Mesh-driven resampling and regularization for robust point cloud-based flow analysis directly on scanned objects

Monu Jaiswal[a], Ashton M. Corpuz[a], Ming-Chen Hsu[a,*]

[a]*Department of Mechanical Engineering, Iowa State University, 2043 Black Engineering, Ames, Iowa 50011, USA*

Abstract

Point cloud representations of three-dimensional objects have remained indispensable across a diverse array of applications, given their ability to represent complex real-world geometry with just a set of points. The high fidelity and versatility of point clouds have been utilized in directly performing numerical analysis for engineering applications, bypassing the labor-intensive and time-consuming tasks of creating analysis-suitable CAD models. However, point clouds exhibit various levels of quality, often containing defects such as holes, noise, and sparse regions, leading to sub-optimal geometry representation that can impact the stability and accuracy of any analysis study. This paper aims to overcome such challenges by proposing a novel method that expands upon our recently developed direct point cloud-to-CFD approach based on immersogeometric analysis. The proposed method features a mesh-driven resampling technique to fill any unintended gaps and regularize the point cloud, making it suitable for CFD analysis. Additionally, ghost penalty stabilization is employed for incompressible flow to improve the conditioning and stability compromised by the small cut elements in immersed methods. The developed method is validated against standard benchmark geometries and real-world point clouds obtained in-house with photogrammetry. Results demonstrate the proposed framework's robustness in facilitating CFD simulations directly on point clouds of varying quality, underscoring its potential for practical applications in analyzing real-world structures.

Keywords: Point cloud, Scanned object, Geometric algorithm, CFD, Immersed method, Ghost penalty

1. Introduction

In the field of geometric modeling and computer vision, the representation of three-dimensional objects remains pivotal for various applications, such as animation, virtual reality, medical imaging, industrial design, and numerical analysis [1–5]. Among different methods, point cloud repre-

*Corresponding authors
Email address: jmchsu@iastate.edu (Ming-Chen Hsu)

The final publication is available at https://doi.org/10.1016/j.cma.2024.117426 *October 17, 2024*

sentation of geometry stands out as a powerful and versatile approach. Point cloud data consists of unstructured points distributed in Euclidean space, each point representing the specific location of the geometry. These points are typically obtained through various methods, including LiDAR [6, 7], structural light techniques [8], or photogrammetry [9–11], sometimes carried with additional geometric information, such as approximated surface normals and color. The advantage of using a point cloud lies in its fidelity to real-world objects and scenes. Conventional geometry representations, such as computer-aided design (CAD), polygon mesh, boundary representation (B-rep), non-uniform rational B-splines (NURBS), and implicit surfaces, often rely on mathematical forms and parametrization [12], which brings in the challenge of accurately representing real-world geometries, such as landscapes, terrains, in-use building structures, organic objects, and environment. Point cloud, on the other hand, excels at such geometry representation [13–17] by exploiting non-manifoldness and non-watertightness, which offers flexibility in representing complex surfaces and shapes, including imperfections. Additional benefits of point clouds are shown in engineering design and manufacturing applications, where the geometry of a part often deviates from its original CAD representation due to design modification or alterations made during the manufacturing process. Similarly, in the context of in-use structures such as historic building structures or landscapes, a readily available CAD may be entirely absent. In these cases, the point cloud emerges as an invaluable tool, providing a first-hand geometric description that is essential for analysis, documentation, and further processing. Moreover, due to recent advancements in scanning procedures [18–21], real-world geometries can be rapidly captured in high detail without manual and laborious tasks of processing a CAD model. Point clouds are an exciting field of modeling that opens the door for new applications that were previously impossible with better geometric representation of the objects of interest.

The flexibility and practicality of point clouds have inspired new methods for engineering analysis, shifting from traditional reliance on CAD models to leveraging point clouds for studying real-world geometry. While analysis using scanned geometries remains a relatively new and unexplored field, it has been gaining popularity with specific applications to structural analysis, additive manufacturing, and flow simulations [22–24], previously deemed difficult with conventional methods. Qian and Lu [25] utilized the discrete gradient method (DGM) [26] to directly perform stress analysis in biological systems by computing geometric and connectivity information from point clouds using Delaunay tessellation [27]. A semi-automatic framework was developed by Rolin et al. [28] to extract an hBIM[1]-oriented model from a laser-scanned point cloud to conduct structural analysis on historic buildings. CFD analysis for urban landscape study was performed by Urech et al. [22] by constructing mesh and voxel grids from laser-scanned point clouds. Bouchiba et al. [29]

[1]hBIM stands for historic building information modeling

extended implicit surface definition obtained from point cloud to perform immersed CFD analysis on real-world complex geometries. Despite facilitating analysis of real-world geometry, these studies typically involve further reconstruction of point clouds into tessellated meshes, voxel grids, or implicit surfaces, introducing additional time-consuming efforts and geometric manipulations for analysis purposes.

Recent efforts in analysis have focused on developing frameworks based on the immersed approach, such as the finite cell method (FCM) [30–35] and immersogeometric analysis (IMGA) [36–46], to perform studies directly on the point cloud without any further modification. Unlike the traditional boundary-fitted method, the immersed method does not require a geometry-conforming mesh to perform analysis. Instead, the geometry is immersed into a non-conforming background mesh, and the analysis is performed on the background mesh by incorporating information from the geometry. Kudela et al. [47] provided a framework based on FCM to perform structural analysis on solids represented by oriented point clouds using a regular grid for the background mesh. This approach seamlessly connects scanned geometry with numerical analysis without additional reconstructions. Hartmann and Kollmannsberger [48] enhanced the work by developing a sharp interface approach to enforce essential boundary conditions. In the field of CFD, Balu et al. [49] introduced a direct point cloud-to-CFD framework based on IMGA. The method computes essential geometric information from the point cloud to enforce necessary boundary conditions on the tetrahedral background mesh. The accuracy of the IMGA approach for point clouds has been demonstrated to be comparable to that of boundary-fitted solutions with similar levels of degrees of freedom. Furthermore, Wang et al. [50] extended the method to simulate flow over scanned geometries of in-use structures, providing time-saving and accurate results for civil applications. However, the stability and accuracy of the immersed method can be impacted by poorly scanned geometry with significant defects.

Despite advancements in scanning methods, point cloud datasets can still suffer from geometric defects. These issues often include noise, incompleteness, and irregular sampling [51]. Noise and artifacts may arise from sensor inaccuracies, environmental factors such as reflections or shadows, or surface properties like transparency. While robust methods exist to mitigate noise [52], incompleteness and irregular sampling continue to pose significant challenges. Incomplete coverage of scanned objects due to occlusions, limited field of view, or scanning limitations can result in holes or sparse regions in point clouds. These incomplete regions can greatly hinder the effectiveness of the analysis methods, rendering the geometry unsuitable for analysis purposes. Moreover, inaccessible or unexplored regions during scanning can also lead to incomplete data. Many applications, such as surveying [21, 22], often focus on capturing the large-scale features in the geometry, while small features may be overlooked, leading to an irregularly sampled and incomplete point cloud. However, despite these challenges, obtaining meaningful results around areas of interest could be

of importance.

Besides completeness, a critical requirement of the immersed method is that intersected background elements must contain a portion of the object boundary to enforce essential conditions effectively [37, 49]. This requirement applies not only to point cloud geometries but also to other types of geometry representations, such as B-rep, NURBS, and polygon meshes. In the case of a point cloud-represented boundary, it was suggested that the intersected element should contain at least one point to satisfy this requirement [49]. With insufficient point density, the background mesh may fail to recognize the immersed boundary, resulting in flow leakage through the object surface in the case of CFD. Unlike boundary-fitted methods that use CAD, where the watertight boundary is refined alongside the volume mesh, no comparable method exists for immersed methods to refine the point cloud representation to match the background mesh. Generating a background mesh that ensures the intersected elements contain a point inside is also non-trivial; to the best of our knowledge, no such method currently exists. Refinement zones (box or sphere) are constructed around the point cloud to create a background mesh with finer elements near the boundary without necessarily containing any point sets. While various machine learning-based approaches [53] and geometric algorithms [51] promise to fill holes and incomplete regions in point clouds, they do not ensure that the point cloud density and distribution are suitable for the given background mesh, making them ineffective for analysis purposes.

In this work, we introduce an innovative mesh-driven resampling method to address the challenges associated with analyzing point clouds of varying quality. The goal of the proposed method is to augment the point cloud density such that each intersected element in the background mesh contains exactly one point, addressing the critical requirement of the immersed method that was not explored before. Through a unique combination of a winding number field [54, 55] and local surface reconstruction, our approach accurately identifies the intersected regions and deploys resampled points based on the local neighborhood information with respect to the background mesh. As a result, the resampling method regenerates a point cloud based on the mesh suitable for analysis while filling the gaps in incomplete and sparse regions. Unlike global mesh reconstruction methods, our approach preserves local features around the high-quality regions of the point cloud while approximating points in incomplete regions. We integrate this mesh-driven resampling method with our previous works of direct point cloud-to-CFD [49] and photogrammetry-based approach [50] to create an expanded point cloud flow analysis framework named NIMBUS: a Novel IMmersed BoUndary method for Scanned geometries. Additionally, NIMBUS incorporates ghost penalty stabilization [56–58] for incompressible flow to improve the conditioning and stability of the immersed method impacted by small cut element issues. With the proposed features, we show that the framework handles complex and incomplete real-world geometries and conducts analysis with robustness and accuracy.

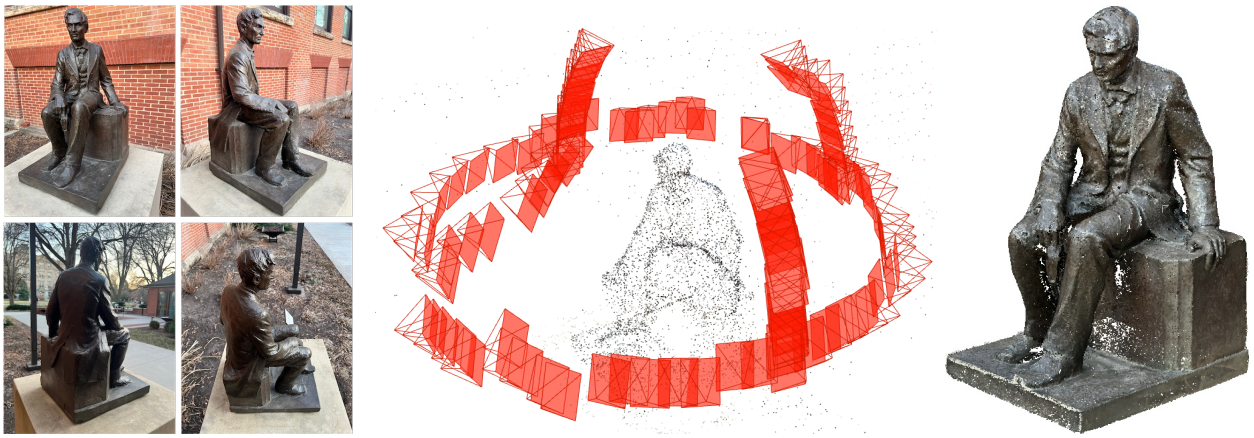
This paper is organized as follows. We provide the details of the NIMBUS framework in Section 2, including photogrammetry reconstruction, IMGA formulation for incompressible flow with ghost penalty stabilization, point cloud processing to obtain geometric quantities, and mesh-driven resampling method. Section 3 provides the details and results of the validation study of the resampling method and incompressible flow. The application of the method in simulating the complex Stanford dragon and a real-world incomplete scan of a statue is discussed in Section 4. In Section 5, we draw conclusions from our work and provide a few directions for future research.

2. The NIMBUS framework

The proposed framework integrates multidisciplinary concepts from photogrammetry-based point cloud reconstruction, immersogeometric analysis for incompressible flow, point cloud processing, and mesh-driven resampling. In the subsequent subsections, we delve into the specifics of each component, discussing their significance and detailing their implementation within NIMBUS. Note that the framework is also applicable to point clouds generated through other methods, such as LiDAR and structured lighting.

2.1. Point cloud reconstruction using photogrammetry

Photogrammetry is a 3D reconstruction process of generating point clouds by using photographic images around objects. In contrast to LiDAR-based approaches, which rely on emitting lights or lasers to measure depth and calculate object positions by considering light reflections, photogrammetry involves the process of extracting precise geometric information about objects and their environments from images. The technique relies on principles of perspective and triangulation, leveraging the overlapping imagery captured from different viewpoints to reconstruct the spatial geometry of the scene. Photogrammetry offers a cost-effective and accessible option, as it does not require expensive depth sensors; a set of images is sufficient for reconstruction. A typical workflow for photogrammetry comprises image collection, structure from motion (SfM) [9], multi-view stereo (MVS) [10], and depth fusion to generate the dense point cloud, as shown in Figure 1. Initially, a set of images is acquired around the object of interest. This can be accomplished using a camera or smartphone for smaller objects and a drone for surveying larger structures. It is crucial that the images are of high quality and captured under similar illumination conditions, ensuring clear visibility of the object without any blurring. Additionally, the images should be taken from different viewpoints such that the subsequent images overlap each other. Next, SfM is used to align the collected images by extracting distinct features and aligning the matching features among the images to estimate the camera poses for each image. Additionally, SfM generates a sparse point cloud representing the extracted features in the spatial scene. MVS then utilizes the output from SfM to calculate depth and normal information for each pixel in an image. The final dense point

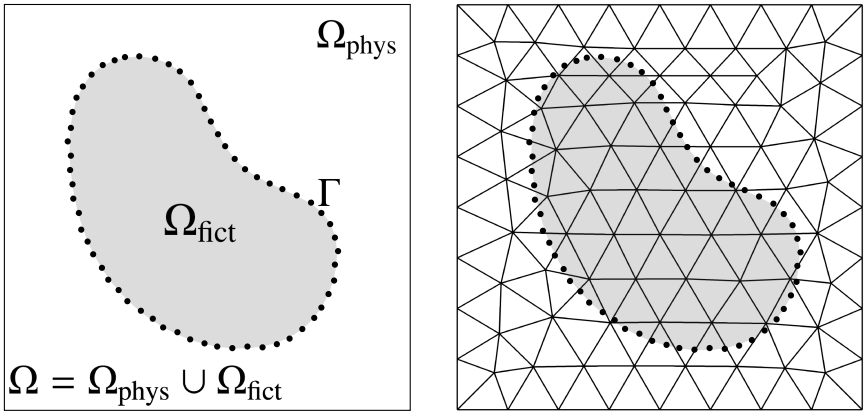


(a) (b) (c)

Figure 1: Photogrammetry workflow: (a) image collection, (b) camera pose estimation and sparse point cloud generated from SfM, and (c) final dense point cloud reconstructed with MVS.

cloud is generated by the fusion of the depth and normal maps from multiple images, as shown in Figure 1c. SfM and MVS are widely recognized techniques in photogrammetry; for further details, interested readers can refer to Refs. [9, 10, 59–61].

One of the primary challenges encountered in MVS is when scanning objects that are composed of weak textures, such as solid colors, transparency, and reflective surfaces. Without identifiable features or key points in the images, traditional MVS struggles to calculate depth maps accurately, leading to a point cloud with holes, incompleteness, and irregularly sampled regions. However, recent advancements in machine learning techniques within computer vision have shown promising capabilities in addressing these challenges. Machine learning-based MVS approaches, as discussed in Refs. [19, 20, 62–69], typically leverage 2D or 3D convolutional networks, or a combination of both, to directly infer depth maps for each image. Once the depth maps are obtained, the standard process of depth map fusion across the image set is performed to generate the dense point cloud. Our previous work [50] employed an MVSNet framework based on the work of Yao et al. [19] and Gu et al. [20] to generate point clouds for conducting photogrammetry-based CFD analysis. Despite its effectiveness in generating complete point clouds, MVSNet produced reconstructions with issues such as multiple registrations and noise, particularly around surfaces with weak textures. In this work, we show that the proposed mesh-driven resampling method solves the issue of multiple registrations obtained with MVSNet. More importantly, our approach effectively resolves the incompleteness and sparsity observed around weak textures in MVS-generated point clouds, resulting in a high-quality point cloud suitable for analysis. For performing SfM and MVS, we utilize COLMAP's existing libraries for the photogrammetry pipeline [9, 10].



(a) (b)

Figure 2: (a) An example of flow over an object represented by a point cloud. The object with boundary Γ is immersed into the domain Ω. The immersed boundary Γ separates the domain Ω into a physical part Ω_{phys} and a fictitious part Ω_{fict}. (b) Background mesh discretizing the domain Ω.

2.2. Point cloud CFD using immersogeometric analysis

This section presents the immersogeometric formulation for incompressible flow directly on point cloud geometries. The method consists of the following key components: (1) variational multiscale (VMS) formulation for incompressible flow using stabilized finite element method [70], (2) Nitsche-based formulation for imposing Dirichlet boundary conditions on the immersed surface [36], (3) subdivision-based adaptive quadrature [71] to capture the immersed boundary more accurately, and (4) ghost penalty stabilization [56] for small cut elements to improve stability and accuracy.

2.2.1. Immersogeometric formulation for incompressible flows

In immersogeometric flow analysis, the computational domain Ω (a subset of $\mathbb{R}^d$, where $d \in 2, 3$) comprises two distinct regions: the physical domain Ω_{phys}, denoting the fluid domain, and the fictitious domain Ω_{fict}, enclosed by the immersed object. Let $\mathcal{P}$ be a point cloud representing the immersed object. The immersed boundary Γ serves as the interface between Ω_{phys} and Ω_{fict}, as illustrated in Figure 2a. The computational domain is discretized into a collection of disjoint elements Ω^e, where the closures of these elements cover Ω, as depicted in Figure 2b. The boundary Γ is discretized into a collection of boundary elements Γ^b, whose coverage is defined by each point in $\mathcal{P}$. We define $\Omega^e_{\text{phys}} = \Omega^e \cap \Omega_{\text{phys}}$ and $\Omega^e_{\text{fict}} = \Omega^e \cap \Omega_{\text{fict}}$, representing the portions of Ω^e belonging to Ω_{phys} and Ω_{fict}, respectively. Let $\mathcal{S}^h_u$ and $\mathcal{S}^h_p$ be the discrete trial function spaces for the fluid velocity and pressure, respectively, and $\mathcal{V}^h_u$ and $\mathcal{V}^h_p$ be the corresponding test function spaces. The VMS discretization of the incompressible Navier–Stokes equations is formulated as follows: Find

velocity $\mathbf{u}^h \in \mathcal{S}_u^h$ and pressure $p^h \in \mathcal{S}_p^h$ such that for all test functions $\mathbf{w}^h \in \mathcal{V}_u^h$ and $q^h \in \mathcal{V}_p^h$,

$$B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) + B^{\text{WBC}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = 0\,, \tag{1}$$

where the bilinear form B^{VMS} and the load vector F^{VMS} represent the terms associated with VMS, and B^{WBC} includes the terms imposing the Dirichlet boundary conditions weakly. In this work, B^{VMS} and F^{VMS} are given as

$$\begin{aligned}
B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) &= \int_{\Omega_{\text{phys}}} \mathbf{w}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega + \int_{\Omega_{\text{phys}}} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(\mathbf{u}^h, p^h) d\Omega \\
&+ \int_{\Omega_{\text{phys}}} q^h \boldsymbol{\nabla}\cdot\mathbf{u}^h d\Omega - \sum_e \int_{\Omega^e_{\text{phys}}} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{w}^h + \frac{\boldsymbol{\nabla} q^h}{\rho}\right) \cdot \mathbf{u}' d\Omega - \sum_e \int_{\Omega^e_{\text{phys}}} p' \boldsymbol{\nabla}\cdot\mathbf{w}^h d\Omega \\
&+ \sum_e \int_{\Omega^e_{\text{phys}}} \mathbf{w}^h \cdot \left(\mathbf{u}' \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega - \sum_e \int_{\Omega^e_{\text{phys}}} \frac{\boldsymbol{\nabla}\mathbf{w}^h}{\rho} : \left(\mathbf{u}' \otimes \mathbf{u}'\right) d\Omega \\
&+ \sum_e \int_{\Omega^e_{\text{phys}}} \left(\mathbf{u}' \cdot \boldsymbol{\nabla}\mathbf{w}^h\right) \bar{\tau} \cdot \left(\mathbf{u}' \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega\,,
\end{aligned} \tag{2}$$

and

$$F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = \int_{\Omega_{\text{phys}}} \mathbf{w}^h \cdot \rho\mathbf{f}\, d\Omega + \int_{\Gamma^{\text{N}}} \mathbf{w}^h \cdot \mathbf{h}\, d\Gamma\,. \tag{3}$$

In the above, ρ is the fluid density, $\partial(\cdot)/\partial t$ is the partial time derivative, $\mathbf{f}$ is the body force per unit mass, $\mathbf{h}$ is the traction vector applied at the Neumann boundary Γ^{N}, and $\boldsymbol{\sigma}(\mathbf{u}, p) = -p\,\mathbf{I} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{u})$ and $\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$ are the Cauchy stress and strain-rate tensors, respectively, where $\mathbf{I}$ is the identity tensor and μ is the dynamic viscosity. $\mathbf{u}'$ and p' are the fine-scale terms associated with the VMS formulation and are given by

$$\mathbf{u}' = -\tau_{\text{M}}\left(\rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla\mathbf{u}^h - \mathbf{f}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\right)\,, \tag{4}$$

$$p' = -\rho\tau_{\text{C}}\boldsymbol{\nabla}\cdot\mathbf{u}^h\,. \tag{5}$$

$\bar{\tau}$, τ_{M}, and τ_{C} are the stabilization parameters, and their detailed expression can be found in Refs. [49, 50]. Other options for the stabilization parameters can be found in Refs. [72–76].

