Computer-Aided Design 160 (2023) 103528

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

XVoxel-Based Parametric Design Optimization of Feature Models

Ming Li [a], Chengfeng Lin [a], Wei Chen [a], Yusheng Liu [a], Shuming Gao [a], Qiang Zou [a,b,*]

[a] State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, 310027, China
[b] Fifth Electronic Research Institute of Ministry of Industry and Information Technology (MIIT), MIIT Key Laboratory of Industrial Software Engineering Application Technology, Guangzhou, 510610, China

ARTICLE INFO

Article history:
Received 23 December 2022
Received in revised form 17 March 2023
Accepted 1 April 2023

Keywords:
Parametric feature modeling
CAD modeling
Design optimization
Extended voxels (XVoxels)
Semantic voxels
CAD/CAE integration

ABSTRACT

Parametric optimization is an important product design technique, especially in the context of the modern parametric feature-based CAD paradigm. Realizing its full potential, however, requires a closed loop between CAD and CAE (i.e., seamless CAD/CAE integration) with automatic design modifications and simulation updates. Conventionally the approach of model conversion is often employed to form the loop, but this way of working is hard to automate and requires manual inputs. As a result, the overall optimization process is too laborious to be acceptable. To address this issue, a new method for parametric optimization is introduced in this paper, based on a unified model representation scheme called eXtended Voxels (XVoxels). This scheme hybridizes feature models and voxel models into a new concept of semantic voxels, where the voxel part is responsible for FEM solving, and the semantic part responsible for high-level information to capture both design and simulation intents. As such, it can establish a direct mapping between design models and analysis models, which in turn enables automatic updates on simulation results for design modifications, and vice versa—effectively a closed loop between CAD and CAE. In addition, robust and efficient geometric algorithms for manipulating XVoxel models and efficient numerical methods (based on the recent finite cell method) for simulating XVoxel models are provided. The presented method has been validated by a series of case studies of increasing complexity to demonstrate its effectiveness. In particular, a computational efficiency improvement of up to 55.8 times the existing FCM method has been seen.

1. Introduction

Design optimization has been recognized as one of the dominant industrial practices for product design due to improved product quality, reduced cost, and shorter time to market [1]. The optimization may be done in various ways, and parametric optimization is among the primary [2]. It optimizes engineering meaningful parameters that are embedded in feature-based CAD models with externally defined objective functions [3]. Design optimization of this sort has seen applications in many fields, including automotive, shipbuilding, and aerospace industries.

While parametric optimization is very relevant and beneficial in the context of modern feature-based CAD [4], its full realization is not trivial. Its working relies on a closed loop between CAD and CAE with automatic design modifications and simulation updates [5,6]. Forming such a loop is difficult because of the different information contents stored in CAD models and CAE models [1]. Specifically, a CAD model is designated to have an accurate description of the design in order to automate any queries from manufacturing and assembling. It usually consists of feature history, geometric constraints, and parameter definitions (which, altogether, encompass design intent) [4]. A CAE model contains data on the boundary conditions, material distribution, and volumetric meshes that are suitable for conducting the finite element method or the like, which encompass simulation intent [5,7].

To solve the discrepancy between CAD models and CAE models, the approach of model conversion is often employed. As illustrated in Fig. 1, a typical conversion begins with a feature model, then goes through steps of boundary representation (B-rep) generation, model simplification, volumetric mesh generation, and boundary condition specification, cumulatively into a simulation model ready for FEM solving. The solving results will then be used to generate parametric modifications on the feature model for the next optimization iteration. Repeating these procedures will lead to an optimized design.

Despite its conceptual simplicity, there are several technical difficulties in the conversion. Typically the model simplification and mesh generation steps are hard to automate and require manual inputs [8–11], the design optimization and adjustment cannot be directly fed back to CAD modeling operations [12], and the important design intent could be lost after conversion [5,7]. In the context of iterative design optimization, such inefficiency will be much amplified and consequently, the overall process is

* Corresponding author.
E-mail address: qiangzou@cad.zju.edu.cn (Q. Zou).

https://doi.org/10.1016/j.cad.2023.103528
0010-4485/

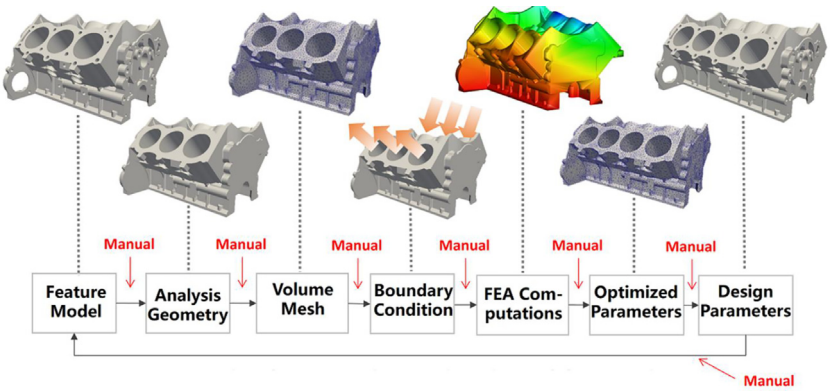


Fig. 1. Traditional conversion-based process of parametric design optimization.

too laborious to be acceptable. It has been reported that manual intervention accounts for about 80% of the overall design time in the conversion-based process described above [5,13].

In view of the above issues, a unified representation scheme that can completely, compactly, and associatively represent the contents of both CAD and CAE models has been recognized as a much-desired method for parametric design optimization [5]. This paper follows this direction and proposes a new representation scheme called Extended Voxel (XVoxel) to address the problem. It essentially makes use of semantic voxels (as will be detailed in Section 3), where the voxel part is responsible for FEM solving, and the semantic part responsible for high-level information to capture both design and simulation intents. In a nutshell, XVoxel models provide the following advantages:

- Design and simulation intents can be preserved in the loop of design, simulation, and optimization, which otherwise are lost in the conversion-based approach and have to be reconstructed. The relevant details can be found in Sections 3.1 and 4.3.
- Generation of analysis geometries and volumetric meshes can be done virtually. As such, labor-intensive and non-robust model simplification and mesh generation can be possibly avoided, and boundary conditions can be well-retained over the course of optimization iterations. The relevant details can be found in Sections 3.3 and 3.4.
- Modifications on design parameters and updates on simulation results can be associated automatically and locally, allowing automatic and efficient looping among design, simulation, and optimization. The relevant details can be found in Sections 3.2 and 4.3.
- Simulation of the feature models can be done efficiently on a coarse XVoxel model while attaining high accuracy through the combination of the fictitious domain technique and material-aware shape functions. The relevant details can be found in Sections 4.1 and 4.2.

The following sections begin with a review on existing parametric optimization methods in Section 2. A detailed description on XVoxel models is given in Section 3. The XVoxel-based simulation and design optimization are presented in Sections 4 and 5, respectively. Application examples and comparisons with existing methods are provided in Section 6, followed by conclusions in Section 7.

2. Related work

Parametric optimization aims to find the optimal design parameters regarding certain performance metrics. Related approaches include conversion-based optimization, unified model-based optimization, and parameter-driven topology optimization.

2.1. Conversion-based parametric optimization

The conversion-based parametric optimization is the de facto standard in practice, but it may require significant manual effort for complex CAD models or boundary conditions to ensure that all conversion steps can be carried out successively [1,6]. A typical conversion procedure involves model simplification, volumetric mesh generation, boundary condition specification, and design modification. Despite the progress on simplifying geometries using methods like feature suppression [8], direct modeling [14], virtual topology [15,16], etc., current methods either have restricted applicability or have robustness issues, thereby requiring considerable manual intervention. Generation of unstructured meshes, e.g., tetrahedron meshes, is an almost solved problem [17,18]. However, automatically generating structured meshes, which are preferable in applications requiring high computational accuracy and efficiency, still remains an open issue [19].

Boundary conditions are largely specified manually in practice, which thus requires huge human efforts in design optimization that loops even thousands of times between the feature model and its simulation. Clearly, this is unacceptable. A common way to address this issue is via assigning fixed boundary conditions or imposing simple varying loads, e.g., in topology optimization [20]. These approaches however would restrict the range of the problem under study. The issue was addressed in a broad sense by defining simulation intent by incorporating concepts of cellular modeling and equivalencing [5,7]. It shares similar spirits with the present work but does not involve standard voxels for performing the simulation. The involvement of manual intervention clearly decreases design efficiency and makes it hard, if not impossible, to automate design optimization.

Note also that in the conversion-based parametric optimization, the underlying FE mesh is varied during each step of the design update, resulting in a varied design space. As a consequence, it usually tends to result in an unstable optimization convergence.

2.2. Unified model-based parametric optimization

Existing unified model-based parametric optimization approaches mainly include isogeometric analysis (IGA), embedded domain, or their combinations.

IGA, initialized by Hughes et al. [21], uses a unified geometric representation scheme, i.e., NURBS (non-uniform rational B-spline), for both design and analysis. Basically, it discards the use of explicit meshes but employs the knot vector and spline basis of a NURBS surface to directly generate the elements and shape functions for FEM solving [22]. As the mesh generation

step is eliminated (in principle), IGA provides a tighter integration between CAD and CAE for automatic design optimization, and the benefits extend beyond integration to higher simulation accuracy and efficiency [2,23–25]. However, IGA only works well on the surface model having a regular parametric domain. For general shapes composed of trimmed NURBS surfaces or 3D volumetric models, quadrilateral meshing of its boundary, or hexahedral meshing of its volume are inevitable, which are challenging research topics in their own right. The XVoxel method to be presented does not have this issue because there are no B-rep models or meshing processes involved. This advantage manifests itself through situations where the B-rep model given to IGA is complex and introduces robustness issues in model simplification and difficulties in quad/hex meshing.

Unlike IGA which revolves around the design model (i.e., NURBS), the embedded domain approach such as finite cell method (FCM) [26] focuses on the other side, i.e., meshes. It uses the same regular background mesh (e.g., a grid) to carry out FEM solving regardless of the design model's variations. As such, no mesh generation is needed when the design model is modified during optimization. This is essentially achieved through high-order finite elements and weak enforcement of unfitted essential boundary conditions. FCM was also used together with IGA to utilize both of their advantages [27,28], most of which did not discuss its work on feature models. Recently, Wassermann et al. [29] studied the problem of conducting FCM on CSG model, which mainly studied the point membership classification problem for different primitives while the present study focuses on the overall integration flowchart for parametric optimization of feature models.

The embedded domain approach is to be combined with the feature-based approach in this work to enable the embedding of design and simulation semantics within the background mesh (which is otherwise purely geometric), where the background mesh serves as a common data structure, and the embedded semantics provide automatic links between design modifications and simulation updates.

2.3. Parameter-driven topology optimization

Research efforts have been devoted toward parametric optimization of CAD models, which mainly focus on finding the optimal shape parameters in describing a specific CAD part but seldom addressed the issue of integrating design semantics into the optimization process.

Chen et al. considered using R-functions for design optimization with topological changes [3,30]. Zhu et al. proposed a direct simulation approach for CAD models undergoing parametric modifications [31] using a model reduction technique called PGD (Proper Generalized Decomposition) [32]. Schulz et al. developed an exploration tool for interactive exploration and optimization of parametric CAD models [33] via pre-computations. More recently, Hafner et al. proposed a generic shape optimization method, called X-CAD, for CAD models based on the eXtended Finite Element Method (XFEM) [34]. These approaches did not involve a complex model-conversion process but worked on an embedded background mesh so as to automate the overall process.

To keep the design intent, the adjustment should be made on feature parameters or the feature history of CAD models [35]. Conducting topology optimization under constraints of specific CAD features has attracted research interests. Zhang and his colleagues have studied extensively the topic [36–39] for practical engineering design. Recently, Guo introduced a novel topology optimization method of MMC (Moving Morphable Components) [40–42], which uses deforming bars as primitive features in topology optimization process for ease of geometric control. However, most of the approaches only studied abstract and single parametric features without design history. Recently, Liu and To [12] first included the feature modeling history of CAD models in the design optimization process. This work follows this direction but employs a more automatic and efficient method, i.e., XVoxel, to carry out the optimization of feature parameters by embedding design intent in the overall optimization process.

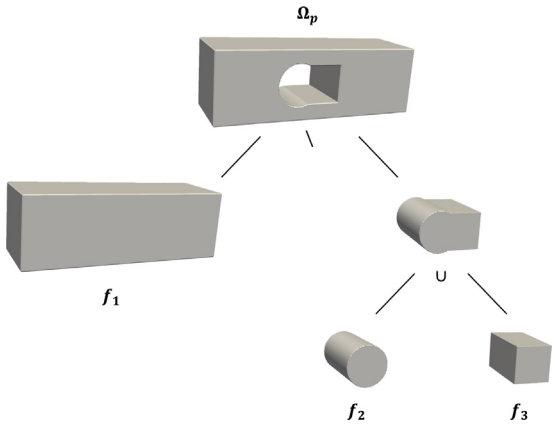


Fig. 2. A CSG example.

3. XVoxel models

This section introduces features, voxels, and their combination into XVoxels, as well as the data structure and algorithms for constructing and manipulating XVoxel models.

3.1. From features and voxels to XVoxels

There is no widely accepted definition of features. The one this work employs is given by Shah [4]: a feature is a generic portion of a model's shape that has certain engineering significance. Roughly speaking, features are clusters of geometric entities in a CAD model, which can be used as information containers to carry domain-specific attributes, e.g., materials and boundary conditions. A feature model is a set of features, combined in a way similar to traditional constructive solid geometry, as shown in Fig. 2. Practically almost all of today's commercial CAD systems use features as an internal representation for constructing and/or editing their CAD models [43]. The user designs a feature by first defining a topology of geometric entities then specifying geometric constraints relating them. A feature can be positioned anywhere in space, or relatively to existing features (through, again, geometric constraints). As such, geometric entities of a CAD model are stored associatively and hierarchically. Changes to the parameters of those features can then be propagated automatically in a pre-defined fashion [44]. This is the basis upon which parametric design optimization becomes possible.

A voxel is a cube-like element in space, and a voxel model is a collection of voxels comprising a three-dimensional geometry of interest. A voxel model can be stored as an array of voxels occupied by the geometry or a grid with binary labels indicating the occupancy relationship between each voxel and the geometry; see also Fig. 3. The former storage scheme is often used to represent static geometries, and the latter used to represent dynamic geometries (and therefore the chosen one in this work).

