicated the virtual decomposition proposed by Woo et al. [12], where the algorithm for obtaining concave edges was implemented by Parasolid API. For the geometric extraction, we replicated the methods of CAT and Zhu et al. [29], with the method by Quadros et al [18] being used as a comparison version of CAT. Notably, when comparing CAT, Zhu et al., and *MidSurfer*, we used the same discretization results as the input for all methods to ensure consistent parameter settings. The Parasolid [13] results were directly obtained by API calls, with the method set to "Medial" and the tolerance set to $1E-5$. Throughout the experiments, *MidSurfer* maintained consistent experimental settings, including the use of an adaptive distance criterion in the face pairing stage, along with default normal (default: 15, soft: 30) and overlap (default: 50%, soft: 30%) criteria. In the geometric extraction stage, the sampling density for medial line generation was set to 1X densification, with 16 sampling points for mid-point extraction (as indicated in Supplementary Materials, Section 2), and the maximum recursion depth was set to 5.

Measures. For the geometric accuracy, we used the R-error (Equation 7) as the metric and compared *MidSurfer* with traditional CAT, Zhu et al. [29], and Parasolid [13] methods. Note that we only included variable wall thickness mid-surfaces that these methods could generate from the dataset. The main results are presented in Table 3. Similarly, to validate the generalizability of *MidSurfer*, we analyzed the completion rates of various methods in the face pairing and geometric extraction stages across the dataset, as detailed in Table 5. For face pairing, we manually intervened to determine whether the generated mid-surfaces had topological omissions or additions compared to the original model. For geometric extraction, we categorized the results into the number of extractable face pairs and the number of models that could be fully abstracted. A face group pair (FGP) was considered extractable if more than 50% of its sampled points fell within the [0, 10‰] R-error interval. A model was deemed correctly abstracted if all its FGPs met this criterion.

Table 3: The statistical results for R-errors of different methods (%), with the default version of each method set to 1X densification.

R-error	[0, 0.5‰]	[0.5‰,1‰]	[1‰, 1%]	[1%, 10%]	[10%,20%]	[20%,1]
CAT	0.49	0.57	10.66	**46.56**	40.59	1.14
CAT (Densified, 3X)	0.66	0.57	12.63	43.84	**41.23**	1.05
CAT (More densified, 10X)	0.88	0.64	13.50	43.91	40.12	0.96
Zhu et al. [29]	0.99	1.50	9.34	21.30	6.92	**59.94**
Zhu et al. [29] (Densified, 3X)	10.03	7.88	**20.30**	3.46	3.02	55.31
Zhu et al. [29] (More densified, 10X)	15.88	13.15	11.15	2.09	3.45	54.28
Parasolid [13]	38.52	**43.28**	3.74	4.95	8.72	0.78
MidSurfer	62.44	24.49	8.36	3.33	1.42	0
MidSurfer (Densified, 3X)	68.25	20.55	8.22	2.42	0.56	0
MidSurfer (More densified, 10X)	**70.89**	19.34	7.14	2.21	0.42	0

Table 4: The performance of different methods at different stages.

	Face Pairing Time(s)	Face Pairing Speedups	Geometry Extraction Time(ms)	Geometry Extraction Speedups
Woo et al. [12]	39.21	3.11	-	
CAT	-		3.41	-1.44
Zhu et al. [29]	-		60.79	12.38
Parasolid [13]	-		19.89	4.05
MidSurfer (single-threaded)	**12.61**	-	20.94	4.26
MidSurfer (parallel)			**4.91**	-

Table 5: The completion rates of different methods at different stages (%).

	Face Pairing All Models	Geometry Extraction All FGPs	Geometry Extraction All Models
Woo et al. [12]	24.41	-	
CAT	-	36.99	19.87
Zhu et al. [29]	-	20.00	15.02
Parasolid [13]	-	51.99	43.66
MidSurfer	**73.71**	**71.90**	**69.01**

5.1. Comparison with Prior Methods

Geometric accuracy. As shown in Table 3, we first analyzed the improvement in geometric accuracy of *MidSurfer* compared to previous works, specifically the CAT method (Fig. 2) and Zhu et al. [29] method. The mid-surfaces generated by the different methods are displayed in Fig. 9. It can be observed that both the CAT and Zhu's methods can extract mid-surfaces from the original models. However, the generated mid-surfaces are either non-smooth or extend beyond the model boundaries. Further analysis of geometric errors in Table 3 reveals that the CAT method performs the worst, as its geometric errors mainly fall within the [1, 20] interval. This is primarily because the CAT method relies on meshing results, leading to inaccurate mid-points with significant positional oscillations. Zhu's method outperforms the CAT method, with geometric errors mainly within the [1, 10] interval. While this method approximates the exact solution by using projection and intersection, it reports substantial geometric errors in the [20, 1%] interval (> 50% across various sampling densities). This is mainly due to its use of the normal direction for projection, which always fails when the angle between two faces exceeds 15. Since most variable wall thickness models contain faces with varying angles, this leads to considerable statistical errors. In contrast, *MidSurfer* significantly outperforms the other two methods, with nearly all sampled points falling within the [0, 1] interval.