In the immersogeometric approach, the Dirichlet boundary conditions are implemented using Nitsche-based formulation [77–79]. Unlike traditional boundary-fitted methods, which enforce the Dirichlet conditions strongly, Nitsche's formulation imposes them weakly on the boundary

elements Γ^b. The bilinear form B^{WBC} in Eq. (1) can be expressed as

$$
\begin{aligned}
B^{\mathrm{WBC}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) = & -\sum_b \int_{\Gamma^b \bigcap \Gamma^{\mathrm{D}}} \mathbf{w}^h \cdot \left(-p^h \mathbf{n} + 2\mu\, \boldsymbol{\varepsilon}(\mathbf{u}^h)\, \mathbf{n}\right) d\Gamma \\
& - \sum_b \int_{\Gamma^b \bigcap \Gamma^{\mathrm{D}}} \left(2\mu\, \boldsymbol{\varepsilon}(\mathbf{u}^h)\, \mathbf{n} + q^h \mathbf{n}\right) \cdot \left(\mathbf{u}^h - \mathbf{g}\right) d\Gamma - \sum_b \int_{\Gamma^b \bigcap \Gamma^{\mathrm{D},-}} \mathbf{w}^h \cdot \rho \left(\mathbf{u}^h \cdot \mathbf{n}\right) \left(\mathbf{u}^h - \mathbf{g}\right) d\Gamma \\
& + \sum_b \int_{\Gamma^b \bigcap \Gamma^{\mathrm{D}}} \tau^B_{\mathrm{TAN}} \left(\mathbf{w}^h - \left(\mathbf{w}^h \cdot \mathbf{n}\right) \mathbf{n}\right) \cdot \left(\left(\mathbf{u}^h - \mathbf{g}\right) - \left(\left(\mathbf{u}^h - \mathbf{g}\right) \cdot \mathbf{n}\right) \mathbf{n}\right) d\Gamma \\
& + \sum_b \int_{\Gamma^b \bigcap \Gamma^{\mathrm{D}}} \tau^B_{\mathrm{NOR}} \left(\mathbf{w}^h \cdot \mathbf{n}\right) \left(\left(\mathbf{u}^h - \mathbf{g}\right) \cdot \mathbf{n}\right) d\Gamma \,, \qquad (6)
\end{aligned}
$$

where $\mathbf{n}$ is the unit normal, $\Gamma^{\mathrm{D},-}$ is the inflow part of Dirichlet boundary Γ^{D}, on which $\mathbf{u}^h \cdot \mathbf{n} < 0$, and τ^B_{TAN} and τ^B_{NOR} are stabilization parameters that act on the tangential and normal components of the velocity at the boundary, respectively. The selection of the stabilization parameters in the presence of cut elements can be found in Refs. [80–82]. In this work, we set $\tau^B_{\mathrm{TAN}} = \tau^B_{\mathrm{NOR}} = 10^3$ [36, 49], which was found to be simple and effective for the type of applications considered.

Remark 1. Over the past two decades, VMS has been extensively studied to explain the origin of stabilized methods [83–87] for CFD and establish a connection between the stabilization operators and the subgrid-scale (SGS) models of turbulence [88–90]. VMS has gained significant prominence as a turbulence model for large-eddy simulation (LES) in the realm of incompressible flow [70, 91–105]. Moreover, similar to traditional LES, VMS requires relatively fine mesh resolution near the boundary layer to accurately capture the sharp velocity gradient and produce accurate results for wall-bounded turbulent flows. However, employing a weakly enforced no-slip condition can mitigate the need for fine boundary layer resolution while preserving accuracy in capturing large scales in the flow [106–114]. While unique to variational methods, the ability of weak no-slip boundary conditions to preserve good solution accuracy on coarser boundary-layer mesh is similar to near-wall modeling approaches in traditional CFD [115], as investigated in Bazilevs et al. [106] and Golshan et al. [116]. As such, the computational methodology presented in this work may be categorized as LES with near-wall modeling.

To accurately capture the immersed boundary Γ of the physical domain Ω_{phys}, a specialized quadrature rule is needed near the interface. Here, we utilize a subdivision-based adaptive quadrature algorithm [36] to improve integration over the physical part of the elements intersected by the immersed boundary without changing the original background mesh. The method recursively refines the quadrature points near the boundary by splitting the quadrature cells of the intersected element into sub-cells until they are completely inside the physical domain or reach a prescribed refinement level. To determine whether an element and its quadrature cells are intersected by the

boundary and whether the quadrature points are inside or outside the physical domain, a point membership classification is required. When the geometry is represented by a point cloud, the point membership classification can be achieved by a winding number test [54, 55], validated by Balu et al. [49]. Furthermore, prior studies have demonstrated that due to the mesh resolution often needed in the boundary layer, two levels of adaptive quadrature refinements are usually sufficient to lead to converged flow quantities of interest [36, 49]. This is consistently applied across all cases in this paper.

2.2.2. *Ghost penalty stabilization on cut elements*

In immersed methods, the object boundary intersects the background mesh in an arbitrary manner, resulting in elements with very small volumes, often referred to as small cut elements or cut cells. The basis functions associated with these small cut elements have limited support within the physical domain, leading to an ill-conditioned system matrix. In conventional boundary-fitted methods, matrix system conditioning can be managed by controlling the quality of the mesh. However, in immersed methods, there is limited control over the shape and size of the cut elements. Moreover, the stability is affected because of the ill-posed Nitsche's formulation in the immersed method; the trace inverse inequality $\left\|\nabla_{\mathbf{n}}\mathbf{u}^h\right\|_{\Gamma^b\cap\Gamma^{\mathrm{D}}} \leq h^{-1/2}\left\|\nabla\mathbf{u}^h\right\|_{\Omega^e_{\mathrm{phys}}}$, where $\nabla_{\mathbf{n}}(\cdot)$ denotes the normal gradient, does not hold for small cut elements. These issues have been extensively investigated by de Prenter et al. [117]. Preconditioning methods, such as the Schwarz preconditioner described in de Prenter et al. [117–119] can improve the conditioning of the matrix system, but they do not inherently address the stability of the formulation. Various techniques have been developed to tackle the stability issues, including the non-symmetric Nitsche's method [120–122], shifted boundary method [123, 124], minimal stabilization procedure [125], and least squares stabilization [126, 127]. However, some of these approaches introduce additional complexities, such as loss of consistency and symmetry, and in some cases, they do not contribute to improving the conditioning.

In recent years, the ghost penalty method [56–58] has emerged as a promising approach to address both stability and conditioning issues. This technique involves adding a penalty term in the elements intersected by the boundary, extending coercivity from the physical domain of the cut element to the entire element volume. By doing so, the ghost penalty method offers additional support to the basis functions of small cut elements. Furthermore, the ghost penalty terms can be directly added to the formulation of VMS and weak Dirichlet boundary conditions, simplifying the implementation. The method has been extended to several applications, including interface problems [128], explicit dynamics [129], heat conduction [130], Stokes [131], Navier–Stokes [132–134], and fluid–structure interaction [135]. It has been modified for problems that utilize higher-order basis functions, such as hierarchical B-splines [40, 136]. In this work, we em-

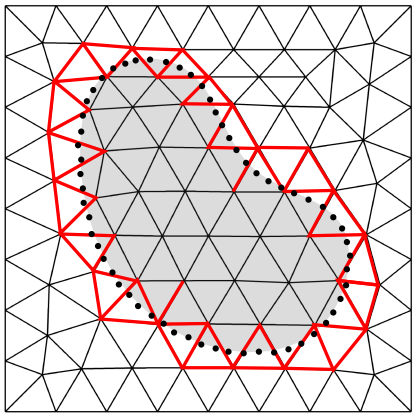


Figure 3: Ghost penalty stabilization: the faces ($\mathcal{F}^f$) where the ghost penalty is applied are highlighted in red. These faces are shared by two neighboring elements that are fully or partially within the physical domain, where at least one of the elements is intersected by the boundary.

ploy ghost penalty stabilization terms for the velocity and pressure based on the work of Schott et al. [133] and Dettmer et al. [136] for incompressible flow. Let $\mathcal{F}^f$ denote the faces shared by two neighboring elements that are fully or partially within the physical domain Ω_{phys}, where at least one of the elements is intersected by the boundary Γ (see Figure 3). The bilinear form of the ghost penalty terms can be written as

$$B^{\text{GhP}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) = \sum_f \int_{\mathcal{F}^f} \tau_{\mathbf{u}}^{\text{GhP}} [\![\nabla_{\mathbf{n}} \mathbf{w}^h]\!] \cdot [\![\nabla_{\mathbf{n}} \mathbf{u}^h]\!] d\Gamma + \sum_f \int_{\mathcal{F}^f} \tau_p^{\text{GhP}} [\![\nabla_{\mathbf{n}} q^h]\!] [\![\nabla_{\mathbf{n}} p^h]\!] d\Gamma \,, \quad (7)$$

where $[\![\cdot]\!]$ denotes the jump operator evaluated at the face $\mathcal{F}^f$, and $\tau_{\mathbf{u}}^{\text{GhP}}$ and τ_p^{GhP} are the ghost penalty stabilization parameters [133] given by

$$\tau_{\mathbf{u}}^{\text{GhP}} = \alpha_{\mathbf{u}}^{\text{GhP}} \mu h_f \,, \quad (8)$$

$$\tau_p^{\text{GhP}} = \alpha_p^{\text{GhP}} \left(\frac{\mu}{h_f} + \frac{\rho U}{6}\right)^{-1} h_f^2 \,. \quad (9)$$

In the above, h_f is the maximum length in the normal direction of each neighboring element at face $\mathcal{F}^f$, U is the freestream velocity magnitude, and $\alpha_{\mathbf{u}}^{\text{GhP}}$ and α_p^{GhP} are non-dimensional penalty parameters. The choice of $\alpha_{\mathbf{u}}^{\text{GhP}}$ and α_p^{GhP} is a trade-off between the stability and accuracy of the solution. For optimal result, $\alpha_{\mathbf{u}}^{\text{GhP}}$, $\alpha_p^{\text{GhP}} \in [0.001, 0.05]$ is recommended for incompressible flow [132–134]. From Eq. (7), it is evident that the ghost penalty method penalizes the jump of the normal gradient of the solution across the cut elements, hence stabilizing the velocity and pressure field within the cut element. Adding the ghost penalty terms, the final semi-discrete problem

becomes

$$B^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) + B^{\mathrm{WBC}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) + B^{\mathrm{GhP}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\mathrm{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = 0\,. \tag{10}$$

2.3. Point cloud processing

Point clouds typically provide only Cartesian coordinates for geometry representation, necessitating additional geometric information for successful utilization in immersogeometric analysis. This includes estimating the normal and area of each point in $\mathcal{P}$, which are used to perform surface integration over the point cloud to impose the weak Dirichlet boundary conditions in Eq. (6). To estimate the normal, we use a jet fitting approach [137, 138] in which a local neighborhood of a query point $\mathbf{q} \in \mathcal{P}$ is selected based on k-nearest neighbor search. The local neighborhood undergoes a quadratic fit to estimate a local surface and compute the normal vector for the query point. The area is computed using the Voronoi diagram of the same nearest neighbors. First, the local point set is projected on a best-fit plane, and Delaunay triangulation is performed. The dual graph of the Delaunay triangulation is used to obtain the Voronoi diagram of the local point set. Then, the associated area of the query point $\mathbf{q}$ is approximated by computing the geodesic area of the Voronoi cell associated with $\mathbf{q}$. This process is repeated for each point in the point cloud to obtain the associated normal and area.

As discussed earlier, it is necessary to distinguish the domain Ω into the physical domain Ω_{phys} and fictitious domain Ω_{fict} for the immersed method. This involves identifying whether a position in Ω is inside or outside the point cloud $\mathcal{P}$. To achieve this, we utilize a winding number test [54, 55] for an effective point membership classification. The winding number test is also used as a preprocessing step to generate adaptive quadrature points near the boundary in the physical domain Ω_{phys} to capture the geometry. For the classification, a threshold winding number value of 0.5 is used. Any point with a winding number smaller than 0.5 is considered outside Ω_{phys}. Detailed validation studies of geometric quantities from point clouds, including normal estimation, area calculation, and winding number computation, can be found in Balu et al. [49].

2.4. Mesh-driven point cloud resampling

The immersed methods rely on the background mesh to solve for the solution field, and refinement of the element size in the background mesh enables detailed and accurate studies. However, a finer background mesh necessitates a denser point representation of the geometry to ensure that each intersected element contains a portion of the object boundary [37, 49]. This is needed to effectively evaluate Eq. (6) using the background basis functions and point cloud surface quadrature to enforce essential boundary conditions. Otherwise, some intersected elements in the background

mesh may fail to recognize the immersed boundary, potentially causing issues such as flow leakage in the case of CFD. This requirement demands a sufficient point density and regularity in the point cloud such that each intersected element in the background mesh adequately contains a point [49]. However, point cloud geometries often come with a fixed number of points, and generating additional points directly is not feasible due to missing connectivity information. To address this issue, one needs a resampling method capable of generating points based on the background mesh. Moreover, it is crucial to avoid generating an excessive number of points, which could lead to issues in point cloud processing and analysis (e.g., speed and memory issues). Our goal is to generate a minimum number of points necessary to satisfy the requirement of having one point for each intersected element. To address this challenge, we propose a resampling method driven by the background mesh information to produce a point cloud with the required point density and distribution.

Furthermore, point clouds obtained from various scanning processes, including photogrammetry and LiDAR, often suffer from sparsity, incompleteness, and missing essential features in severe cases, as previously discussed in Section 2.1. To address these shortcomings, numerous methods have been proposed in the field of computer vision. These methods aim to augment point clouds by filling sparse regions and holes using various machine learning-based approaches [53] and geometric algorithms [51], such as boundary-detection [139], surface-based [140, 141], and volume-based hole filling [142, 143]. The proposed mesh-driven resampling acts as a hole-filling algorithm as well, combining both volume and surface-based methods to regenerate points globally while filling the incomplete region and providing an analysis-suitable point cloud for the immersed method.

The method combines two important ingredients: the winding number field and local surface reconstruction. The winding number field serves as a globally consistent field that includes information about the regions that are inside and outside the point cloud geometry. Additionally, the winding number field can represent a watertight geometry by capturing the transition at the interface of inside and outside regions. Surface topology and geometry of the point cloud can be inferred with isosurface extraction, enabling the reconstruction of a continuous surface. This approach of reconstructing a watertight mesh from a raw point cloud has been widely employed in various studies [55, 144, 145]. However, relying solely on the winding number field to approximate boundaries may overlook intricate geometric features and prove computationally intensive in accurately identifying sharp interface regions. Furthermore, noisy point clouds can significantly impact the winding number field at these interfaces, leading to sub-optimal geometry representation. To solve this issue, we utilize local surface reconstruction to better represent the boundary in the vicinity of intersected elements. The local surface reconstruction builds upon the winding number field information. Once the regions of the interface (or intersected elements) have been identified by the winding number test, the local surface can effectively capture the geometry pass-

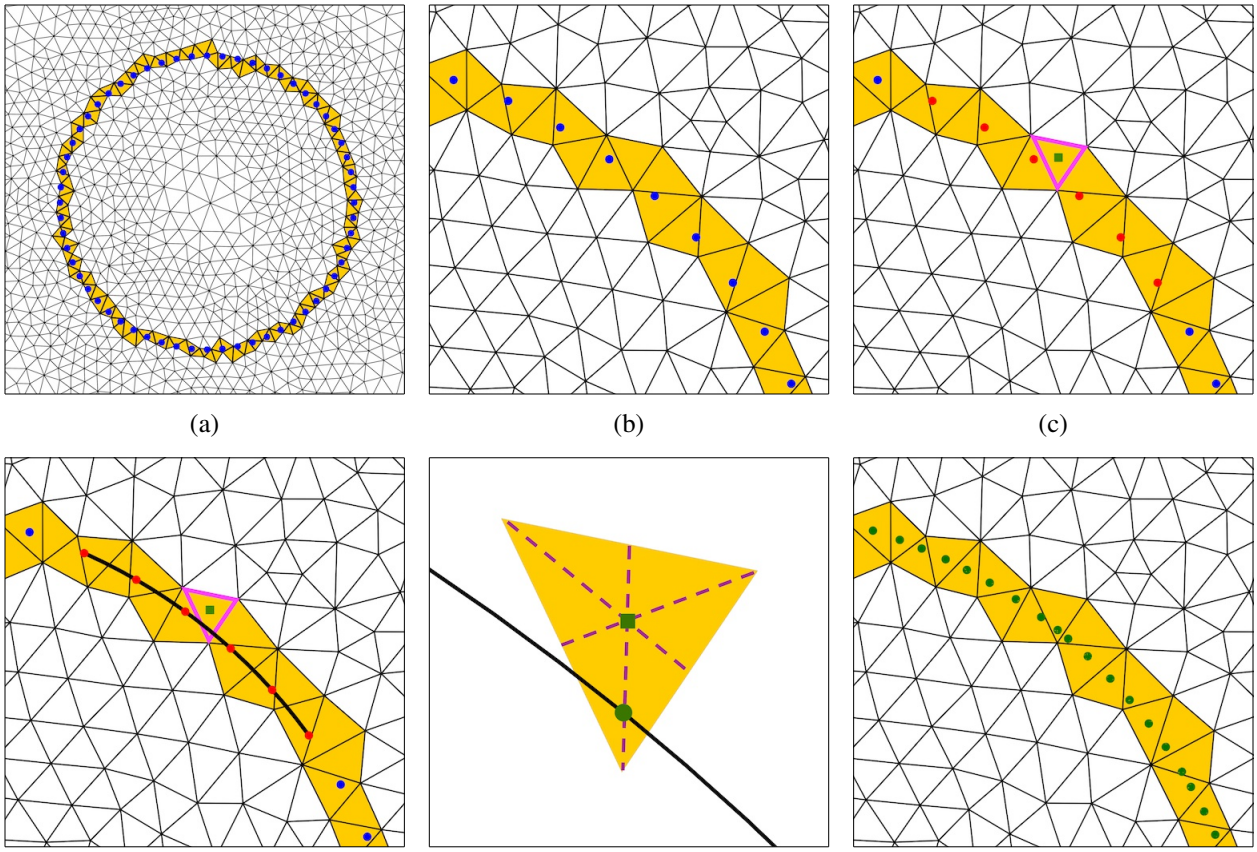


Figure 4: (a) Overlaying the original point cloud onto the background mesh and identifying intersected elements using the winding number test. (b) A closer look reveals incomplete filling of intersected elements. Note that the winding number test identifies these elements based on the global representation of the point cloud and does not require a point to be inside the element in order to classify the element as intersected. (c) Selection of an intersected element, determining nearest neighboring points using its centroid. (d) Fitting a local quadratic surface to the neighboring points. (e) Testing each median against the local surface for intersection, accepting only the closest point within element bounds as a resampled point. (f) Repeating steps (c)–(e) for all intersected elements to ensure at least one point per element.

ing through these elements.

The proposed mesh-driven resampling method draws inspiration from established techniques, such as moving least squares (MLS) [146], which employ local surface reconstruction to create a smooth surface representation and sample points uniformly. However, our approach distinguishes itself in how points are sampled on the reconstructed surface. We leverage background mesh information to project points, ensuring that each intersected element contributes exactly one point to the resampled point cloud. Figure 4 outlines the step-by-step procedure for acquiring the resampled point cloud. Initially, the original point cloud is overlaid onto the background mesh, and the intersected elements are identified by executing the winding number test (Figures 4a and 4b). As discussed earlier, the winding number test yields a watertight representation of the intersected region. Next, a local surface is constructed for each intersected element by considering a set of

Algorithm 1 Mesh-Driven Point Cloud Resampling

Input: Background mesh, Point cloud
Output: Resampled point cloud
procedure ResamplePointCloud(backgroundMesh, pointCloud)
    for element e in backgroundMesh **do**
        $N_e \leftarrow$ Nodes of element e
        $W \leftarrow$ CalculateWindingNumber(N_e, pointCloud)
        $I \leftarrow \{i \,|\, W_i < 0.5\}$ ▷ Nodes inside point cloud
        if $|I| > 0$ and $|I| < |N_e|$ **then**
            intersectedElements $\leftarrow e$
        end if
    end for
    for element e_i in intersectedElements **do**
        $C \leftarrow$ Centroid(e_i)
        $N_k \leftarrow$ NearestNeighbors(pointCloud, C)
        $S \leftarrow$ LocalSurfaceFitting(N_k)
        for median of e_i **do**
            $P_{intersect} \leftarrow$ IntersectWithSurface(S, median)
            if Distance($P_{intersect}$, C) < Distance(closestIntersection, C) **then**
                closestIntersection $\leftarrow P_{intersect}$
            end if
        end for
        resampledPointCloud $\leftarrow$ closestIntersection
    end for
end procedure

nearest neighbor points in the original point cloud. The nearest neighbors are determined using the centroid of the intersected element as the query point (Figures 4c and 4d). Subsequently, the query point is projected to the fitted local surface. Projection involves selecting medians extending from the centroid of the face of the intersected element to its opposite vertex. There are four medians in a tetrahedral element. These bisectors are tested against the locally fitted surface to identify intersections. At least one median must intersect with the reconstructed surface. The closest intersection is then chosen as the resampled point (Figure 4e). This process iterates for each intersected element, resulting in exactly one point being generated for every intersected element (Figure 4f). A pseudocode for the proposed method is presented in Algorithm 1. It is important to highlight that our resampling method addresses the noise within the point cloud with regularization through local surface reconstruction. The robustness of the method against point clouds of various quality is demonstrated in the next section.

Remark 2. The winding number test is utilized to determine element intersections based on the global representation of a point cloud. It evaluates whether a region is inside or outside an object

by calculating the number of times the point cloud winds around a query point. This method allows for the identification of intersected elements based on the overall geometry, without requiring the element to contain a point in order to be classified as intersected.

Remark 3. In the process of local surface reconstruction of the nearest points, a quadratic surface is reconstructed using the least squares method. To achieve this, the direction of the z-axis is determined through principal component analysis (PCA), selecting the direction with the least eigenvalue as the z-direction. Subsequently, the points are surface fitted to a height function $z = f(x, y)$ using the least squares method. This ensures a unique solution for surface fit. A higher-order polynomial surface fit can also be performed; however, a quadratic fit was found sufficient for most point clouds.

Remark 4. Solving for the intersection between the local quadratic surface and the median line presents a nonlinear problem, for which we employ Newton's method. The median is parameterized between the centroid of the face and the opposite vertex of the element. The intersection is solved iteratively until convergence is achieved.

Remark 5. The local neighborhood of points utilized to construct the local surface is determined using the k-nearest neighbor search with a fixed number of neighbors. In this study, a neighborhood size of 50 nearest neighbors, reported by Balu et al. [49], was deemed suitable for the scale and initial point density of the point cloud.

3. Validation studies on NIMBUS

In this section, we validate the proposed resampling method using analytical and complex geometries to show how it fixes sparse and noisy point clouds and generates analysis-suitable point clouds for the immersed method. Also, we present a convergence study of the point cloud CFD using the Stanford bunny and a soda can scanned using MVSNet to demonstrate the robustness and accuracy of the proposed NIMBUS framework. Validation of the implementation of ghost penalty stabilization for immersogeometric analysis is provided in Appendix A.

3.1. Validation of point cloud resampling

3.1.1. Randomly sampled sphere: hole-filling and robustness to noise

In this study, the analytical geometry of a sphere serves as a suitable model to demonstrate the effectiveness of the resampling method, offering clear visualization and quantifiable geometric properties. The sphere is randomly sampled by generating points and projecting them onto the sphere by dividing their distance to the sphere center. This approach allows to generate an arbitrary number of points required for the resampling study. The background mesh required for resampling

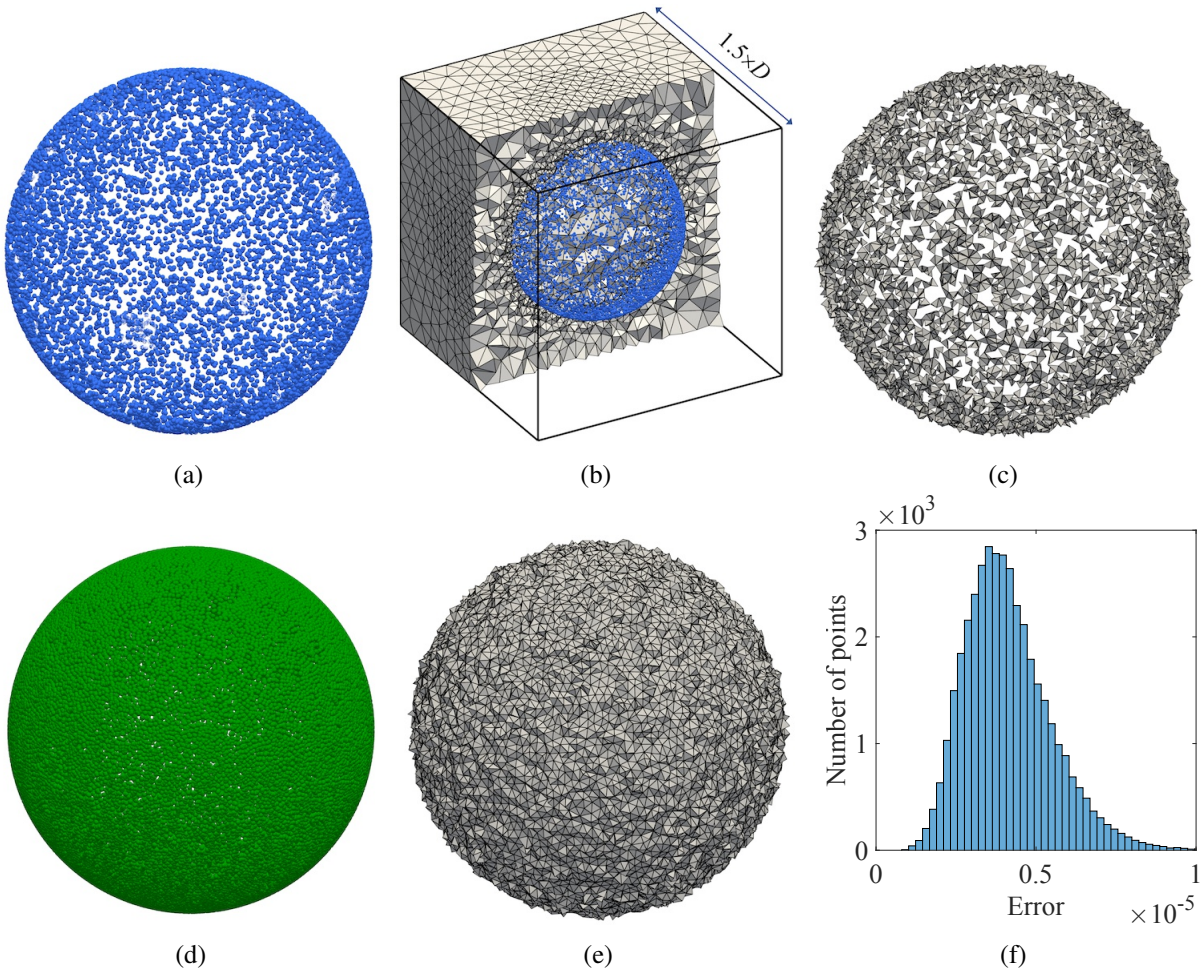


Figure 5: (a) Randomly sampled sphere with 10,000 points. (b) The background mesh (RM1) of a cube domain is used to perform the resampling study. The dimension of the mesh is based on the diameter ($D = 1$) of the sphere. (c) Intersected elements in the background mesh containing at least one point. (d) Resampled point cloud with 39,768 points obtained for mesh RM1. (e) The updated intersected elements after resampling. (f) Histogram plot of the error distribution of the resampled point cloud.

is obtained using Gmsh [147], incorporating local refinements around the points. The process of generating an immersed mesh with local refinement around geometry using Gmsh has been elaborated in Xu et al. [36].

Initially, we explore the primary objective of the resampling method, aiming to generate points based on the background element size. To achieve this, we consider a randomly sampled sphere with 10,000 points, as shown in Figure 5a, and two background meshes, RM1 and RM2, featuring different element sizes. The mesh statistics and element sizes utilized to construct the background mesh are detailed in Table 1. Figure 5b illustrates the mesh domain (a cube) discretized with the coarsest mesh RM1, which showcases refinement along the point cloud geometry. Figure 5c highlights the intersected elements in the mesh containing at least one point, revealing visible gaps between elements due to the insufficient number of points for mesh RM1. This gap can lead to

Table 1: Resampling statistics for the randomly sampled sphere.

Mesh	Near boundary element size	Outer box element size	Total number of elements	Number of resampled points
RM1	0.02	0.2	244,497	39,768
RM2	0.01	0.1	978,833	159,851

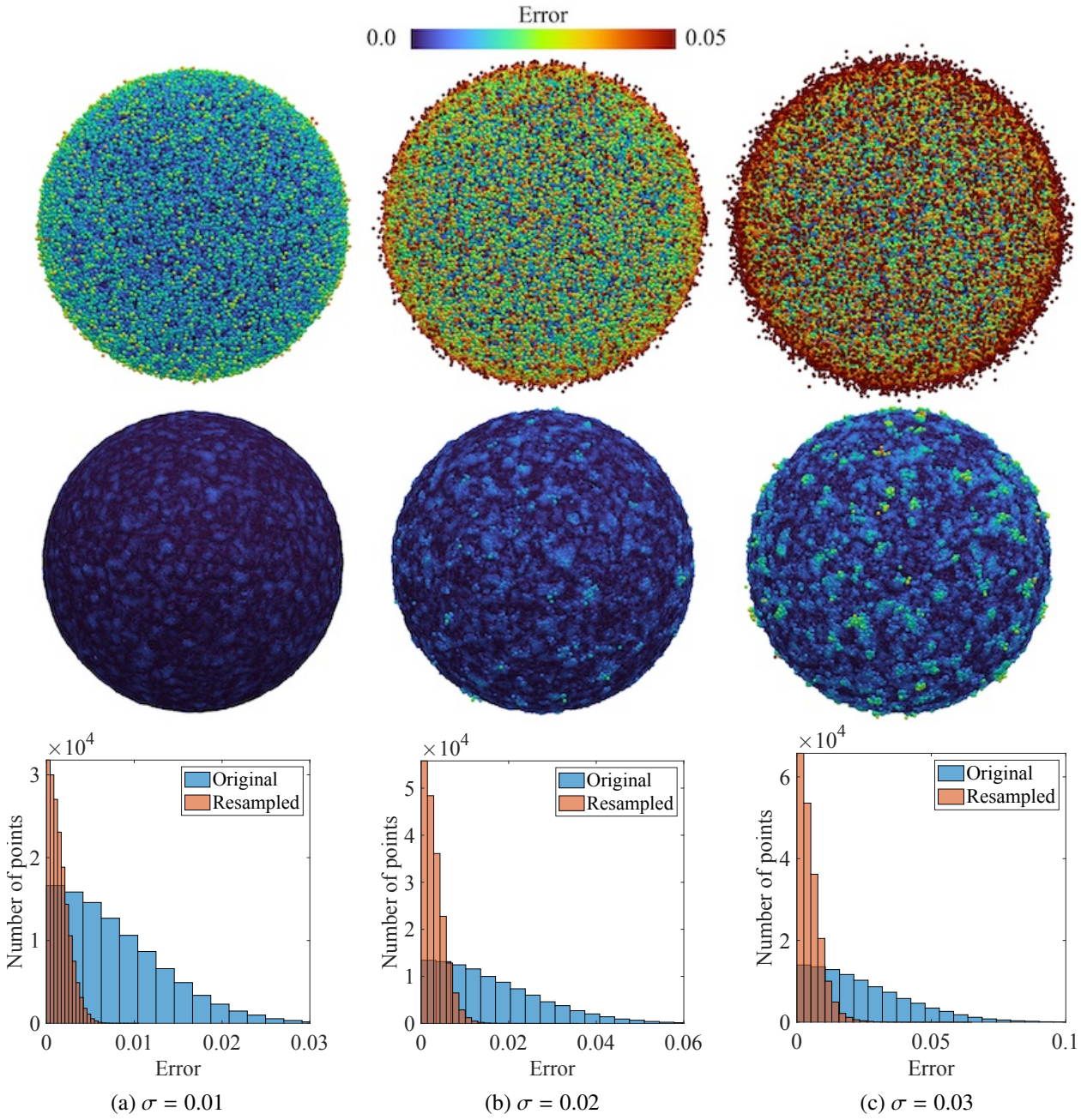
issues such as flow leakage through the boundary, impacting solution accuracy, as further discussed in the subsequent section.

Using the proposed resampling method, we successfully increased the original point cloud size from 10,000 points to 39,768 points for RM1 and 159,851 points for RM2. This indicates that the number of resampled points increases with a decrease in the element size in the background mesh. Figures 5d and 5e present the resampled point cloud and updated intersected elements obtained for mesh RM1, respectively, revealing no visible gaps between intersected elements. This ensures a watertight-like geometry representation for the immersed method. Moreover, the number of points generated equals the number of intersected elements, confirming that each intersected element contains exactly one point based on the algorithm. The accuracy of the resampled point cloud is illustrated by error distribution as shown in Figure 5f. The error is defined as $\left|r - \|\mathbf{p}\|\right|$, where r is the radius of the sphere and $\mathbf{p} \in \mathcal{P}$. The error distribution is remarkably low, with the slight error due to round-off errors arising from the least squares method used for the quadratic fit, despite the original points being distributed on a quadratic surface.

Next, we aim to illustrate the efficacy of the resampling method in addressing noisy point clouds. To achieve this, we utilize a sphere comprising 100,000 points and introduce Gaussian noise at varying standard deviations. Specifically, we consider three cases with standard deviations of 0.01, 0.02, and 0.03. To provide a high-density point cloud for the background mesh, we employ mesh RM2 for the sphere. Figure 6 presents the resampled point cloud recovered from the original noisy point cloud. It is evident that a significant portion of the noise has been mitigated, and the geometric shape of the sphere has been successfully recovered. Furthermore, the histogram plot of the error for both the noisy and resampled point clouds shows that the error is substantially reduced for point clouds following resampling. This highlights the method's effectiveness in handling geometries with noise, which is crucial in regularizing noisy point clouds resulting from multiple registrations and geometric defects.

3.1.2. Scanned geometry: complex and sparse point cloud

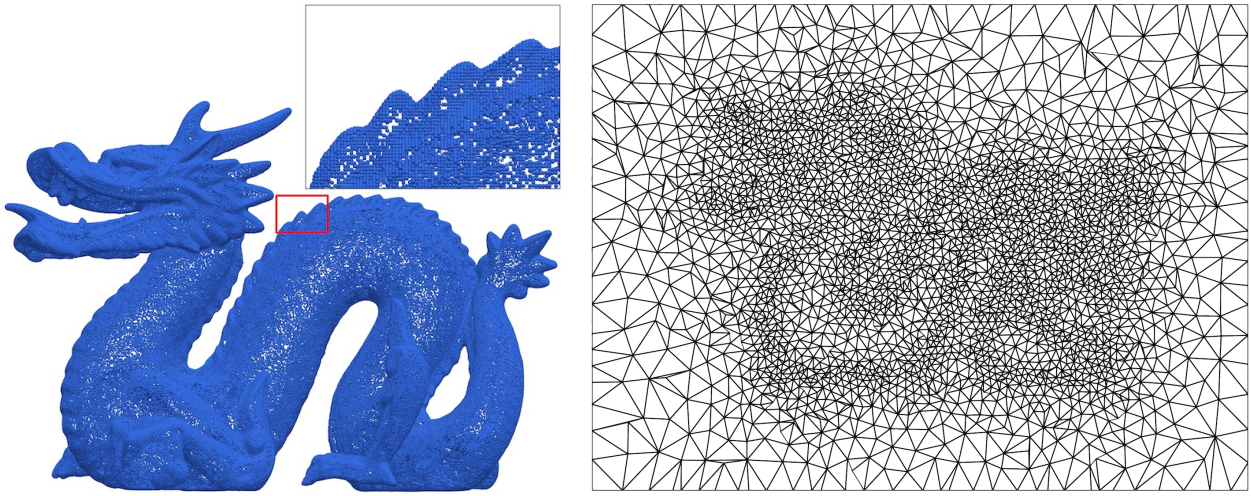
This section focuses on demonstrating the effectiveness of the proposed approach in resampling a real-world complex geometry, specifically the Stanford dragon. Given its intricate shape and presence of numerous geometric irregularities [148], the dragon serves as an ideal candidate to validate the efficacy of the proposed resampling method to preserve geometric complexities and



(a) $\sigma = 0.01$ (b) $\sigma = 0.02$ (c) $\sigma = 0.03$

Figure 6: Comparing the distribution of resampled and original point clouds with Gaussian noise of different standard deviations. The first and second rows show the original noisy point cloud and resampled point cloud, respectively, colored by the error.

produce an analysis-suitable point distribution. The Stanford dragon was created with a Cyberware 3030 Model Shop (MS) Color 3D Scanner at Stanford University in 1996 [149]. It is a well-known 3D test model widely used to benchmark geometric algorithms, such as polygonal simplification, compression, and surface smoothing [150–152]. For the resampling purpose, we utilize the reconstructed mesh of the dragon available at The Stanford 3D Scanning Repository [148], which



(a) (b)

Figure 7: (a) Original point cloud obtained using the vertices of the Stanford dragon mesh reconstruction obtained from The Stanford 3D Scanning Repository [148]. (b) Background mesh (RM1) for resampling.

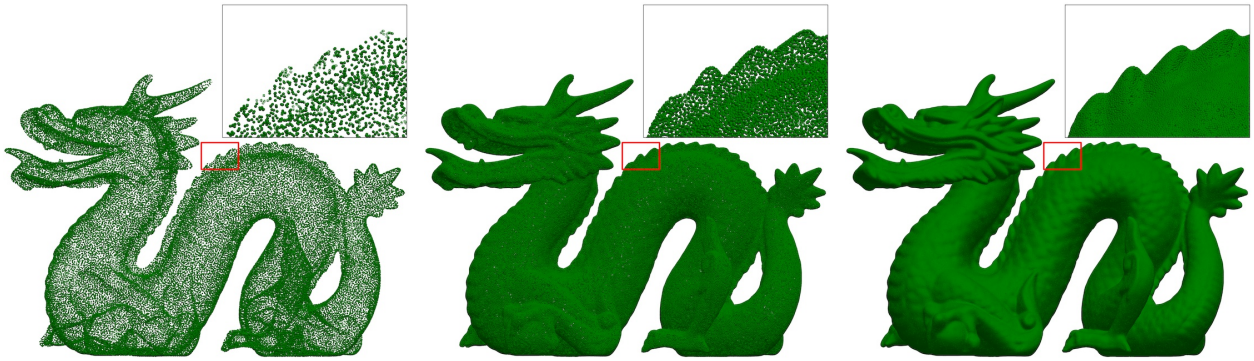
Table 2: Background mesh statistics for the Stanford dragon. The element sizes are scaled with respect to the width of the dragon ($W = 0.0916$).

Mesh	Near boundary element size ($\times W$)	Outer box element size ($\times W$)	Total number of elements
RM1	0.02	0.2	605,242
RM2	0.01	0.1	2,508,990
RM3	0.005	0.05	10,066,800

consists of 871,306 triangles and 435,545 vertices[2]. The vertices of the reconstructed mesh are used as the original point dataset for the resampling algorithm, as presented in Figure 7a. We construct three background meshes, RM1, RM2, and RM3, featuring varying element sizes, detailed in Table 2. Figure 7b illustrates the coarsest mesh RM1, highlighting refinements near the dragon's boundary. Following resampling, the number of resampled points is 107,854 for RM1, 433,400 for RM2, and 1,678,269 for RM3, as shown in Figure 8. Initial visual inspection shows that the local features, such as the dragon scales, are preserved in the resampled point clouds without any significant geometry simplification.

To assess the improvement of the resampled point cloud distribution with respect to their background mesh, we determine the point spacing of the point cloud and compare it with the mesh element size. Point spacing is calculated based on the surface point density of local neighborhoods of every point obtained using a fixed radius search of $0.04 \times W$ ($W = 0.0916$ is the width of the dragon). The local neighborhood selection corresponds to twice the size of the background element of the coarsest mesh RM1 and is applied to all point cloud cases of the dragon for a consistent

[2]Although the repository web page reported 1,132,830 triangles, the actual .ply file contains only 871,306 triangles.



(a) RM1 (b) RM2 (c) RM3

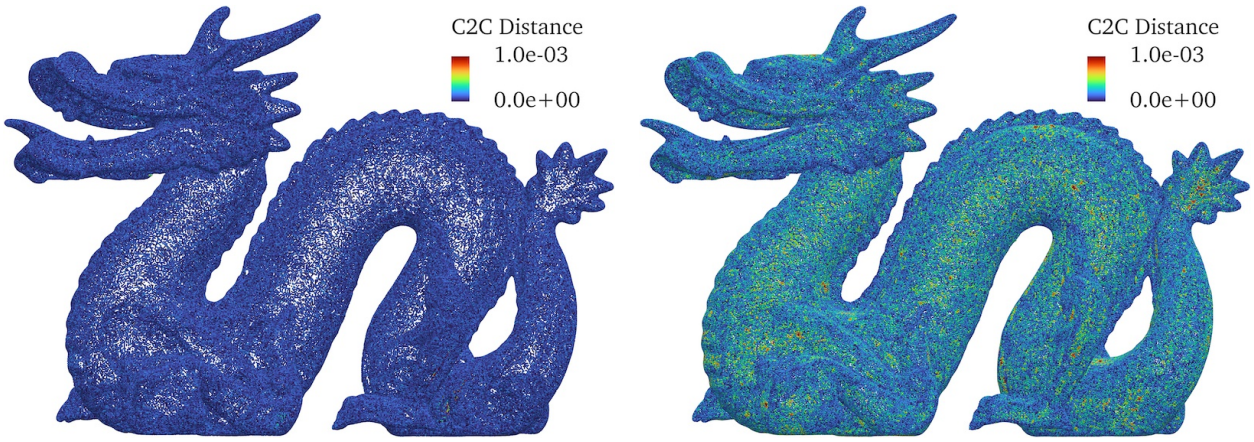
Figure 8: Resampled point cloud obtained for the Stanford dragon using different background meshes. The zoomed-in view shows that the local features along the dragon's back are captured consistently.

Table 3: Point cloud statistics for the Stanford dragon. $\bar{X}$ denotes the mean and σ denotes the standard deviation of the point spacing. The element sizes and point spacing are scaled with respect to the width of the dragon ($W = 0.0916$).

Point cloud	Number of points	Point spacing $\bar{X} \pm \sigma$ ($\times W$)	Maximum point spacing ($\times W$)	Background element size ($\times W$)
RM1	107,854	0.0087 ± 0.0004	0.0134	0.02
RM2	433,400	0.0044 ± 0.0002	0.0065	0.01
RM3	1,678,269	0.0022 ± 0.0001	0.0031	0.005
Original	437,645	0.0042 ± 0.0008	0.0129	

comparison. Table 3 summarizes the point spacing statistics of the resampled point clouds and the original point cloud of the dragon. The average point spacing of the resampled point cloud consistently decreases relative to the background element size. Moreover, the maximum point spacing within each point set is less than the corresponding background element size, ensuring that the point clouds maintain sufficient density and distribution for each mesh. The original point cloud exhibits a similar number of points and translates to comparable average point spacing as the point cloud resampled using RM2. However, the variation in the point spacing is significantly higher than in any of the resampled point clouds, indicating a more irregular point distribution originally. Also, the maximum point spacing of the original point cloud only meets the density requirement of the coarsest mesh RM1, but not RM2 and RM3. This highlights that even if the number of points in the original point cloud exceeds the density requirement for RM2, the irregular point distribution makes it unsuitable for analysis purposes.

To evaluate the efficacy of the resampling in capturing the original geometry while regularizing geometric defects and sparsity, we utilize the cloud-to-cloud (C2C) distance between the original point cloud and the resampled point cloud obtained for RM2, both having a comparable number of points. C2C distance is a measure of the distance of a target point to its nearest point in the reference point cloud. Figure 9a presents the C2C distance of the points in the original cloud, with



(a) Orignal to resampled (b) Resampled to original

Figure 9: Comparison of cloud-to-cloud (C2C) distance of (a) the original point cloud to the resampled point cloud for RM2 and (b) vice versa. The color represents the nearest distance of the target point to the reference point cloud. The low C2C distance in (a) suggests a close match between the target and reference point clouds, indicating that the original geometry is accurately represented in the new resampled point cloud. The higher C2C distance in (b) results from resampling in sparse regions where new points are farther from the original. This demonstrates the method's effectiveness in generating points in sparse areas while closely capturing the overall geometry.

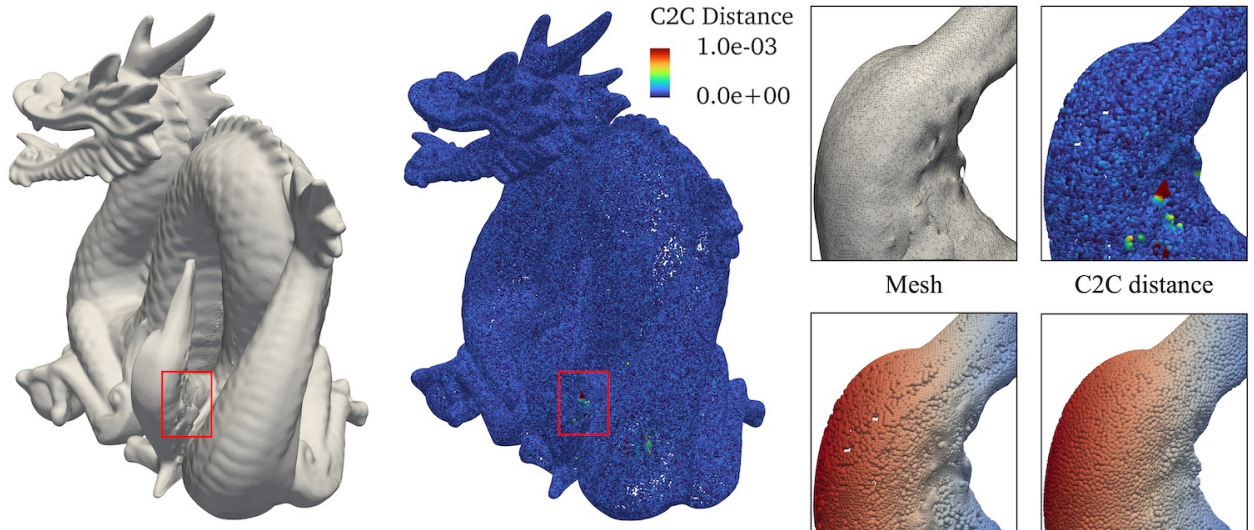


Figure 10: Regularization of the dragon point cloud near geometric defects. The higher C2C distance from the original to the resampled point cloud indicates that the points associated with the defect are not present in the resampled point cloud. The color gradient in the original and resampled point cloud represents the depth, providing a clear visualization of the improvement.

the resampled point cloud serving as a reference. The consistently low C2C distance for the original point cloud suggests that the target points closely match the reference point cloud, indicating that the original geometry is accurately represented in the new resampled point cloud. Figure 9b demonstrates the C2C distance of the points in the resampled point cloud with the original point

cloud as the reference. The higher C2C distance arises from points that are resampled in regions where the original point cloud was sparse. As a result, these new points are relatively farther from the original point cloud. This observation shows the effectiveness of the method in generating points in sparse regions while capturing the geometry very closely. Figure 10 illustrates the impact of the resampling method in regions where the point cloud originally suffered geometric defects, such as hanging surfaces, self-intersections, and noise stemming from mesh reconstruction. The inflicted regions have been regularized by the resampling process to provide a watertight-like point cloud with the original geometric defects removed.

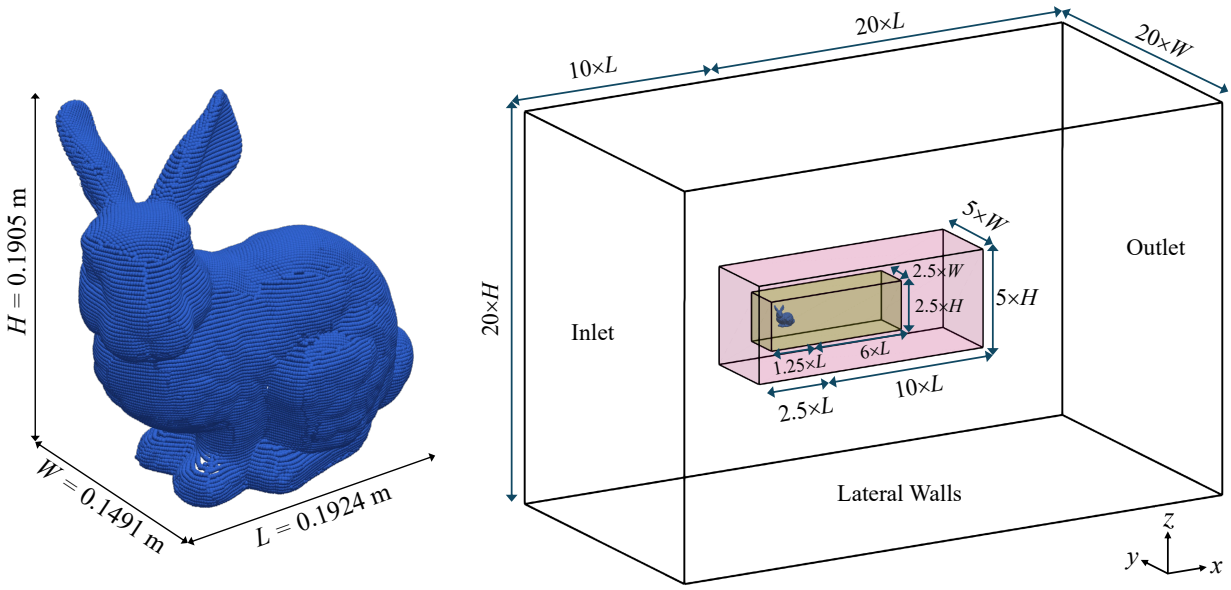
3.2. Validation of point cloud CFD

In this section, we use the NIMBUS framework to perform direct flow analysis on two different point cloud geometries: the Stanford bunny and an MVSNet-generated soda can. We aim to show the effectiveness of the proposed resampling method in facilitating a direct mesh convergence study and generating analysis-suitable point clouds corresponding to background meshes. Additionally, we demonstrate that a regularized point cloud obtained after resampling a noisy point cloud can improve the solution quality.

3.2.1. Flow past the Stanford bunny

In this study, we utilize a point cloud sampled from the renowned Stanford bunny mesh reconstruction, originally created by Greg Turk and Mark Levoy in 1994 at Stanford University [153]. The iconic model has served as a cornerstone in the field of 3D computer vision, extensively studied and benchmarked with various geometric algorithms. Furthermore, its recent surge in popularity has expanded its utility into physics-based simulations encompassing linear elasticity [154], large deformation [155], flow problems [29], soft-solid behavior [156], fractures [157], and reaction-diffusion [158]. In this work, we focus on conducting a mesh convergence study of the flow past the Stanford bunny, providing valuable and accurate benchmark results for future reference. To the best of our knowledge, such a study has not been reported previously.

The original mesh reconstruction of the bunny [148] is scaled to a height of 0.1905 m (7.5 inches), which is the true height of the original clay bunny used for scanning. The 35,947 vertices of the mesh reconstruction are sampled as the original point cloud dataset, as depicted in Figure 11a. The bounding domain size of the point cloud is $(-0.1170, 0.0754) \times (-0.0765, 0.0726) \times (-0.0209, 0.1696)$ m^3, representing the length (L), width (W), and height (H) along the x, y, and z axes, respectively. Figure 11b illustrates the computational domain constructed based on the bunny's dimensions and the boundary conditions. The width of the bunny ($W = 0.1491$ m) serves as the characteristic length for the calculation of the Reynolds number. The density and viscosity are set to $\rho = 1.184$ kg/m^3 and $\mu = 1.845 \times 10^{-5}$ N·s/m^2, respectively. The inflow and lateral no-penetration boundary conditions are enforced strongly, while the no-slip condition on the surface



(a) (b)

Figure 11: (a) Dimensions of the Stanford bunny point cloud with 35,947 points. The bounding domain of the object is $(-0.1170, 0.0754) \times (-0.0765, 0.0726) \times (-0.0209, 0.1696)$ m^3, representing the length (L), width (W), and height (H) along the x, y, and z axes, respectively. (b) Computational domain showing boundary conditions and mesh refinement regions for the background mesh. The dimensions of the domain are based on L, W, and H reported in (a).

Table 4: Element sizes in the background mesh for the Stanford bunny. The element sizes are scaled with the characteristics length of the bunny ($W = 0.1491$ m).

Mesh	Total number of elements	Near boundary ($\times W$)	Inner refinement ($\times W$)	Outer refinement ($\times W$)	Outer box ($\times W$)	Resampled points
IM1	554,727	0.02	0.2	$0.8/\sqrt{2}$	1.2	49,991
IM2	3,846,468	0.01	0.1	$0.4/\sqrt{2}$	1.0	309,645
IM3	9,921,344	0.005	0.05	$0.2/\sqrt{2}$	0.8	757,867

of the bunny is enforced weakly. Freestream velocities of $U = 0.0105$ m/s and $U = 0.0314$ m/s are applied at the inlet to simulate a flow past the stationary bunny at $Re = 100$ and 300, respectively, with a consistent time stepping of 1×10^{-2} s. The outflow boundary is traction-free.

To conduct a convergence study, we discretize the computational domain using three background meshes denoted as IM1, IM2, and IM3, each exhibiting different levels of refinement (see Table 4 for mesh statistics). Figure 12 illustrates the coarsest mesh (IM1) with refinements near the geometry. Next, we generate the resampled point cloud of the bunny for the three background meshes to ensure sufficient point cloud density crucial for accurate and stable simulation. Using the proposed resampling method, the original point cloud of the bunny is processed to generate 49,991 points for IM1, 309,645 for IM2, and 757,867 for IM3, as depicted in Figure 13.

Observing from the simulation results, both $Re = 100$ and 300 cases reach a steady state. Figure 14 shows the velocity field and streamlines for $Re = 300$, where the formation of vortices

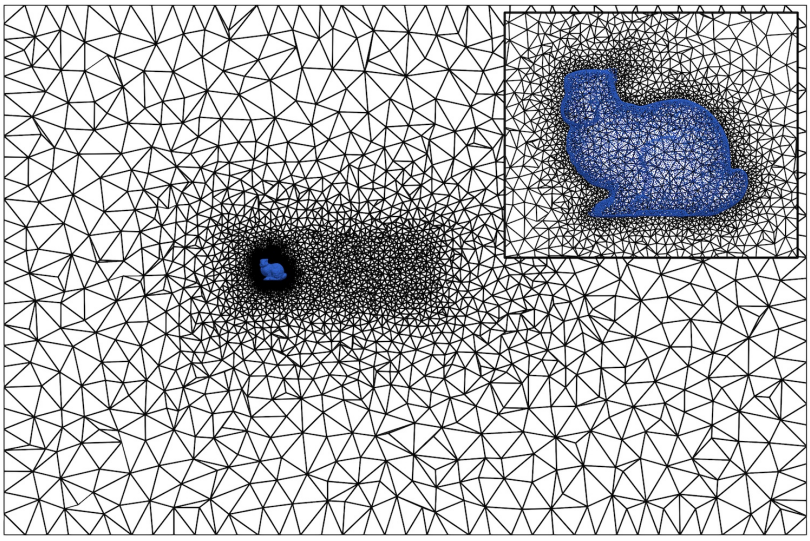
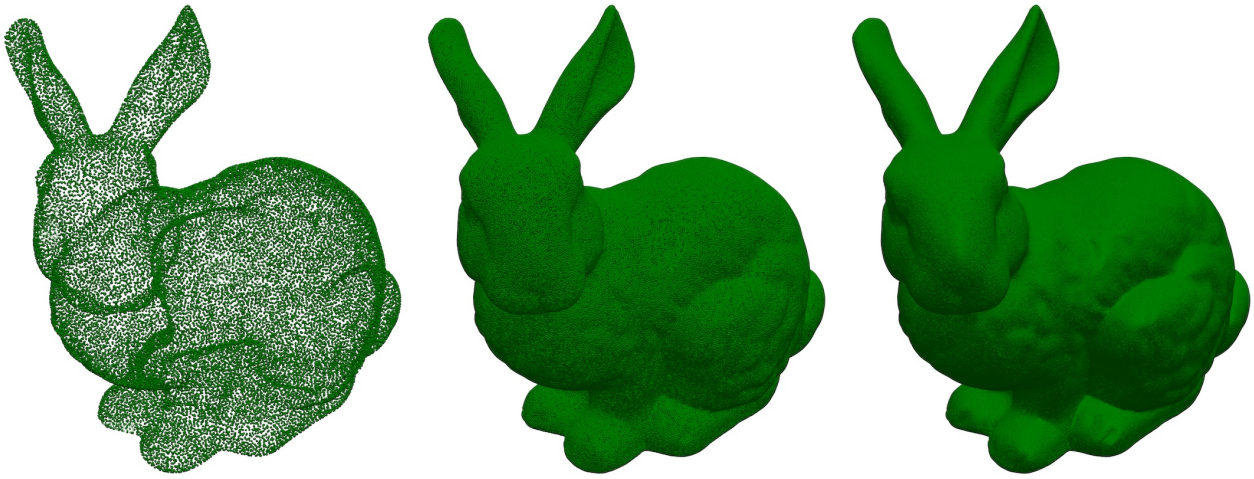


Figure 12: Central cross-section view of the coarsest background mesh (IM1) with a zoomed-in view of elements around the bunny.



(a) IM1 (b) IM2 (c) IM3

Figure 13: Resampled points obtained for Stanford bunny using different background meshes.

behind the bunny is observed. To demonstrate convergence, we utilize characteristic quantities commonly used in fluid mechanics, focusing on the drag and pressure coefficients. The drag coefficient is computed as $C_D = 2F_D/(\rho U^2 A)$, where F_D is the drag force, U is the magnitude of inlet velocity, and A is the frontal area of the bunny. The drag force is evaluated using a conservative definition of traction [36, 108]. To calculate the frontal area of the bunny, we project the point cloud onto a 2D plane along the x-axis and generate its alpha shape [159], which yields a non-convex polygon tightly encapsulating the 2D point set. The frontal area ($A = 0.0171$ m^2) is then determined by computing the area enclosed by the non-convex polygon. Table 5 presents the drag coefficients obtained using different background mesh sizes at $Re = 100$ and 300. Convergence is

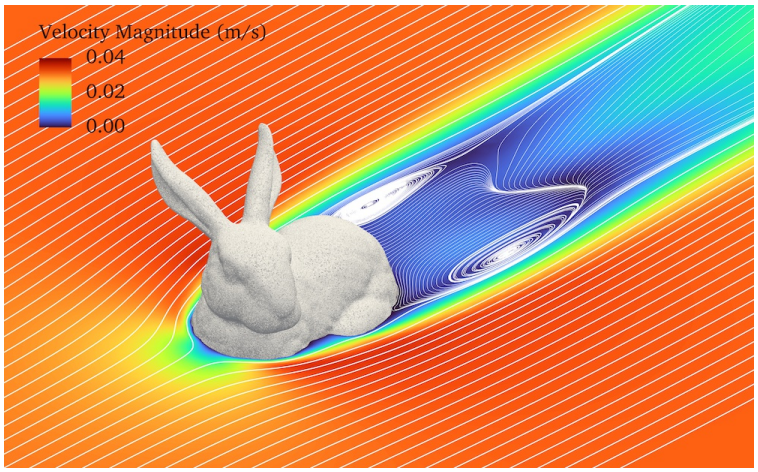


Figure 14: Velocity magnitude and streamlines at the central cross-section of the bunny for $Re = 300$ obtained using IM3.

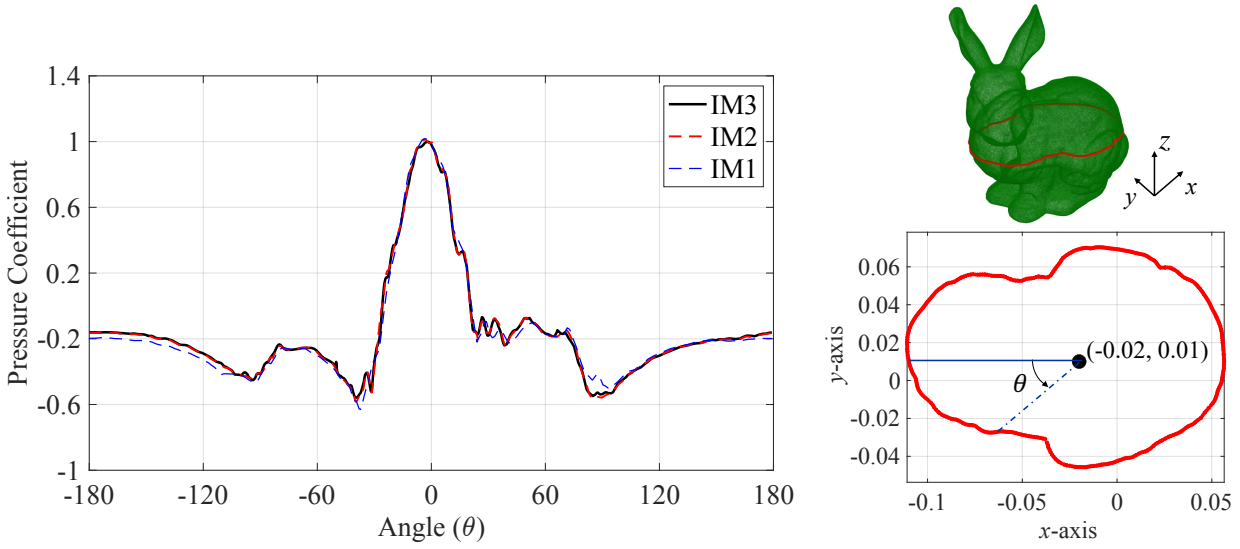
Table 5: Drag coefficients obtained using different background meshes for the Stanford bunny at different Reynolds numbers.

Mesh	$Re = 100$	$Re = 300$
IM1	1.426	0.931
IM2	1.348	0.858
IM3	1.334	0.846

observed for both cases.

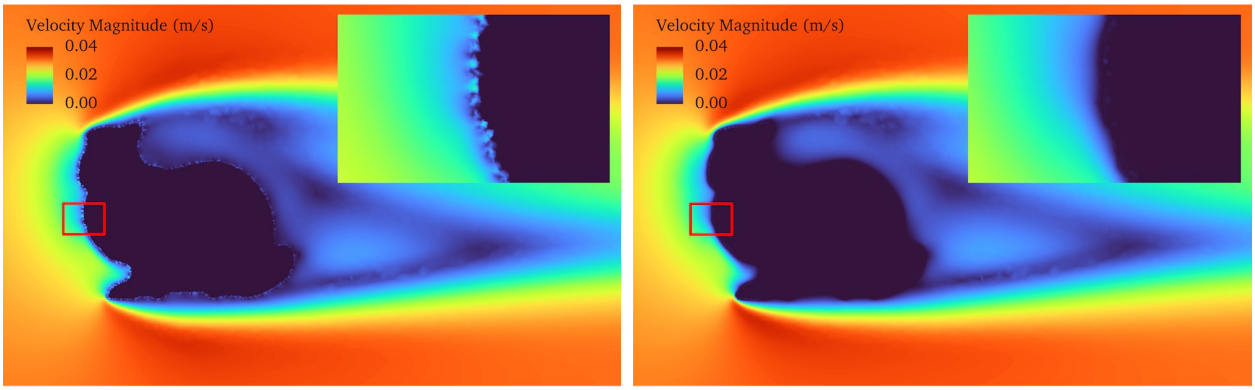
Next, we calculate the pressure coefficient, a non-dimensional parameter that characterizes the relative pressure distribution in the field. Figure 15 shows the pressure coefficient plot along the cross-section of the bunny at $z = 0.05$ m for $Re = 300$. The complexity and sharp corners within the boundary are evident in the cross-section profile at $z = 0.05$ m, demonstrated in Figure 15b. The pressure coefficient plot in Figure 15a effectively captures the pressure variations along the intricate geometry. Furthermore, excellent convergence is observed under mesh refinements. This study emphasizes the significance of the NIMBUS framework in facilitating a direct mesh convergence study, which previously would have been deemed challenging. Additionally, it provides an opportunity to conduct an extremely fine study where the original reconstruction may not be sufficient for accurate analysis.

To demonstrate the effect of insufficient point cloud density for conducting an immersed analysis, we utilize the finest background mesh IM3 and the coarsest resampled point cloud of IM1 to simulate the flow at $Re = 300$. The resulting velocity field around the bunny, as presented in Figure 16a, reveals severe impacts near the boundary with significant oscillations. This phenomenon arises due to inadequate point density, rendering some intersected elements without enforcing essential no-slip boundary conditions effectively, thereby leading to flow leakage. On the contrary, Figure 16b showcases the velocity field near the boundary using the background mesh IM3 with



(a) (b)

Figure 15: (a) Pressure coefficients along the bunny's boundary at $z = 0.05$ m obtained using different background meshes for $Re = 300$. (b) The cross-sectional profile at $z = 0.05$ m; $(-0.02, 0.01)$ m is taken as the origin to measure the angle (θ).



(a) (b)

Figure 16: Comparison of velocity magnitude near the bunny's boundary obtained using background mesh IM3 with (a) coarser point cloud (resampled using IM1) and (b) finer point cloud (resampled using IM3) for $Re = 300$.

its corresponding resampled point cloud. With sufficient point density and distribution, each intersected element contains a point inside, effectively preventing flow leakage and mitigating spurious oscillations near the boundary. This observation is further validated through the pressure coefficient plot depicted in Figure 17, where oscillations along the cross-section at $z = 0.05$ m are evident. However, the use of resampled point clouds corresponding to the background mesh significantly removes these oscillations, offering a more accurate and stable solution. This insight highlights the importance of the proposed resampling method that generates the necessary density and distribution needed for a given computational setting.

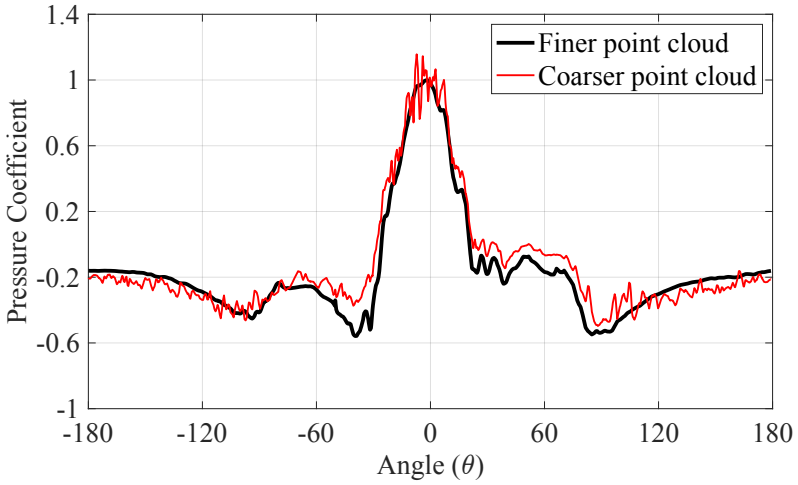


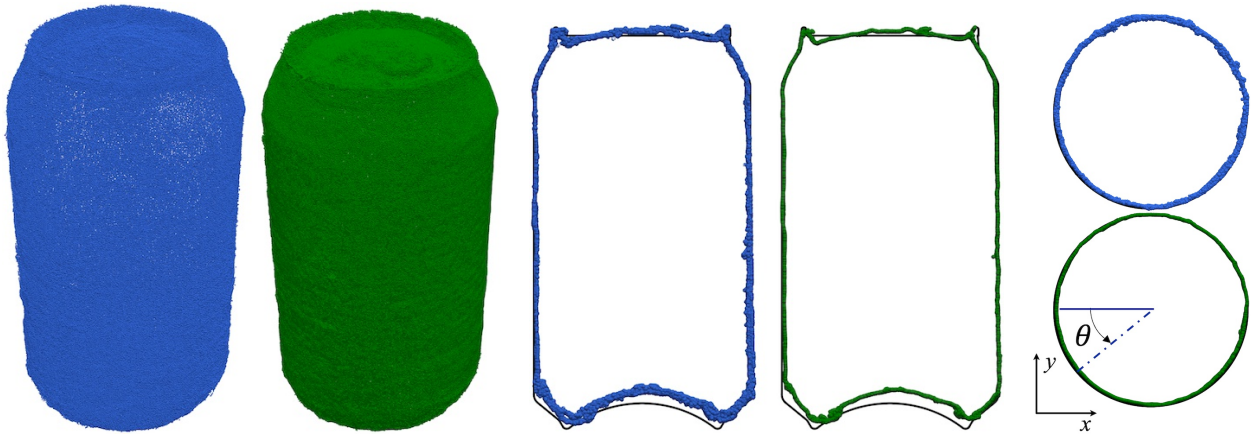
Figure 17: Pressure coefficients along the bunny's boundary at $z = 0.05$ m obtained using background mesh IM3 with a coarser point cloud (resampled using IM1) and a finer point cloud (resampled using IM3) for $Re = 300$.

3.2.2. Flow past a soda can

In this study, our focus is on leveraging the resampling method to regularize a raw output point cloud of a soda can obtained using MVSNet. The flow past the soda can has been previously investigated by Wang et al. [50], who conducted a thorough validation study at different Reynolds numbers using boundary-fitted and point cloud CFD. Here, we employ the proposed resampling method to enhance the quality of the point cloud, aiming for improved accuracy and stability. Specifically, our aim is to address the noise resulting from multiple point registrations generated by MVSNet and achieve a better point distribution in sparse regions. To evaluate the improvement in solution accuracy with resampled points, we select the flow past the soda can at $Re = 300$ and 5000, for which reference immersed and boundary-fitted results are available [50]. The computational settings, including the background mesh (IM3) and the boundary conditions, are identical to the study of Wang et al. [50].

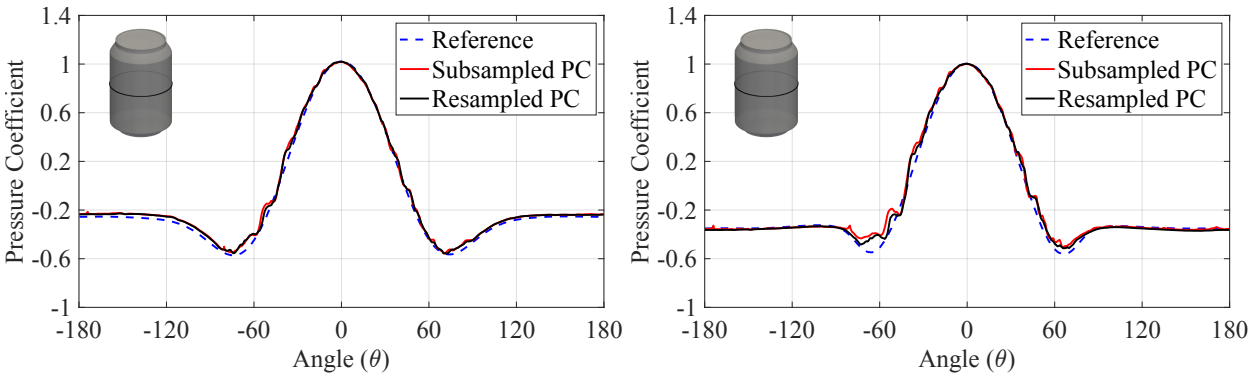
The original soda can point cloud generated by MVSNet in Wang et al. [50] comprises approximately 4.6 million points. The point density was considered unnecessarily large for analysis and was consequently subsampled using an octree-based method. Despite the subsampling procedure, the point cloud still suffered the issue of noise. Here, we utilize the mesh-driven resampling method to regularize the original soda can point cloud and generate essential points required for mesh IM3, resulting in a total of 685,786 resampled points. Figure 18 compares the subsampled and resampled point clouds, showing that the latter has a more uniform distribution with reduced noise levels observed in the cross-section views. The resampled point cloud provides a skin-layer point representation, which is advantageous for the immersed method as it provides a clear definition of the boundary interface and reduces the number of cut elements.

Figure 19 presents the distribution of pressure coefficients plotted around the circumference of



(a) (b) (c)

Figure 18: Comparison between the subsampled soda can point cloud from Wang et al. [50] (blue) and resampled point cloud used in this study (green) obtained from the original dataset for background mesh IM3. (a) Isometric view. (b) Central cross-section along the x-axis. (c) Central cross-section along the z-axis.

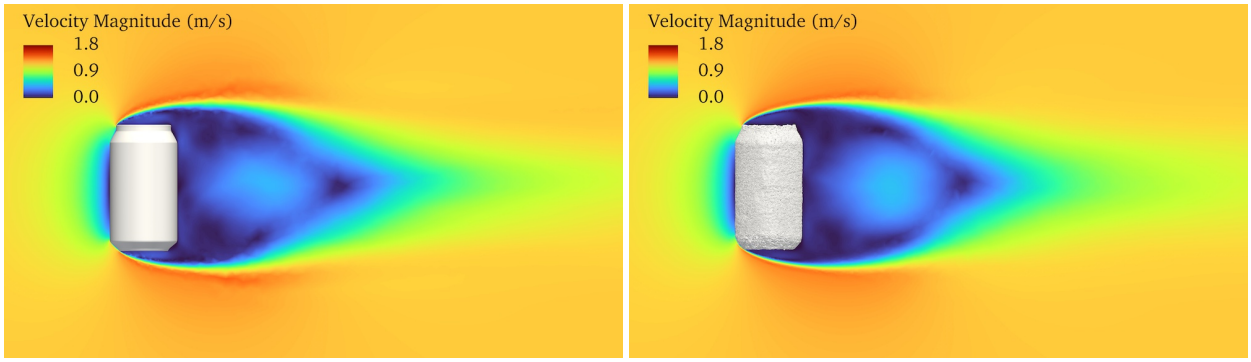


(a) Re = 300 (b) Re = 5000

Figure 19: Pressure coefficients along the soda can's circumference at the central cross-section in z direction obtained using resampled and subsampled point clouds (PCs) for Re = 300 and 5000. The reference solution is obtained using boundary-fitted CFD analysis of an idealized CAD model.

the soda can at the central cross-section for Re = 300 and 5000. The pressure coefficient obtained using the resampled point cloud is consistent with the reference results. Notably, employing the resampled point cloud leads to reduced oscillations compared to employing the subsampled point cloud. This is particularly evident near the $-60°$ and $60°$ regions, especially for Re = 5000. This improvement matches the results obtained using the resampled point cloud more closely with boundary-fitted analyses compared to the subsampled point cloud. Figure 20 compares the time-averaged velocity magnitude around the soda can, obtained with boundary-fitted analysis and point cloud CFD using resampled points for Re = 5000, showcasing excellent agreement between the two methods. Similarly, Figure 21 presents the time-averaged pressure projected on the point cloud, comparing it with the boundary-fitted CFD result.

It should be noted that geometric discrepancies exist between the point cloud representation



(a) Boundary-fitted CFD (b) Point cloud CFD

Figure 20: Time-averaged velocity magnitude at the central cross-section of the soda can obtained for $Re = 5000$.

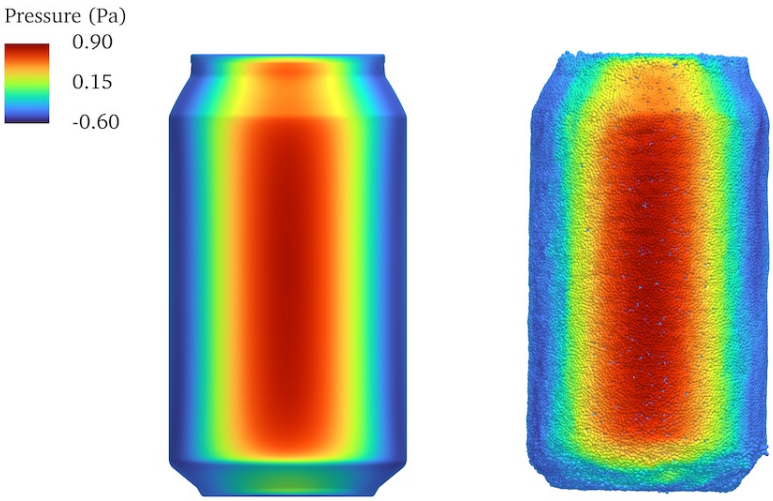


Figure 21: Front view of time-averaged pressure distribution on the soda can obtained for $Re = 5000$ using boundary-fitted (left) and point cloud (right) CFD.

of the soda can and the reference CAD model, causing minor differences between the solutions. However, the point cloud CFD yields results that closely match the boundary-fitted analysis. This highlights the robustness and adaptability of the proposed framework, particularly suitable for engineering problems associated with complex geometries where exact CAD information is difficult to obtain. In the next section, we delve into more challenging geometries where conducting boundary-fitted analysis may be infeasible. However, the combination of point cloud CFD with the proposed resampling method proves to be highly effective.

4. Applications

4.1. Flow past the Stanford dragon

The Stanford dragon serves as a complex computational geometry for flow analysis. The original mesh reconstruction of the dragon exhibits numerous geometric defects, such as holes,

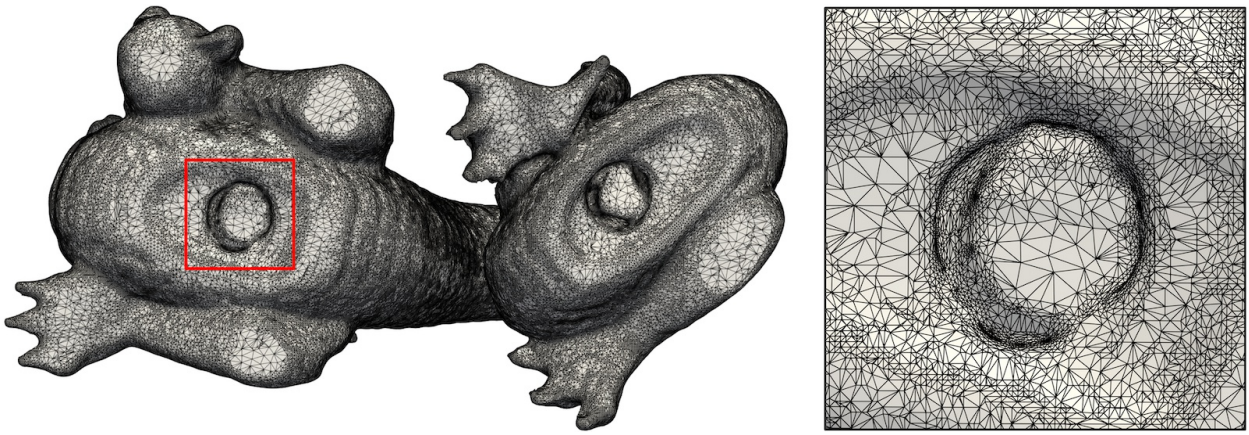


Figure 22: Bottom view of the surface mesh reconstruction of the Stanford dragon obtained from the repository [148]. The zoomed-in view around the mesh shows an irregular distribution of element sizes that can hinder boundary-fitted analysis.

self-intersection, and hanging surfaces [148]. Utilizing such geometry presents challenges for boundary-fitted analysis, which requires additional time-consuming and labor-intensive efforts to fix the inherent geometric defects. Moreover, the irregular distribution of element sizes, shown in Figure 22, could significantly impact the fluid mesh quality, resulting in skewed elements and inconsistent boundary layer resolution. Attempting to adjust the triangles to achieve a coarser or finer mesh poses additional challenges and complexities. Hence, generating a boundary-fitted volume mesh from such geometry is non-trivial, and conducting a meaningful analysis would be difficult. The point cloud CFD, on the other hand, presents an effective solution for handling such intricate geometries, simplifying the generation of watertight volume mesh with a non-conforming background mesh and representing the surface mesh with a point cloud. Leveraging the proposed resampling method, the surface representation can be precisely adjusted, either coarsened or refined, based on the background mesh. This flexibility is not attainable with a boundary-fitted method, as the reconstructed mesh may not be easily coarsened or refined to suit the simulation requirements.

Figure 23a illustrates the computational domain, the location of the dragon, and the boundary conditions used to simulate the flow past the dragon at $Re = 10^6$. The dragon's dimension – length, width, and height – serves as the basis for constructing the computational domain. The no-penetration boundary conditions are applied at the lateral walls strongly, while the no-slip condition on the surface of the dragon is enforced weakly. The outflow boundary is traction-free. Since standard units for these dimensions are unavailable in the literature, a non-dimensional study is conducted. For this purpose, the width of the dragon ($W = 0.0916$) is considered as the characteristic length. To simulate a flow at $Re = 10^6$, the inlet velocity and density are set to one, while the viscosity is set to $\mu = 9.16 \times 10^{-8}$. The computational domain is discretized into a background mesh shown in Figure 23b, consisting of 12,011,610 elements, with an element size of $0.005 \times W$

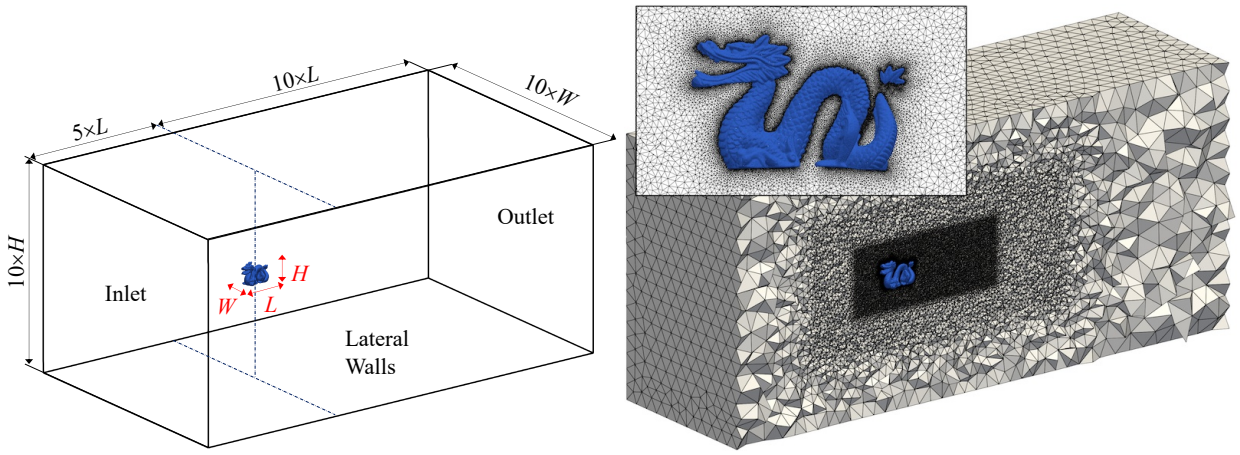


Figure 23: (a) Computational domain for the flow past the dragon. The dimensions of the domain are based on the height ($H = 0.144$), length ($L = 0.205$), and width ($W = 0.0916$) of the dragon. (b) Central cross-section view of the immersed mesh with a zoomed-in view of elements around the dragon.

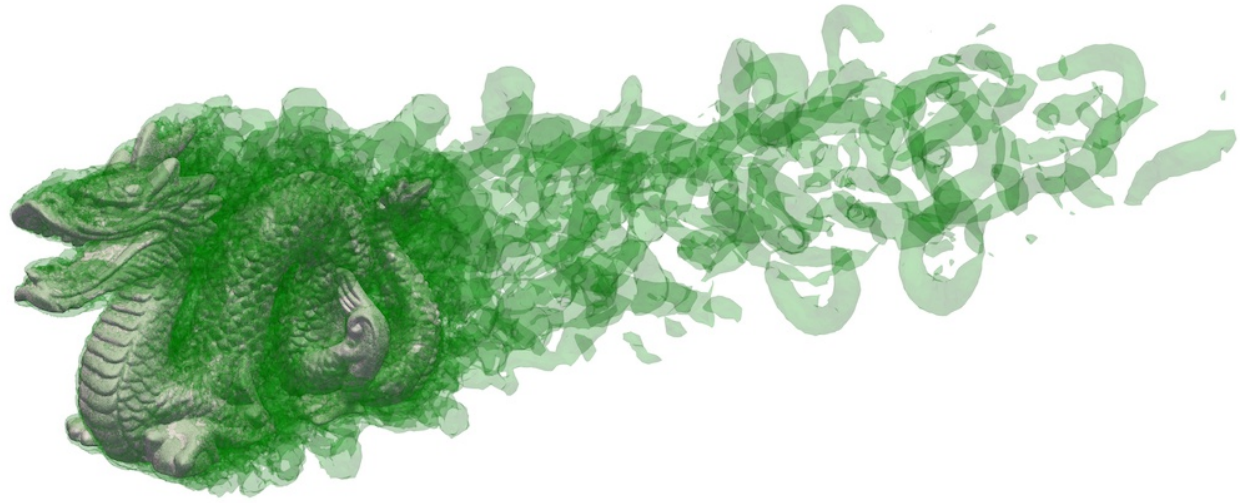
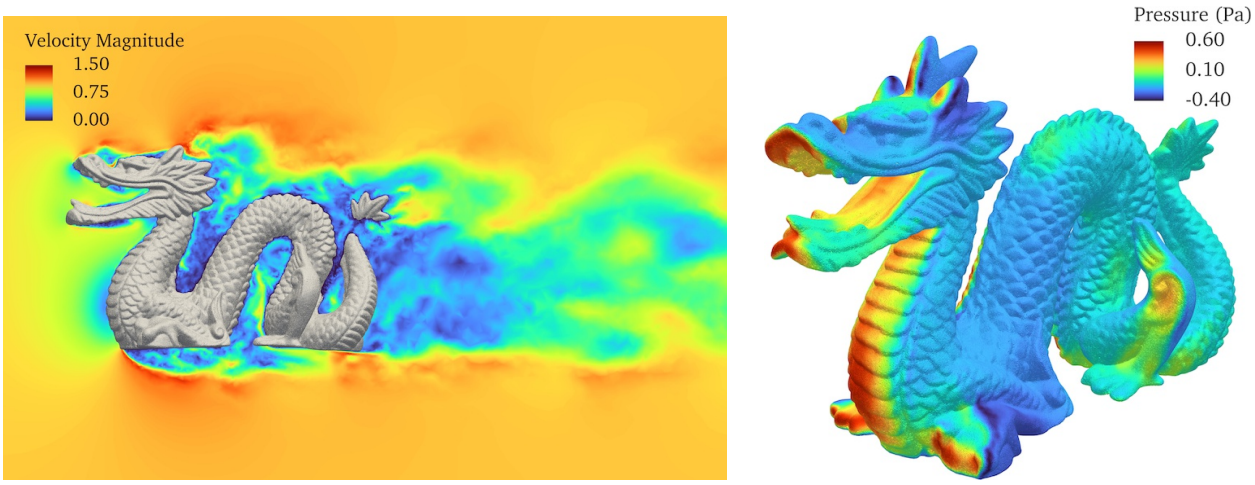


Figure 24: Isosurface of instantaneous Q-criterion for the flow past the dragon.

near the object boundary. The dragon point cloud is resampled for the background mesh, resulting in $1,212,294$ new sets of points. The accuracy of resampled points has already been demonstrated in Section 3. For the simulation, a time-step size of 1×10^{-3} is used.

Figure 24 presents a visualization of isosurfaces of instantaneous Q-criterion, clearly demonstrating the presence of significant wakes behind the dragon. Moreover, Figure 25a illustrates the instantaneous velocity magnitude along the central cross-section of the dragon, effectively capturing the sharp boundary layer around the intricate geometry. Figure 25b shows the instantaneous pressure projected on the dragon's point cloud. The pressure varies across the dragon's local features, particularly noticeable along the scales on the back and neck. Figure 26 demonstrates the time-averaged velocity magnitude and 2D streamlines along the central cross-section.



(a) (b)

Figure 25: (a) Instantaneous velocity magnitude along the central cross-section of the dragon. (b) Instantaneous pressure projected on the Stanford dragon's point cloud.

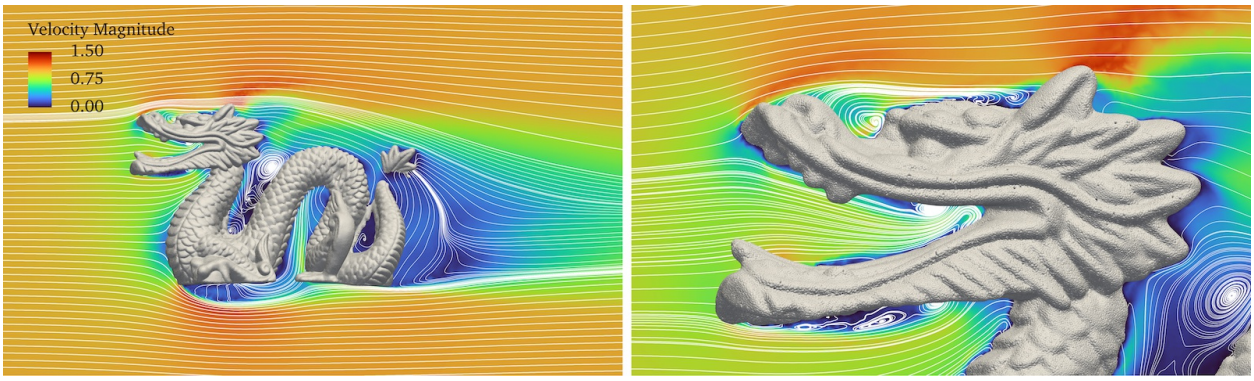


Figure 26: Streamlines projected on the central cross-section for the time-averaged velocity field of the flow past the dragon.

The zoomed-in view clearly demonstrates the complexity of the flow around the dragon's head, which has been captured well by the proposed framework. These comprehensive visualizations underscore the efficacy of our approach in accurately simulating flow phenomena around complex geometries without any labor-intensive and time-consuming efforts.

4.2. Flow past a scanned statue

In this section, we show the capability of the framework in resampling and directly simulating a scanned point cloud obtained in-house using photogrammetry. The photogrammetry reconstruction (Figure 1) is based on 108 photographic images captured using an iPhone 13 Pro from various angles around the statue located on the campus of Iowa State University. The reconstruction is processed with traditional MVS using the COLMAP library [9, 10]. The statue's metallic surface, which exhibits reflectivity, poses challenges for traditional MVS algorithms in accurately capturing depth information. Additionally, the statue contains internal regions that are only partially

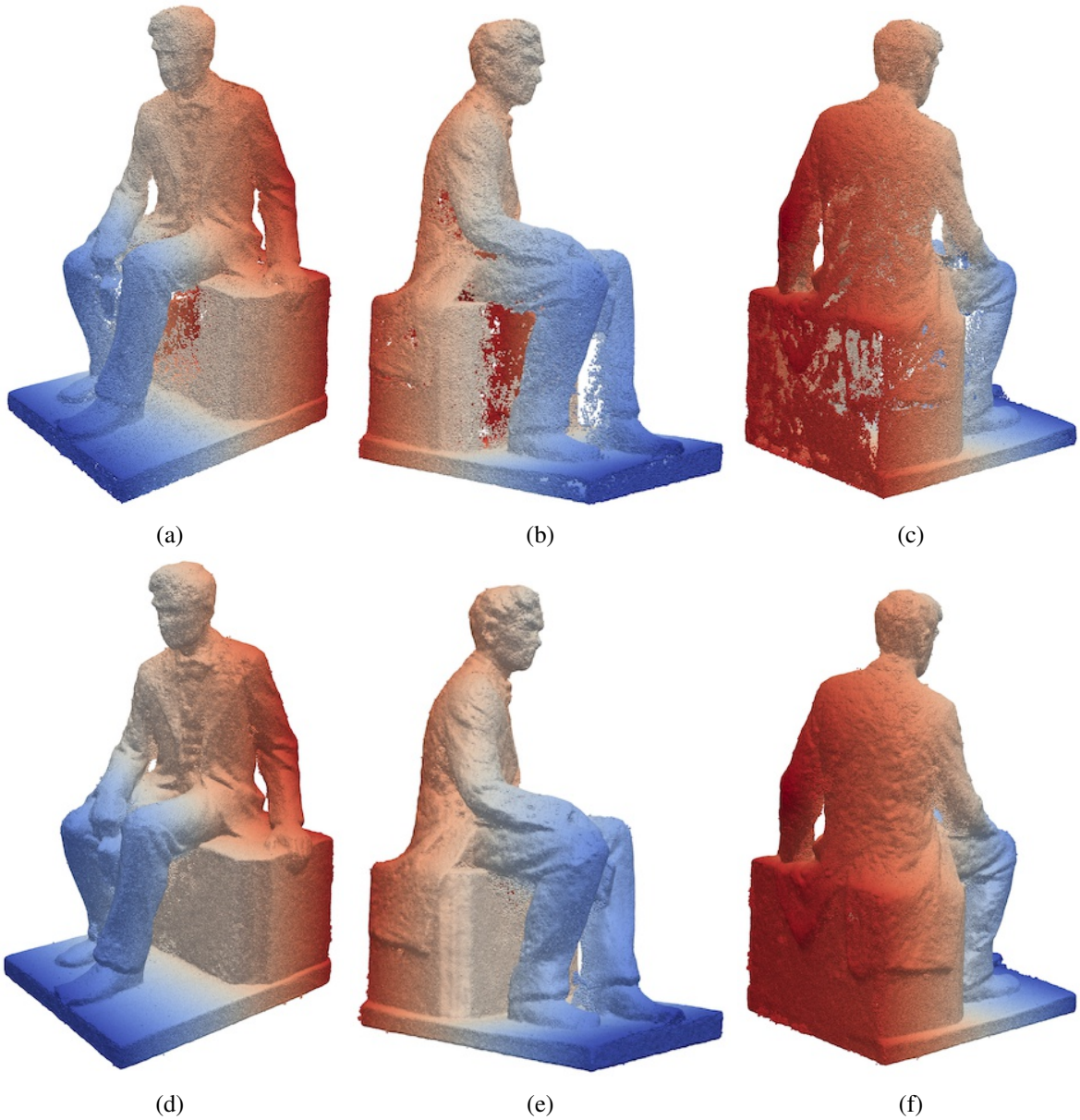
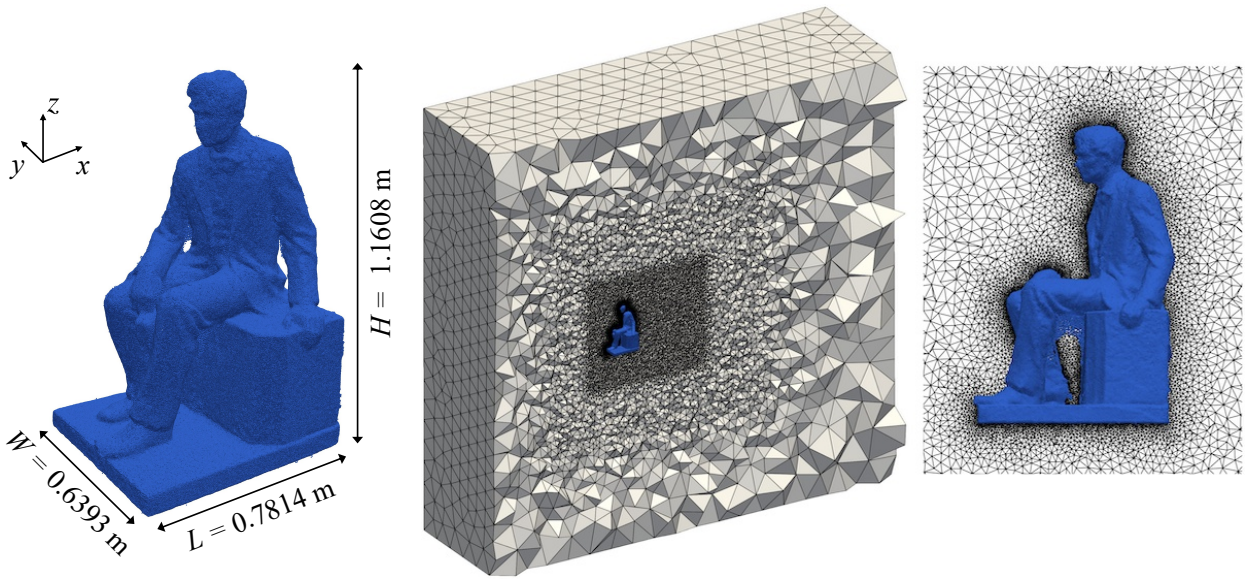


Figure 27: Different views of the (a)–(c) scanned point cloud of the statue showing sparse and incomplete regions and (d)–(f) resampled point cloud showing filled gaps and sparse regions. The color gradient, ranging from 0 (blue) to 1 (red), represents the normalized depth along the x-axis.

accessible for photography, leading to sparsity and incomplete regions within the obtained point cloud. Figures 27a, 27b, and 27c present the different views of the raw point cloud of the statue obtained after photogrammetry, comprising of 480,243 points. It highlights the insufficient point cloud density near the regions of weak textures, such as the seat and the legs. These deficiencies indicate that the geometry representation is not optimal and is not suitable for analysis.

We demonstrate that by applying the proposed resampling method, an analysis-suitable point cloud with a watertight-like representation can be obtained from the raw MVS output. First,



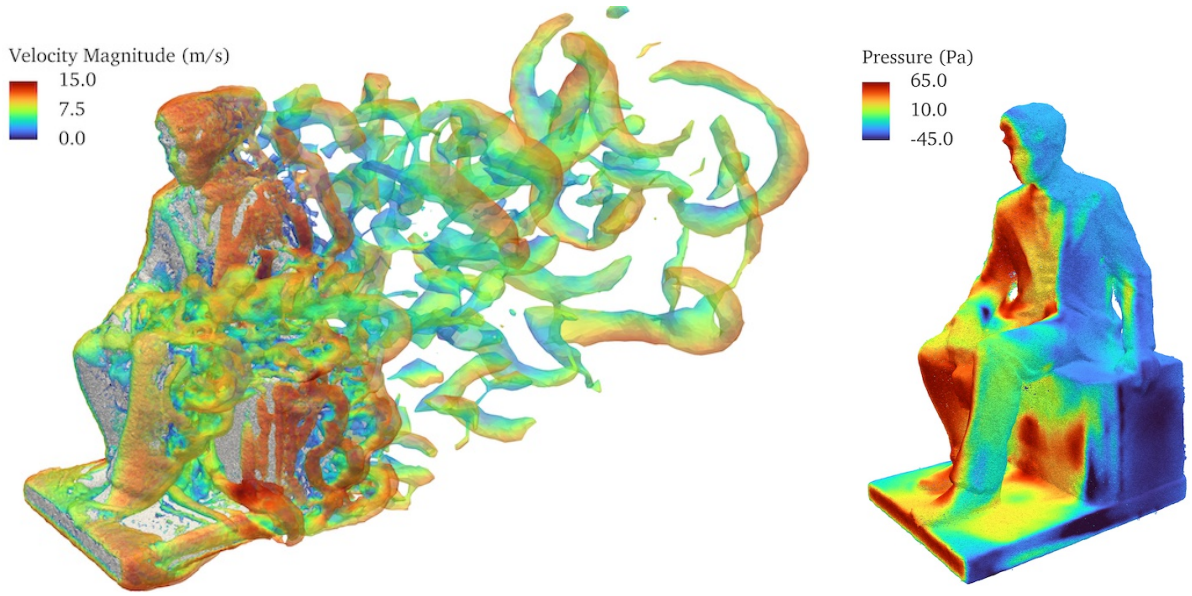
(a) (b)

Figure 28: (a) Resampled point cloud of the scanned statue. The object is enclosed within a domain of dimensions $(-0.4975, 0.2839) \times (-0.2611, 0.3782) \times (-0.2842, 0.8766)$ m^3, representing the length (L), width (W), and height (H) along the x, y, and z axes, respectively. (b) Central cross-section view of the immersed mesh with a zoomed-in view of elements around the statue.

we create a background mesh that is utilized both in the resampling process and for the subsequent CFD study. The point cloud is enclosed within a domain of dimensions $(-0.4975, 0.2839) \times (-0.2611, 0.3782) \times (-0.2842, 0.8766)$ m^3, representing the length (L), width (W), and height (H) along the x, y, and z axes, respectively. This domain serves as the basis for constructing the computational domain and mesh. With a near-boundary element size of $0.005 \times W$ and refinement regions designed to capture the wake, the generated background mesh consists of $7,307,977$ elements, as shown in Figure 28. After resampling, the resulting point cloud contains a total of 810,588 points, as illustrated in Figures 27d, 27e, and 27f. Regions previously afflicted by gaps and holes are now fully filled, resulting in a robust and analysis-ready resampled point cloud.

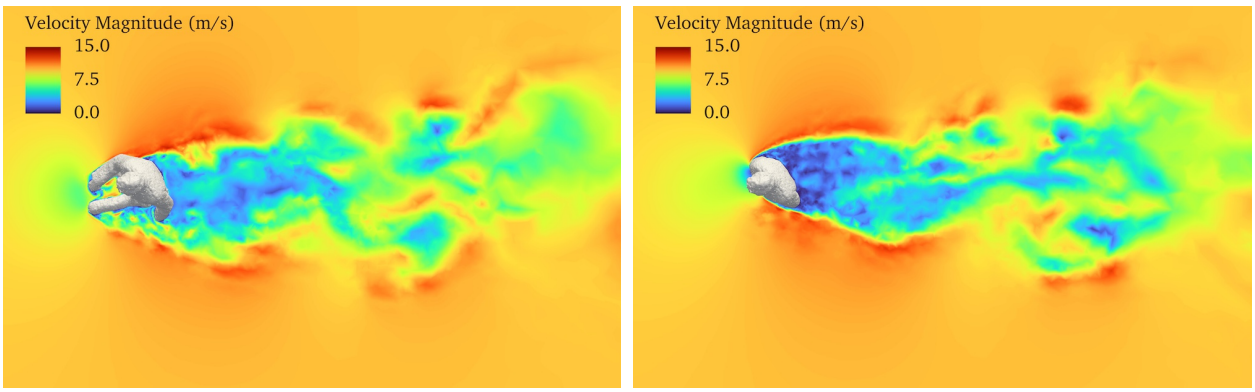
Next, we employ the resampled point cloud for direct flow analysis using the same background mesh. The resampling method guarantees sufficient points for every intersected element in the given mesh. For the flow analysis, we adopt boundary conditions similar to those used for the analysis of the dragon. The inlet velocity magnitude is set to 10 m/s, representing typical wind speed on a casual day, while the surface of the statue is enforced with no-slip boundary conditions. Density and viscosity are set to $\rho = 1.184$ kg/m^3 and $\mu = 1.845 \times 10^{-5}$ N·s/m^2 respectively. The width of the statue ($W = 0.64$ m) is considered as the characteristic length to obtain $Re = 4.1 \times 10^5$, indicating a turbulent flow regime. A time-step size of 1×10^{-3} s is used for the analysis.

Figure 29a provides a visualization of the Q-criterion isosurface, indicating the presence of a significant wake behind the statue resulting from the flow. Furthermore, Figure 29b illustrates the



(a) (b)

Figure 29: (a) Isosurface of instantaneous Q-criterion for the flow past the statue. (b) Time-averaged pressure projected on the statue's point cloud.



(a) z = 0.33 m (b) z = 0.5 m

Figure 30: Instantaneous velocity magnitude at different heights of the statue.

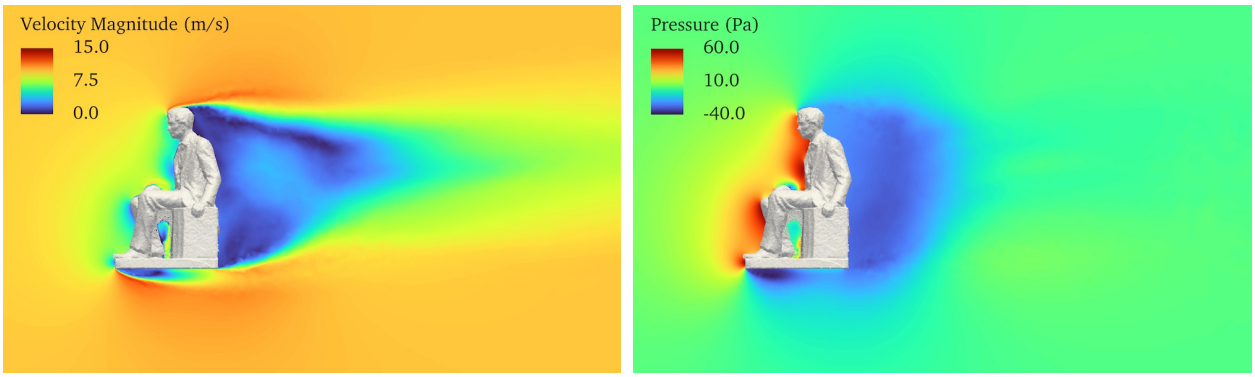


Figure 31: Time-averaged velocity magnitude (left) and pressure (right) at the central cross-section of the statue.

variation in pressure on the statue's geometry. Additionally, Figure 30 showcases the instantaneous velocity distribution at different heights. Notably, regions of incomplete point cloud coverage, such as the legs and seat of the statue, are successfully captured in the simulation, indicating the robustness of the resampling method in improving the point cloud quality for accurate flow analysis. The time-averaged velocity magnitude and pressure distribution are presented in Figure 31, highlighting wake regions and stagnation points around the statue. These observations underscore the effectiveness of the framework in capturing complex flow phenomena around the statue and obtaining quantities of interest on the point cloud essential for various engineering applications.

The resampling procedure has proven instrumental in transforming incomplete point cloud data, such as that obtained from MVS, into an analysis-ready representation that is suitable for CFD simulations. By effectively filling the gaps and addressing the incompleteness inherent in scanned geometries, the proposed NIMBUS framework enables meaningful results to be obtained where traditional approaches would have likely faltered. The ability to efficiently and accurately simulate such geometries holds significant promise, offering a cost-effective and time-efficient solution for a wide range of engineering applications.

5. Conclusions

In this paper, we present an innovative framework, NIMBUS, to generate an analysis-suitable point cloud representation from a raw output of photogrammetry and perform direct CFD analysis on it. The framework comprises several key stages: initial photogrammetry reconstruction for point cloud, point cloud processing to obtain necessary geometric quantities, mesh-driven resampling to fix geometric defects, and immersogeometric formulation to conduct flow analysis. The algorithm for the mesh-driven resampling method is discussed in detail, and its validation against geometries with various complexity and defects is conducted. We integrate ghost penalty stabilization into the immersed method to address stability and ill-conditioning issues with small cut elements. Our results demonstrate the capability to perform mesh refinement studies on point cloud-represented objects, even with intricate geometry such as the Stanford bunny, offering valuable benchmarking data for future investigations. Real-world point clouds with fine local features (Stanford dragon) and hole defects (scanned statue) are simulated using the proposed method to yield meaningful results. The proposed approach has unlocked the simulation of geometries previously deemed infeasible. With these enhancements, our framework emerges as a robust tool for conducting analysis directly on point cloud geometries, irrespective of complexity or defects, promising significant advancements in this field.

In the future, we plan to extend the proposed framework to address direct multiphysics analysis of objects represented by point cloud, particularly focusing on fluid–structure interaction using point-based approaches such as reproducing kernel particle methods (RKPM), smoothed particle

hydrodynamics (SPH), and peridynamics [160–168] for various industrial applications [169–172]. We also plan to incorporate higher-order background mesh capabilities with local refinements, such as T-splines [173] and truncated hierarchical B-splines [174–176], and explore cut-cell stabilization based on extended B-splines [177–179]. Finally, we will utilize recent novel view synthesis methods, such as NeRF [180] and Gaussian splatting [181], for reconstructing real-world objects and environments with higher fidelity and efficiency.

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.

Acknowledgments

This work was supported in part by the National Science Foundation under award number DMS-2436623. This support is gratefully acknowledged. We thank the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing high-performance computing resources that contributed to the results presented in this paper.

Appendix A. Validation of ghost penalty stabilization

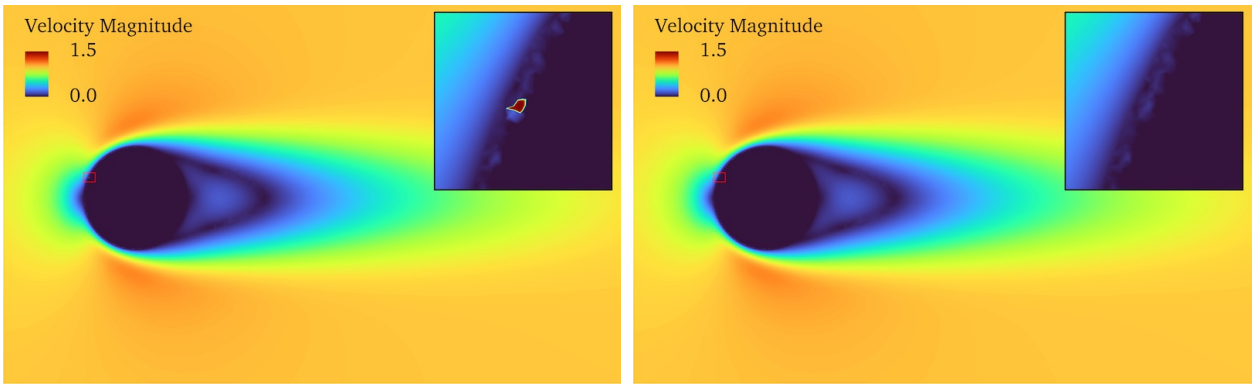
In this appendix, the implementation of ghost penalty stabilization for immersogeometric analysis is validated using a standard benchmark case of flow over a sphere at $Re = 100$. For this purpose, we utilize a similar setup and boundary conditions as employed by Xu et al. [36] and Balu et al. [49]. The sphere is initially sampled with 100,000 random points and then resampled using the proposed NIMBUS framework. A convergence study is conducted using three different background meshes, as detailed in Table A.1. The ghost penalty stabilization parameter $\alpha_{\mathbf{u}}^{\mathrm{GhP}} = \alpha_{p}^{\mathrm{GhP}} = 0.05$ is chosen based on the recommendations in Refs. [132–134]. Using this parameter value, the drag coefficient obtained for the sphere at $Re = 100$ is summarized in Table A.2, where the converged solution using mesh IM3 matches with the reference result. Finally, Figure A.1 demonstrates the velocity magnitude comparisons with and without ghost penalty stabilization. The implementation of the ghost penalty has effectively resolved the issue related to small cut elements, resulting in a stable solution. It should be noted that these issues could have a more pronounced impact on the solution, especially in complex geometries or when considering a higher level of adaptive quadrature, which captures very small volumes. In this study, the adaptive quadrature of level 2 was used for a valid comparison between the solutions obtained with and without ghost penalty. Using a higher level of adaptive quadrature led to solution divergence and failure

Table A.1: Element sizes in the background mesh for the flow over a sphere problem.

Mesh	Total number of elements	Near boundary	Inner refinement	Outer refinement	Outer box
IM1	345,294	0.02	0.2	$0.8/\sqrt{2}$	1.2
IM2	1,868,820	0.01	0.1	$0.4/\sqrt{2}$	1.0
IM3	8,066,240	0.005	0.05	$0.2/\sqrt{2}$	0.8

Table A.2: Drag coefficients obtained for flow over the sphere at $Re = 100$ with ghost penalty parameter $\alpha_{\mathbf{u}}^{\text{GhP}} = \alpha_{p}^{\text{GhP}} = 0.05$.

Mesh	C_D
IM1	1.129
IM2	1.096
IM3	1.093
Xu et al. [36]	1.093



(a) Without ghost penalty stabilization

(b) With ghost penalty stabilization

Figure A.1: Time-averaged velocity magnitude obtained for flow over the sphere at $Re = 100$ with and without ghost penalty stabilization. The zoomed-in view shows the small cut element issue, which has been resolved with the ghost penalty.

of analysis because of ill-conditioning without ghost penalty. However, with the ghost penalty, we were able to obtain consistent results even with a higher level of adaptive quadrature.

References

[1] J. Wang, D. Gu, Z. Yu, C. Tan, L. Zhou, A framework for 3D model reconstruction in reverse engineering, Computers & Industrial Engineering 63 (2012) 1189–1200.

[2] F. Bruno, S. Bruno, G. De Sensi, M.-L. Luchi, S. Mancuso, M. Muzzupappa, From 3D reconstruction to virtual reality: A complete methodology for digital archaeological exhibition, Journal of Cultural Heritage 11 (2010) 42–49.

[3] G. L. Zeng, Medical Image Reconstruction: A Conceptual Tutorial, Springer, Berlin, Heidelberg, 2010.

[4] B. D. Upadhyay, S. S. Sonigra, S. D. Daxini, Numerical analysis perspective in structural shape optimization: A review post 2000, Advances in Engineering Software 155 (2021) 102992.

[5] F. Pomerleau, F. Colas, R. Siegwart, A Review of Point Cloud Registration Algorithms for Mobile Robotics, Now Publishers, 2015.

[6] S. Reutebuch, H.-E. Andersen, R. Mcgaughey, Light detection and ranging (LiDAR): An emerging tool for multiple resource inventory, Journal of Forestry 103 (2005) 286–292.

[7] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, A. Hussain, A Survey on LiDAR Scanning Mechanisms, Electronics 9 (2020) 741.

[8] T. Bell, B. Li, S. Zhang, Structured light techniques and applications, in: Wiley Encyclopedia of Electrical and Electronics Engineering, John Wiley & Sons, Ltd, 2016, pp. 1–24.

[9] J. L. Schönberger, J.-M. Frahm, Structure-from-motion revisited, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4104–4113.

[10] J. L. Schönberger, E. Zheng, J.-M. Frahm, M. Pollefeys, Pixelwise view selection for unstructured multi-view stereo, in: Computer Vision–ECCV 2016, Springer International Publishing, Cham, 2016, pp. 501–518.

[11] W. Linder, Digital Photogrammetry, Springer, Berlin, Heidelberg, 2009.

[12] N. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer, Berlin, Heidelberg, 2002.

[13] P. R. Urech, M. A. Dissegna, C. Girot, A. Grêt-Regamey, Point cloud modeling as a bridge between landscape design and planning, Landscape and Urban Planning 203 (2020) 103903.

[14] C. Lucas, W. Bouten, Z. Koma, W. D. Kissling, A. C. Seijmonsbergen, Identification of linear vegetation elements in a rural landscape using LiDAR point clouds, Remote Sensing 11 (2019) 292.

[15] Y. Xu, X. Tong, U. Stilla, Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry, Automation in Construction 126 (2021) 103675.

[16] J. M. Jurado, J. L. Cárdenas, C. J. Ogayar, L. Ortega, F. R. Feito, Semantic segmentation of natural materials on a point cloud using spatial and multispectral features, Sensors 20 (2020) 2244.

[17] X. Chen, N. Ravikumar, Y. Xia, R. Attar, A. Diaz-Pinto, S. K. Piechnik, S. Neubauer, S. E. Petersen, A. F. Frangi, Shape registration with learned deformations for 3D shape reconstruction from sparse and incomplete point clouds, Medical Image Analysis 74 (2021) 102228.

[18] X. Wang, C. Demartino, Y. Narazaki, G. Monti, B. F. Spencer Jr, Rapid seismic risk assessment of bridges using UAV aerial photogrammetry, Engineering Structures 279 (2023) 115589.

[19] Y. Yao, Z. Luo, S. Li, T. Fang, L. Quan, MVSNet: Depth inference for unstructured multi-view stereo, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, Cham, 2018, pp. 767–783.

[20] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, P. Tan, Cascade cost volume for high-resolution multi-view stereo and stereo matching, in: Proceedings of the IEEE/CVF Conference on Computer Vision and

Pattern Recognition, Seattle, WA, USA, 2020, pp. 2495–2504.

[21] X. Li, C. Liu, Z. Wang, X. Xie, D. Li, L. Xu, Airborne LiDAR: state-of-the-art of system design, technology and application, Measurement Science and Technology 32 (2020) 032002.

[22] P. R. Urech, M. O. Mughal, C. Bartesaghi-Koc, A simulation-based design framework to iteratively analyze and shape urban landscapes using point cloud modeling, Computers, Environment and Urban Systems 91 (2022) 101731.

[23] M. Gouda, J. Mirza, J. Weiß, A. Ribeiro Castro, K. El-Basyouny, Octree-based point cloud simulation to assess the readiness of highway infrastructure for autonomous vehicles, Computer-Aided Civil and Infrastructure Engineering 36 (2021) 922–940.

[24] N. Korshunova, G. Alaimo, S. Hosseini, M. Carraturo, A. Reali, J. Niiranen, F. Auricchio, E. Rank, S. Kollmannsberger, Image-based numerical characterization and experimental validation of tensile behavior of octet-truss lattice structures, Additive Manufacturing 41 (2021) 101949.

[25] J. Qian, J. Lu, Point-cloud method for image-based biomechanical stress analysis, International Journal for Numerical Methods in Biomedical Engineering 27 (2011) 1493–1506.

[26] J. Lu, J. Qian, W. Han, Discrete gradient method in solid mechanics, International Journal for Numerical Methods in Engineering 74 (2008) 619–641.

[27] M. Berg, M. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, Springer, Berlin, Heidelberg, 2008.

[28] R. Rolin, E. Antaluca, J.-L. Batoz, F. Lamarque, M. Lejeune, From point cloud data to structural analysis through a geometrical hBIM-oriented model, Journal on Computing and Cultural Heritage 12 (2019) 9.

[29] H. Bouchiba, S. Santoso, J.-E. Deschaud, L. Rocha-Da-Silva, F. Goulette, T. Coupez, Computational fluid dynamics on 3D point set surfaces, Journal of Computational Physics: X 7 (2020) 100069.

[30] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Computer Methods in Applied Mechanics and Engineering 197 (2008) 3768–3782.

[31] E. Rank, M. Ruess, S. Kollmannsberger, D. Schillinger, A. Düster, Geometric modeling, isogeometric analysis and the finite cell method, Computer Methods in Applied Mechanics and Engineering 249-250 (2012) 104–115.

[32] D. Schillinger, M. Ruess, The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models, Archives of Computational Methods in Engineering 22 (2015) 391–455.

[33] S. Duczek, F. Duvigneau, U. Gabbert, The finite cell method for tetrahedral meshes, Finite Elements in Analysis and Design 121 (2016) 18–32.

[34] F. de Prenter, C. V. Verhoosel, G. J. van Zwieten, E. H. van Brummelen, Condition number analysis and preconditioning of the finite cell method, Computer Methods in Applied Mechanics and Engineering 316 (2017) 297–327.

[35] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, E. Rank, Integrating CAD and numerical

analysis: 'Dirty geometry' handling using the Finite Cell Method, Computer Methods in Applied Mechanics and Engineering 351 (2019) 808–835.

[36] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, M.-C. Hsu, The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries, Computers & Fluids 141 (2016) 135–154.

[37] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, A. Krishnamurthy, Direct immersogeometric fluid flow analysis using B-rep CAD models, Computer Aided Geometric Design 43 (2016) 143–158.

[38] C. Wang, F. Xu, M.-C. Hsu, A. Krishnamurthy, Rapid B-rep model preprocessing for immersogeometric analysis using analytic surfaces, Computer Aided Geometric Design 52–53 (2017) 190–204.

[39] S. Xu, F. Xu, A. Kommajosula, M.-C. Hsu, B. Ganapathysubramanian, Immersogeometric analysis of moving objects in incompressible flows, Computers & Fluids 189 (2019) 24–33.

[40] T. Hoang, C. V. Verhoosel, C.-Z. Qin, F. Auricchio, A. Reali, E. H. van Brummelen, Skeleton-stabilized immersogeometric analysis for incompressible viscous flow problems, Computer Methods in Applied Mechanics and Engineering 344 (2019) 421–450.

[41] F. Xu, Y. Bazilevs, M.-C. Hsu, Immersogeometric analysis of compressible flows with application to aerodynamic simulation of rotorcraft, Mathematical Models and Methods in Applied Sciences 29 (2019) 905–938.

[42] Q. Zhu, F. Xu, S. Xu, M.-C. Hsu, J. Yan, An immersogeometric formulation for free-surface flows with application to marine engineering problems, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112748.

[43] S. Xu, B. Gao, A. Lofquist, M. Fernando, M.-C. Hsu, H. Sundar, B. Ganapathysubramanian, An octree-based immersogeometric approach for modeling inertial migration of particles in channels, Computers & Fluids 214 (2021) 104764.

[44] F. Xu, C. Wang, K. Hong, Y. Liu, Immersogeometric thermal analysis of flows inside buildings with reconfigurable components, Journal of Thermal Analysis and Calorimetry 143 (2021) 4107–4117.

[45] K. Saurabh, B. Gao, M. Fernando, S. Xu, M. A. Khanwale, B. Khara, M.-C. Hsu, A. Krishnamurthy, H. Sundar, B. Ganapathysubramanian, Industrial scale Large Eddy Simulations with adaptive octree meshes using immersogeometric analysis, Computers & Mathematics with Applications 97 (2021) 28–44.

[46] J. E. Fromm, N. Wunsch, K. Maute, J. A. Evans, J.-S. Chen, Interpolation-based immersogeometric analysis methods for multi-material and multi-physics problems, Computational Mechanics (2024). https://doi.org/10.1007/s00466-024-02506-z.

[47] L. Kudela, S. Kollmannsberger, U. Almac, E. Rank, Direct structural analysis of domains defined by point clouds, Computer Methods in Applied Mechanics and Engineering 358 (2020) 112581.

[48] F. Hartmann, S. Kollmannsberger, Enforcing essential boundary conditions on domains defined by point clouds, Computers & Mathematics with Applications 113 (2022) 13–23.

[49] A. Balu, M. R. Rajanna, J. Khristy, F. Xu, A. Krishnamurthy, M.-C. Hsu, Direct immersogeometric

fluid flow and heat transfer analysis of objects represented by point clouds, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115742.

[50] X. Wang, M. Jaiswal, A. M. Corpuz, S. Paudel, A. Balu, A. Krishnamurthy, J. Yan, M.-C. Hsu, Photogrammetry-based computational fluid dynamics, Computer Methods in Applied Mechanics and Engineering 417 (2023) 116311.

[51] X. Guo, J. Xiao, Y. Wang, A survey on algorithms of hole filling in 3D surface reconstruction, The Visual Computer 34 (2018) 93–103.

[52] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, L. Xiao, A review of algorithms for filtering the 3D point cloud, Signal Processing: Image Communication 57 (2017) 103–112.

[53] R. A. Tabib, Y. V. Jadhav, S. Tegginkeri, K. Gani, C. Desai, U. Patil, U. Mudenagudi, Learning-based hole detection in 3D point cloud towards hole filling, Procedia Computer Science 171 (2020) 475–482.

[54] A. Jacobson, L. Kavan, O. Sorkine-Hornung, Robust inside-outside segmentation using generalized winding numbers, ACM Transactions on Graphics 32 (2013) 33.

[55] G. Barill, N. G. Dickson, R. Schmidt, D. I. W. Levin, A. Jacobson, Fast winding numbers for soups and clouds, ACM Transactions on Graphics 37 (2018) 43.

[56] E. Burman, Ghost penalty, Comptes Rendus Mathematique 348 (2010) 1217–1220.

[57] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, Applied Numerical Mathematics 62 (2012) 328–341.

[58] E. Burman, S. Claus, P. Hansbo, M. G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, International Journal for Numerical Methods in Engineering 104 (2015) 472–501.

[59] Y. Furukawa, C. Hernández, Multi-view stereo: A tutorial, Foundations and Trends in Computer Graphics and Vision 9 (2015) 1–148.

[60] N. Snavely, S. M. Seitz, R. Szeliski, Modeling the world from internet photo collections, International Journal of Computer Vision 80 (2008) 189–210.

[61] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2004) 91–110.

[62] M. Ji, J. Gall, H. Zheng, Y. Liu, L. Fang, SurfaceNet: An end-to-end 3D neural network for multiview stereopsis, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2307–2315.

[63] W. Hartmann, S. Galliani, M. Havlena, L. Van Gool, K. Schindler, Learned multi-patch similarity, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1586–1594.

[64] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, J.-B. Huang, DeepMVS: Learning multi-view stereopsis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2821–2830.

[65] R. Chen, S. Han, J. Xu, H. Su, Point-based multi-view stereo network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South), 2019, pp. 1538–1547.

[66] Y. Hou, J. Kannala, A. Solin, Multi-view stereo by temporal nonparametric fusion, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2651–2660.

[67] Y. Xue, J. Chen, W. Wan, Y. Huang, C. Yu, T. Li, J. Bao, MVSCRF: Learning multi-view stereo with conditional random fields, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 4312–4321.

[68] K. Luo, T. Guan, L. Ju, H. Huang, Y. Luo, P-MVSNet: Learning patch-wise matching confidence aggregation for multi-view stereo, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 10452–10461.

[69] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, L. Quan, Recurrent MVSNet for high-resolution multi-view stereo depth inference, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5525–5534.

[70] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering 197 (2007) 173–201.

[71] V. Varduhn, M.-C. Hsu, M. Ruess, D. Schillinger, The tetrahedral finite cell method: Higher-order immersogeometric analysis on adaptive non-boundary-fitted meshes, International Journal for Numerical Methods in Engineering 107 (2016) 1054–1079.

[72] T. E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, Computer Methods in Applied Mechanics and Engineering 190 (2000) 411–430.

[73] M.-C. Hsu, Y. Bazilevs, V. M. Calo, T. E. Tezduyar, T. J. R. Hughes, Improving stability of stabilized and multiscale formulations in flow simulations at small time steps, Computer Methods in Applied Mechanics and Engineering 199 (2010) 828–840.

[74] K. Takizawa, T. E. Tezduyar, Y. Otoguro, Stabilization and discontinuity-capturing parameters for space–time flow computations with finite element and isogeometric discretizations, Computational Mechanics 62 (2018) 1169–1186.

[75] D. Jia, M. Esmaily, A time-consistent stabilized finite element method for fluids with applications to hemodynamics., Scientific Reports 13 (2023) 19120.

[76] K. Takizawa, Y. Otoguro, T. E. Tezduyar, Variational multiscale method stabilization parameter calculated from the strain-rate tensor, Mathematical Models and Methods in Applied Sciences 33 (2023) 1661–1691.

[77] Y. Bazilevs, T. J. R. Hughes, Weak imposition of Dirichlet boundary conditions in fluid mechanics, Computers & Fluids 36 (2007) 12–26.

[78] T. Rüberg, F. Cirak, Subdivision-stabilised immersed b-spline finite elements for moving boundary flows, Computer Methods in Applied Mechanics and Engineering 209-212 (2012) 266–283.

[79] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, T. J. R. Hughes, An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves, Computer Methods in Applied Mechanics and Engineering 284 (2015) 1005–1053.

[80] A. Embar, J. Dolbow, I. Harari, Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements, International Journal for Numerical Methods in Engineering 83 (2010) 877–898.

[81] W. Jiang, C. Annavarapu, J. Dolbow, I. Harari, A robust Nitsche's formulation for interface problems with spline-based finite elements, International Journal for Numerical Methods in Engineering 104 (2015) 676–696.

[82] I. Harari, E. Grosu, A unified approach for embedded boundary conditions for fourth-order elliptic problems, International Journal for Numerical Methods in Engineering 104 (2015) 655–675.

[83] A. N. Brooks, T. J. R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1982) 199–259.

[84] T. E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Advances in Applied Mechanics 28 (1992) 1–44.

[85] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, J. B. Quincy, The variational multiscale method–A paradigm for computational mechanics, Computer Methods in Applied Mechanics and Engineering 166 (1998) 3–24.

[86] T. J. R. Hughes, G. Scovazzi, L. P. Franca, Multiscale and stabilized methods, in: E. Stein, R. de Borst, T. J. R. Hughes (Eds.), Encyclopedia of Computational Mechanics, Volume 3: Fluids, John Wiley & Sons, 2004.

[87] T. J. R. Hughes, G. Sangalli, Variational multiscale analysis: the fine-scale Green's function, projection, optimization, localization, and stabilized methods, SIAM Journal of Numerical Analysis 45 (2007) 539–557.

[88] T. J. R. Hughes, L. Mazzei, K. E. Jansen, Large eddy simulation and the variational multiscale method, Computing and Visualization in Science 3 (2000) 47–59.

[89] T. J. R. Hughes, L. Mazzei, A. A. Oberai, A. Wray, The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence, Physics of Fluids 13 (2001) 505–512.

[90] T. J. R. Hughes, A. A. Oberai, L. Mazzei, Large eddy simulation of turbulent channel flows by the variational multiscale method, Physics of Fluids 13 (2001) 1784–1799.

[91] A. Masud, R. Calderer, A variational multiscale method for incompressible turbulent flows: Bubble functions and fine scale fields, Computer Methods in Applied Mechanics and Engineering 200 (2011) 2577–2593.

[92] K. Takizawa, D. Montes, S. McIntyre, T. E. Tezduyar, Space–time VMS methods for modeling of incompressible flows at high Reynolds numbers, Mathematical Models and Methods in Applied

Sciences 23 (2013) 223–248.

[93] A. Masud, R. Calderer, Residual-based turbulence models for moving boundary flows: Hierarchical application of variational multiscale method and three-level scale separation, International Journal for Numerical Methods in Fluids 73 (2013) 284–305.

[94] Y. Bazilevs, J. Yan, M. de Stadler, S. Sarkar, Computation of the flow over a sphere at Re = 3700: A comparison of uniform and turbulent inflow conditions, Journal of Applied Mechanics 81 (2014) 121003.

[95] Y. Bazilevs, A. Korobenko, J. Yan, A. Pal, S. M. I. Gohari, S. Sarkar, ALE–VMS formulation for stratified turbulent incompressible flows with applications, Mathematical Models and Methods in Applied Sciences 25 (2015) 2349–2375.

[96] R. Calderer, L. Zhu, R. Gibson, A. Masud, Residual-based turbulence models and arbitrary Lagrangian–Eulerian framework for free surface flows, Mathematical Models and Methods in Applied Sciences 25 (2015) 2287–2317.

[97] L. Yang, S. Badia, R. Codina, A pseudo-compressible variational multiscale solver for turbulent incompressible flows, Computational Mechanics 58 (2016) 1051–1069.

[98] J. Yan, A. Korobenko, A. E. Tejada-Martínez, R. Golshan, Y. Bazilevs, A new variational multiscale formulation for stratified incompressible turbulent flows, Computers & Fluids 158 (2017) 150–156.

[99] A. Korobenko, Y. Bazilevs, K. Takizawa, T. E. Tezduyar, Computer modeling of wind turbines: 1. ALE-VMS and ST-VMS aerodynamic and FSI analysis, Archives of Computational Methods in Engineering 26 (2019) 1059–1099.

[100] S. Xu, N. Liu, J. Yan, Residual-based variational multi-scale modeling for particle-laden gravity currents over flat and triangular wavy terrains, Computers & Fluids 188 (2019) 114–124.

[101] L. Aydinbakar, K. Takizawa, T. E. Tezduyar, D. Matsuda, U-duct turbulent-flow computation with the ST-VMS method and isogeometric discretization, Computational Mechanics 67 (2021) 823–843.

[102] M. Ravensbergen, T. A. Helgedagsrud, Y. Bazilevs, A. Korobenko, A variational multiscale framework for atmospheric turbulent flows over complex environmental terrains, Computer Methods in Applied Mechanics and Engineering 368 (2020) 113182.

[103] Q. Zhu, J. Yan, A. E. Tejada-Martínez, Y. Bazilevs, Variational multiscale modeling of Langmuir turbulent boundary layers in shallow water using Isogeometric Analysis, Mechanics Research Communications 108 (2020) 103570.

[104] H. Cen, Q. Zhou, A. Korobenko, Simulation of stably stratified turbulent channel flow using residual-based variational multiscale method and isogeometric analysis, Computers & Fluids 214 (2021) 104765.

[105] L. Aydinbakar, K. Takizawa, T. E. Tezduyar, T. Kuraishi, Space–time VMS isogeometric analysis of the Taylor–Couette flow, Computational Mechanics 67 (2021) 1515–1541.

[106] Y. Bazilevs, C. Michler, V. M. Calo, T. J. R. Hughes, Weak Dirichlet boundary conditions for wall-bounded turbulent flows, Computer Methods in Applied Mechanics and Engineering 196 (2007)

4853–4862.

[107] Y. Bazilevs, C. Michler, V. M. Calo, T. J. R. Hughes, Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes, Computer Methods in Applied Mechanics and Engineering 199 (2010) 780–790.

[108] Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and the residual-based variational multiscale method, Journal of Computational Physics 229 (2010) 3402–3414.

[109] M.-C. Hsu, I. Akkerman, Y. Bazilevs, Wind turbine aerodynamics using ALE–VMS: Validation and the role of weakly enforced boundary conditions, Computational Mechanics 50 (2012) 499–511.

[110] M.-C. Hsu, I. Akkerman, Y. Bazilevs, Finite element simulation of wind turbine aerodynamics: Validation study using NREL Phase VI experiment, Wind Energy 17 (2014) 461–481.

[111] Y. Bazilevs, A. Korobenko, J. Yan, A. Pal, S. M. I. Gohari, S. Sarkar, ALE–VMS formulation for stratified turbulent incompressible flows with applications, Mathematical Models and Methods in Applied Sciences 25 (2015) 2349–2375.

[112] K. Takizawa, T. E. Tezduyar, T. Kuraishi, S. Tabata, H. Takagi, Computational thermo-fluid analysis of a disk brake, Computational Mechanics 57 (2016) 965–977.

[113] F. Xu, G. Moutsanidis, D. Kamensky, M.-C. Hsu, M. Murugan, A. Ghoshal, Y. Bazilevs, Compressible flows on moving domains: Stabilized methods, weakly enforced essential boundary conditions, sliding interfaces, and application to gas-turbine modeling, Computers & Fluids 158 (2017) 201–220.

[114] S. Xu, B. Gao, M.-C. Hsu, B. Ganapathysubramanian, A residual-based variational multiscale method with weak imposition of boundary conditions for buoyancy-driven flows, Computer Methods in Applied Mechanics and Engineering 352 (2019) 345–368.

[115] D. C. Wilcox, Turbulence Modeling for CFD, DCW Industries, La Canada, CA, 1998.

[116] R. Golshan, A. E. Tejada-Martínez, M. Juha, Y. Bazilevs, Large-eddy simulation with near-wall modeling using weakly enforced no-slip boundary conditions, Computers & Fluids 118 (2015) 172–181.

[117] F. de Prenter, C. V. Verhoosel, S. B. E. H. van Brummelen, M. G. Larson, Stability and conditioning of immersed finite element methods: Analysis and remedies, Archives of Computational Methods in Engineering 30 (2023) 3617–3656.

[118] F. de Prenter, C. Verhoosel, E. van Brummelen, Preconditioning immersed isogeometric finite element methods with application to flow problems, Computer Methods in Applied Mechanics and Engineering 348 (2019) 604–631.

[119] F. de Prenter, C. Verhoosel, H. Van Brummelen, J. Evans, C. Messe, J. Benzaken, K. Maute, Multigrid solvers for immersed finite element methods and immersed isogeometric analysis, Computational Mechanics 65 (2020) 807–838.

[120] E. Burman, A penalty-free nonsymmetric Nitsche-type method for the weak imposition of boundary conditions, SIAM Journal on Numerical Analysis 50 (2012) 1959–1981.

[121] T. Boiveau, E. Burman, A penalty-free Nitsche method for the weak imposition of boundary conditions in compressible and incompressible elasticity, IMA Journal of Numerical Analysis 36 (2015) 770–795.

[122] D. Schillinger, I. Harari, M.-C. Hsu, D. Kamensky, S. K. F. Stoter, Y. Yu, Y. Zhao, The non-symmetric Nitsche method for the parameter-free imposition of weak boundary and coupling conditions in immersed finite elements, Computer Methods in Applied Mechanics and Engineering 309 (2016) 625–652.

[123] A. Main, G. Scovazzi, The shifted boundary method for embedded domain computations. Part I: Poisson and Stokes problems, Journal of Computational Physics 372 (2018) 972–995.

[124] A. Main, G. Scovazzi, The shifted boundary method for embedded domain computations. Part II: Linear advection–diffusion and incompressible Navier–Stokes equations, Journal of Computational Physics 372 (2018) 996–1026.

[125] P. Antolín, A. Buffa, R. Puppi, X. Wei, Overlapping multipatch isogeometric method with minimal stabilization, SIAM Journal on Scientific Computing 43 (2021) A330–A354.

[126] D. Elfverson, M. G. Larson, K. Larsson, A new least squares stabilized Nitsche method for cut isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 349 (2019) 1–16.

[127] K. Larsson, S. Kollmannsberger, E. Rank, M. G. Larson, The finite cell method with least squares stabilized Nitsche boundary conditions, Computer Methods in Applied Mechanics and Engineering 393 (2022) 114792.

[128] C. Gürkan, A. Massing, A stabilized cut discontinuous Galerkin framework for elliptic boundary value and interface problems, Computer Methods in Applied Mechanics and Engineering 348 (2019) 466–499.

[129] S. K. F. Stoter, S. C. Divi, E. H. van Brummelen, M. G. Larson, F. de Prenter, C. V. Verhoosel, Critical time-step size analysis and mass scaling by ghost-penalty for immersogeometric explicit dynamics, Computer Methods in Applied Mechanics and Engineering 412 (2023) 116074.

[130] S. C. Divi, P. Zuijlen, T. Hoang, F. de Prenter, F. Auricchio, A. Reali, H. Van Brummelen, C. Verhoosel, Residual-based error estimation and adaptivity for stabilized immersed isogeometric analysis using truncated hierarchical B-splines, Journal of Mechanics 38 (2022) 204–237.

[131] A. Massing, M. Larson, A. Logg, M. Rognes, A stabilized Nitsche fictitious domain method for the Stokes problem, Journal of Scientific Computing 61 (2014) 604–628.

[132] B. Schott, W. Wall, A new face-oriented stabilized XFEM approach for 2D and 3D incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 276 (2014) 233–265.

[133] B. Schott, U. Rasthofer, V. Gravemeier, W. Wall, A face-oriented stabilized Nitsche-type extended variational multiscale method for incompressible two-phase flow, International Journal for Numerical Methods in Engineering 104 (2015) 721–748.

[134] B. Schott, S. Shahmiri, R. Kruse, W. Wall, A stabilized Nitsche-type extended embedding mesh

approach for 3D low- and high-Reynolds-number flows, International Journal for Numerical Methods in Fluids 82 (2016) 289–315.

[135] B. Liu, A Nitsche stabilized finite element method: Application for heat and mass transfer and fluid–structure interaction, Computer Methods in Applied Mechanics and Engineering 386 (2021) 114101.

[136] W. Dettmer, C. Kadapa, D. Perić, A stabilised immersed boundary method on hierarchical B-spline grids, Computer Methods in Applied Mechanics and Engineering 311 (2016) 415–437.

[137] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, Computer Aided Geometric Design 22 (2005) 121–146.

[138] F. Cazals, M. Pouget, Algorithm 889: Jet_fitting_3: A generic C++ package for estimating the differential properties on sampled surfaces via polynomial fitting, ACM Transactions on Mathematical Software 35 (2008) 24.

[139] G. Bendels, R. Schnabel, R. Klein, Detecting holes in point set surfaces, Journal of WSCG 14 (2006) 89–96.

[140] J. Wang, M. M. Oliveira, A hole-filling strategy for reconstruction of smooth surfaces in range images, in: 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), Sao Carlos, Brazil, 2003, pp. 11–18.

[141] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, ACM SIGGRAPH Computer Graphics 26 (1992) 71–78.

[142] H. Xie, K. T. McDonnell, H. Qin, Surface reconstruction of noisy and defective data sets, in: IEEE Visualization 2004, Austin, Texas, 2004, pp. 259–266.

[143] E. Franchini, S. Morigi, F. Sgallari, Implicit shape reconstruction of unorganized points using PDE-based deformable 3D manifolds, Numerical Mathematics: Theory, Methods and Applications 3 (2010) 405–430.

[144] G. Wyvill, C. McPheeters, B. Wyvill, Data structure for *soft* objects, The Visual Computer 2 (1986) 227–234.

[145] W. Lorensen, H. Cline, Marching cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer Graphics 21 (1987) 163–169.

[146] R. B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, Shanghai, China, 2011, pp. 1–4.

[147] C. Geuzaine, J.-F. Remacle, Gmsh: A 3D finite element mesh generator with built-in pre-and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (2009) 1309–1331.

[148] The Stanford 3D Scanning Repository, http://graphics.stanford.edu/data/3Dscanrep/, Accessed April 24, 2024.

[149] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIG-

GRAPH '96, Association for Computing Machinery, New York, USA, 1996, pp. 303–312.

[150] T. Jones, F. Durand, M. Desbrun, Non-iterative, feature-preserving mesh smoothing, ACM Transactions on Graphics 22 (2003) 943–949.

[151] P. O'Neil, T. Wanner, Analyzing the squared distance-to-measure gradient flow system with k-order voronoi diagrams, Discrete & Computational Geometry 61 (2019) 91–119.

[152] I. Tobor, P. Reuter, C. Schlick, Multi-scale reconstruction of implicit surfaces with attributes from large unorganized point sets, in: Proceedings Shape Modeling Applications, Genova, Italy, 2004, pp. 19–30.

[153] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, New York, USA, 1994, pp. 311–318.

[154] C.-H. Yang, K. Saurabh, G. Scovazzi, C. Canuto, A. Krishnamurthy, B. Ganapathysubramanian, Optimal surrogate boundary selection and scalability studies for the shifted boundary method on octree meshes, Computer Methods in Applied Mechanics and Engineering 419 (2024) 116686.

[155] C. Dick, J. Georgii, R. Westermann, A real-time multigrid finite hexahedra method for elasticity simulation using CUDA, Simulation Modelling Practice and Theory 19 (2011) 801–816.

[156] S. LeBlanc, P. Boyer, C. Joslin, Modelling and animation of impact and damage with smoothed particle hydrodynamics, The Visual Computer 30 (2014) 909–917.

[157] J. F. O'Brien, J. K. Hodgins, Animating fracture, Communications of the ACM 43 (2000) 68–75.

[158] C. Landsberg, A. Voigt, A multigrid finite element method for reaction-diffusion systems on surfaces, Computing and Visualization in Science 13 (2010) 177–185.

[159] H. Edelsbrunner, Alpha shapes – a survey, in: R. van de Weygaert, G. Vegter, J. Ritzerveld, V. Icke (Eds.), Tessellations in the Sciences: Virtues, Techniques and Applications of Geometric Tilings, Springer, 2011.

[160] J.-S. Chen, C. Pan, C.-T. Wu, W. K. Liu, Reproducing Kernel Particle Methods for large deformation analysis of non-linear structures, Computer Methods in Applied Mechanics and Engineering 139 (1996) 195–227.

[161] J.-S. Chen, M. Hillman, S.-W. Chi, Meshfree methods: Progress made after 20 years, Journal of Engineering Mechanics 143 (2017) 04017001.

[162] Y. Bazilevs, G. Moutsanidis, J. Bueno, K. Kamran, D. Kamensky, M. C. Hillman, H. Gomez, J. S. Chen, A new formulation for air-blast fluid–structure interaction using an immersed approach: part II—coupling of IGA and meshfree discretizations, