Simplified Dense and Coarse Higher-Order Mesh Generation Using Moving Least Squares

Vijai Kumar Suriyababu*
Delft University of Technology

Cornelis Vuik†
Delft University of Technology

Matthias Möller‡
Delft University of Technology

Abstract

Higher-order surface and volume meshes are essential for better approximating PDEs. While a large body of literature focuses on PDE solvers for higher-order methods, there is still a need for robust grid generators that can produce higher-order surface/volume meshes. We propose practical workflows that provide fully automatic processes for higher-order mesh generation. We utilize the MLS algorithm, demonstrating its robustness in higher-order surface reconstruction. We also show its effectiveness in generating higher-order meshes in geometries with non-uniform triangulation, aided by intermediate support structures. Building on recent advancements in surface multigrid methods, we propose the simplest coarse higher-order mesh generation approach to date. We illustrate the proposed workflows with examples to showcase their utility and effectiveness.

1 Introduction and Related Works

Discrete surface meshes, particularly triangular meshes, are widely used as CAD representations and play a crucial role in industrial applications. These meshes are typically generated by discretizing NURBS geometries in Computer-Aided Engineering (CAE). Once created, these linear meshes can be converted into volumetric meshes for use in numerical methods such as Finite Element and Finite Volume methods. While higher-order numerical solvers have seen notable advancements, the development of higher-order mesh generation techniques has not progressed at the same pace.

Over the past two decades, higher-order CFD methods have gained considerable popularity, as highlighted by [9]. This paper focuses on techniques that use only surface meshes, without requiring access to CAD data. In many cases, CAD data may be unavailable, proprietary, or impractical to use, especially when dealing with 3D scans or reverse engineering tasks. Essential geometric information, such as feature curves and curvatures, can be derived directly from surface meshes [6]. This work focuses on such techniques and excludes methods that do not cater to engineering simulations, as covered in [5].

One of the most comprehensive workflows for generating higher-order meshes was introduced by [7], with their WALF (Weighted Average of Local Fittings) technique. It involves surface reconstruction and local polynomial fitting, combined with a weighted average approach for global fitting. This method was later refined by [4], who released a practical tool called meshCurve, one of the few free tools available for higher-order mesh generation. WALF was further extended into H-WALF, incorporating Hermite interpolation for improved mesh quality. Our work builds on this foundation.

Similar research includes the work by [3], who used B-Splines for curved mesh generation, and [11], who employed subparametric transformations to produce higher-order meshes compatible with third-party mesh generators. These techniques have been successfully applied in aerodynamic simulations. Recently, the graphics community has also shown interest in analysis-suitable higher-order meshes. For instance, [10] used Isogeometric basis functions to generate higher-order meshes, applying them in their Poly Spline FEA solver.

Additionally, [5] introduced a method for generating coarse higher-order meshes from dense linear meshes, maintaining a bijective projection from the original surface. Our approach provides a simpler and more practical alternative for coarse higher-order mesh generation. Another recent work by [1] focuses on higher-order multi-block meshes specifically for tubular structures.

2 Series of Local Fittings

In this study, we suggest the potential for generalizing methodologies akin to Weighted Average of Local Fittings (WALF) [4] to a broader spectrum of local surface fitting techniques. We introduce a workflow that

*v.k.suriyababu@tudelft.nl, vijaikumar.in
†c.vuik@tudelft.nl
‡m.moller@tudelft.nl

employs the Moving Least Squares (MLS) method for generating higher-order meshes. Although the authors of the WALF method have outlined specific drawbacks of the MLS approach in comparison to WALF, we argue that these limitations can be effectively addressed and integrated within the MLS framework. This is particularly feasible given the diverse array of MLS variants that exist (as discussed in [2]), many of which have not been thoroughly investigated for higher-order mesh generation. We utilize the simplest MLS techniques along with point cloud support to perform higher-order surface reconstruction and extend this to produce coarse higher-order meshes in a more general way. We rely on the surface multigrid method proposed by [8] for coarse meshes, where they show a provable way to achieve a bijective mapping between coarse and fine meshes. This allows us to extend the proposed MLS-based surface fitting approach to produce coarse higher-order meshes.

3 Higher-Order Mesh Generation Using the SOLF Method

Higher-order mesh generation using the Series of Local Fittings (SOLF) method involves the workflow outlined below. Each step of the algorithm is explained in the subsequent subsections.

Algorithm 3.1. Higher Order Mesh Generation Workflow

Require: Input surface mesh
1: Calculate a feature distance field
2: Analyze surface density for uniformity
3: Build an intermediate support structure (random point sampling)
4: **if** mesh density is globally uniform **then**
5: Proceed with mesh stencils alone
6: **else**
7: Include points from intermediate support structures (for local fittings)
8: **end if**
9: Estimate local fittings using dynamic stencils (based on the feature distance field)
10: Determine initial positions for higher-order points
11: Apply a weighted Moving Least Squares (MLS) method to refine the higher-order point positions

This technique is similar to the surface reconstruction approach described in [4], but it incorporates intermediate support structures and feature distance fields. Using feature distance fields, instead of heuristic-based stencil selection, eliminates the need for special handling of specific geometries.

3.1 Feature Distance Field In existing literature, complex heuristic-based techniques [6] were used. We found that it is sufficient to estimate a distance field from feature curves onto the mesh and use it to select stencils. The presence of points from the intermediate support structure ensures that there are always enough points for building the local surface.

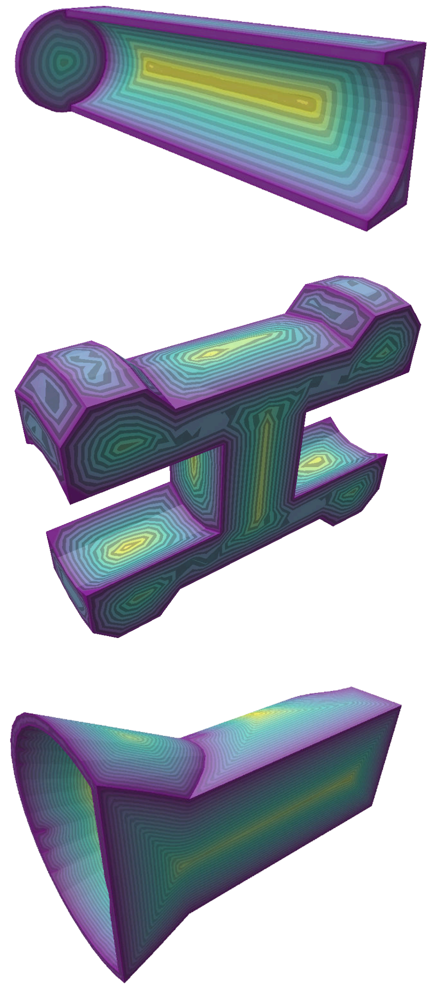

Figure 1: Feature distance fields for selected geometries. We use a viridis color map here. Purple indicates a distance field of zero.

3.2 Uniformity Check (or Remeshing) This check is straightforward and involves a two-step process. First, we evaluate the edge lengths in the mesh, followed by the surface areas of individual triangles.

$$e_{\text{len}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{3.1}$$

$$A = \frac{1}{2} \left\| \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \times \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} \right\| \tag{3.2}$$

Having a reasonably uniform mesh is advisable, as it significantly aids in the generation of coarse higher-order meshes. One may opt to use a surface remeshing tool or proceed with the intermediate support structure outlined in the subsequent sections.

3.3 Intermediate Support Structures This is one of the key differentiators between existing techniques and ours. Rather than relying solely on vertices from the input mesh, we build an intermediate support structure. If required, we can combine points from the intermediate support structure for local polynomial fitting. In our experiments, we found that even random point sampling [12] works well enough. This, combined with feature distance fields, allows us to dynamically handle non-uniform meshes.

3.4 Reconstruction with MLS Given a point x_i and its neighboring points chosen from the stencil or a combination of the stencil along with points from the support structure, the MLS local approximation at a given point x_i is achieved by fitting a polynomial $P(x;\beta)$ to the values y_j at neighboring points x_j.

This is formalized as minimizing the weighted sum of squared differences:

$$\min_{\beta} \sum_{j=1}^{n} w(x_i, x_j)(y_j - P(x_j;\beta))^2 \tag{3.3a}$$

$$w(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{3.3b}$$

Here, $w(x_i, x_j)$ is the weight function that decreases with the Euclidean distance between x_i and x_j, emphasizing closer points more significantly in the fit, and σ is a scale parameter. This weighting scheme ensures that the fitting function P smoothly adapts to local variations in the mesh.

One of the key drawbacks stated in the WALF paper is that MLS relies on Euclidean measures, while theirs relies on stencils. However, MLS can use any measure to find neighbors. Indeed, we use this combination of stencil along with Euclidean measures (and an additional filtering mechanism that discards points from triangles not in the same local planar neighborhood). This turns a traditional MLS-based workflow into a useful surface fitting approach.

3.5 Reconstruction with MLS Algorithm The MLS-based surface reconstruction method, integrated with local stencil-based fittings and intermediate support structures, is outlined below. This method addresses the limitations of traditional MLS workflows by leveraging both stencils and Euclidean distance measures to find neighbors, ensuring effective handling of non-uniform meshes.

ALGORITHM 3.2. Reconstruction with MLS

Require: Input surface mesh, feature distance field, intermediate support structure (optional), LGL points

for each mesh node x_i **do**
- Identify neighboring points for x_i using a combination of stencil and Euclidean measure
- Construct a local polynomial fitting $P(x_j;\beta)$ using neighboring points x_j
- Minimize the weighted least squares error:

$$\min_{\beta} \sum_{j=1}^{n} w(x_i, x_j)(y_j - P(x_j;\beta))^2 \tag{3.4}$$

- Compute the weight function $w(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$, where σ is a scale parameter
- Update higher-order point positions using the local polynomial fit $P(x_i)$

end for

Return the reconstructed surface with higher-order points

Once the local fits are established, the new positions for the higher-order nodes can be estimated by weighting these fits within their local neighborhood. We generate the higher-order points using Lebesgue-Gauss-Lobatto points, as outlined in [4]. Alternatively, any generic technique for generating higher-order points can be used, depending on the application. Various strategies are discussed in the numerical experiments section.

4 Coarse Mesh Generation

For coarse meshes, we rely on the multigrid surface generation method proposed by [8]. Originally developed for solving surface PDEs efficiently, it also provides a way to decimate a dense mesh and create a bijective map between points in the coarse and dense meshes. By combining intermediate support structures with the coarse-to-fine mapping provided by the multigrid surface technique, we can generate a coarse higher-order mesh without using complex variational methods.

4.1 Coarse Mesh Generation Using the Multigrid Framework

ALGORITHM 4.1. **Require:** A dense linear surface mesh

Ensure: Positions of higher-order points on a coarse SOLF surface

```
Decimate the mesh.
Generate a coarse-to-fine mesh mapping using the multigrid framework.
for each point in the coarse mesh do
    Estimate a SOLF surface for the point, using the corresponding mapped points on the dense mesh to preserve detail.
    Find nearby points from the intermediate support structure and use them to generate the local MLS surface as discussed earlier.
end for
Estimate the positions of the higher-order points based on the coarse SOLF surface using the MLS reconstruction algorithm.
```

The preservation of feature curves in mechanical parts depends on the decimation strategy and the number of decimation cycles. While the coarse-to-fine mapping allows us to retrieve points from the input mesh, control over the level of decimation is necessary. Therefore, we recommend using these techniques primarily on smooth surfaces. However, as demonstrated in the numerical experiments, with careful parameter selection, it is possible to produce coarse higher-order meshes that are suitable for analysis.

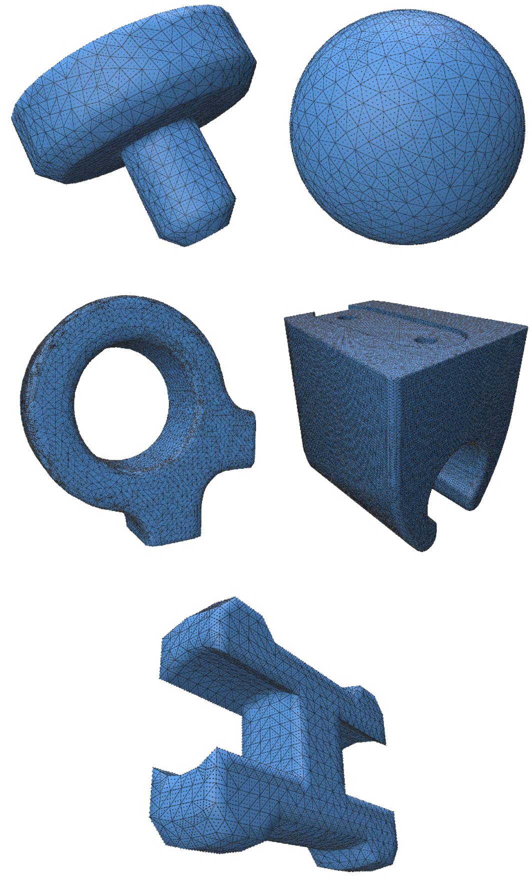

Figure 2: Dense higher-order meshes (all meshes are generated with $P = 4$) with an L_2-norm deviation within 3% of the original mesh.

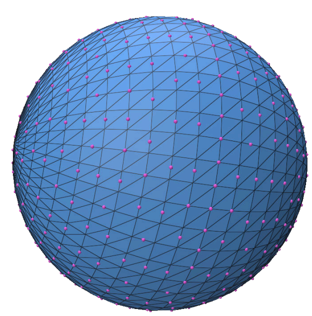

Figure 3: Coarse points mapped onto the dense mesh.

5 Numerical Experiments

This section presents various numerical experiments to demonstrate the algorithms outlined in previous sections. The code for the entire paper was implemented using a combination of C++ and Python, and benchmarks were conducted on a laptop with an 8-core Intel i5-8350U CPU and 64 GB of RAM. For dense meshes, the workflow described in the first algorithm was applied directly. We demonstrate the reconstruction algorithm on a selection of geometries from the Thingi10k [13] dataset. All experiments were reconstructed with $P = 4$.