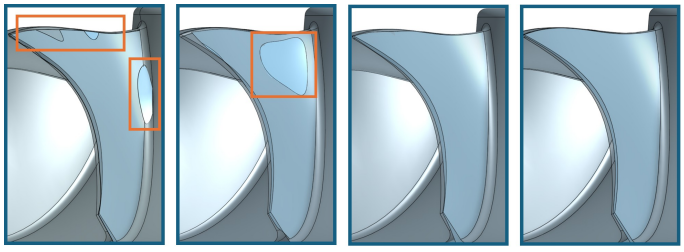

Figure 9: The comparison of geometric accuracy across different methods. Both CAT and Zhu's methods extract mid-surfaces but introduce errors—non-smooth surfaces or boundary extensions—highlighted in red rectangles.

To further demonstrate the impact of sampling density, we conducted additional experiments with different sampling point densities. It can be observed that as the density of points increases, the geometric accuracy of all methods improves, especially for methods reliant on sampling rates, except for the CAT method, which achieves densification by increasing mesh density. However, it is worth noting that strategies to improve geometric accuracy through densification come with high computational costs. In our method, due to its parallel nature, the time complexity increases linearly with the sampling rate, so we default to only 1X densification. Based on this, densification strategies for improving geometric accuracy are not ideal. The proposed method in this study significantly enhances the geometric accuracy of mid-surface models at a lower computational cost.

Performance. Table 4 provides a comparison of the efficiency across different methods. In the face pairing stage, Woo et al. [12] involves a virtual decomposition method. For each model, the concave edges are first identified, followed by the extension and intersection of these edges to generate several half-spaces. These half-spaces are then merged into convex cells before entering the pairing operation. This process heavily depends on basic geometric operations and iterative calls, leading to poor performance. Additionally, this method only applies to constant wall-thickness models; for variable wall-thickness models, it may produce abnormal intersection partitions, generating a vast number of half-spaces that either cause the program to crash or result in a sharp increase in computation time. In contrast, while *MidSurfer* also takes longer, it still outperforms Woo et

al.'s method. This is mainly because *MidSurfer* effectively utilizes face-adjacent information and eliminates the need for secondary decomposition. This feature is particularly beneficial for models with variable wall thickness. Furthermore, *MidSurfer*'s more targeted pairing criteria save significant computational effort, yielding an average performance improvement of 3.11X.

In the geometric extraction stage, *MidSurfer* was compared to both the CAT [18] and Zhu et al.'s [29] methods. The results show that even *MidSurfer*'s parallel version struggles to match CAT's performance, primarily due to CAT's simplicity in computation. The core principle of CAT involves systematically shifting vertices and meshes according to predefined rules for each discretized mesh form, reducing computational costs significantly. However, when compared to Zhu et al.'s method, *MidSurfer* demonstrated a significant acceleration in computation. This is because Zhu et al.'s method requires projection and intersection for each sampling point, resulting in a substantial computational burden, which *MidSurfer* efficiently reduces, achieving an average 12.38X speedup. Additionally, we tested *MidSurfer*'s performance with and without parallelization. The results showed that *MidSurfer*'s design is well-suited for parallel computation, achieving a 4.26X speedup.

Generalizability. Further analysis of *MidSurfer*'s generalizability across various models, as shown in Table 5, reveals that for face pairing, the traditional Woo et al. [12] method correctly identifies only 24.41% of the models in the dataset, while *MidSurfer* achieves 73.31% accuracy. This is because Woo's method only supports constant wall thickness face pairing and all FGP types are 1-1 face pairs, making it largely unable to reflect the topological features of the models. In contrast, *MidSurfer*'s enhanced face pairing capabilities effectively handle most real-world models. In the geometric extraction stage, the CAT and Zhu's methods perform poorly as they cannot handle 1-n/n-n FGPs. However, since we consider the $[0, 10]$ interval as correct geometric mid-surfaces, the CAT method performs better than Zhu's method. *MidSurfer* demonstrates excellent performance across all FGPs and models, with success rates of 71.9% and 69.01%, respectively. This highlights the method's ability to effectively extract most variable wall thickness face pair scenarios and its robust performance across all models, even though there are cases where *MidSurfer* cannot extract mid-surfaces (discussed in limitations). Overall, *MidSurfer* exhibits strong performance across the board.

5.2. Comparison with Parasolid

We also compared *MidSurfer* with the current commercial solution provided by Parasolid [13]. It is worth noting that Parasolid's method outperforms all existing algorithms, achieving a geometric accuracy of over 80% in the $[0, 1]$ interval (Table 3). Due to this advantage, its geometric extraction completion rate in real-world scenarios is higher than that of the CAT and Zhu et al, as shown in Table 5 and Fig. 9. Similarly, its performance shows significant advantages, with improvements of 1.05X and 3.06X compared to the single-threaded versions of *MidSurfer* and Zhu et al, respectively, as shown in Table 4. Nevertheless, *MidSurfer* still holds its own advantages, particularly in the $[0, 0.5]$ interval (*MidSurfer*: 70.89%, Parasolid: 38.52%), performance ($MidSurfer_{parallel}$: 4.91s, Parasolid: 19.89s, Speedup: 4.05X), and in terms of extraction rates for all FGPs (*MidSurfer*: 71.90%, Parasolid: 51.99%) and all models (*MidSurfer*: 69.01%, Parasolid: 43.66%).