This work proposes to combine features with voxels, i.e., embedding features into voxels. Traditional CAD/CAE integration methods consistently use features as information containers to store design intent (e.g., shape parameterization) and simulation

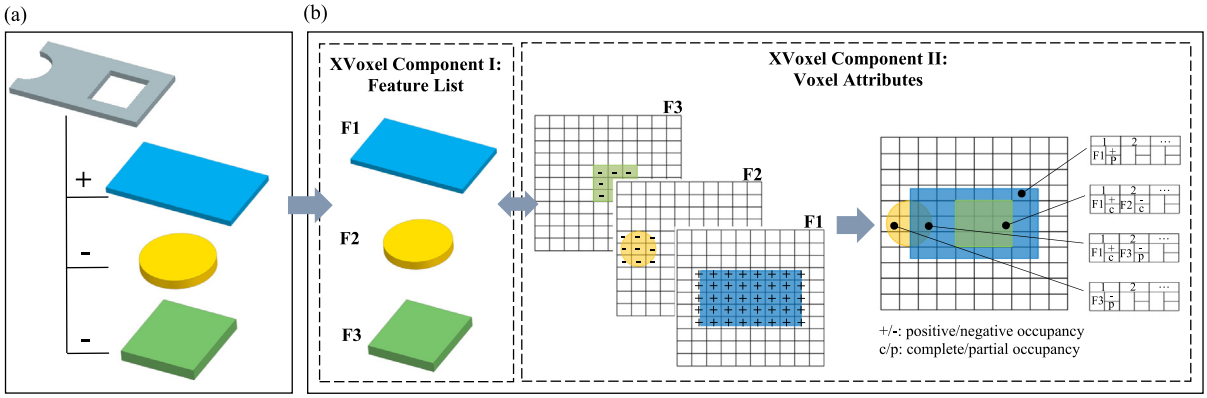


Fig. 3. Illustration of the XVoxel data structure: (a) a feature model (the plus sign means Boolean addition, and the minus sign Boolean subtraction); and (b) its corresponding XVoxel model.

intent (e.g., meshing procedures and boundary conditions) [45]. In this work, voxels are information containers where design intent and simulation intent reside. This shift leads to the notion of semantic voxels (named XVoxels in this work). The primary benefit of doing so is that explicit generation of analysis geometry and meshes can be mostly avoided, and then an automatic, closed loop between CAD and CAE can be achieved. This will be demonstrated in the next few subsections. We begin with the specific data structure used to represent XVoxel models and some primitive operations used to manipulate them.

3.2. XVoxel representation and operations

An XVoxel model consists of two components: a list of features and an array of voxel attributes, as shown in Fig. 3. The feature list is nothing but an unordered set of features (with boundary conditions, material properties, etc. already associated). The voxel attributes associate each voxel with the features occupying it. Three feature attributes are stored: feature occupancy, feature nature, and feature history. For a voxel, feature occupancy describes whether it is completely or partially occupied by a feature; feature nature indicates whether an occupying feature is adding material or subtracting material; feature history refers to the precedence of all occupying features of the voxel.

Consider, for example, the model in Fig. 3, and focus on feature F1. It occupies the voxels colored blue. Voxels at its boundary have partial occupancy, while those in its interior have complete occupancy. The plus signs in Fig. 3b indicate that the occupied voxels are positive (the same as F1's nature). Following the same principle, two additional arrays of voxel attributes can be generated for features F2 and F3, as shown in Fig. 3b. Combining these three arrays of voxel attributes in their chronological order (i.e., F1 $\rightarrow$ F2 $\rightarrow$ F3) results in an XVoxel model, where each voxel maintains an ordered list of 3-tuples (*feature index*, *feature nature*, *occupancy completeness*), as shown by the rightmost four lists in Fig. 3b.

In XVoxel models, determining a model's actual shape relies merely on XVoxel nature, which refers to the nature of the last feature in the attributes list of individual XVoxels. This is because whether the last feature adds or subtracts material, it will override any preceding operations. One exceptional situation is when the last feature partially occupies an XVoxel; this XVoxel's nature is a compound result of the last few features in the attribute list, from the last feature with a complete occupancy to the end. In the following, an XVoxel of this kind is referred to as compound nature. Special algorithms will be developed in the next subsection to handle this situation when using XVoxel to conduct its property simulation.

The above statements seemingly imply that there is no need for storing all historical feature natures of an XVoxel, but only the last one (or ones). They are actually saved for providing easy ways to carry out XVoxel operations, as detailed below. In particular, the novel idea of constantly storing negative feature nature, rather than immediately discarding it after feature Booleans as in conventional feature modeling approaches, allows all operations to work locally, efficiently, and robustly.

Feature Addition This operation creates a new feature by instantiating a chosen feature class with user-specified feature parameter values. After instantiation, the feature's shape extent is used to determine which voxels it occupies, then append the feature's attributes (i.e., the 3-tuple described above) to the end of those voxels' attribute lists. This addition operation is the basis of constructing an XVoxel model from a given feature model. We simply repeat this operation over all features of the model in their chronological order.

Feature Deletion The selected feature is simply removed from the XVoxel model's feature list, with feature dependencies updated accordingly and its attributes removed from relevant XVoxels' attribute lists. To facilitate the retrieval of relevant XVoxels, we further associate each feature with a list of XVoxel indices it occupies in the XVoxel data structure (which can be easily recorded during feature addition). For every single relevant XVoxel, we linearly search the corresponding feature entry in its attribute list and, once found, simply remove it from its current position. (Note that in practice, because attributes in each XVoxel are stored as a linked list, a postprocessing step to correct the linking pointers of remaining entries in the list is needed.) If parallel computing is enabled, we can search and do the removal for all XVoxels simultaneously, without the need for the associativity from features to relevant voxels. Multiple features can also be deleted in parallel. It should, however, be noted that to avoid race conditions when deleting feature attributes at the same XVoxel, we lock the list when the entry removal operation is being carried out for a feature.

Parameter Editing This operation modifies features' parameter values. In the background, we first delete it from the XVoxel model, then re-add its modified version to the XVoxel model according to its original precedence in the feature history. As such, no additional algorithms are needed. Considering that features are often interdependent [46], the above two procedures are modified to include the dependent features of the feature being edited.

Feature Rearrangement This operation modifies the order of features (under the condition that feature dependencies will not be broken). What we need to do is simply updating the

orders in individual XVoxels' attribute lists to accommodate the rearrangement.

As can be seen, there is no time-consuming and non-robust geometric computing involved in the above operations, except for the determination of voxels occupied by a feature to be added in the addition operation. All operations boil down to manipulating entries in a certain linked list, which is easy to implement, robust, and efficient. For the determination of occupied voxels, the essential task involved is to voxelize the shape of a given feature, using the same resolution as the XVoxel model. Note that voxelization is done on individual feature shapes here, which are usually primitives like cuboids or spheres, not on the overall combined shape of all features, which is otherwise complex. Many algorithms exist to voxelize a feature's B-rep model, and the method developed by Young and Krishnamurthy [47] is employed in this work due to its high efficiency. The B-rep model of a feature is often made readily available during feature instantiation, a function provided by almost all modern commercial CAD modelers.

3.3. Virtual model simplification

Model simplification[1] is to remove some design features (e.g., small drilled holes) that are of little significance to simulation. This task becomes straightforward if XVoxel models are used. What we need to do is applying the delete operation described in the previous subsection. The only issue is that, similar to traditional feature-based model simplification approaches, directly removing a feature may cause the persistent naming problem, ultimately breaking the design-analysis cycle. To be more specific, features are made interdependent in feature modeling to enable automatic propagation of parameter changes [44]. Removing a feature makes any inter-dependencies related to it undefined, and then the whole model becomes invalid, which is the so-called persistent naming problem [50].

To solve this issue, we customize the delete operation slightly. The delete operation in Section 3.2 directly removes a feature from the XVoxel model's feature list. Instead, we retain it but make it transparent to voxel attributes by associating the feature list with a bitmask whose 0-elements indicate that features at their positions have been removed, virtually. As such, the difficult persistent naming problem is avoided and meanwhile, there are no real geometric operations involved in model simplification. Another benefit of doing so is that boundary conditions, once associated with certain features, can retain over the course of optimization iterations regardless of design modifications because those features are completely stored in XVoxel models.

3.4. Point membership classification

Traditionally, what comes next after model simplification is generating boundary-conformed meshes for downstream task of simulations. This is, however, a field not all major questions have been answered [21]. This work reformulates the problem as an underlying problem of point membership classification (PMC) for Gaussian integral point selection. It is to be further combined with the recently developed method of FCM for physical simulation, which embeds the physical domain of computation (i.e., the geometry of the feature model) in a larger, regular mesh like a grid, and then transforms the FE computation onto the embedding meshes [26]. (A detailed introduction to FCM will be given in the next section.) This way of working is a perfect match for XVoxel models.

If an XVoxel has positive nature and complete occupancy, it is completely within the model shape. Then the stiffness matrix for this XVoxel can be computed in the exact same way as conventional FEM does. If an XVoxel has compound nature, the XVoxel crosses the boundary of the model shape. According to FCM, voxel subdivision is needed to generate stiffness matrices for such boundary XVoxels, which in turn relies on the operator of point membership classification (PMC) to determine if a sample integration point within a boundary XVoxel is IN/ON/OUT the model shape.

Because XVoxel models have prepared the history of feature occupancy for every XVoxel, the problem of PMC against the overall model shape can be converted to a sequence of much simpler PMCs against individual features [51]. The conversion consists of three major steps: (1) screening relevant features; (2) evaluating PMC against each screened feature; and (3) compiling evaluation results to the final IN/ON/OUT decision. Clearly, not all features occupying an XVoxel contribute to its final shape (i.e., which portion of the XVoxel is solid or void). According to the XVoxel's attribute list, candidate features include those ranging from the last feature with a complete occupancy to the end feature (see Section 3.2). For this reason, the screening step can be simply done by tracing from the back of the attribute list up to the first entry having the complete occupancy attribute.

Having relevant features in place, we next determine the IN/ON/OUT relationship between a query integration point and each of the features. Let the relevant features be denoted by $f_1, f_2, \ldots, f_n$, and their corresponding implicit representation denoted by $\phi_1, \phi_2, \ldots, \phi_n$. In this work, feature implicitization is done by first triangulating its B-rep model with a sufficient high accuracy, then building a KD-tree for the triangles to allow fast query of the (approximated) signed distance between a given point and the feature, similar to the method presented in [52]. Note that alternative methods surely exist [53], and we choose this one for its simplicity and efficiency. Whether a given point $\mathbf{x}$ is IN/ON/OUT feature f_i is determined by the sign of $\phi_i(\mathbf{x})$:

$$\begin{cases} \phi_i(\mathbf{x}) > 0 \quad \rightarrow \quad \mathbf{x} \text{ IN } f_i, \\ \phi_i(\mathbf{x}) = 0 \quad \rightarrow \quad \mathbf{x} \text{ ON } f_i, \\ \phi_i(\mathbf{x}) < 0 \quad \rightarrow \quad \mathbf{x} \text{ OUT } f_i. \end{cases} \tag{1}$$

To compile individual classification results to the final IN/ON/OUT decision, we again make use of the feature history stored in each XVoxel. First, the features classified as OUT are filtered out from the relevant feature set because they contribute nothing to the process of adding/removing material. Then, the final IN/ON/OUT decision is the same as the nature of the last remaining relevant features: if the nature is positive, the material is added to the query point, and the final decision is IN/ON; otherwise, the final decision is OUT. This is because the last material removing/adding operation overrides all the preceding operations.

Altogether, they yield a method to generate a "mesh" suitable for FCM solving from an XVoxel model. The mesh is not explicitly generated but through combining the fixed grid carrying the XVoxel model and an implicit PMC operator developed specifically for XVoxel models. The method is thus easy to implement. It should, however, be noted that, the use of triangulation in feature implicitization will introduce errors, and therefore possible misclassifications in PMC. In fact, this is generally acceptable since we can triangulate at a high accuracy. Also, due to the integral nature of FCM, it is not very sensitive to such misclassifications.

[1] It should be noted that a more general concept than model simplification is model idealization, which includes an additional dimension reduction task [10,48,49]. As XVoxels models are three-dimensional, they are not able to handle dimension-reduced geometries in their current form. The authors wish to extend XVoxels to representing low-dimension geometries in our future studies.

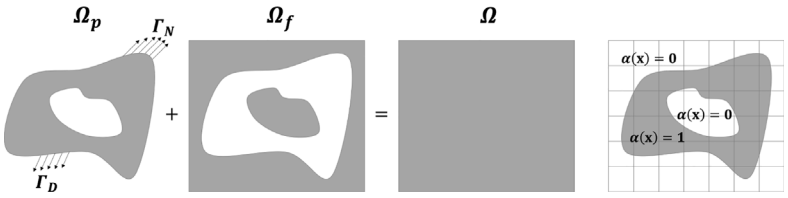


Fig. 4. The embedded domain Ω consists of the physical domain Ω_p and the fictitious domain Ω_f, and the influence of Ω_f is penalized by material parameter 10^{-q}.

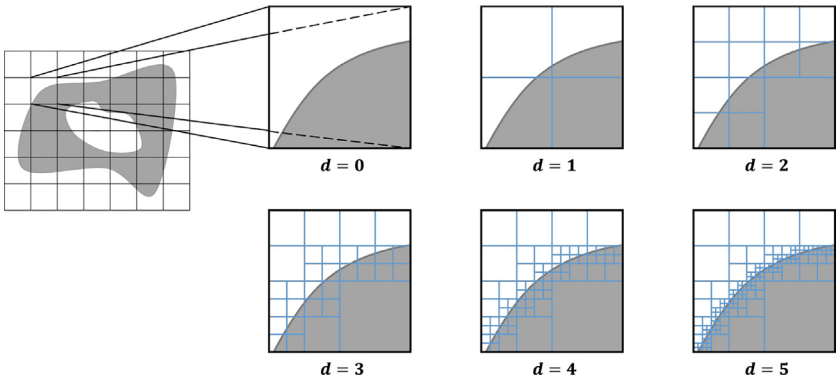


Fig. 5. In 2D, adaptive Gaussian integration is recursively refined towards boundary of physical domain (gray domain), yielding a quadtree structure (thin blue grids) of finite cells (bold black grids); Quadtrees depth ranging from 0 to 5 are shown here. The voxels along the boundary are called cut voxels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. XVoxel-based simulation

In this work, simulation is to be carried out using a fictitious domain approach following a FCM-like framework [26], which can work directly on voxel models. This approach's low computational efficiency is improved in two aspects: (1) by introducing material-aware piecewise matrix-valued shape functions, called CBN (Curved Bridge Node) shape functions following the previous study in [54]; (2) by utilizing the local computation of XVoxel models.

4.1. Finite cell method (FCM) for XVoxel-based simulation

The basic idea of FCM is to use a simple regular structured mesh to approximate the solution fields. This is achieved by combining the fictitious domain idea with the benefits of high-order finite elements, thus avoiding the costly and even labor-intensive meshing process.