As outlined in Section 4.1, we use the surface multigrid technique. Here, we show the points of the sphere geometry, decimated and mapped onto the original mesh. These points were obtained using the coarse-to-fine mapping provided by the surface multigrid technique.

6 Conclusion

We have developed and demonstrated practical workflows for generating higher-order meshes. Our examples highlight the practicality and efficiency of these methods. We emphasize the usefulness of feature distance fields in estimating a local neighborhood that

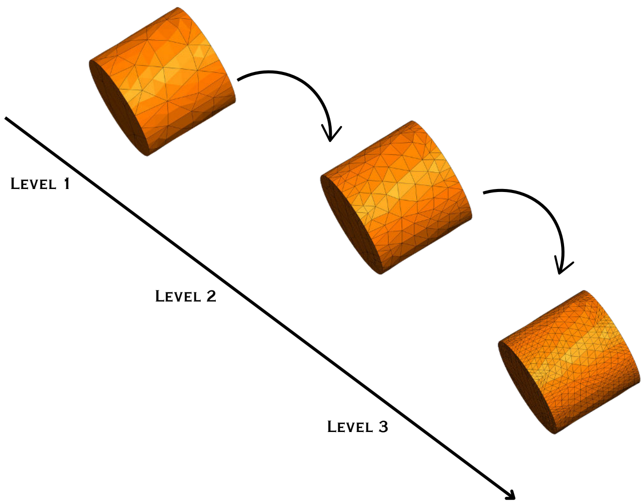

Figure 4: Coarse higher-order mesh of a cylinder visualized with refined elements to showcase the smoothness and reconstruction of the mesh.

preserves the features of the input surface. Additionally, we demonstrate how combining surface multigrid techniques with sensible decimation strategies can effectively produce coarse, higher-order meshes.

It is essential to start with the best possible level of detail, allowing the grid generator to handle the coarsening process while preserving critical information. However, we have not generalized our algorithms to work for every possible geometry; they require fine-tuning for each new case. Nevertheless, this workflow can be further improved to become more robust for real-world applications.

References

[1] D. Bošnjak, A. Pepe, R. Schussnig, D. Schmalstieg, and T.-P. Fries, *Higher-order block-structured hex meshing of tubular structures*, Engineering with Computers, 40 (2024), pp. 931–951.

[2] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, and S.-Y. Jin, *A survey of methods for moving least squares surfaces*, in Proceedings of the Fifth Eurographics / IEEE VGTC Conference on Point-Based Graphics, SPBG'08, Goslar, DEU, 2008, Eurographics Association, p. 9–23.

[3] F. Hindenlang, T. Bolemann, and C.-D. Munz, *Mesh curving techniques for high order discontinuous galerkin simulations*, in IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach, Springer, 2015, pp. 133–152.

[4] J. Ims and Z. J. Wang, *Automated low-order to high-order mesh conversion*, Engineering with Computers, 35 (2019), pp. 323–335.

[5] Z. Jiang, Z. Zhang, Y. Hu, T. Schneider, D. Zorin, and D. Panozzo, *Bijective and coarse high-order tetrahedral meshes*, ACM Trans. Graph., 40 (2021).

[6] X. Jiao and N. R. Bayyana, *Identification of c1 and c2 discontinuities for surface meshes in cad*, Computer-Aided Design, 40 (2008), pp. 160–175.

[7] X. Jiao and D. Wang, *Reconstructing high-order surfaces for meshing*, Engineering with Computers, 28 (2012), pp. 361–373.

[8] H.-T. D. Liu, J. E. Zhang, M. Ben-Chen, and A. Jacobson, *Surface multigrid via intrinsic prolongation*, ACM Trans. Graph., 40 (2021).

[9] R. Löhner, *High-Order Methods for Simulations in Engineering*, Springer International Publishing, Cham, 2021, pp. 277–307.

[10] T. Schneider, D. Panozzo, and X. Zhou, *Isogeometric high order mesh generation*, Computer Methods in Applied Mechanics and Engineering, 386 (2021), p. 114104.

[11] T. V. Smitha and K. V. Nagaraja, *Automated 3D higher-order tetrahedral mesh generator utilizing the subparametric transformations in MATLAB*, AIP Conference Proceedings, 2316 (2021), p. 040003.

[12] G. Turk, *Generating Random Points in Triangles*, Academic Press Professional, Inc., USA, 1990, p. 24–28.

[13] Q. Zhou and A. Jacobson, *Thingi10k: A dataset of 10, 000 3d-printing models*, ArXiv, abs/1605.04797 (2016).