However, the Parasolid API also has some limitations. Firstly, Parasolid cannot automatically identify left and right faces; users must manually select and input all left and right faces into the API for extraction. Moreover, as the number of left and right faces increases, the pairing results are often incorrect. Secondly, Parasolid's adaptability for geometric extraction is suboptimal. For example, swapping the left and right faces for the same face pair often results in extraction failure (report error codes). Additionally, Parasolid struggles with scenarios involving quadric surfaces (cylinders, cones, spheres, etc.) and freeform surfaces, likely due to its reliance on geometric parameters to calculate mid-surfaces. In contrast, *MidSurfer* employs a method that discretizes surfaces, performs parallel mid-point extraction, and fits the final surface, thereby avoiding the aforementioned issues. Based on this, *MidSurfer* demonstrates superior geometric accuracy and model generalizability compared to both previous methods and commercial solutions.

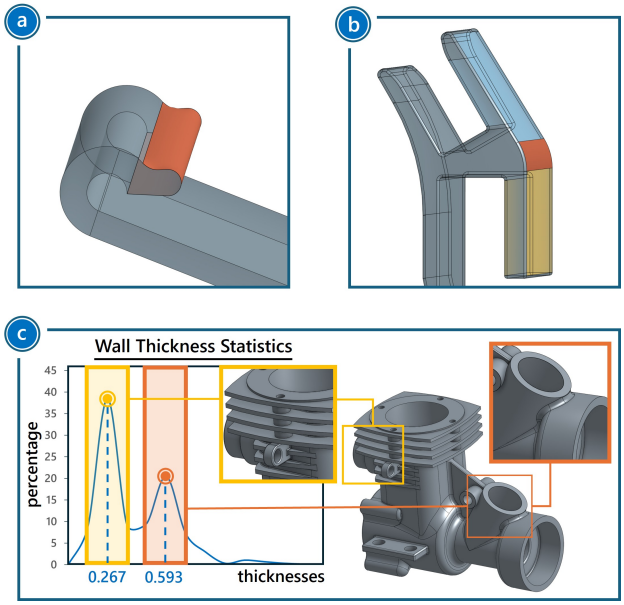

Figure 10: *MidSurfer* abstraction failure cases: a) Boundary single faces, b) Topological deficiency, and c) Multiple wall thickness peaks.

6. Conclusion and Limitations

We introduce *MidSurfer*, a mid-surface abstraction method for variable-thickness thin-walled models. Our approach leverages an improved face-pairing method with extended face types and a parallelized geometric extraction strategy based on two-step mid-point extraction, enabling rapid and efficient generation of high-quality simplified models.

Experiments on six real-world benchmarks demonstrate the accuracy and efficiency of *MidSurfer*. Additionally, studies on a manually constructed dataset of 213 thin-walled models further highlight *MidSurfer*'s superior usability and generalizability compared to existing methods. To our knowledge, no publicly available geometric modeling algorithm has previously achieved satisfactory results for this specific task while simultaneously improving performance. Thus, our method makes a valuable contribution by addressing this challenge. Despite these advancements, it has several limitations that warrant further exploration.

Abstraction failures. While *MidSurfer* delivers accurate results in most scenarios, it still exhibits limitations in following cases:

- Boundary single faces: it fails to generate mid-surfaces for boundary cases with single faces (Fig. 10-a). A potential

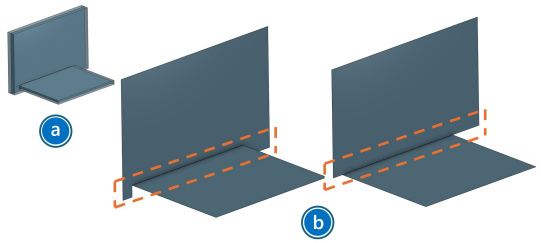

Figure 11: The cases that *MidSurfer* cannot address, with the red parts indicating the unprocessable scenarios.

solution is to segment the single face and form a new FGP, but determining the segmentation boundaries is non-trivial.

- Topological deficiency: for variable-thickness transitions with rib structures (Fig. 10-b), partial face mispairing may occur, leading to topological discontinuities at junctions. Extending adjacent FGP's mid-surface could mitigate this issue, but the loss of fine topological details may impact subsequent analysis tasks.

- Multiple wall thickness peaks: In cases where the model has multiple local wall thicknesses (Fig. 10-c), *MidSurfer* cannot automatically determine the wall thickness threshold for the distance criterion to perform face pairing. For such models, using the smaller wall thickness peak by default generally yields better results. However, selecting a smaller wall thickness may prevent face pairs with larger thicknesses from being matched, while choosing a larger wall thickness may result in mis-matching of non-face pairs (such as lateral faces). The current solution is that *MidSurfer* returns all candidate wall thickness peaks and their pairing results, allowing users to manually select the most accurate one.

Precision. Precision limitations arise from topological and geometric aspects. Topologically, while mid-surface extraction should be unique under thin-wall criteria, conflicting topological semantics may emerge. Fig. 11 illustrates a simplified example: *MidSurfer*'s result (Fig. 11-a) is algorithmically correct but may not align with user expectations, as alternative mid-surfaces (Fig. 11-b) are also valid. This issue stems from the face-pairing stage, suggesting the need for broader pairing rules to offer multiple topology-semantic options. Geometrically, while *MidSurfer* achieves high accuracy for most models, fitting errors accumulate when an FGP contains too many faces, drastically reducing geometric fidelity. Additionally, precision loss during the discretization is inevitable. Addressing these issues will be crucial for future improvements.

Performance. Efficiency remains a key concern. In the face-pairing stage, multi-loop judgments account for most of the computational cost. To enhance performance, an optimized face adjacency graph search algorithm [20] for variable wall thickness should be considered. In the geometric extraction stage, although the method benefits from data-independent parallelism, current CPU-based parallelization has room for improvement. GPU acceleration presents a promising avenue, particularly for parallel searches [32] and distance computing [33], which are integral to our algorithm. Exploring these optimizations could further advance mid-surface abstraction techniques.

References

[1] P. Dabke, V. Prabhakar, S. Sheppard, Using features to support finite element idealizations, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 13805, American Society of Mechanical Engineers, 1994, pp. 183–193.

[2] C. G. Armstrong, Modelling requirements for finite-element analysis, Computer-aided design 26 (7) (1994) 573–578.

[3] S. Bouix, J. C. Pruessner, D. L. Collins, K. Siddiqi, Hippocampal shape analysis using medial surfaces, Neuroimage 25 (4) (2005) 1077–1089.

[4] Y. Li, W. Wang, X. Liu, Y. Ma, Definition and recognition of rib features in aircraft structural part, International Journal of Computer Integrated Manufacturing 27 (1) (2014) 1–19.

[5] A. Thakur, A. G. Banerjee, S. K. Gupta, A survey of CAD model simplification techniques for physics-based simulation applications, Computer-Aided Design 41 (2) (2009) 65–80.

[6] R. Donaghy, C. G. Armstrong, M. A. Price, Dimensional reduction of surface models for analysis, Engineering with computers 16 (2000) 24–35.

[7] M. Ramanathan, B. Gurumoorthy, Generating the mid-surface of a solid using 2D mat of its faces, Computer-Aided Design and Applications 1 (1-4) (2004) 665–674.

[8] W. R. Quadros, An approach for extracting non-manifold mid-surfaces of thin-wall solids using chordal axis transform, Engineering with computers 24 (3) (2008) 305–319.

[9] D. C. Nolan, C. M. Tierney, C. G. Armstrong, T. T. Robinson, J. E. Makem, Automatic dimensional reduction and meshing of stiffened thin-wall structures, Engineering with Computers 30 (2014) 689–701.

[10] M. Rezayat, Midsurface abstraction from 3D solid models: general theory and applications, Computer-Aided Design 28 (11) (1996) 905–915.

[11] D.-P. Sheen, T.-g. Son, C. Ryu, S. H. Lee, K. Lee, Dimension reduction of solid models by mid-surface generation, International Journal of CAD/CAM 7 (1).

[12] Y. Woo, Abstraction of mid-surfaces from solid models of thin-walled parts: A divide-and-conquer approach, Computer-Aided Design 47 (2014) 1–11.

[13] S. D. I. Software, Parasolid, `https://plm.sw.siemens.com/en-US/plm-components/parasolid` (2022).

[14] H. Blum, A transformation for extracting new descriptions of shape, Models for the perception of speech and visual form (1967) 362–380.

[15] D. J. Sheehy, C. G. Armstrong, D. J. Robinson, Computing the medial surface of a solid from a domain Delaunay triangulation, in: Proceedings of the third ACM symposium on Solid modeling and applications, 1995, pp. 201–212.

[16] Y.-G. Lee, K. Lee, Computing the medial surface of a 3-D boundary representation model, Advances in Engineering Software 28 (9) (1997) 593–605.

[17] L. Prasad, Morphological analysis of shapes, CNLS newsletter 139 (1) (1997) 1997–07.

[18] W. R. Quadros, K. Shimada, Hex-layer: Layered all-hex mesh generation on thin section solids via chordal surface transformation., in: IMR, 2002, pp. 169–180.

[19] K. Kwon, B. C. Lee, S. Chae, Medial surface generation using chordal axis transformation in shell structures, Computers & structures 84 (26-27) (2006) 1673–1683.

[20] H. Lee, Y.-Y. Nam, S.-W. Park, Graph-based midsurface extraction for finite element analysis, in: 2007 11th International Conference on Computer Supported Cooperative Work in Design, IEEE, 2007, pp. 1055–1058.

[21] D.-P. Sheen, T.-g. Son, D.-K. Myung, C. Ryu, S. H. Lee, K. Lee, T. J. Yeo, Transformation of a thin-walled solid model into a surface model via solid deflation, Computer-Aided Design 42 (8) (2010) 720–730.

[22] C. S. Chong, A. S. Kumar, K. Lee, Automatic solid decomposition and reduction for non-manifold geometric model generation, Computer-Aided Design 36 (13) (2004) 1357–1369.

[23] H. Zhu, Y. Shao, Y. Liu, J. Zhao, Automatic hierarchical mid-surface abstraction of thin-walled model based on rib decomposition, Advances in Engineering Software 97 (2016) 60–71.

[24] T. Robinson, C. Armstrong, R. Fairey, Automated mixed dimensional modelling from 2D and 3D CAD models, Finite elements in analysis and design 47 (2) (2011) 151–165.

[25] Y. H. Kulkarni, A. Sahasrabudhe, M. Kale, Computation of midsurface by feature-based simplification–abstraction–decomposition, Journal of Computing and Information Science in Engineering 17 (1) (2017) 011006.

[26] Y. H. Kulkarni, A. Sahasrabudhe, M. Kale, Leveraging feature generalization and decomposition to compute a well-connected midsurface, Engineering with Computers 33 (1) (2017) 159–170.

[27] J. R. Shewchuk, Delaunay refinement mesh generation, Carnegie Mellon University, 1997.

[28] J. R. Shewchuk, Delaunay refinement algorithms for triangular mesh generation, Computational geometry 22 (1-3) (2002) 21–74.

[29] H. Zhu, Y. Shao, Y. Liu, C. Li, Mid-surface abstraction for complex thin-

wall models based on virtual decomposition, International Journal of Computer Integrated Manufacturing 29 (8) (2016) 821–838.
[30] OpenMP ARB, OpenMP, https://www.openmp.org/ (2024).
[31] L. Piegl, W. Tiller, The NURBS Book, Springer Science & Business Media, 2012.
[32] H. Zhu, Y. Liu, H. Wang, J. Zhao, Efficient construction of the medial axis for a CAD model using parallel computing, Engineering with Computers 34 (2018) 413–429.
[33] P. Fan, W. Wang, R. Tong, H. Li, M. Tang, gDist: efficient distance computation between 3D meshes on GPU, in: SIGGRAPH Asia 2024 Conference Papers, 2024, pp. 1–11.

MidSurfer: Efficient Mid-surface Abstraction from Variable Thin-walled Models Supplementary Material

1. Theorems and Lemmas

Theorem 1. *Let S_{left} and S_{right} be two surfaces in FG_{left} and FG_{right}, respectively, and let L be a line intersecting both surfaces at points $A \in S_{left}$ and $B \in S_{right}$. There exists at least one point $Q_0 \in L$ between A and B such that Q_0 lies on the mid-surface between S_{left} and S_{right}.*

Proof. For any point $Q \in L$, define the shortest distance functions:

$$\mathrm{Dist}(Q, S_{left}) = \inf_{P \in S_{left}} \|Q - P\|, \quad \mathrm{Dist}(Q, S_{right}) = \inf_{P \in S_{right}} \|Q - P\|. \tag{1}$$

S_{left} and S_{right} are closed sets, so their distance functions $\mathrm{Dist}(Q, S_{left})$ and $\mathrm{Dist}(Q, S_{right})$ are continuous over $\mathbb{R}^3$, thus both distance functions are continuous on L. Construct the function $f(Q) = \mathrm{Dist}(Q, S_{left}) - \mathrm{Dist}(Q, S_{right})$. At endpoints:

$$f(A) = \mathrm{Dist}(A, S_{left}) - \mathrm{Dist}(A, S_{right}) = 0 - \mathrm{Dist}(A, S_{right}) < 0, \tag{2}$$

$$f(B) = \mathrm{Dist}(B, S_{left}) - \mathrm{Dist}(B, S_{right}) = \mathrm{Dist}(B, S_{right}) - 0 > 0, \tag{3}$$

The function $f(Q)$ is continuous on the line segment $[A, B] \subset L$. Since $f(A) < 0$ and $f(B) > 0$, by the Intermediate Value Theorem, there exists at least one point $Q_0 \in (A, B)$ such that:

$$f(Q_0) = 0 \implies \mathrm{Dist}(Q_0, S_{left}) = \mathrm{Dist}(Q_0, S_{right}). \tag{4}$$

Thus, Q_0 lies on the mid-surface. □

Lemma 1 (Convex Projection Invariance)**.** *Let T be a convex triangle in M, and let $A, B \in L$ be two points on a line segment such that their closest points on T lie strictly inside T. Then, for any $Q \in AB$, the closest point $P_Q \in T$ to Q also lies strictly inside T.*

Proof. By convexity of T and projective continuity, the line segment $P_A P_B \subset T$. Parametrize $Q = (1 - \lambda)A + \lambda B$ and define $P_Q = (1 - \lambda)P_A + \lambda P_B$, by convexity, $P_Q \in T$, and the distance of Q to P_Q follows:

$$\mathrm{Dist}_{\min}(\mathbf{Q}, \mathbf{P_Q}) = Q - P_Q = (1 - \lambda)(\mathbf{A} - \mathbf{P}_A) + \lambda(\mathbf{B} - \mathbf{P}_B). \tag{5}$$

