Research And Implementation of Isosurface Extraction Algorithm Based on Flexicubes

Yuyang Wang

Silesian University of Technology

yw304865@student.polsl.pl

Abstract. Surface meshes are fundamental in the representation, transmission, and generation of three-dimensional geometric figures across various domains, such as computer graphics and robotics. The utilization of surface meshes brings about numerous advantages, including efficient hardware-accelerated rendering, support for solving equations in physical simulation, and facilitation of geometric processing. These attributes contribute to the creation of concise and accurate representations of arbitrary surfaces. It's essential to note that not all meshes possess these favorable characteristics; they are typically associated with high-quality grids. The efficiency of hardware-accelerated rendering and the ability to solve equations in physical simulation and geometric processing are often features implemented on such high-quality grids. Conversely, when a grid contains an excessive number of elements, issues like self-intersecting and sliding elements may arise, or the underlying geometry might not be accurately captured. In such cases, the mesh becomes unsuitable for downstream tasks, underscoring the importance of generating high-quality meshes tailored to specific shapes. Nonetheless, this task is often tedious and demands significant manual effort.The recent proliferation of algorithmic content generation and generative 3D modeling tools has given rise to an increased demand for automatic mesh generation. Traditionally, the creation of high-quality grids has been the domain of skilled technical artists and modelers. However, with the advent of automatic algorithm pipelines, the task is increasingly being addressed more efficiently and autonomously.Automatic mesh generation is now at the forefront of advancements, providing solutions to challenges that were previously reliant on manual intervention. These algorithms leverage computational techniques to analyze and understand the desired shape parameters, allowing for the generation of high-quality meshes without the need for extensive manual involvement. This shift is transforming the landscape of 3D modeling, making it more accessible and efficient for a broader range of users.The implications of automatic mesh generation extend beyond simplifying the creation process. They contribute to democratizing access to sophisticated 3D modeling capabilities, enabling individuals with varying levels of technical expertise to engage in content creation. Moreover, the efficiency of automatic mesh generation tools is invaluable in scenarios where rapid prototyping, iterative design processes, or large-scale content generation is required. Surface meshes serve as crucial elements in various computational fields, and their quality significantly influences their applicability in downstream tasks. The advent of automatic mesh generation, driven by algorithmic content creation tools, has revolutionized the traditional manual approaches, making 3D modeling more accessible and efficient. As technology continues to advance, the role of automatic mesh generation in shaping the future of computer graphics and related fields becomes increasingly prominent. Automatic algorithm pipeline is usually based on differentiable grid generation, that is, parameterization of three-dimensional surface grid space, and optimization of various targets through gradient-based technology. For example, reverse rendering, structural optimization and 3D modeling all make use of this basic building block. In a perfect world, such applications would perform simple gradient descent on some grid representations to optimize their desired goals. However, there are many obstacles in this workflow, from the basic problem of how to optimize grids with different topologies to the lack of stability and robustness of existing formulas, which lead to the output of irreparable low-quality grids. In this work, we put forward a new formula, which makes it closer to this goal and significantly improves the simplicity and quality of differentiable grid generation in various downstream tasks.Unless very careful initialization, local subdivision and regularization are used, it is easy to have degeneracy and local minima when directly optimizing the vertex position of the grid. The optimization of scalar fields or signed distance functions (SDFs) to extract triangular meshes plays a crucial role in various fields such as photogrammetry, generative modeling, and inverse physics. The choice of representation method for the scalar function and the mesh extraction algorithm significantly impacts the overall

optimization pipeline's performance.One prevalent approach involves defining and optimizing a scalar field or SDF in space, followed by the extraction of a triangular mesh resembling the level set of the function. The challenge lies in the fact that the space of possible meshes is constrained. The algorithm chosen for mesh extraction directly influences the properties of the generated shapes.Currently, gradient-based mesh optimization often represents the three-dimensional surface mesh as the equivalent surface of a scalar field. This technique has gained popularity in applications like photogrammetry due to its effectiveness in generating realistic shapes. Classical isosurface extraction algorithms, such as Marching Cubes or Dual Contouring, are commonly employed in existing implementations. However, these algorithms are typically designed for extracting meshes from fixed known fields. In the context of optimization, they struggle to maintain mesh flexibility and may encounter issues of numerical instability.In response to these challenges, a novel isosurface representation method called FlexiCubes has been introduced. FlexiCubes is specifically designed to optimize unknown meshes concerning geometric, visual, and even physical properties. The core concept involves introducing carefully selected parameters into the representation method, allowing for local flexible adjustments to the extracted mesh geometry and connectivity. During optimization of downstream tasks, these parameters are updated concurrently with the underlying scalar field through an automatic discrimination process.FlexiCubes addresses the limitations of traditional isosurface extraction algorithms by providing a more adaptive and versatile framework. By incorporating additional parameters, it enables the representation of complex shapes with high-quality features, addressing issues of mesh freedom and numerical stability encountered in previous methods. This innovation opens up new possibilities for mesh optimization in various applications, contributing to advancements in the fields of computer graphics, simulation, and beyond. We propose this scheme based on the method of improving the topological properties of Dual Marching Cubes, and extend it to selectively generate tetrahedrons and hierarchical adaptive grids. In the implementation of the algorithm, we demonstrated how to use the FlexiCubes algorithm to extract grids from fixed signed distance fields (SDFs) without optimization. The algorithm extracts grids from the initial FlexiCubes grid, including the initial grid topology and cut grid feature points. Then, the algorithm is used for grid optimization, and the final generated image can intuitively see how the contour surface of FlexiCube evolves during the optimization process. It can be seen that it smoothly converges to the reference grid and successfully restores all sharp features.Experiments verify the success of FlexiCubes in synthetic benchmark test and practical application, and show that our proposed method has significantly improved the mesh quality and geometric fidelity.

Keywords: Isosurface extraction; mesh model; mesh optimization; photogrammetry; three-dimensional reconstruction.

1. Introduction

Surface meshes are ubiquitous in the representation, transmission, and generation of 3D geometric graphics across various domains of computer graphics and robotics. Among many other advantages, surface meshes benefit from efficient hardware-accelerated rendering and support equation solving in physical simulations and geometric processing, thus offering a concise and precise encoding of arbitrary surfaces.

However, not all meshes are created equal—the aforementioned attributes are typically realized only on high-quality meshes. In fact, if a mesh contains too many elements, issues such as self-intersections and sliver elements may arise, or it might fail to capture the underlying geometry, rendering it entirely unsuitable for downstream tasks. Hence, generating a high-quality mesh for a specific shape is crucial, but this task is often tedious and usually requires a significant amount of manual effort.

Recently, the explosive growth in algorithmic content generation and generative 3D modeling tools has led to an increased demand for automatic mesh generation. In fact, the task of creating a high-quality mesh, traditionally the domain of skilled technical artists and modelers, is increasingly being addressed through automatic algorithm pipelines. These pipelines typically rely on differentiable mesh generation, i.e., parameterizing the space of 3D surface meshes and optimizing various objectives through gradient-based techniques. For instance, inverse rendering, structural

optimization, and generative 3D modeling all utilize this fundamental building block (Hasselgren et al., 2022; Munkberg et al., 2022). In an ideal world, such applications would perform simple gradient descent on some mesh representation to optimize their desired objectives. However, this workflow presents numerous obstacles, from the fundamental issue of optimizing meshes of different topologies to existing formulations lacking stability and robustness, resulting in irremediably low-quality meshes. In this work, we introduce a new formulation that moves closer to this goal, significantly enhancing the simplicity and quality of differentiable mesh generation for various downstream tasks.

Unless very careful initialization, local re-parametrization, and regularization are used, directly optimizing the positions of mesh vertices is prone to degeneracies and local minima. Therefore, a common paradigm is to define and optimize a scalar field or a signed distance function (SDF) in space and then extract a triangulated mesh approximating the level set of that function. The choice of scalar function representation method and mesh extraction scheme greatly impacts the performance of the overall optimization pipeline. The challenge of extracting meshes from scalar fields is that the possible space of generated meshes can be restrictive.

To address these challenges, we identified two key properties that the mesh generation process should provide to enable simple, efficient, and high-quality optimization for downstream tasks: (1) Gradients. The differentiation regarding the mesh is explicit, allowing gradient-based optimization to effectively converge in practice. (2) Flexibility. Mesh vertices can be individually and locally adjusted to accommodate surface features and find high-quality meshes with a minimal number of elements.

However, these two properties are inherently conflicting. Increased flexibility provides more capacity to represent degenerate geometries and self-intersections, but it hinders the convergence of gradient-based optimization. Thus, existing techniques often ignore one of these two properties. For example, the widely used Marching Cubes program is inflexible, with its vertices always located on a fixed grid, (Subedi et al., 2020) hence the generated mesh can never align with non-axis-aligned sharp features. Common marching techniques allow underlying mesh deformation but still do not permit individual vertex adjustment, resulting in slicing elements and imperfect matches. On the other hand, Dual Contouring is popular for its capability to capture sharp features, but its gradient-based optimization convergence is poor, and the linear systems used for vertex positioning are unstable and lead to ineffective optimization.

In this work, we employed a new technique, dubbed FlexiCubes, that satisfies both desired properties. Its principle adjusts a specific Dual Marching Cubes formulation and introduces additional degrees of freedom to flexibly position each extracted vertex within its dual cell. By carefully constraining the formulation, it still produces mostly intersect-free manifolds and watertight meshes in the vast majority of cases and achieves good differentiation (GRAD) regarding the underlying mesh (Gao et al., 2022; Lin et al., 2022).

The most crucial property of this formulation is the practical success of gradient-based mesh optimization. To assess this inherent empirical issue, a significant part of this work was dedicated to an extensive evaluation of FlexiCubes for several downstream tasks. Specifically, the formulation provided significant advantages for various mesh generation applications, including inverse rendering, optimization of physical and geometric energies, and generative 3D modeling. The resulting meshes concisely capture the desired geometry with a low element count and are easily optimized through gradient descent (Lorensen and Cline, 1987; Remelli et al., 2020; Shen et al., 2021). Furthermore, extensions of FlexiCubes were proposed, such as adaptively adjusting the mesh resolution through hierarchical refinement, and the internal tetrahedralization of domains. Benchmarks and experiments demonstrated the value of this technique compared to past methods, and we believe this will serve as a valuable tool for high-quality mesh generation across many application domains.

2. Related Work

2.1. Isosurface Extraction

Traditional isosurface processing methods extract a polygonal mesh representing the level set of a scalar function, a problem that has been extensively studied in various domains. Here, we review particularly relevant work and refer the reader to the research by De Araujo et al. According to De Araujo et al., we classify isosurface processing methods into three categories and categorize the most commonly used methods (De Araújo et al., 2015).

Spatial Decomposition. The first category of methods obtains isosurfaces through spatial decomposition, which divides space into cells such as cubes or tetrahedra and creates polygons within cells containing the surface. Marching Cubes (MC) is the most representative of these methods. As originally proposed, Marching Cubes suffers from topological ambiguities and difficulty representing sharp features (Lorensen and Cline, 1987). Subsequent work improved lookup tables assigned polygon types to cubes or divided cubes into tetrahedra and used Marching Tetrahedra-like approaches for isosurface extraction. To better capture sharp features, Dual Contouring (DC) moved to a dual representation, where each cell extracts mesh vertices and proposed estimating vertex positions based on local isosurface details. Dual Contouring was extended to adaptive mesh division and can be used to output tetrahedral meshes. Another improved method is Dual Marching Cubes (DMC), which combines the benefits of Marching Cubes and Dual Contouring. Recently, Neural Marching Cubes and Neural Dual Contouring (NDC) introduced a data-driven approach, positioning the extracted mesh as a function of the input field (Chen et al., 2022). Despite significant progress in extracting from known scalar fields, applying isosurface methods to gradient-based mesh optimization remains challenging.

The second category of methods uses surface tracking, leveraging adjacency information between surface samples to extract isosurfaces. Marching Triangles was one of the first representative methods, triangulating surfaces iteratively from initial points under Delaunay constraints (McCormick and Fisher, 2002). Subsequent work aimed to incorporate adaptivity into it or align with sharp features. However, within the framework of surface tracking, gradient-based mesh optimization needs to be differentiated through a discrete iterative update process, which is a significant undertaking.

Shrink Wrapping. The third category relies on shrinking a spherical mesh or inflating critical points to match the isosurface. By default, these methods are only applicable to a limited set of topologies and require manual selection of critical points to support arbitrary topologies. Moreover, differentiating through the shrink-wrapping process is not straightforward, making these methods less suitable for gradient-based optimization.

2.2. Machine Learning-based Mesh Optimization

With the latest advancements in machine learning (ML), some researchers have explored generating 3D meshes with neural networks, optimizing their parameters under certain loss functions through gradient-based optimization. Early methods sought to predefine the topology of generated shapes, such as spheres, primitives' combinations, or a set of segmented parts. However, their ability to generalize to objects with complex topologies is limited. To address this issue, AtlasNet represents 3D shapes as a collection of parameterized surface elements, although it does not encode a coherent surface. Mesh R-CNN first predicts a rough structure and then refines it into a surface mesh. This two-stage approach can generate meshes with different topological structures, but since the second stage still relies on mesh deformation, topological errors initiated in the first stage cannot be corrected. PolyGen automatically generates mesh vertices and edges, but they require 3D ground truth data, thus are limited. CvxNet and BSPNet attempt to use convex decomposition of shapes or binary space partitioning for spatial division but extending them to various objectives defined on meshes is not straightforward.

Recently, many works have explored differentiable mesh reconstruction schemes from an implicit function, typically encoded through convolutional networks or implicit neural representations. Deep

Marching Cubes calculates the expected topology within a cube, but its scalability deteriorates with increasing mesh resolution. MeshSDF proposed a specialized scheme for sampling gradients through mesh extraction, while Mehta et al. carefully formulated level set evolution in a neural context. DefTet predicts a deformable tetrahedral mesh to represent 3D objects. Most similar to the method presented in this paper is DMTet, which utilizes a differentiable layer of Marching Tetrahedra for mesh extraction (Chen et al., 2019; Hanocka et al., 2020; Kato et al., 2018; Wang et al., 2018).

3. Core Method

This paper describes the FlexiCubes representation method for differentiable mesh optimization. At the core of this method is a scalar function on the mesh, from which a triangulated mesh is extracted using Dual Marching Cubes. The main contribution is the introduction of three sets of additional parameters, carefully chosen to increase the flexibility of the mesh representation while maintaining robustness and ease of optimization: interpolation weights present in each mesh cell, aimed at positioning the dual vertices in space; segmentation weights for each mesh, aimed at controlling how quadrilaterals are divided into triangles; and deformation vectors for each vertex of the underlying mesh, used for spatial alignment.

These parameters are optimized alongside the scalar function through auto-differentiation, to mold the mesh towards the desired target. The paper also proposes an extension of the FlexiCubes representation to extract tetrahedral meshes of volumes and to represent hierarchical meshes at adaptive resolutions.

3.1. Dual Marching Cubes Mesh Extraction

We first extract the connectivity of Dual Marching Cubes based on the scalar function values at each grid vertex. The sign at the cube's corners determines the connectivity and adjacency relations (Figure 1). Unlike regular Marching Cubes, which extract vertices along the edges of the grid, Dual Marching Cubes extract a vertex for each original face within the cell; typically, one vertex, but up to four in certain cases (Figure 1, Case C13). Vertices extracted in adjacent cells are connected through edges, forming a dual grid composed of quadrilateral faces (Figure 2). The resulting mesh is guaranteed to be manifold, although it may contain self-intersections due to the additional degrees of freedom described below (Mehta et al., 2022; Remelli et al., 2020).

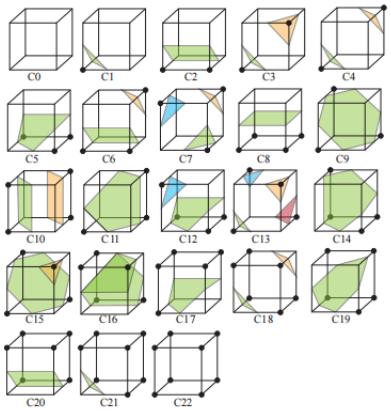

Fig. 1 Grid extraction diagram

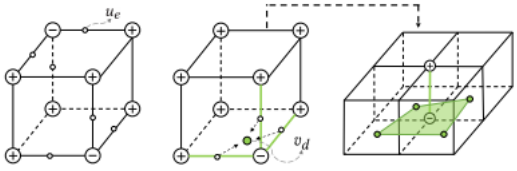

Fig. 2 Double grid

3.2. Flexible Dual Vertex Positioning

Our method extends how the extracted mesh vertex positions are calculated in the standard Dual Marching Cubes. Recalling that the original vertices in Marching Cubes are located at scalar zero crossings along the edges of mesh cells, ordinary Dual Marching Cubes defines the position of each extracted vertex as the centroid of its original face. To introduce additional flexibility into this representation, we first define a set of weights within each mesh cell, associating a positive scalar with each corner of the cube. In the implementation process, an activation function was applied to limit and no convergence problems caused by degeneracy were observed. Similarly, the algorithm does not merely position dual vertices at the centroids of original faces but introduces a set of weights within each cell, associating a positive scalar with each edge of the cube. These weights adjust the position of dual vertices within each face. In experiments, this method was used again to restrict the scope, similar to. These weights total up to 20 scalars provided for each mesh cell. In both cases, weights are defined independently for each cell, rather than shared at adjacent corners or edges; independent weights offer more flexibility, and continuity conditions of adjacent elements are not maintained in our dual setting. Note that the extracted vertex positions are necessarily within the convex hull of their mesh cell vertices. Moreover, when a convex cell emits multiple dual vertices (Figure 1), the corresponding original faces of the dual vertices are non-intersecting, eliminating almost all self-intersections in the resulting mesh.

3.3. Flexible Quadrilateral Segmentation

2.3 Flexible Quadrilateral Segmentation Meshes with non-planar pure quadrilaterals extracted by Dual Marching Cubes and FlexiCubes are typically segmented into triangles for processing in downstream applications. Simple segmentation along an arbitrary diagonal lead to significant artifacts in curved areas (Figure 3), and generally, there is no single ideal strategy to segment non-planar quads to represent unknown geometries (Ju et al., 2002). We define a weight in each mesh cell, which is propagated to the vertices emitted into the extracted mesh. Only during optimization, by inserting a midpoint vertex, is each quadrilateral mesh face split into 4 triangles (Figure 3).

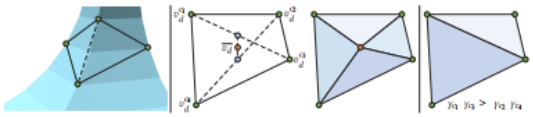

Fig. 3 Quadrilateral mesh and segmentation

This is a weighted combination of the midpoints of the two possible diagonals of the face, with the weights coming from the parameters at the corresponding vertices. Intuitively, adjusting the weights can smoothly interpolate between the two possible segmentations generated. The optimization makes the result evolve in a direction that aligns with the targeted interest. For the final extraction method upon optimization completion, the algorithm does not insert a midpoint vertex but simply segments each quadrilateral along the diagonal with the greater product of values.

3.4. Flexible Mesh Deformation

Inspired by DefTet and DMTet, this algorithm further allows the deformation of the underlying mesh vertices according to the displacement at each mesh vertex. These deformations allow the mesh to align locally with thin features and provide additional flexibility in vertex positioning. We limit deformations to half the mesh spacing to ensure that mesh cells never invert.

3.5. Tetrahedral Mesh Extraction

Many applications, such as physical simulations and character animation, require the tetrahedralization of a shape body. Therefore, the algorithm optimized FlexiCubes to additionally output a tetrahedral mesh when needed, which completely conforms to the boundaries of the extracted

surface and can extract surface-based automatic differentiation. This method adopts the strategy proposed by Liang and Zhang for Dual Contouring (Liang and Zhang, 2014). The vertex set of the tetrahedral mesh is the union of the mesh vertices, our extracted mesh vertices, and the midpoints of any cells that do not have an extracted surface vertex. We then output the tetrahedral structure as shown in Figure 4.

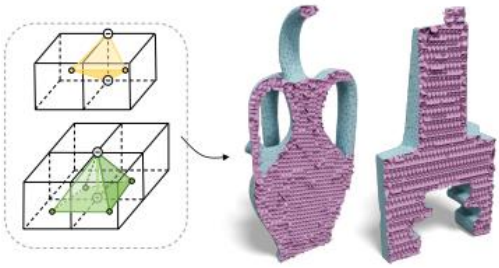

Fig. 4 Tetrahedral mesh output result

For each mesh edge connecting two vertices with the same sign, four tetrahedra are generated, each composed of two mesh vertices and two vertices from consecutive adjacent cells. For each mesh edge connecting two vertices with different signs, two quadrilateral pyramids are generated, each composed of a mesh vertex and one vertex from each adjacent cell. Subsequent steps then split the base of the pyramid, making each part produce two tetrahedra (Jon et al., 2022). When using Dual Marching Cubes connections, there is an issue that a cell might contain multiple extracted mesh vertices, so we must choose the correct vertices when outputting tetrahedra. In most cases, this selection method can be read from Figure 1.

3.6. Adaptive Mesh Resolution

The algorithm also optimized FlexiCubes to leverage adaptive hierarchical meshes, able to variably increase spatial resolution in areas of high geometric detail (Jon et al., 2021). The strategy of refining an octree mesh representation is application-specific, such as local curvature thresholds in geometric fitting or visual error in inverse rendering; our algorithm can extract hierarchical adaptive meshes while maintaining flexibility and the key properties of gradient-based optimization. The method involves locally refining the background mesh into a hierarchical octree with different resolutions. Most algorithms apply to octrees unless they encounter the problem of connecting adjacent dual vertices across different layers of the octree to form quadrilateral mesh faces. On a general octree, there might not be any double-sided connectivity producing a closed manifold mesh (Figure 5).