The FCM concept is interpreted by a 2D linear elasticity problem in Fig. 4. Let $\Omega_p \in \mathbb{R}^2$ be the physical domain, Γ_D the Dirichlet boundary, and Γ_N the Neumann boundary under external loading $\boldsymbol{\tau}$. A linear elasticity analysis problem on Ω_p is studied to find the displacement $\mathbf{u}$ satisfying

$$a(\mathbf{u}, \mathbf{v}) = l(\mathbf{v}), \quad \forall\, \mathbf{v} \in H_0^1(\Omega), \tag{2}$$

where

$$a(\mathbf{u}, \mathbf{v}) = \int_{\Omega_p} \boldsymbol{\varepsilon}(\mathbf{u})^T \mathbf{D} \boldsymbol{\varepsilon}(\mathbf{v})\, \mathrm{dV} = \int_{\Omega} H(\mathbf{x}) \boldsymbol{\varepsilon}(\mathbf{u})^T \mathbf{D} \boldsymbol{\varepsilon}(\mathbf{v})\, \mathrm{dV}, \tag{3}$$

and

$$l(\mathbf{v}) = \int_{\Omega_p} \mathbf{f}\cdot\mathbf{v}\, \mathrm{dV} + \int_{\Gamma_N} \boldsymbol{\tau}\cdot\mathbf{v}\, d\Gamma = \int_{\Omega} H(\mathbf{x}) \mathbf{f}\cdot\mathbf{v}\, \mathrm{dV} + \int_{\Gamma_N} \boldsymbol{\tau}\cdot\mathbf{v}\, d\Gamma, \tag{4}$$

where $H^1(\Omega)$ and $H_0^1(\Omega)$ are the usual Sobolev vector spaces, $\mathbf{f}$ is the body force, $\boldsymbol{\sigma}(\mathbf{u})$ is the second-order *stress tensor* defined via Hooke's law,

$$\boldsymbol{\sigma}(\mathbf{u}) = \mathbf{D} : \boldsymbol{\varepsilon}(\mathbf{u}), \quad \boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{5}$$

for a fourth-order elasticity tensor $\mathbf{D}$.

Here in Eqs. (3) and (4), the computation domain is converted from Ω_p to the embedded domain Ω by incorporating the fictitious domain material which is defined via a Heaviside function $H(\Phi(\mathbf{x}))$,

$$H(\Phi(\mathbf{x})) = \begin{cases} 1, & \text{if } \Phi(\mathbf{x}) > 0, \\ \alpha, & \text{otherwise}\ , \end{cases} \tag{6}$$

where Φ is the SDF (Signed Distance Function) of the feature model Ω_p and α is a small positive coefficient, say 10^{-8}, to avoid ill-conditionedness on the stiffness matrix. The Nitsche's method [55] was usually adopted to weakly impose Dirichlet boundary conditions in FCM; we are not going into details here.

Following a classical Galerkin FE method, the solution $\mathbf{u}(\mathbf{x})$ to Eq. (2) is approximated as a linear combination of *higher-order* shape (base) functions $\mathbf{N}^\alpha(\mathbf{x})$ for each regular grid (or voxel) $\Omega^\alpha \subset \Omega$. Specifically, the overall displacement on any point of $\mathbf{x} \in \Omega^p$ can be interpolated from an assembly sum

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{N}(\mathbf{x})\mathbf{Q} = \sum_{\alpha=1}^{M} \mathbf{N}^\alpha(\mathbf{x})\, \mathbf{Q}^\alpha, \quad \mathbf{x} \in \Omega, \tag{7}$$

where $\mathbf{N}(\mathbf{x})$ is the collection of bases $\mathbf{N}^\alpha(\mathbf{x})$, $\mathbf{Q}$ is the collection of $\mathbf{Q}^\alpha$, a displacement vector per voxel Ω_α.

Accordingly, the displacement $\mathbf{Q}$ to Eq. (2) is computed as the solution to a linear system

$$\mathbf{K}\mathbf{Q} = \mathbf{F}, \tag{8}$$

where the stiffness matrix and load vector are assembly from their element stiffness matrix $\mathbf{K}_\alpha$ and element load vector on a regular grid (or XVoxels)

$$\mathbf{K} = \sum_\alpha \mathbf{K}^\alpha, \quad \mathbf{F} = \sum_\alpha \mathbf{F}^\alpha. \tag{9}$$

FCM transfers the challenges of mesh generation to the numerical integration of discontinuous integrands in Eqs. (3) and (4). An adaptive Gauss integration is usually applied to improve its accuracy; see Fig. 5. The high-order shape functions $\mathbf{N}^\alpha(\mathbf{x})$ in

FCM requires a huge number of Gaussian points, which involves huge computational costs as compared with FEA and occupies the dominant computational costs of FCM.

4.2. CBN shape functions for efficient FCM computation

Following previous study [54], material-aware CBN shape functions are introduced in this work to replace the higher-order shape functions in FCM to accelerate the computations.

In our adopted version of CBN, a 4×4 grid is formed by introducing twelve additional virtual nodes on each face of a voxel element besides its original 4 nodes (all together 80 nodes for a 3D XVoxel). Displacement on these CBN nodes are collected into a vector $\mathbf{Q}$ and taken as DOFs for solution computation.

The higher order shape function $\mathbf{N}^{\alpha}$ in Eq. (7) is replaced by the following one composed of linear shape functions on fine mesh via a transformation matrix $\tilde{\mathbf{\Phi}}^{\alpha}$:

$$\mathbf{N}^{\alpha}(\mathbf{x}) = \mathbf{N}^{\alpha,h}(\mathbf{x})\ \tilde{\mathbf{\Phi}}^{\alpha}, \quad \mathbf{x} \in \Omega^{\alpha} \tag{10}$$

where $\mathbf{N}^{\alpha,h}(\mathbf{x})$ is an assembly of the nodal shape functions on the fine mesh of Ω^{α}.

The CBN transformation matrix $\tilde{\mathbf{\Phi}}^{\alpha}$ aims to map the CBN nodal values to the interior values in Ω^{α}. It is derived as a product of *boundary interpolation matrix* $\mathbf{\Psi}$ and *boundary–interior transformation matrix* $\tilde{\mathbf{M}}^{\alpha}$, as follows,

$$\tilde{\mathbf{\Phi}}^{\alpha} = \tilde{\mathbf{M}}^{\alpha}\ \mathbf{\Psi}, \tag{11}$$

where $\mathbf{\Psi}$ and $\tilde{\mathbf{M}}^{\alpha}$ maps the displacements from the CBNs to the boundary nodes and then to the full fine nodes in Ω^{α}.

The boundary interpolation matrix $\mathbf{\Psi}$ maps the CBN nodal values to the fine mesh boundary nodal values of Ω^{α}. It is derived by constructing a bi-cubic Bézier interpolation surface over the face of interest, taking the CBN as control points. The matrix $\mathbf{\Psi}$ is derived by evaluating the surfaces at fine mesh nodes within the face, and collecting them in a matrix form all the values row by row for the six faces of the mesh α.

The transformation matrix $\tilde{\mathbf{M}}^{\alpha}$ maps the boundary node values to those of the fine mesh in Ω^{α}. It is derived from the local simulation on the fine mesh of Ω^{α} with the equilibrium equation

$$\begin{bmatrix} \mathbf{k}_b & \mathbf{k}_{bi} \\ \mathbf{k}_{ib} & \mathbf{k}_i \end{bmatrix} \begin{bmatrix} \mathbf{q}_b \\ \mathbf{q}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ 0 \end{bmatrix}, \tag{12}$$

where $\mathbf{k}_b$, $\mathbf{k}_i$, $\mathbf{k}_{bi}$, $\mathbf{k}_{ib}$ are the sub-matrices of the local stiffness matrix $\mathbf{k}^{\alpha}$ on Ω^{α}, $\mathbf{q}_b$, $\mathbf{q}_i$ is respectively vector of the boundary, interior nodes, and $\mathbf{f}_b$ is vector of exposed forces on the boundary nodes formed by harmonic analysis [54].

We have from the second-row the relation of $\mathbf{q}_i = \mathbf{M}^{\alpha}\mathbf{q}_b$, for $\mathbf{M}^{\alpha} = -\mathbf{k}_i^{-1}\mathbf{k}_{ib}$. Accordingly, assembling $\mathbf{q}_i$ and $\mathbf{q}_b$ as $\mathbf{q} = [\mathbf{q}_b, \mathbf{q}_i]^T$, we have the form of $\tilde{\mathbf{M}}^{\alpha}$,

$$\tilde{\mathbf{M}}^{\alpha} = [\mathbf{I}_{2b}, -\mathbf{k}_i^{-1}\mathbf{k}_{ib}]^T, \tag{13}$$

where $\mathbf{I}_{2b}$ is the $2b \times 2b$ identity matrix.

Once the CBN shape functions are derived, the solution to the linear elasticity problem in Eq. (2) can be similarly attained, following a classical Galerkin FE method. More technical details are referred to [54].

4.3. XVoxel-based local simulation of feature models

Based on the approach of FCM for simulation, in combination with CBN, the XVoxel-based approach for simulation of modified feature model is developed below.

The feature model is generally modified via updating feature parameters, which may change the topology and geometry of the final B-rep model. As long as the feature model are updated, element stiffness matrix $\mathbf{K}^{\alpha}$ of each voxel need to be re-computed, which accounts most for the computation costs. FCM equipped with local voxel updates can accelerate the computations in two ways: (a) the element stiffness matrix of each full-voxel or void-voxel is identical; the voxels along the boundary are called *cut voxels*; (b) the element stiffness matrices of voxels not affected by updated features remain unchanged. Case (a) can be easily resolved via a pre-computation strategy. For case (b), the element stiffness matrix can be incrementally updated by updating and querying voxel-feature membership table via the PMC algorithm described in Section 3.4, where voxels affected by the updated features can be quickly located, called *active voxels*, and consequently only their stiffness matrices are re-computed. This can significantly reduce computation costs.

5. XVoxel-based parametric design optimization

Using XVoxels, the parametric design optimization works over a fixed regular grid under controlled simulation accuracy, and on direct updates of feature parameters. During the process, the sensitivities with respect to the design parameters is derived for parameter updates. The locality information of XVoxel provides an efficient sensitivity computation either via finite difference or via a derived analytical expressions.

Let Ω_p be a CAD model with features $f_1, f_2, \ldots, f_n$. For ease of explanation, each feature f_i is assumed to take only one parameter p_i. The classical compliance minimization problem is studied to find the optimized design parameters $\mathbf{p} = (p_1, p_2, \ldots, p_n)$:

$$\min_{\mathbf{p}} C(\mathbf{u}, \mathbf{p}) = \mathbf{u}^T\mathbf{K}\mathbf{u},$$
$$\text{s.t.} \begin{cases} \mathbf{K}\mathbf{u} = \mathbf{F}, \\ V = \int_{\Omega} H(\Phi(\mathbf{x}, \mathbf{p}))\mathrm{d}\Omega \le \bar{V}, \\ \underline{p}_i \le p_i \le \bar{p}_i, i = 1, 2 \ldots, n, \end{cases} \tag{14}$$

in which V and $\bar{V}$ are the total structural volume and maximum volume constraint, the Heaviside function $H(\cdot)$ is used to indicate structural boundary, $\underline{p}_i$ and $\bar{p}_i$ are lower and upper bounds of the design variable p_i.

The optimization problem Eq. (14) is to be solved following a numerical gradient-based approach Globally Convergent Method of Moving Asymptotes (GCMMA) [56] for its robust convergence in design optimization. It approximates the original nonconvex problem through a set of convex sub-problems by using the gradients of the optimization objective and constraints with respect to the design variables $\mathbf{p}$ derived below.

The gradient computation follows the chain rule. First consider the sensitivities of stiffness matrix $\mathbf{K}$ with respect to design parameter p_i. Rewriting $\psi(\mathbf{x}) = \mathbf{B}^T\mathbf{D}\mathbf{B}$ for conciseness, we have

$$\begin{aligned} \frac{\partial \mathbf{K}}{\partial p_i} &= \frac{\partial}{\partial p_i}\int_{\Omega} \mathbf{B}^T\mathbf{D}\mathbf{B}H(\Phi(\mathbf{x}, \mathbf{p}))\mathrm{d}\Omega \\ &= \int_{\Omega} \mathbf{B}^T\mathbf{D}\mathbf{B}\frac{\partial H(\Phi)}{\partial \Phi}\frac{\partial \Phi}{\partial p_i}\,\mathrm{d}\Omega \\ &= \int_{\Omega} \psi(\mathbf{x})\frac{\partial H(\Phi)}{\partial \Phi}\frac{\partial \Phi}{\partial p_i}\,\mathrm{d}\Omega. \end{aligned} \tag{15}$$

The key point of above equation is to compute derivative of Heaviside function. We bring in Dirac delta function $\hat{\delta}(\Phi)$

$$\hat{\delta}(\Phi) = \nabla H(\Phi) \cdot \frac{\nabla \Phi}{\|\nabla \Phi\|} = \frac{\mathrm{d}H(\Phi)}{\mathrm{d}\Phi}\nabla\Phi \cdot \frac{\nabla \Phi}{\|\nabla \Phi\|} = \frac{\mathrm{d}H(\Phi)}{\mathrm{d}\Phi}\|\nabla \Phi\|, \tag{16}$$

where

$$\|\nabla\Phi\| = \sqrt{\left(\frac{\partial\Phi}{\partial x}\right)^2 + \left(\frac{\partial\Phi}{\partial y}\right)^2 + \left(\frac{\partial\Phi}{\partial z}\right)^2}. \tag{17}$$

Consequently, Eq. (15) is rewritten as

$$\begin{aligned}\frac{\partial\mathbf{K}}{\partial p_i} &= \int_\Omega \psi(\mathbf{x})\frac{\partial H(\Phi)}{\partial\Phi}\frac{\partial\Phi}{\partial p_i}\,\mathrm{d}\Omega \\ &= \int_\Omega \psi(\mathbf{x})\frac{\partial\Phi}{\partial p_i}\frac{1}{\|\nabla\Phi\|}\left(\frac{\partial H(\Phi)}{\partial\Phi}\|\nabla\Phi\|\right)\mathrm{d}\Omega \\ &= \int_\Omega \psi(\mathbf{x})\frac{\partial\Phi}{\partial p_i}\frac{1}{\|\nabla\Phi\|}\hat{\delta}(\Phi)\mathrm{d}\Omega \\ &= \int_{\partial\Omega_p} \psi(\mathbf{x})\frac{\partial\Phi}{\partial p_i}\frac{1}{\|\nabla\Phi\|}\mathrm{d}\Gamma,\end{aligned} \tag{18}$$

where $\partial\Omega_p$ denotes boundary of feature model Ω_p. This way, the volume integral of sensitivities is transformed into a boundary integral.

According to the expression of $\Phi(\mathbf{x}, \mathbf{p})$ in Eq. (6), we further have for Eq. (18),

$$\frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial p_i} = \sum_{j=1}^{n}\frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial\phi_j}\cdot\frac{\partial\phi_j}{\partial p_i}, \tag{19}$$

$$\|\nabla\Phi(\mathbf{x},\mathbf{p})\| = \left\|\sum_{j=1}^{n}\frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial\phi_j}\nabla\phi_j\right\|. \tag{20}$$

Noting that design variable p_i is only associated to one feature f_i, we have

$$\frac{\partial\phi_j}{\partial p_i} = 0, \quad j \neq i. \tag{21}$$

Let $S_i = \frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial\phi_i}$ be the logical operation defined by parent bifurcation nodes of f_i in CSG tree, and $S_i \in \{-1, 1\}$. Accordingly, the integral domain of Eq. (18) can be reduced from boundary $\partial\Omega_p$ of whole feature model Ω_p to boundary ∂f_i of a feature f_i, that is, computations.

$$\frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial p_i} = \frac{\partial\Phi(\mathbf{x},\mathbf{p})}{\partial\phi_i}\cdot\frac{\partial\phi_i}{\partial p_i} = S_i\frac{\partial\phi_i}{\partial p_i}, \tag{22}$$

which avoids redundant integration.

Similarly, we have

$$\|\nabla\Phi(\mathbf{x},\mathbf{p})\| = \|S_i\nabla\phi_i\| = \|\nabla\phi_i\| \tag{23}$$

for quadrature points along boundary of feature f_i.

Accordingly, the sensitivities in Eq. (18) is reduced to

$$\begin{aligned}\frac{\partial\mathbf{K}}{\partial p_i} &= \int_{\partial\Omega_p}\psi(\mathbf{x})\frac{\partial\Phi}{\partial p_i}\frac{1}{\|\nabla\Phi\|}\mathrm{d}\Gamma \\ &= \int_{\partial f_i} S_i\psi(\mathbf{x})\frac{\partial\phi_i}{\partial p_i}\frac{1}{\|\nabla\phi_i\|}\mathrm{d}\Gamma.\end{aligned} \tag{24}$$

Afterwards, we consider sensitivities of structural compliance C,

$$\begin{aligned}\frac{\partial C}{\partial p_i} &= \frac{\partial\mathbf{F}^{\mathrm{T}}}{\partial p_i}\mathbf{u} + \mathbf{F}^{\mathrm{T}}\frac{\partial\mathbf{u}}{\partial p_i} \\ &= \frac{\partial\mathbf{F}^{\mathrm{T}}}{\partial p_i}\mathbf{u} + \mathbf{F}^{\mathrm{T}}\mathbf{K}^{-1}\left(\frac{\partial\mathbf{F}}{\partial p_i} - \frac{\partial\mathbf{K}}{\partial p_i}\mathbf{u}\right) \\ &= 2\frac{\partial\mathbf{F}^{\mathrm{T}}}{\partial p_i}\mathbf{u} - \mathbf{u}^{\mathrm{T}}\frac{\partial\mathbf{K}}{\partial p_i}\mathbf{u}.\end{aligned} \tag{25}$$

Assuming for simplicity the independence of load $\mathbf{F}$ and feature design variables, the first term in the above equation is zero.

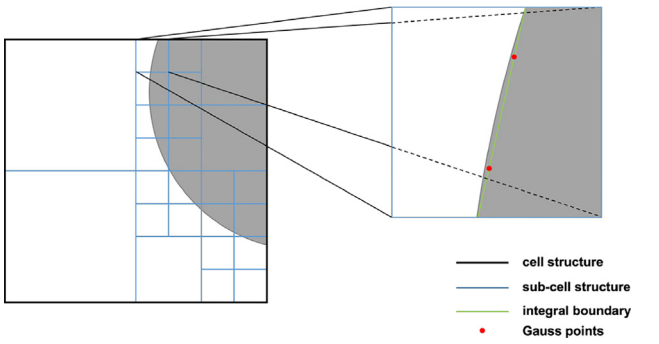


Fig. 6. Sensitivity computation (in 2D) as adaptive boundary integration of finite cell (bold black grid): In each sub-cell (thin blue grid), the boundary is approximated with line segments (green), along which Gauss quadrature points (red) are taken. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Based on sensitivities of stiffness matrix in Eq. (24), we have

$$\begin{aligned}\frac{\partial C}{\partial p_i} &= -\mathbf{u}^T\frac{\partial\mathbf{K}}{\partial p_i}\mathbf{u} \\ &= -\mathbf{u}^T\left(\int_{\partial f_i} S_i\psi(\mathbf{x})\frac{\partial\phi_i}{\partial p_i}\frac{1}{\|\nabla\phi_i\|}\mathrm{d}\Gamma\right)\mathbf{u} \\ &= -\mathbf{u}^T\left(\int_{\partial f_i} S_i\mathbf{B}^T\mathbf{D}\mathbf{B}\frac{\partial\phi_i}{\partial p_i}\frac{1}{\|\nabla\phi_i\|}\mathrm{d}\Gamma\right)\mathbf{u}.\end{aligned} \tag{26}$$

In our numerical implementation, we use triangles in 3D (lines in 2D) to approximate structural boundary of model for adaptive integration; see Fig. 6 for an illustration.

6. Numerical examples and discussions

The proposed XVoxel-based method for parametric design optimization of feature models has been implemented in Matlab on a computer with Intel Core i7-12700 3.6 GHz CPU, 64 GB RAM. Five different examples are shown to demonstrate its effectiveness: the first three on simulation during interactive editing to test its computational accuracy and efficiency, and the last two on its usage for feature-based design optimization. The CAD model sizes are all measured in micrometer, and the material has a Young's modulus $E = 2e^{11}$ Pa and Poisson's ratio $\nu = 0.3$. In FCM computing, all examples have the octree refinement depth $d = 3$ and the shape function order $p = 2$, except for the first example where $d = 4$ and $p = 3$.

The locality characteristics of XVoxel are measured in terms of the number of active voxels (i.e. voxels affected by local feature updates) against that of FCM. The fidelity of XVoxel or FCM is measured via its displacement residual (in terms of top 10%) against the benchmark:

$$r_u = \frac{\|\mathbf{u}_1 - \mathbf{u}_0\|}{\|\mathbf{u}_0\|}, \tag{27}$$

where $\mathbf{u}_1$, $\mathbf{u}_0$ are the computed and the benchmark displacements, respectively.

The experimental settings and results are summarized in Table 1, including mesh size, DOFs, timing (per step/iter), the number of active voxels and relative error $r_{\mathbf{u}}$. In all these examples, FEA simulation results on tetrahedral meshes (in Ansys Workbench 22R1) were taken as the benchmark. Three other approaches were tested to show the method's simulation accuracy and efficiency: standard FCM approach [26], XVoxel-FM combining FCM with XVoxel, XVoxel-CBN combining CBN-based FCM [54]; the last two are our approaches. The DOFs of FEA and

Table 1
Summary of the performance of XVoxel on the tested numerical examples in comparison with those using FEA and FCM (standard FCM, XVoxel-FCM combining FCM with XVoxel, XVoxel-CBN combining CBN-based FCM and XVoxel). Here, d is the depth of octree refinement for element integration and p is the order of shape functions.

Example	Step (iter)	Mesh size		DOFs			Timings (per step/iter)				Active voxel number		$r_{\mathbf{u}}$ (%)
		FEA	FCM(XVoxel)	FEA	FCM (XVoxel-FCM)	XVoxel-CBN	FEA	FCM	XVoxel-FCM	XVoxel-CBN	FCM	XVoxel	
#1	1	13,801	675	63,372	63,480	47,280	5.3	35.1	35.1	6.0	33	33	0.0061
	2	14,118		64,833			5.2	28.8	28.9	2.9	21	21	0.0169
	3	13,807		63,600			5.2	23.2	23.7	1.8	21	21	0.0263
	4	13,844		63,714			6.2	17.3	17.7	1.3	15	15	0.0049
	5	13,908		64,029			5.2	11.6	11.8	0.8	9	9	0.0047
#2	1	22,789	4004	109,632	108,135	255,024	8.5	44.9	44.9	13.8	1182	1182	0.0328
	2	22,579		108,777			7.3	44.2	6.3	1.1	1182	121	0.0825
	3	22,326		108,006			7.2	42.6	9.4	1.8	1182	183	0.0812
	4	22,465		109,350			7.3	44.3	9.8	2.1	1185	189	0.1555
	5	22,611		109,569			7.3	43.7	5.6	1.4	1191	128	0.3221
	6	22,326		108,336			8.5	40.2	8.5	1.4	1141	227	0.0263
#3	1	222,816	39,775	1,019,364	998,325	2,365,080	22.1	140.1	143.1	80.9	6040	6040	0.0928
	2	217,244		1,001,976			19.7	151.5	33.9	5.2	6048	397	1.0381
	3	218,659		1,006,104			21.6	152.0	35.0	6.0	6156	491	1.1859
	4	215,383		996,989			23.1	166.2	35.3	5.4	6356	412	1.3995
	5	215,185		996,513			20.8	161.3	28.0	1.1	6369	97	1.4215
	6	214,206		991,353			23.5	167.1	35.8	4.5	6545	458	1.4720
	7	216,983		1,002,681			21.0	168.0	31.0	3.3	6561	216	1.4834
	8	215,132		997,344			20.9	174.4	34.8	3.9	6681	325	2.5318
	9	214,288		993,975			18.0	181.3	30.3	2.0	6789	190	2.3235
	10	216,411		1,004,442			21.3	210.0	29.5	2.2	6633	102	2.4980
	15	214,690		992,787			23.7	201.5	48.2	9.0	5837	677	2.7177
	20	213,749		992,295			23.5	223.4	42.6	5.5	6485	459	3.0647
#4	1		9100		240,975	250,638		1,490.8	130.3	55.2	2256	2256	
	10							1411.0	109.6	25.2	2174	308	
	30							1401.8	101.3	24.8	2166	274	
	100							1467.5	102.4	25.3	2172	255	
#5	1		33,150		279,265	282,006		146.0	148.0	60.4	4592	4,592	
	10							141.5	85.8	30.7	3904	1644	
	30							156.0	92.7	33.2	3952	1611	
	100							166.6	97.2	32.6	3962	1486	

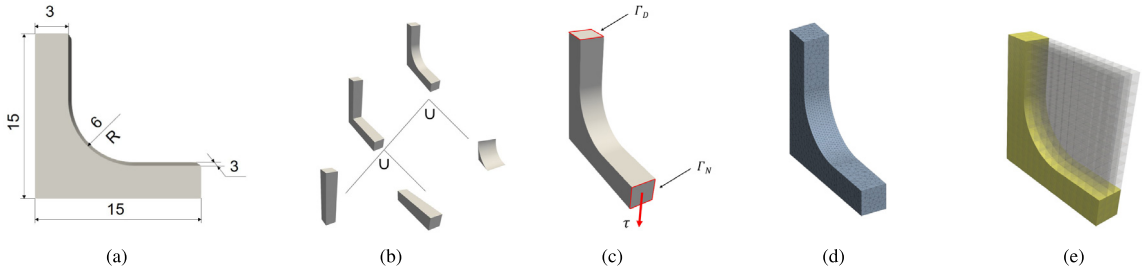


Fig. 7. Example #1. (a) Parameters (mm) of the L-shaped model, where the radius R gradually varies from 6 to 2 with a step size of −1; (b) The CSG of the model; (c) Boundary conditions, where γ_D is fixed and $\tau = 100$ N/mm^2 in γ_N; (d) FEA mesh with 6 325 tetrahedral elements; (e) FCM (XVoxel) mesh with $3 \times 15 \times 15$ voxels.

XVoxel were set approximately same for the comparisons to be fair.

As can be seen from Table 1, the error of FCM (XVoxel) is as low as 0.005%, demonstrating its high accuracy. FCM and XVoxel-FCM always have the same simulation accuracy, and CBN-FCM is very close to them. Other examples may have higher error due to the need for balancing accuracy and efficiency. Note that FCM (XVoxel) can reach a prescribed accuracy voxel refinement or degree elevation [26]. In all examples, FEA is much more efficient than FCM while XVoxel-FCM improves the efficiency, resulting in a similar computation time to FEA, due to its local computations. XVoxel-CBN greatly improves the efficiency of XVoxel-FCM due to its usage of piecewise linear shape functions. We have to mention again that in comparison with FEA FCM or XVoxel (either FCM or CBN version) has a prominent advantage in its much easier and more robust voxelization than FEA's tetrahedral meshing.

6.1. Example #1: an L-shaped model for simulation accuracy testing

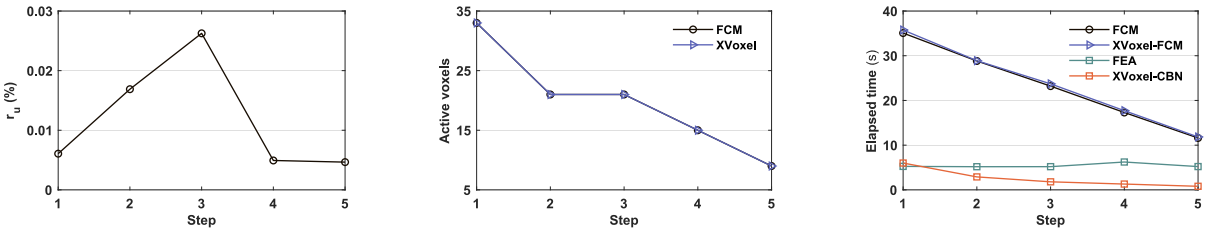
The accuracy of the proposed method was first tested on a classic L-shaped model as shown in Fig. 7, constructed by combining two cubes and one rounded corner. The model is fixed on its upper face and subject to a downward traction of $\tau = 100$ N/mm^2 on its right face. The tetrahedral mesh of FEA has 6.3K elements, and the FCM has 768 voxels.

The rounded corner radii was varied from 6 mm to 2 mm at a step of −1 mm. The simulation error, the number of active voxels, and timings for each step were respectively plotted in Figs. 8(a), (b), (c). FEA and XVoxel-FCM has a very close approximation at an error $r_{\mathbf{u}} = 0.03\%$, as can also be observed from distributions of their displacement norm and von Mises stress in Figs. 9(a), (b). We also notice from Figs. 9(c) and 8(b), (c) that FCM and XVoxel-FCM have exactly the same number of active voxels and computational timings in this example. This is because the cut cells in FCM are just the active voxels due to the regular shape of the L-shaped model.

6.2. Example #2: a connector model for simulation efficiency testing

The second test was conducted on the engine connector as shown in Fig. 10 to test XVoxel's ability in handling more complex models. The model is fixed on its left hole, subject to horizontal and vertical tractions on its right hole. The FEA tetrahedral mesh has 23K elements while the XVoxel has 7.9K voxels. The connector model was modified in Fig. 12(a) by the following six steps of feature operations:

1. Add a pair of inner groove made of two cylinders and their tangents. Note that the two sides of connector are symmetrical, and we only consider design parameters on one side.



(a) Relative error $r_{\mathbf{u}}$ (b) The number of active voxels (c) Timing

Fig. 8. Performance statistics of Example #1 in Fig. 7: (a) Displacement residual r_u between XVoxel and FCM; (b) The number of active voxels by FCM and XVoxel; (c) Timing of four methods FEA, FCM, XVoxel-FCM (XVoxel based on FCM) and XVoxel-CBN (XVoxel based on CBN).