Since

$$\mathbf{A} - \mathbf{P}_A \perp T, \quad \mathbf{B} - \mathbf{P}_B \perp T, \tag{6}$$

and the vector $Q - P_Q$ is a linear combination, it remains **orthogonal** to the T at P_Q, thereby ensuring $\mathbf{P}_Q$ is the closest point to Q on T. □

Theorem 2. *Let M_{left} and M_{right} be triangular meshes generated by two face groups (FG), and let L be a line segment intersecting both meshes. Suppose two points $A, B \in L$ (between the intersections of L with M_1 and M_2) are such that:*

1. *A and B share the same closest triangle indices $T_a \in M_{left}$ and $T_b \in M_{right}$.*
2. *T_a and T_b are convex.*

Then, for any point $Q \in AB$, the closest triangle indices of Q to M_{left} and M_{right} remain T_a and T_b, respectively.

Proof. For M_{left}, by Lemma 1, if the closest points of A and B to T_a lie inside of T_a, then the closest points of all points on the segment AB also lie inside of T_a. If the closest points of A and B lie on the edge or vertex of T_a, then the closest points of points on AB may move along the edge or vertex but still belong to T_a. Therefore, for all points on AB, the index of the closest face to M_{left} is T_a. Same proof for M_{right}. □

Table 1: The statistical results of efficiency at different sampling points and parallelism for different benchmarks (s).

		M2	M3	M4	M5	M6
N=8	parallel	0.031	0.355	**0.068**	0.384	0.644
	single-threaded	0.122	0.973	**0.220**	1.593	2.011
N=16	parallel	**0.029**	**0.252**	0.085	**0.342**	**0.589**
	single-threaded	**0.108**	**0.963**	0.251	**1.363**	**1.801**
N=32	parallel	0.047	0.321	0.105	0.509	0.798
	single-threaded	0.204	1.779	0.368	2.624	3.313

Note: The data in the table represents the results of the Mid-point Extraction stage. Model 1, being a constant wall thickness model, is not listed in the table.

2. Performance of Different Sampling Points

We evaluated the computational efficiency in the mid-point extraction based on the number of sampling points and the use of parallel computing, as detailed in Table 1. For all models, the best efficiency was achieved with 16 sampling points (i.e., $N = 16$) and parallel acceleration (the default setting used in the algorithm). Notably, we pre-tested the impact of different thread counts on the algorithm, revealing only a ±2% variation in efficiency. Thus, we used the default thread count (i.e., the CPU's core count). When $N = 8$, the parallel strategy achieved a speedup of $2.7X - 4.2X$, but the overall time was generally longer than with $N = 16$. This is primarily due to insufficient sampling points leading to increased recursion and more iterations in the binary search. For example, in Model 3, with $N = 8$, the average recursion count was 1.32, and the average binary search count was 6.75, compared to 1.12 and 6.03, respectively, with $N = 16$. Similarly, with $N = 32$, the average speedup ranged from $3.5X$ to $5.5X$, demonstrating enhanced parallelism with more sampling points. However, despite the improved parallel efficiency and favorable recursion (1.03) and binary search counts (4.94), this configuration was slower than the $N = 16$ case across all models. This is mainly because excessive sampling points reduce computational efficiency, as more points with no impact on the final result are calculated. An exception in Table 1 is Model 4, which performed better with $N = 8$ than $N = 16$. This is attributed to

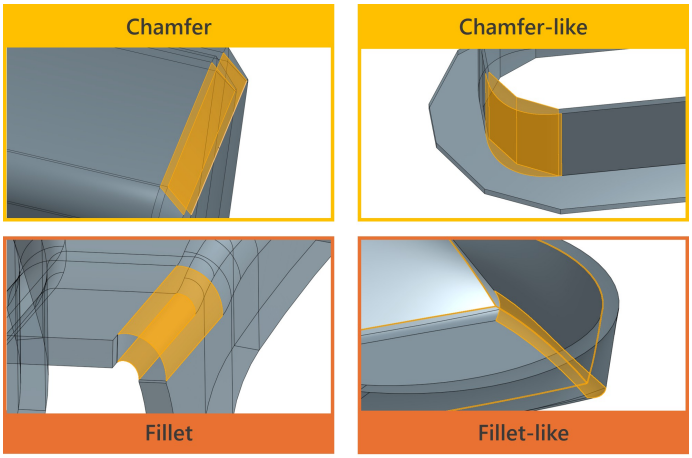

Figure 1: Examples of different transition scenarios, where the highlighted areas are transition areas.

the gentle transitions in Model 4's variable wall thickness (low curvature variation), allowing fewer sampling points to quickly identify triangular meshes with the same shortest distance. Additionally, the model's low discretization level, resulting in larger average triangular mesh areas, made it easier to hit the same meshes. However, this configuration significantly reduces the geometric accuracy of the generated mid-surface. Based upon this, $N = 16$ was finalized as the default value.

3. Construction of Thin-walled Model Dataset

All models were selected from various datasets using keywords such as "thin-walled parts," "sheet metal parts," and "plastic shells." We searched databases including the Onshape public dataset[1], GrabCAD Library [2], and the ABC dataset [1], covering categories such as aerospace, automotive, industrial design, and machine design. After the initial search, we manually curated the collected models, resulting in a dataset of 213 representative models comprising 7894 faces. All models are thin-walled and include various wall thickness types, transition scenarios, and face pair combinations.

Further, the models were manually classified and annotated. To reflect the diversity of the database, the models were divided into four categories based on their corresponding domains: aerospace, automotive, industrial/mechanical design, and others, with the quantity and proportion of each category presented in Table 2. Moreover, To illustrate the dataset's composition, the count of constant and variable wall thickness models were recorded, and categorized into 1-1 and 1-n/n-n FGP types, along with various transition scenarios, such as chamfers(-like), fillets(-like) structures (see Fig. 1). These statistics highlight the dataset's complexity, as shown in Table 3. Moreover, 12 representative models and their extracted mid-surfaces were visually displayed with corresponding model data in Figs. 2-3, showcasing the diversity and complexity of the model dataset across different domains, wall thicknesses, and FGP types, ensuring the dataset's validity for evaluation.

[1] https://cad.onshape.com/documents?nodeId=3&resourceType=filter

[2] https://grabcad.com/library

Table 2: The statistical data for each domain of the model in the dataset.

	Count	Percentage
Aerospace	45	21.12%
Automotive	52	24.41%
Industry/Machine Design	75	35.21%
Others	41	19.25%
Total	213	-

Table 3: The statistical data of each face group pair (FGP) type and transition scenarios in the dataset.

	Constant Wall-thickness		Variable Wall-thickness	
	Count	Percentage	Count	Percentage
1-1 *FGPs*	38	17.84%	30	14.08%
1-n/n-n *FGPs*	27	12.68%	118	55.40%
Chamfer/Fillet (-like)	22	10.33%	121	56.81%
Total	65	30.52%	148	69.48%

References

[1] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, D. Panozzo, ABC: A big CAD model dataset for geometric deep learning, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 9601–9611.