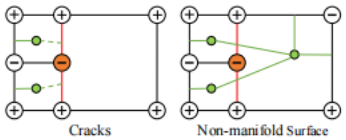

Fig. 5 Schematic diagram without any double-sided connectivity

Existing methods generally solve this problem by constraining the topology of the octree or the sign of the implicit function (Ju et al., 2002; Schaefer et al., 2007). However, these rules do not apply to algorithm optimization, where the topology is unknown and constantly changing. We adopt the method shown in Figure 6; refined octree mesh vertices adjacent to coarse cells always interpolate their values from the coarse face vertices to ensure sign consistency.

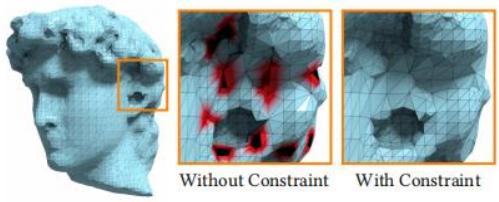

Fig. 6 The cracks and non-manifold vertices generated by FlexiCubes with octree representation are solved by applying constraints.

In the experiments cited in the literature, Sheng et al. (2023) demonstrated that this projection produced almost perfectly suitable adaptive meshes. Employing this method, the combination of all possible configurations on adjacent octree nodes at different levels, as illustrated in Figure 1, also resulted in instances where a few extracted meshes contained defects. Despite this, the method significantly improved the production of adaptively refined meshes, as depicted in Figure 7.

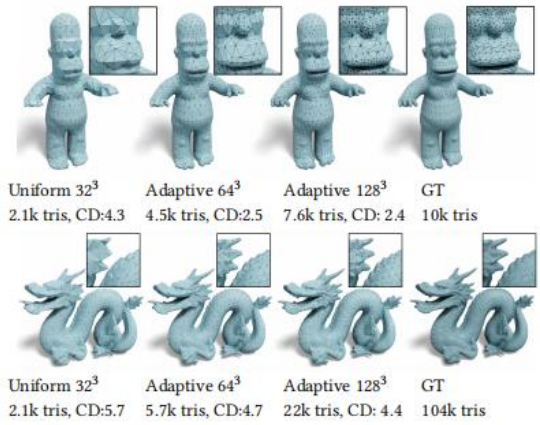

Fig. 7 Using FlexiCubes to optimize grid topology and octree structure for adaptive grids in references.

3.7. Regularization

The method used by this algorithm is a universal tool that can be optimized for specific targets and regularization, including geometric depth and SDF loss, loss from image space rendering, and mesh quality regularization (Chen et al., 2022a). The FlexiCubes representation in the algorithm is sufficiently flexible, primarily relying on the extracted mesh itself. Regularization can be directly evaluated through automatic differentiation and incorporated into gradient-based optimization. Some surface-based regularizations, such as surface area, can be easily directly expressed as functions of the underlying scalar field, while others, particularly those dependent on the mesh discretization itself, have no direct analogs. The same simple strategy does not succeed for rigid representations like Marching Cubes, as the extracted mesh cannot automatically adapt to arbitrary targets (Chen et al., 2022b).

4. Experiments

In the implementation of the algorithm, we demonstrated how to extract meshes from a fixed signed distance field (SDF) using FlexiCube without the need for optimization. In this example, the extraction scheme used is the original Dual Marching Cubes algorithm, with slight improvements in terms of splitting. Initially, we establish two functions: one to compute the SDF of the cube, and another to determine its analytical gradient. In specific applications, the SDF may be predicted by a network, with gradients calculated via finite differences or automatic differentiation, among other methods. Next, we call FlexiCubes to extract a mesh from this SDF, regardless of whether gradient information is provided. We then visualize these two meshes (Figure 8). Without gradient information

(left), the extracted vertex positions would be located at the centroids of the original (marching cubes) mesh. As a result, this method fails to reconstruct the sharp features present within the cube.

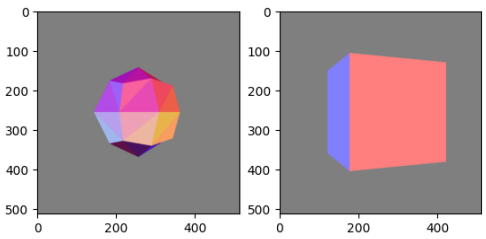

Fig. 8 Visual grid extraction

We can also use other interactive visualization tools for interactive visualization by moving around the camera and adjusting the wireframe to inspect the mesh's topological structure.

FlexiCubes is an isosurface representation method specifically designed for gradient-based mesh optimization. We iteratively optimize the 3D surface mesh by representing it as an isosurface of a scalar field. Essentially, this method allows direct evaluation of objectives on the extracted surface, enabling flexible optimization of meshes with varying topological structures.

We attempted to reconstruct the target mesh using FlexiCube with multi-view masks and depth supervision. In the discussion above, we analyzed how FlexiCube could be optimized for various applications and multiple objectives.

Then we loaded a reference mesh using the corresponding algorithm and initialized a FlexiCubes object. We will optimize its SDF, weights, and deformation variables to fit the corresponding reference mesh (Figure 9). In this experiment, we directly perform gradient descent on these parameters.

The optimization starts randomly, and then the algorithm extracts a mesh from the initial FlexiCubes mesh. The initial mesh topology and the segmented mesh features (Figure 9 left) differ significantly from the reference image we provided (Figure 9 right), which will ultimately be reconstructed to resemble the reference image.

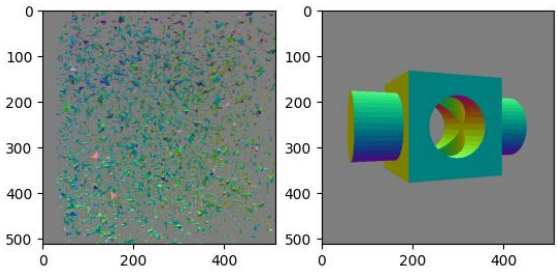

Fig. 9 The segmented target topological grid points and the original reference image

Interactive visualization using the interactive visualization tool mentioned in the references [kaolin] (Sheng et al., 2023) is also possible, by moving around the camera and adjusting the wireframe to view the mesh's topological structure.

The last thing to do before starting optimization is to set up the optimizer and the differentiable renderer. Thus, the algorithm performs the actual optimization cycle. In each iteration, the algorithm performs the following steps:

Random cameras pose samples to render the reference image and the ground-truth live image.

Extract the mesh using FlexiCube, as we did above.

Render the mesh and evaluate the reconstruction and regularization losses.

The final generated image intuitively shows how the isosurface of FlexiCube evolves during the optimization process. It can be seen smoothly converging to the reference mesh, successfully recovering all sharp features.

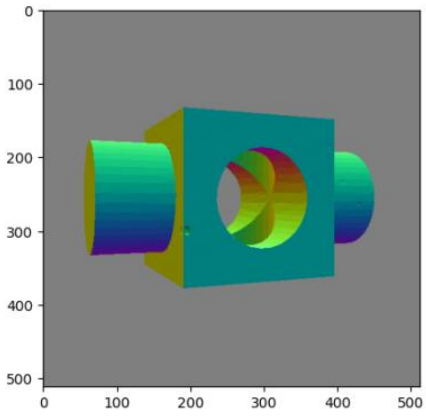

Fig. 10 Reconstruction result image

5. Applications

In this chapter, we mainly introduce some potential applications of this algorithm in real life, identified through our research.

5.1. Photogrammetry with Differentiable Rendering

DMTet, a core work within differentiable isosurface technologies, encompasses the optimization of shape, material, and image lighting. By simply replacing FlexiCubes with DMTet in the topology optimization step, allowing the rest of the pipeline to remain unchanged, improved geometric reconstruction can be achieved (Hasselgren et al., 2022; Munkberg et al., 2022). Theoretically, it could produce fewer slicing triangles and a minimal angle histogram in visualization examples. Furthermore, better triangulation makes it easier to create unique texture coordinates (UV mapping), and when running the extracted mesh through off-the-shelf unwrapping tools, an improved UV layout can be achieved (Mildenhall et al., 2020).

5.2. Mesh Simplification of Animated Objects

FlexiCube allows for the differential skinning and deformation of meshes through off-the-shelf tools while optimizing across an entire animation sequence, rather than fitting a single mesh in a reference pose (Mildenhall et al., 2020). This contrasts with neural volumetric or implicit surface representations, whose geometric deformation systems either need to be redesigned for specific neural representations, e.g., D-NeRF, or only apply skinning after mesh optimization, without end-to-end mesh optimization and optimization through gradients of deformation. Therefore, the FlexiCubes version introduced here further optimizes topology, providing a more flexible approach to simplifying animated product meshes (Wang et al., 2021).

5.3. 3D Mesh Generation

The generation of 3D meshes, typically aimed at facilitating the construction of 3D models, is an important task in computer graphics and vision and benefits industries such as gaming and social platforms. Recent 3D generative models present 3D representations as 2D images and combine them with classic generative adversarial frameworks, synthesizing 3D content using only 2D image supervision (Chan et al., 2022; Gao et al., 2022; Gu et al., 2022; Schwarz et al., 2022; Zhou et al., 2021). The latest state-of-the-art GET3D directly synthesizes high-quality textured 3D meshes through the differentiable isosurface module DMTet (Gao et al., 2022). In this application, theoretically, FlexiCubes could serve as a plug-and-play differentiable mesh extraction module in 3D generative models, significantly improving mesh quality (Chang et al., 2015).

6. Conclusion

Only high-quality meshes can be better applied to various fields of computer graphics and robotics for the representation, transmission, and generation of 3D geometric shapes. To address issues such as excessive element count leading to self-intersections and sliver elements, or the inability to capture the underlying geometric shapes, we research and attempt to implement this isosurface extraction algorithm based on FlexiCubes. Our experimental analysis indicates that introducing additional degrees of freedom in the extraction representation leads to a moderate increase in runtime and memory usage. However, in many applications, the cost of mesh extraction is usually smaller compared to the overall computation, and using a more concise mesh extraction method might ultimately reduce the total memory requirements. Experimental results show that FlexiCubes are indeed slower and more memory intensive than DMTet but offer advantages over the traditional Marching Cubes method (all these costs are small). The maximum mesh resolution of this method is not limited by the isosurface extraction but by other components of the application, such as rendering or neural network evaluation.

However, we also identify some issues with this method during our research. While this method typically produces high-quality meshes with improved element shapes in practice and the core algorithm guarantees manifoldness, it does not ensure non-self-intersecting output; although we considered differentiable mesh extraction, this method is not truly globally continuous. Mesh discontinuities can occur when the isosurface slides onto mesh vertices, inheriting characteristics from the Dual Contouring and Dual Marching Cubes methods.

For future improvements, we believe that integrating volumetric rendering with mesh-based representations to improve gradient approximations in visual tasks is a promising direction. Additionally, we note that four-dimensional spatiotemporal meshes hold significant application value in dynamic geometry representation and optimization. Lastly, we hope to integrate adaptive hierarchical mesh extraction methods into generative modeling applications. Overall, the isosurface extraction algorithm based on FlexiCubes represents a novel approach with significant implications for both research and practical applications, warranting further in-depth study.

References

[1] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. IEICE Transactions on Information and Systems, 74(1): 214–224, 1991.

[2] Andrea Bottino, Wim Nuij, and Kees Van Overveld. How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. In Proc. Implicit Surfaces, volume 96, pages 53–72, 1996.

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012, 2015.

[4] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. arXiv preprint arXiv:1909.05736, 2019.

[5] Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. A survey on implicit surface polygonization. ACM Computing Surveys (CSUR), 47(4):1–39, 2015.

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient Geometry-aware 3D Generative Adversarial Networks. In CVPR, 2022.

[7] Evgeni Chernyaev. Marching Cubes 33: Construction of topologically correct isosurfaces. Technical report, Institute for High Energy Physics, Moscow, 1995.

[8] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. ICCV 2019, 2019.

[9] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In International Conference on Learning Representations, 2022.

[10] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-Driven Automatic 3D Model Simplification. In Eurographics Symposium on Rendering, 2021.

[11] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

[12] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. GET3D: A Generative Model of High-Quality 3D Textured Shapes Learned from Images. In Advances In Neural Information Processing Systems, 2022.

[13] Jules Bloomenthal. Polygonization of implicit surfaces. Computer Aided Geometric Design, 5(4):341–355, 1988.

[14] Jules Bloomenthal. An implicit surface polygonizer. Graphics gems, 4:324–350, 1994.

[15] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. In IEEE International Conference on Computer Vision (ICCV), 2021.

[16] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2Mesh: A Self-Prior for Deformable Meshes. ACM Trans. Graph., 39(4), 2020.

[17] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. In Computer Graphics Forum, volume 20:2, pages 67–80, 2001.

[18] Sergei Azernikov and Anath Fischer. Anisotropic meshing of implicit surfaces. In International Conference on Shape Modeling and Applications 2005 (SMI'05), pages 94–103. IEEE, 2005.

[19] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. ACM Trans. Graph. 42, 4, Article 37 (August 2023), 16 pages. https://doi.org/10.1145/3592430

[21] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In Advances in Neural Information Processing Systems, 2019.

[22] Zhiqin Chen and Hao Zhang. Neural Marching Cubes. ACM Trans. Graph., 40(6), 2021.

[23] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[24] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. arXiv preprint arXiv:2208.00277, 2022a.

[25] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural Dual Contouring. ACM Trans. Graph., 41(4), 2022b.