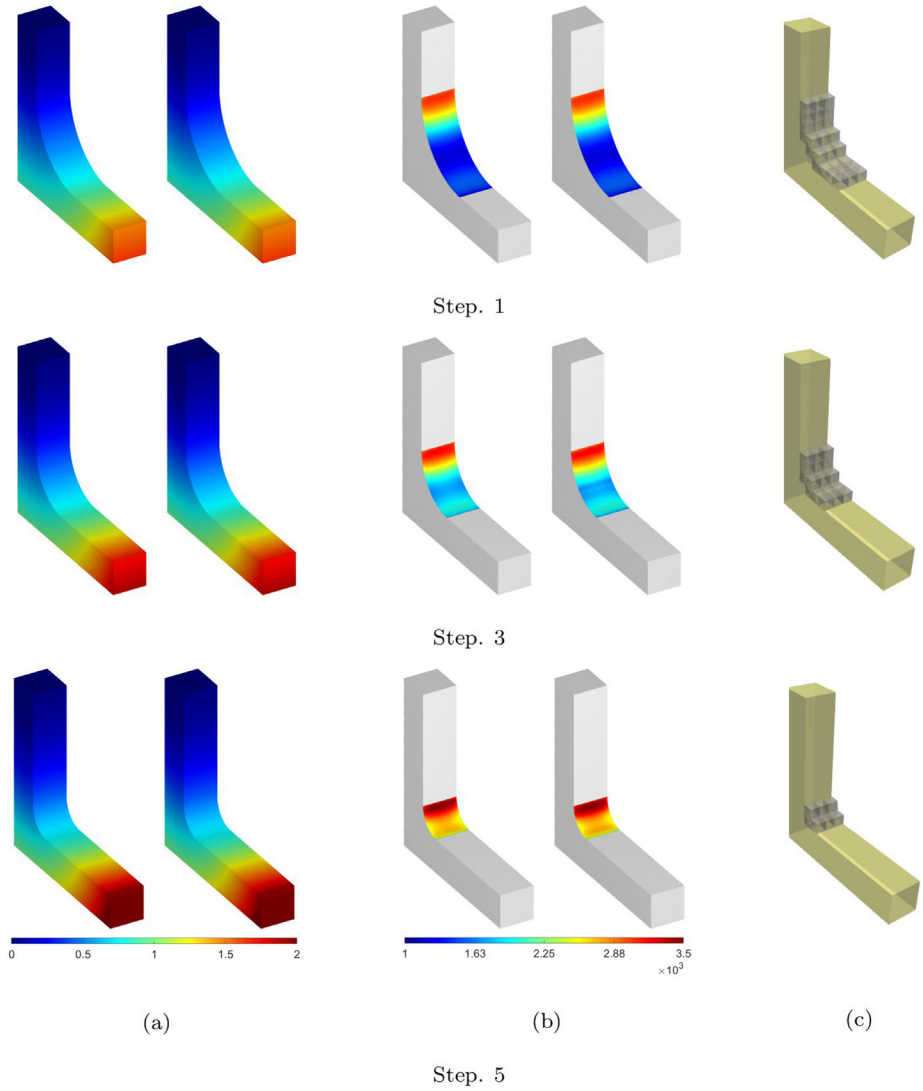
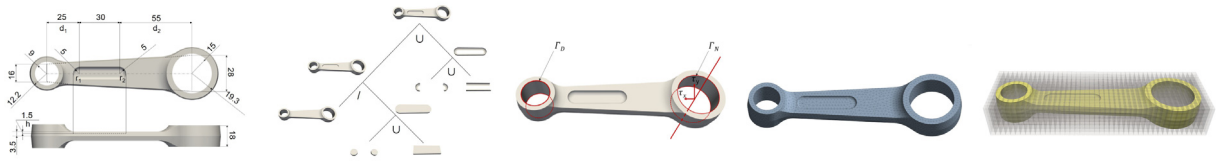


Fig. 9. Results for an L-shaped model in editing step 1, 3 and 5. (a): Displacement norm (mm) from FEA (left) and FCM/XVoxel (right); (b) Von Mise stress (MPa) from FEA (left) and FCM/XVoxel (right); (c) Active voxels (in gray) of XVoxel.

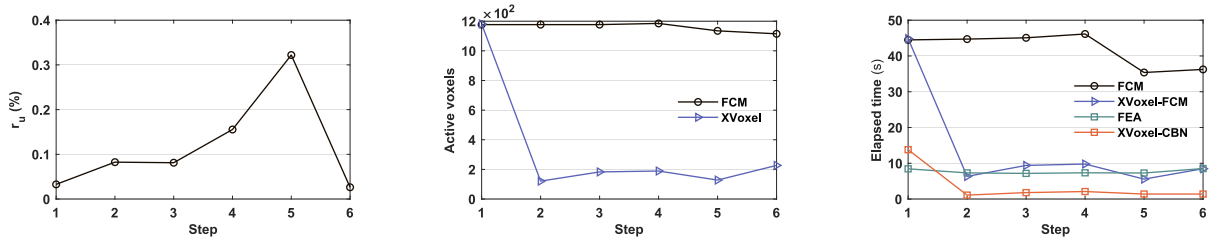
2. Translate the two cylinders by changing parameters d_1 from 25 to 30, d_2 from 55 to 40.
3. Modify the right cylinder's radius r_2 from 5 to 7.5.
4. Modify the inner groove's depth h from 1.5 to 2.5.
5. Modify design parameter h from 2.5 to 3.5 and remove the round corner so that the inner groove goes through the whole model.
6. Modify design parameters r_1 from 5 to 3, r_2 from 7.5 to 5.

XVoxel has a close approximation to FEA with a maximal $r_{\mathbf{u}} = 0.33\%$ as observed from plot in Fig. 11(a), or the comparison of simulation results in Figs. 12(b) and (c). We also noticed from Figs. 12(d) and 11(b) and (c) that the XVoxel (XVoxel-FCM and XVoxel-CBN) has much less computational time than FCM during the model modifications (after the first step) as its local voxel update immensely decreases the number of active voxels.



(a) (b) (c) (d) (e)

Fig. 10. Example #2. (a) Parameters of the piston model, where modified variables are d_1, d_2, r_1, r_2 and h; (b) The CSG of the model; (c) Boundary conditions where Γ_D is fixed and $\tau_x = 100$ N/m^2 and $\tau_y = 200$ N/m^2 were applied to Γ_N as (sinusoidal) bearing loads; (d) FEA mesh of 22 789 tetrahedral elements in step 1; (e) FCM (XVoxel) mesh of $55 \times 16 \times 9$ voxels.



(a) Relative error $r_{\mathbf{u}}$ (b) The number of active voxels (c) Timing

Fig. 11. Performance statistics of Example #2 in Fig. 10: (a) Displacement residual r_u between XVoxel and FCM; (b) The number of active voxels by FCM and XVoxel; (c) Timing of four methods FEA, FCM, XVoxel-FCM and XVoxel-CBN.

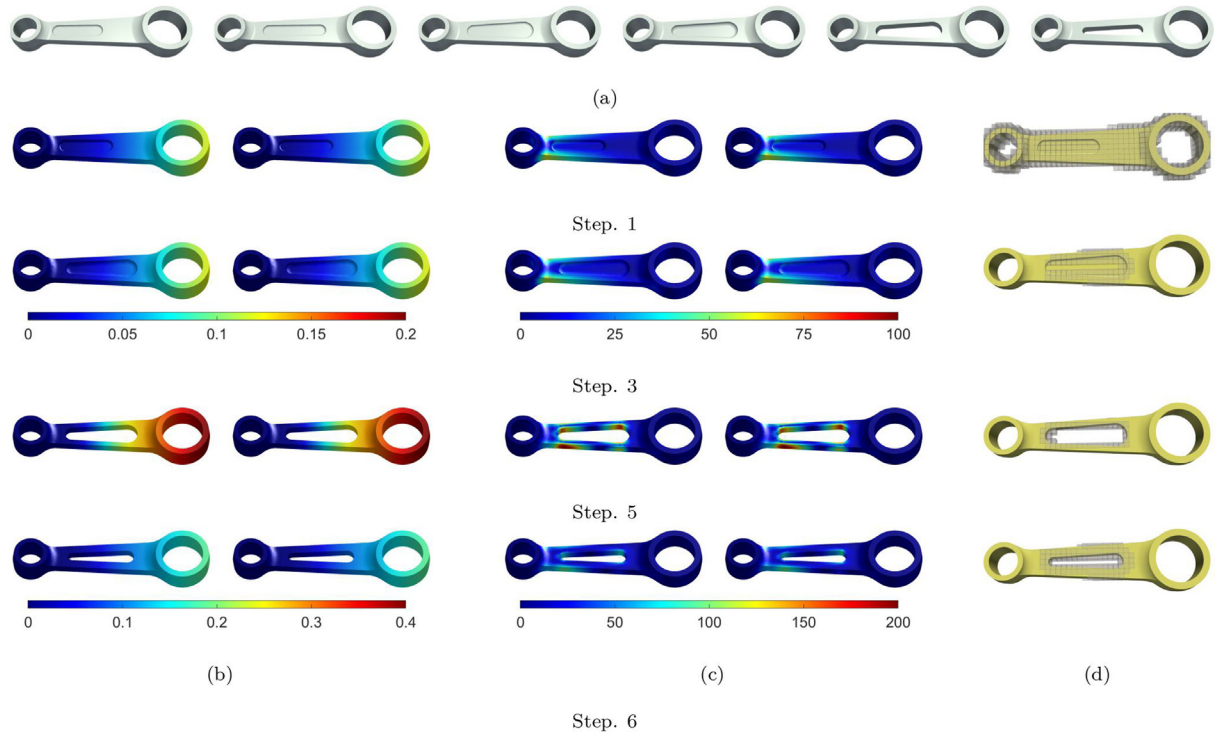


Fig. 12. (a) The models for steps 1 to 6: Initial model with a pair of symmetrical grooves; Modify $d1$ and $d2$ to translate grooves; Increase the right cylinder's radius $r2$; Increase grooves' depth h; Increase groove' depth h to run through the model; Decrease radii r_1 and r_2 to narrow the groove; Results for a connector model in editing steps 1, 3, 5 and 6 (b) Displacement norm (mm) by FEA (left) and FCM/XVoxel (right), (c) Von Mise stress (MPa) from FEA (left) and FCM/XVoxel (right), (d) Active voxels (in gray) of XVoxel.

6.3. Example #3: a pump model under drastic topology variation and varying loads

The proposed method's potentiality in handling drastic topology variations and varying loads was further tested on the complex pump model in Fig. 13. The pump's bottom is fixed and its top and outer side are exerted by forces of 200 N, 100 N respectively. The FEA has 216K tetrahedral elements while FCM (XVoxel) has 40K voxels. The model was edited by the following steps, during which both the model's topology and external loadings are varied:

1. Input an initial model consisting of different cylinders.
2. Add a round corner feature f_1.
3. Add feature f_2 which consists a cube and a cylinder.
4. Add a negative feature f_3 as a union of four cylinders.
5. Add a negative feature f_4.

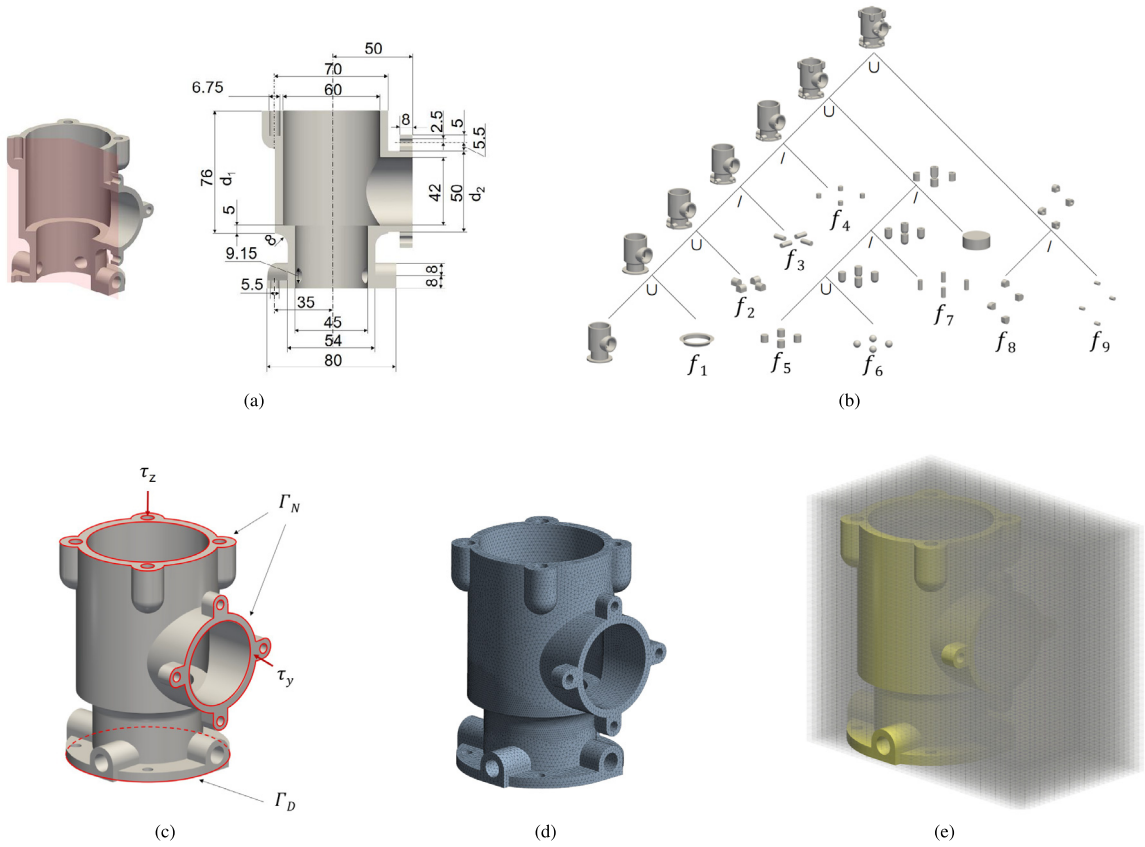


Fig. 13. Example #3. (a) Parameters (mm) of the pump model at the 10th step, where d_1 and d_2 are the variables to modify in step $11-20$; (b) The CSG of the model at the 10th step, where f_i are the features to add in step $1-10$, $i=1,\ldots,9$; (c) Boundary condition, where γ_D is fixed and $\tau_z = 200$ N and $\tau_y = 100$ N; (d) FEA mesh with 216 411 tetrahedron at the 10th step; (e) FCM (XVoxel) mesh with $25 \times 43 \times 37$ voxels.

6. Add a feature f_5.
7. Add a feature f_6 consisting of four spheres.
8. Add a negative feature f_7.
9. Add a feature f_8 consisting of a cube and a cylinder.
10. Add a negative feature f_9.

11–15. Modify design parameter d_1 from 76 to 56 at a step of -4.

16–20. Modify design parameter d_1 from 50 to 70 at a step of 4.

The resulting models during the modification were shown in Figs. 14(a)–(c), with the associated active voxels given in Fig. 14(b). Statistics of the relative errors, numbers of active voxels, timings were compared in Fig. 15, which indicates XVoxel's high simulation accuracy and efficiency, as already confirmed in Examples #1 and #2.

We in particular observed from Fig. 15(b) that the number of active voxels of XVoxel is only around 1/4 of FCM's, demonstrating XVoxel's strong ability in properly selecting active voxels regardless of the complex feature shape and feature operations. The nice property in turn resulted in a 4-time efficiency improvement of XVoxel-FCM in comparison to FCM; XVoxel-CBN even achieved a 50 times efficiency improvement. Such efficiency is very useful in obtaining interactive simulation feedback in modifying designs.

6.4. Example #4: a bracket model for parametric design optimization

The proposed method's ability in feature-based parametric design optimization was first tested on an asymmetric bracket model in Fig. 16, which has a negative groove feature composed of several cylinders and prisms. The model was fixed on its left hole, and subject to an axial load of $100\sqrt{5}$N on its right hole. The model is discretized into 9.1K voxels for FCM (XVoxel) based simulation.

The design goal is to minimize the bracket's compliance under a volume ratio of 0.9 by varying the locations and sizes of the groove. The design variables are: circle centers x_i, y_i and radii r_i (i=1, 2, 3, 4) of the four corner circles, their inscribed circle radius r_5 and circumscribed circles' radii r_6, r_7, r_8 (in 2D plane). The associated geometric constraints are formulated as follows so as to produce a valid geometry:

$$
\begin{cases}
\mathbf{Ku} = \mathbf{F}, & \\
V \le 0.9V_0, & \\
\|(x_i, y_i) - O_5\| + r_i = r_5, & i = 1, 2 \\
\|(x_i, y_i) - O_6\| - r_i = r_6, & i = 3, 4 \\
\|(x_i, y_i) - O_7\| - r_i = r_7, & i = 1, 4 \\
\|(x_i, y_i) - O_8\| - r_i = r_8, & i = 2, 3 \\
x_i^{min} \le x_i \le x_i^{max}, & i = 1, \ldots, 4 \\
y_i^{min} \le y_i \le y_i^{max}, & i = 1, \ldots, 4 \\
r_j^{min} \le r_j \le r_j^{max}, & j = 1, \ldots, 8
\end{cases}
$$

where V_0 is the volume of the original model, O_i's are the 2D circle centers' coordinates, and ranges of x_i, y_i, r_j were set so that the features would not move out of the bracket.

Some intermediate structures during optimization were shown in Fig. 17(a), where the groove gradually enlarged its size to meet the volume constraint while moving to the left side for performance improvement. Stress distributions were also plotted in Fig. 17(b), and the active voxels were shown in gray in Fig. 17(c).

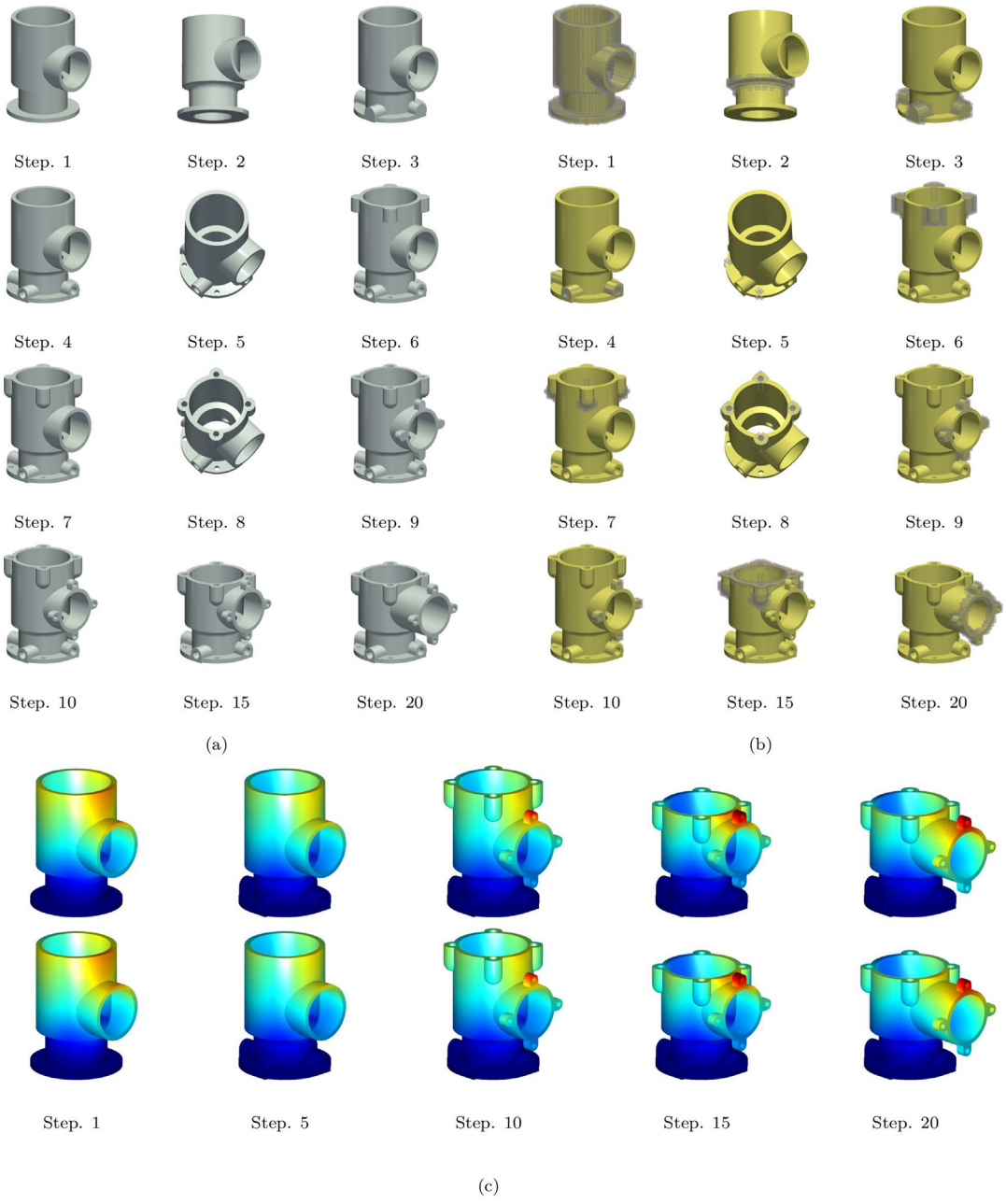
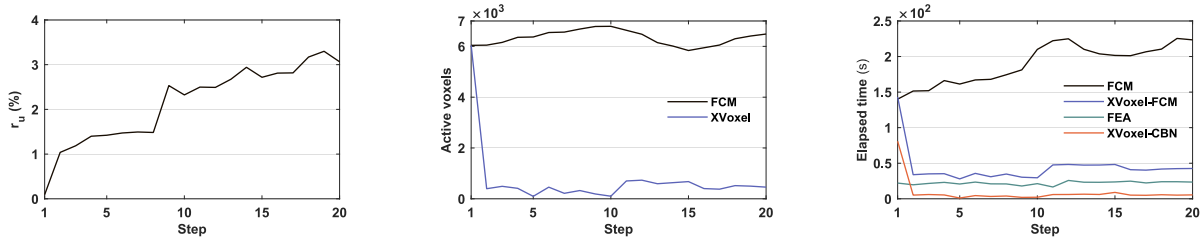


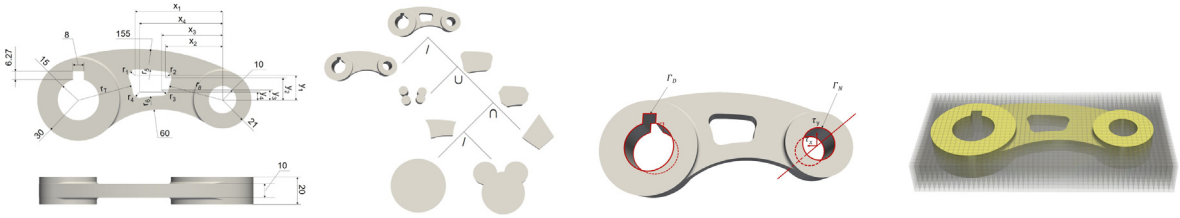
Fig. 14. (a) The models in editing steps 1–10, 15 and 20, (b) Active voxels (in gray) of XVoxel in editing steps 1–10, 15 and 20, (c) The simulation results for a pump model in editing steps 1, 5, 10, 15 and 20. Displacement norm (mm) of FEA (up) and of FCM/XVoxel (down).



(a) Relative error $r_{\mathbf{u}}$ (b) The number of active voxels (c) Timing

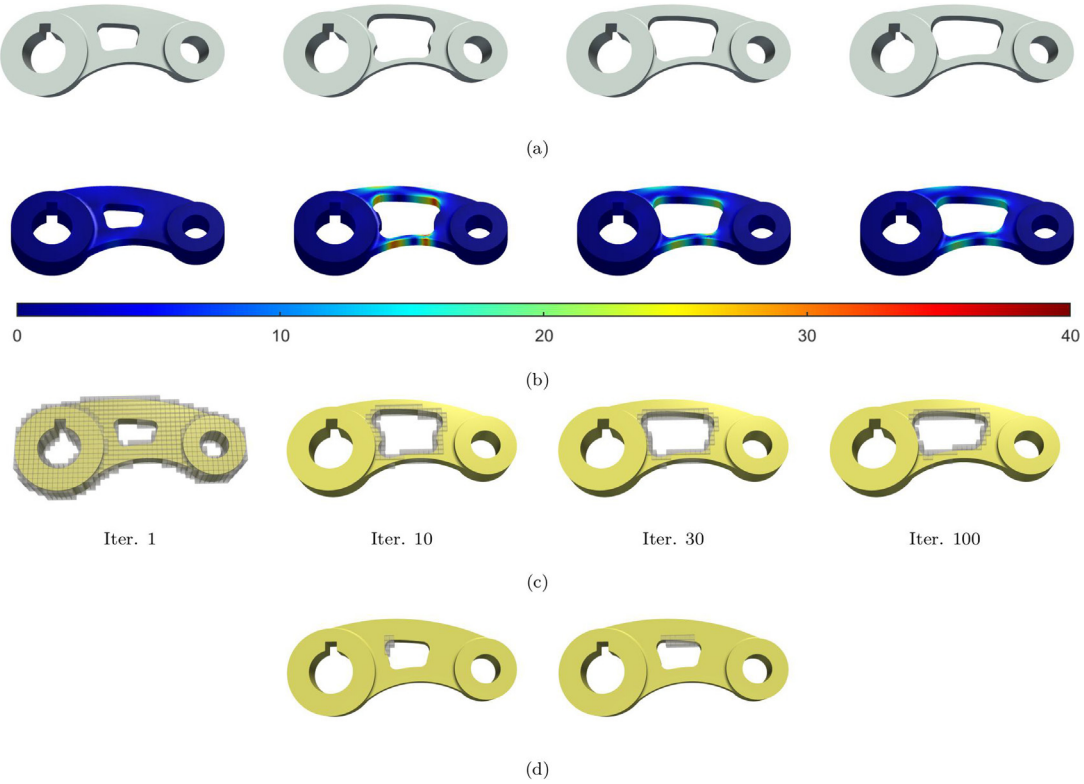
Fig. 15. Performance statistics of Example #3 in Fig. 13: (a) Displacement residual r_u between XVoxel and FCM; (b) The number of active voxels by FCM and XVoxel; (c) Timing of four methods FEA, FCM, XVoxel-FCM and XVoxel-CBN.

The optimization was stopped after 100 iterations, where the features and their relative constraints were all maintained during the optimization; the convergence curve was plotted in Fig. 18(a). During the optimization, XVoxel only had approximately 1/8



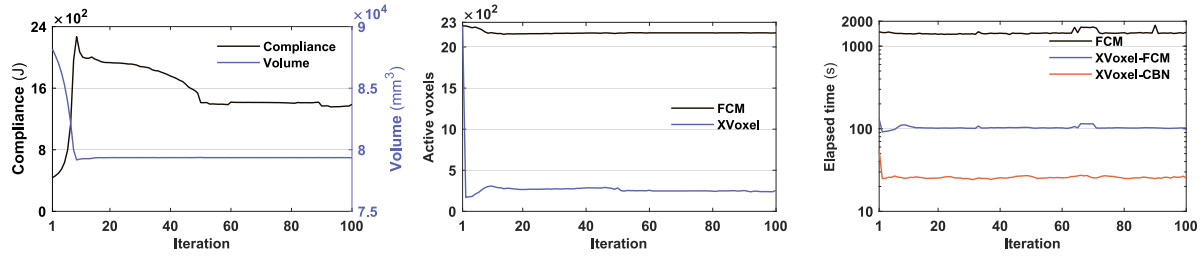
(a) (b) (c) (d)

Fig. 16. Example #4. (a) Parameters (mm) of the model. The position and size of a groove are optimized by changing the position and radius of the tangential cylinders that form it, where the design variables are x_i, y_i and r_j(i=1, 2, …, 4; j=1, 2, …, 8); (b) The CSG of the model; (c) Boundary conditions, where Γ_D is fixed and $\tau_x = 100$ N, $\tau_y = 200$ N are applied on Γ_N as (sinusoidal) bearing loads; (d) FCM (XVoxel) mesh with 52 × 25 × 7 voxels.



(a) (b) (c) (d)

Fig. 17. Results for a bracket model in optimization iterations of steps 1, 10, 30 and 100 (a) The model, (b) Von Mise stress (MPa) of FCM/XVoxel, (c) Active voxels (in gray) of XVoxel, (d) Active voxels (in gray) for calculating the sensitivities with respect to x_1, y_1, r_1 (left) and r_5 (right) in the first iteration.



(a) Convergence curves of compliance and vol- (b) The number of active voxels (c) Timing

Fig. 18. Performance statistics of Example #4 in Fig. 16: (a) Convergence curves of compliance and volume by XVoxel and FCM; (b) The number of active voxels by FCM and XVoxel; (c) Timing of three methods FCM, XVoxel-FCM and XVoxel-CBN.

active voxels of FCM, with a much-improved efficiency; see also Figs. 18(b) and (c). XVoxel-FCM and XVoxel-CBN respectively achieved around 13× and 50× efficiency improvements compared to FCM. By maintaining the feature lists during optimization, the simulation was greatly accelerated by local recomputations for active voxels.

As can be seen from Fig. 18(a), the compliance curve did not go steadily, but first went up quickly to the peak, then went down. This is because compliance is highly sensitive to volume changes, and the volume factor dominates the optimization before reaching the specified volume limit. That is, the optimization will quickly reduce the volume toward the specified volume limit

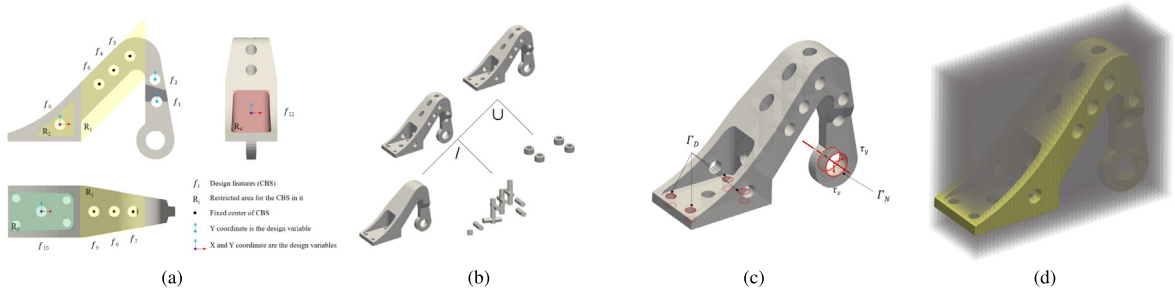


Fig. 19. Example #5. (a) The CBS (Closed B-Spline) design features $f_i,\ i = 1, \ldots, 11$ and restricted area R_1 (in yellow) for $f_j,\ j = 3, 4, 5, 7, 8, 9$, R_2 (in yellow) for f_6, R_3 (in green) for f_{10} and R_4 (in red) for f_{11}. The x-coordinate of $f_i,\ i = 1, 2$ and x-, y- coordinates of $f_j,\ i = 3, 4, 5, 7, 8, 9$ are fixed. There are 250 design variables in total, and each feature contains 22 control radii and unfixed x-, y-coordinates; (b) The CSG of the model; (c) Boundary conditions, where Γ_D is fixed and $\tau_x = 100$ N and $\tau_y = 100$ N are exerted on Γ_N as (sinusoidal) bearing loads; (d) FCM (XVoxel) mesh with $50 \times 17 \times 39$ voxels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at the beginning (see the volume curve in the same figure). After getting the peak (at the 10th iteration), the optimization algorithm (i.e., the globally convergent method of moving asymptotes) reduces compliance effectively while keeping the volume above the limit. This process corresponds to the going down phase in Fig. 18(a).

In conducting the optimization, the sensitivities for this example were derived using finite differences. Without proper handling, the computation would be very expensive as it requires a complete FE recomputation for all 16 design parameters. Instead, XVoxel requires much fewer active voxels for the finite difference computations, as only one feature parameter was varied in each finite difference process; see also Fig. 17(d) for an illustration.

6.5. Example #5: a bearing bracket model for parametric design optimization with varied topology

The proposed method's robustness in handling complex parametric design optimization with varied structural topology was tested on a bearing bracket model in Fig. 19, where the design model, CSG tree, boundary conditions, discrete voxels were respectively shown in Figs. 19(a), (b), (c), (d). The FCM (XVoxel) has 33K voxels, and the volume ratio constraint was set to be 0.6.

The model has 11 CBS (Closed B-Spline) negative features, each with 24 control radii and 2 position coordinates, where the first and last radii of each feature are driven parameters for higher-order geometric continuity. In order to maintain the features to produce a valid structure of complex topology, features f_1 and f_2 were restricted to move only along the y-axis, features $f_3, f_4, f_5, f_7, f_8, f_9$ were fixed, and features f_6, f_{10} and f_{11} only moved along a prescribed plane; see also Fig. 19(a). Altogether, there are in total $(24 - 2 + 2) \times 11 - 14 = 250$ design variables.

The intermediate results of the examples are given in Fig. 20 and performance statistics of the example in Fig. 21. As can be seen from Fig. 20, during the optimization, the CBS features gradually expanded to meet the volume constraint, resulting in drastic topology variations in the arm and base part of the bearing bracket, before reaching convergence after 35 iterations. The XVoxel-based optimization successfully handled the topological changes.

The XVoxel model has around half active voxels of FCM, and XVoxel-FCM is about 1.3× faster than FCM in each iteration. The speedup is much smaller in comparison with Example #4, as the CBS features were distributed more broadly within the bracket model, and therefore resulted in more active voxels. Nevertheless, XVoxel-CBN has gained about 5× efficiency improvement. Again, the proposed method remains robust in the feature-based design optimization framework in handling such a large-scale model with a large number of design variables and drastic topology changes.

6.6. Example #6: bearing plates containing varying cylindrical supports

In the end, we further test the approach's ability in handling features of varied locations, sizes or orientations using the example in Fig. 22. The approach's potentiality is also tested in removing unnecessary features in the construction history during the optimization. The model has two plates supported by 9 cylinders in the middle. Each cylindrical feature contains 5 design variables, including position coordinates x_i, y_i, directions α_i, β_i and a radius r_i for $i = 1, \ldots, 9$. The upper surface of the top plate is exerted by a radiant force F defined as $F(x, y) = F_c \cos\left(\sqrt{(x - x_c)^2 + (y - y_c)^2}/(9\sqrt{2}) \cdot \frac{\pi}{2}\right)^8$, where $F_c = 50$ N, $x_c = x_c = 9$ mm, $0 \le x, y \le 12$ while the lower surface of the bottom plate is fixed, as shown in Fig. 22(c). Three different tests are conducted at a volume fraction of 1.2 times the original volume of the cylinders by optimizing x_i, y_i and r_i; α_i, β_i and r_i; all the variables.

The optimized structures and convergence curves are shown in Fig. 23. All the cases lead to reliable convergence and produce structures respectively of compliance 39.3, 47.1, and 36.0. It can be observed that the full variable optimization has the best convergence rate. Meanwhile, note in Fig. 24(c) that three design features are able to run out of the design area during optimization, demonstrating the approach's ability in removing unnecessary features automatically. In this case, we can conveniently remove the features from the feature list. The variations of the features are also shown in Fig. 24.

6.7. Discussion and limitations

As one may have noticed from the above examples, although the final results of our XVoxel method can admit topology changes, their overall shapes still follow a similar structural pattern to those of the initial designs before optimization. This is because the method is designated for feature-based CAD and parametric optimization, where feature semantics (and therefore the overall shapes) often need to be respected. For example, the boundary representation of the final results is consistently composed of smooth parametric surfaces; there is no way the boundary takes discrete, free shapes (which is the case for SIMP or the voxel density method [20]).

For this reason, our method works better for situations like fine or semi-fine design tuning. Dimensional variations and topology changes can be large (as demonstrated by the example in Fig. 20) but cannot be radically different. If design changes of this sort are desired, other methods, e.g., the voxel density method, should be used. For the same reason, our method needs to take as input an initial design that does not deviate too much from the

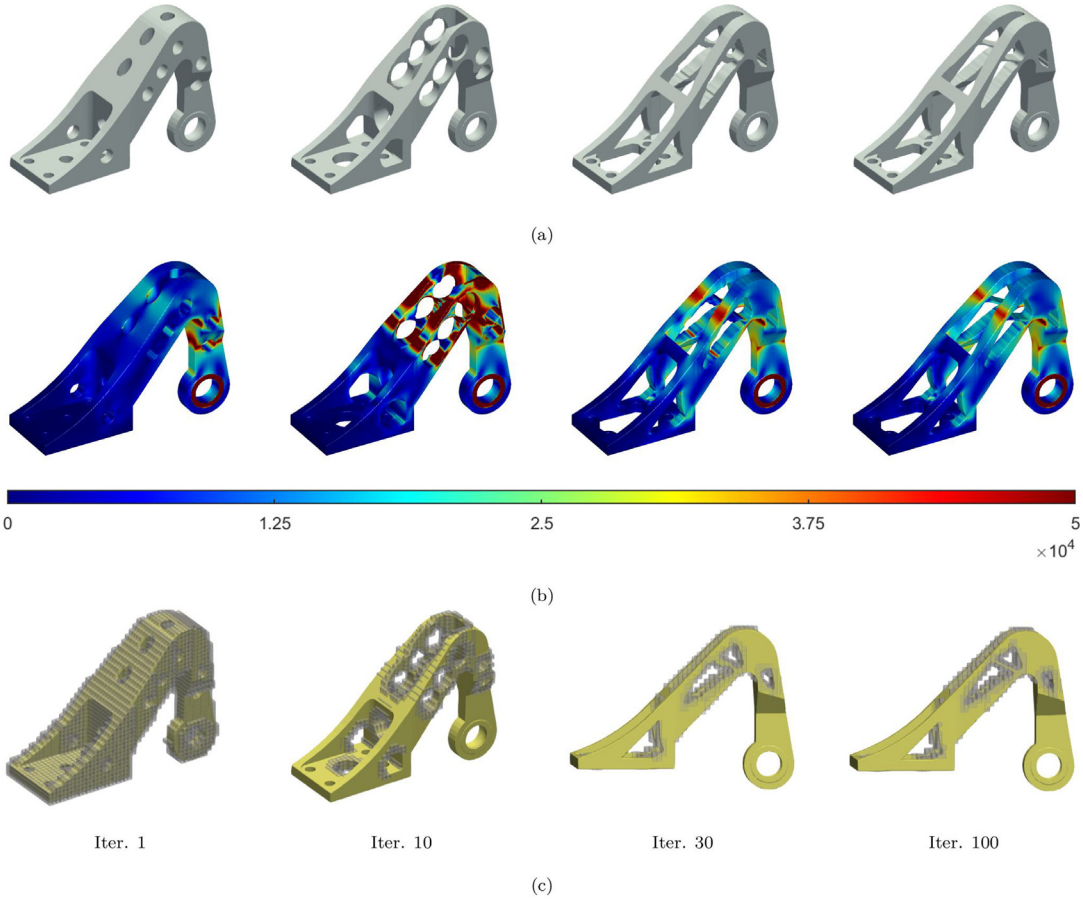
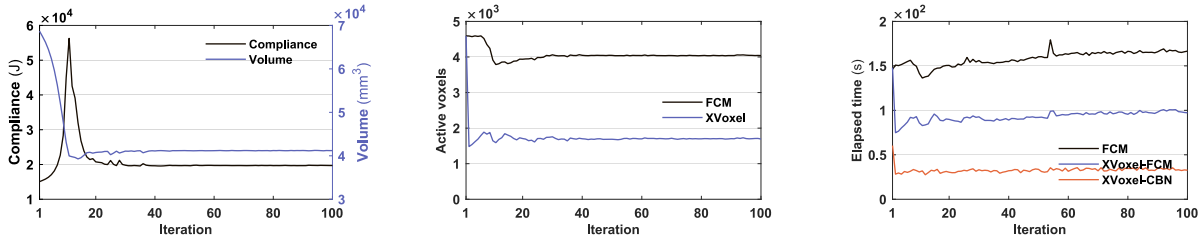


Fig. 20. Results for a bearing bracket model in optimization iterations of steps 1, 10, 30 and 100: (a) the models, (b) Von Mise stress (MPa) of FCM/XVoxel, (c) active voxels (in gray) of XVoxel.

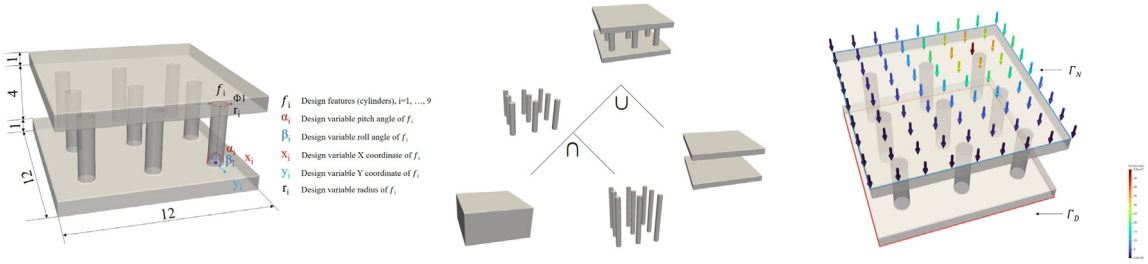


(a) Convergence curves of compliance and volume

(b) The number of active voxels

(c) Timing

Fig. 21. Performance statistics of Example #5 in Fig. 19: (a) Convergence curves of compliance and volume by XVoxel and FCM; (b) The number of active voxels by FCM and XVoxel; (c) Timing of three methods FCM, XVoxel-FCM and XVoxel-CBN.



(a) (b) (c)

Fig. 22. Example #6. (a) Parameters (mm) of the model, where the design variables are $x_i, y_i, \alpha_i, \beta_i$ and r_i for $i = 1, 2, \ldots, 9$; (b) The CSG of the model; (c) Boundary conditions, where Γ_D is fixed and Γ_N is exerted by a radially decayed force field $F(x, y) = F_c \cos\left(\sqrt{(x - x_c)^2 + (y - y_c)^2}/(9\sqrt{2}) \cdot \frac{\pi}{2}\right)^8$, where $F_c = 50$ N, $x_c = x_c = 9$ mm, $0 \le x, y \le 12$.

Table 2
Performance comparison of FCM, XVoxel-FCM and XVoxel-CBN on the tested numerical examples. The table shows the size of mesh, the number of DOFs, the total time and the total acceleration ratio..

Example	Mesh size	DOFs		Timings (total steps/iterations)			Total acceleration ratio (based on FCM)	
		FCM(XVoxel-FCM)	XVoxel-CBN	FCM	XVoxel-FCM	XVoxel-CBN	XVoxel-FCM	XVoxel-CBN
#1	675	63,480	42,280	116.0	117.2	12.8	0.990	9.06
#2	4,004	108,135	255,024	259.9	84.5	21.6	3.08	12.0
#3	39,775	998,325	2,365,080	3,801.4	883.6	174.6	4.30	21.8
#4	9100	240,975	250,638	145,022.8	10,321.3	2597.2	14.1	55.8
#5	33,150	279,265	282,006	15,849.0	9,358.5	3278.5	1.69	4.83

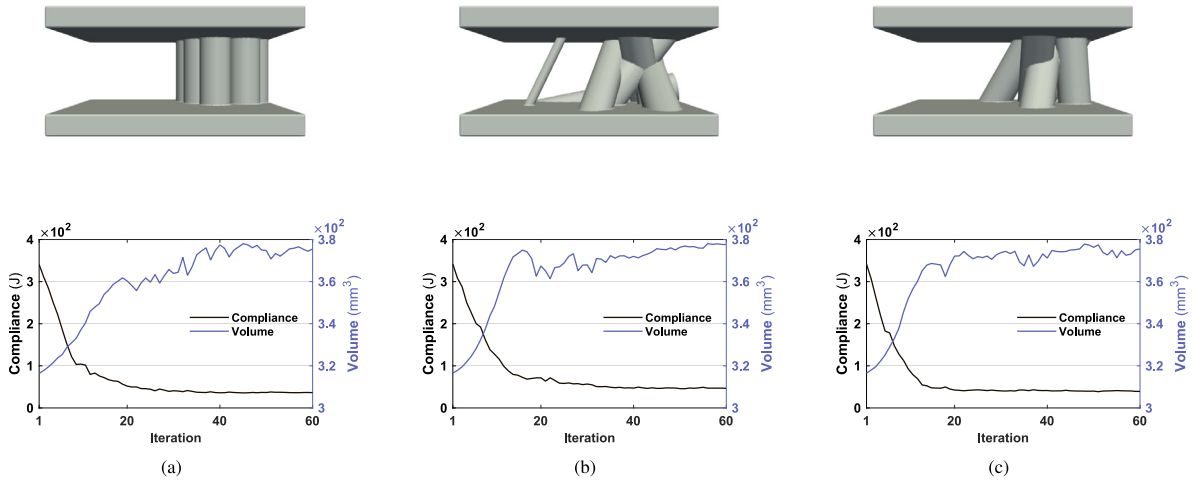


Fig. 23. Optimized models and convergence curves of structural compliance and cylindrical volumes: (a) x_i, y_i and r_i as design variables; (b) α_i, β_i and r_i as design variables; (c) all 5 design variables; in Fig. 22.

final result. This is different from methods like the voxel density method. Their input can even be a block without bearing any similarity to the final result.

7. Conclusions

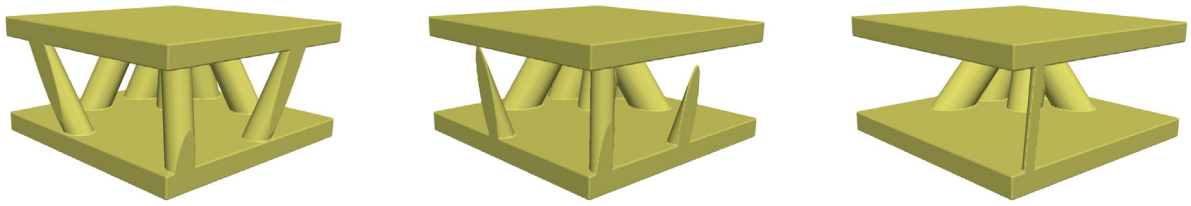
An XVoxel-based parametric design optimization method for feature models has been presented in this paper. The proposed method combines the local regularity of voxel models and the global semantic information of feature models to facilitate the automatic linking between CAD and CAE. By further integrating XVoxels, FCM and CBN, design modifications and simulation updates can be looped in an efficient and robust manner, without involving labor-intensive conversion between CAD models and CAE models. The effectiveness of the proposed method has been validated by various numerical examples with complex topology variations and varying loads. And a computational efficiency improvement of up to 55.8 times the existing FCM method has been achieved, see Table 2.

We consider the proposed XVoxel method as an alternative attempt toward the long-standing research objective of a unified model representation scheme that can completely, compactly, and associatively represent the contents of both CAD and CAE models. The method builds itself upon a new concept called semantic voxels to provide the advantage of avoiding B-rep model simplification and mesh generation. These two procedures could present a particular challenge for existing methods; for example, the quad/hex meshing required by IGA is never easy if the geometry is complex. For this reason, a typical use case where XVoxel is preferable over the others is when the design to be optimized is given as a feature model and its overall shape is complex.

A couple of interesting improvement directions for the XVoxel method are noted here. As already noted in Sections 3.3 and 4.3, XVoxel models can much reduce the dependence of simulation on model simplification and can be directly used to guide the simulation-suitable model simplification process. Nevertheless, the method, in its current form, is still not able to handle dimension reduction, which is the other important step in model idealization [10,48,49]. Extending the method to including dimension reduction is among the research studies to be carried out in our group. Another interesting improvement direction is that the proposed method has only been implemented in Matlab for proof of concept. Its further implementation on basis of commercial/opensource feature modelers, e.g., Open CASCADE, and then release as an open-source plugin is of great interest to our future research work.

In industrial design optimization, innovative designs often require heavy optimization within a large design space but time resources are limited. This entails the use of dimensionality reduction techniques on the design space during optimization. The proposed XVoxel method has a good potential to integrate with dimensionality reduction methods, e.g., parametric model embedding [57], due to its generality on the input model. Such an interesting integration is among our future work. Another improvement direction lies in the efficiency of the proposed method. Currently, our use of octrees in finding Gaussian points leads to a time-consuming simulation. If augmented with some adaptive meshing method (e.g., [58]), the proposed method can be much accelerated.

It is also worth noting that a model may correspond to multiple construction ways using Boolean operations. For different construction ways, their design variation spaces may be different, so do the corresponding optimization processes. Therefore, the construction way needs to be carefully thought out before using our method. The proposed method, in its current form, only focuses on parametric optimization on given models, and it cannot automatically find an appropriate construction way. Such an automatic selection mechanism is of great interest to future work. In addition, If a B-Rep model is provided, a Boundary-to-CSG conversion procedure (e.g., the method presented in [59]) is necessary for the proposed method to work. Another limitation of the proposed method is that the optimization result is affected by



(a) loop 14 (b) loop 17 (c) loop 20

Fig. 24. The disappearance of features during optimization in Fig. 20(c).

the size of the cells used in the simulation, a consequence of using FCM for simulation (see [60] for a detailed discussion). Currently, there is no principled way to choose the best cell size, and the usual solution is using empirical tuning to find a good cell size.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

This work has been partially supported by the National Key Research and Development Program of China (No. 2020YFC2201303), the National Natural Science Foundation of China (No. 62102355, 61872320), the Natural Science Foundation of Zhejiang Province (No. LQ22F020012), Key Research and Development Program of Zhejiang Province (No. 2022C01025), and the fund from the Fifth Electronic Research Institute of Ministry of Industry and Information Technology (No. HK07202200877).

References

[1] Shapiro V, Tsukanov I, Grishin A. Geometric issues in computer aided design/computer aided engineering integration. J Comput Inf Sci Eng 2011;11(2).
[2] Daxin SD, Prajapati JM. Parametric shape optimization techniques based on meshless methods: A review. Struct Multidiscip Optim 2017;56(5):1197–214.
[3] Chen J, Shapiro V, Suresh K, Tsukanov I. Shape optimization with topological changes and parametric control. Internat J Numer Methods Engrg 2007;71(3):313–46.
[4] Shah JJ, Mäntylä M. Parametric and feature-based CAD/CAM: concepts, techniques, and applications. John Wiley and Sons; 1995.
[5] Boussuge F, Tierney CM, Vilmart H, Robinson TT, Armstrong CG, Nolan DC, et al. Capturing simulation intent in an ontology: CAD and CAE integration application. J Eng Des 2019;30(10–12):688–725.
[6] Boussuge F, Armstrong CG, Tierney CM, Robinson TT. Application of tensor factorisation for CAE model preparation from CAD assembly models. Comput Aided Des 2022;152:103372.
[7] Nolan DC, Tierney CM, Armstrong CG, Robinson TT. Defining simulation intent. Comput Aided Des 2015;59:50–63.
[8] Zhu H, Menq C-H. B-rep model simplification by automatic fillet/round suppressing for efficient automatic feature recognition. Comput Aided Des 2002;34(2):109–23.
[9] Turevsky I, Gopalakrishnan SH, Suresh K. An efficient numerical method for computing the topological sensitivity of arbitrary-shaped features in plate bending. Internat J Numer Methods Engrg 2009;79(13):1683–702.
[10] Fine L, Remondini L, Leon JC. Automated generation of FEA models through idealization operators. Internat J Numer Methods Engrg 2000;49(1–2):83–108.
[11] Li M, Zhang B, Martin RR. Second-order defeaturing error estimation for multiple boundary features. Internat J Numer Methods Engrg 2015;100(5):321–46.
[12] Liu J, To AC. CAD-based topology optimization system with dynamic feature shape and modeling history evolution. J Mech Des 2019;142(7):1–25.
[13] Bazilevs Y, Calo VM, Cottrell JA, Evans JA, Hughes TJR, Lipton S, et al. Isogeometric analysis using T-splines. Comput Methods Appl Mech Engrg 2015;199(5–8):229–63.
[14] Zou Q, Feng H-Y. Push-pull direct modeling of solid CAD models. Adv Eng Softw 2019;127:59–69.
[15] Sheffer A, Blacker T, Clements J, Bercovier M. Virtual topology construction and applications. In: Strasser W, Klein R, Rau R, editors. Geometric modeling: theory and practice: the state of the art. Berlin, Heidelberg: Springer Berlin Heidelberg; 1997, p. 247–59.
[16] Tierney CM, Sun L, Robinson TT, Armstrong CG. Using virtual topology operations to generate analysis topology. Comput Aided Des 2017;85:154–67.
[17] Si H. TetGen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans Math Software 2015;41(2):1–36.
[18] Hu Y, Schneider T, Wang B, Zorin D, Panozzo D. Fast tetrahedral meshing in the wild. ACM Trans Graph 2020;39(4):1–18.
[19] Blacker T. Automated conformal hexahedral meshing constraints, challenges and opportunities. Eng Comput 2001;17.
[20] Bendsoe MP, Sigmund O. Topology optimization: theory, methods and applications. Springer; 2003.
[21] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Engrg 2005;194(39–41):4135–95.
[22] Li K, Qian X. Isogeometric analysis and shape optimization via boundary integral. Comput Aided Des 2011;43(11):1427–37.
[23] Qian X. Full analytical sensitivities in NURBS based isogeometric shape optimization. Comput Methods Appl Mech Engrg 2010;199(29):2059–71.
[24] Seo Y-D, Kim H-J, Youn S-K. Isogeometric topology optimization using trimmed spline surfaces. Comput Methods Appl Mech Engrg 2010;199(49):3270–96.
[25] Xie X, Wang S, Xu M, Jiang N, Wang Y. A hierarchical spline based isogeometric topology optimization using moving morphable components. Comput Methods Appl Mech Engrg 2020;360:112696.
[26] Schillinger D, Ruess M. The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. Arch Comput Methods Eng 2015;22(3):391–455.
[27] Rank E, Ruess M, Kollmannsberger S, Schillinger D, Düster A. Geometric modeling, isogeometric analysis and the finite cell method. Comput Methods Appl Mech Engrg 2012;249–252:104–15.
[28] Yingjun W, Benson DJ. Isogeometric analysis for parameterized LSM-based structural topology optimization. Comput Mech 2016;57(1):19–35.
[29] Wassermann B, Kollmannsberger S, Bog T, Rank E. From geometric design to numerical analysis: A direct approach using the finite cell method on constructive solid geometry. Comput Math Appl 2017;74(7):1703–26.
[30] Chen J, Freytag M, Shapiro V. Shape sensitivity of constructively represented geometric models. Comput Aided Geom Design 2008;25(7):470–88.
[31] Zhu L, Li M, Martin RR. Direct simulation for CAD models undergoing parametric modifications. Comput Aided Des 2016;78:3–13.
[32] Chinesta F, Ammar A, Cueto E. Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. Arch Comput Methods Eng 2010;17(4):327–50.
[33] Schulz A, Xu J, Zhu B, Zheng C, Grinspun E, Matusik W. Interactive design space exploration and optimization for CAD models. ACM Trans Graph 2017;36(4):157.
[34] Hafner C, Schumacher C, Knoop E, Auzinger T, Bcher M. X-CAD: optimizing CAD models with extended finite elements. ACM Trans Graph 2019;38(6):1–15.
[35] Zou Q, Feng H-Y. A decision-support method for information inconsistency resolution in direct modeling of CAD models. Adv Eng Inform 2020;44:101087.

[36] Zhu J-H, Zhang W-H, Xia L. Topology optimization in aircraft and aerospace structures design. Arch Comput Methods Eng 2016;23(4):595–622.
[37] Jiu L, Zhang W, Meng L, Zhou Y, Chen L. A CAD-oriented structural topology optimization method. Comput Struct 2020;239:106324.
[38] Zhang W, Zhou Y, Zhu J. A comprehensive study of feature definitions with solids and voids for topology optimization. Comput Methods Appl Mech Engrg 2017;325:289–313.
[39] Zhou Y, Zhang W, Zhu J, Xu Z. Feature-driven topology optimization method with signed distance function. Comput Methods Appl Mech Engrg 2016;310:1–32.
[40] Guo X, Zhang W, Zhong W. Doing topology optimization explicitly and geometrically-a new moving morphable components based framework. J Appl Mech-Trans Asme 2014;81(8).
[41] Guo X, Zhao X, Zhang W, Yan J, Sun G. Multi-scale robust design and optimization considering load uncertainties. Comput Methods Appl Mech Engrg 2015;283:994–1009.
[42] Zhang W, Chen J, Zhu X, Zhou J, Xue D, Lei X, et al. Explicit three dimensional topology optimization via moving morphable void (MMV) approach. Comput Methods Appl Mech Engrg 2017;322:590–614.
[43] Zou Q, Feng H-Y. A robust direct modeling method for quadric B-rep models based on geometry–topology inconsistency tracking. Eng Comput 2022;38(4):3815–30.
[44] Shah JJ. Designing with parametric CAD: Classification and comparison of construction techniques. In: International workshop on geometric modelling. Springer; 1998, p. 53–68.
[45] Lee SH. A CAD–CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. Comput Aided Des 2005;37(9):941–55.
[46] Zou Q, Feng H-Y. Variational B-rep model analysis for direct modeling using geometric perturbation. J Comput Des Eng 2019;6(4):606–16.
[47] Young G, Krishnamurthy A. GPU-accelerated generation and rendering of multi-level voxel representations of solid models. Comput Graph 2018;75:11–24.
[48] Armstrong CG. Modelling requirements for finite-element analysis. Comput Aided Des 1994;26(7):573–8.
[49] Chong CS, Kumar AS, Lee K. Automatic solid decomposition and reduction for non-manifold geometric model generation. Comput Aided Des 2004;36(13):1357–69.
[50] Shapiro V, Vossler DL. What is a parametric family of solids? In: Proceedings of the third ACM symposium on solid modeling and applications. 1995, p. 43–54.
[51] Rossignac J. IBNC: integrated boundary and natural CSG for polyhedra (review, simplifications, and integration of prior art). Comput Aided Des 2022;50:103296.
[52] Wang C, Chen Y. Thickening freeform surfaces for solid fabrication. Rapid Prototyp J 2013;19(6):395–406.
[53] Jones MW, Baerentzen JA, Sramek M. 3D distance fields: A survey of techniques and applications. IEEE Trans Vis Comput Graphics 2006;12(4):581–99.
[54] Li M, Hu J. Analysis of heterogeneous structures of non-separated scales using curved bridge nodes. Comput Methods Appl Mech Engrg 2022;392:114582.
[55] Juntunen M, Stenberg R. Nitsche's method for general boundary conditions. Math Comp 2009;78(267):1353–74.
[56] Zillober C. A globally convergent version of the method of moving asymptotes. Struct Optim 1993;6(3):166–74.
[57] Serani A, Diez M. Parametric model embedding. Comput Methods Appl Mech Engrg 2023;404:115776.
[58] Qian J, Zhang Y. Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies. Eng Comput 2012;28:345–59.
[59] Shapiro V, Vossler DL. Separation for boundary to CSG conversion. ACM Trans Graph 1993;12(1):35–55.
[60] Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. Comput Methods Appl Mech Engrg 2008;197(45–48):3768–82.