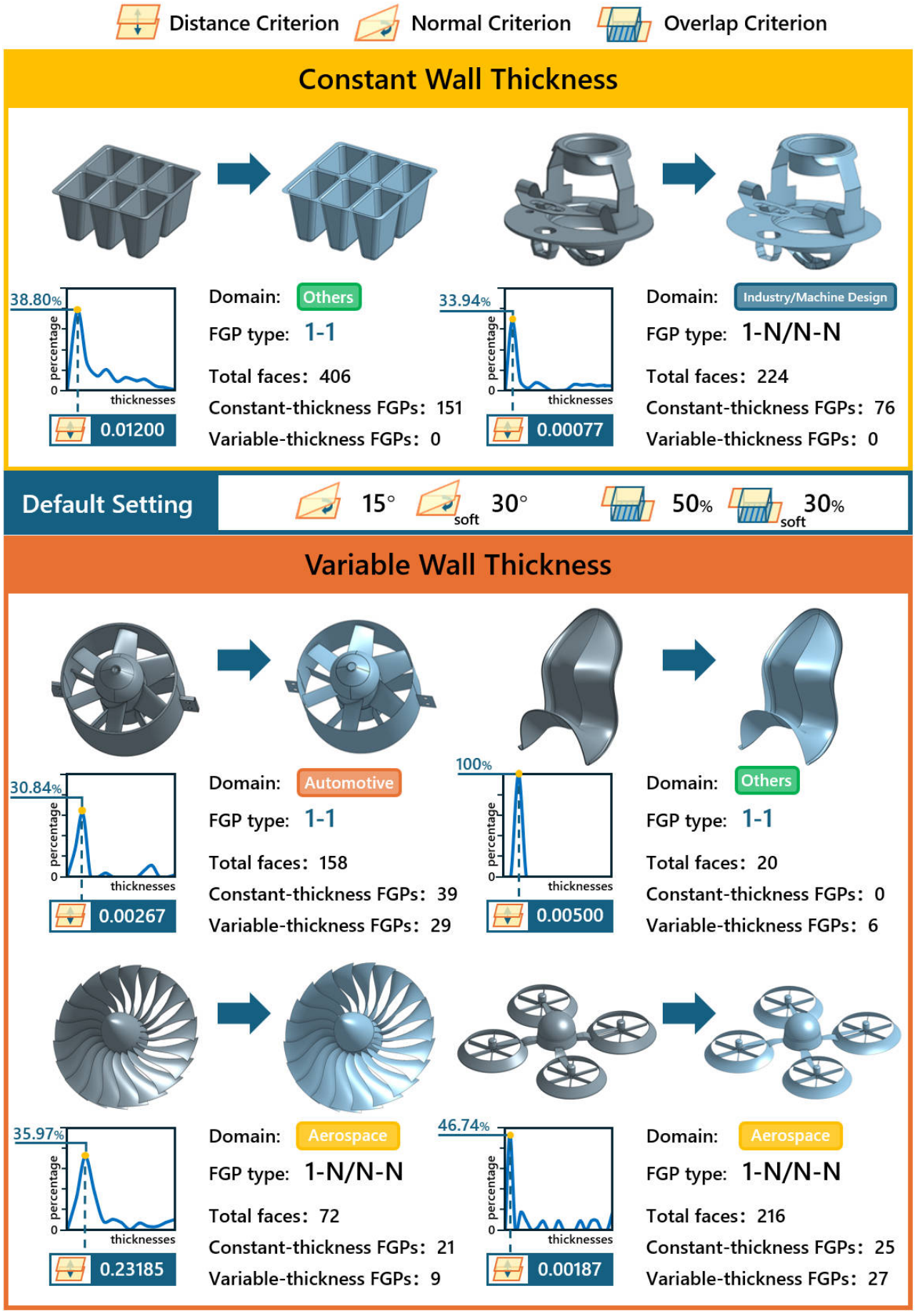

Figure 2: Visual examples of 12 representative models in the dataset. The yellow boxes indicate constant wall-thickness models, and the orange boxes indicate variable wall-thickness models. The domain, FGP type, and parameter settings for each model are listed. The blue boxes show the default parameter settings of our method.

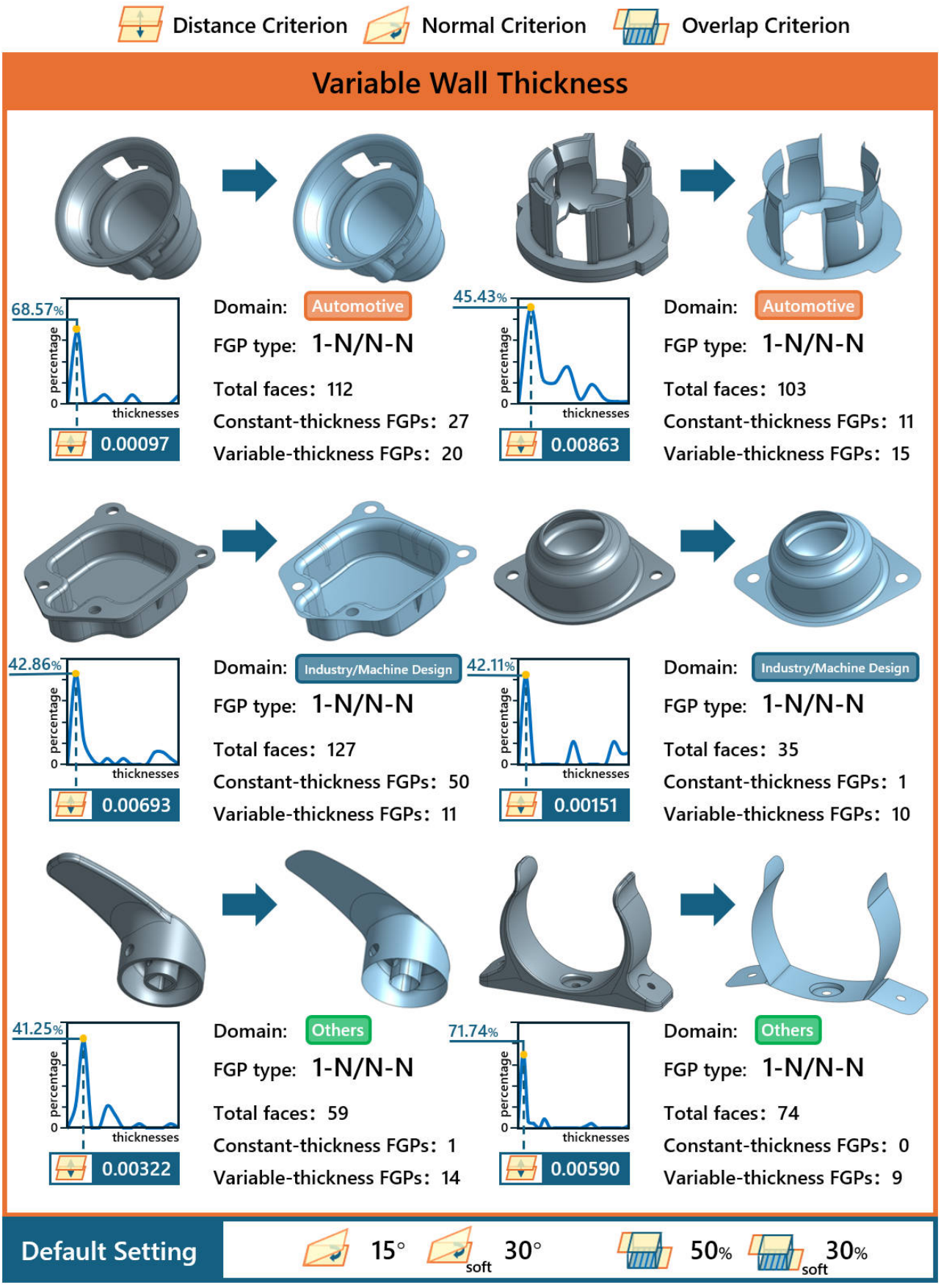

Figure 3: Visual examples of 12 representative models in the dataset. The orange boxes indicate variable wall-thickness models. The domain, FGP type, and parameter settings for each model are listed. The blue boxes show the default parameter settings of our method.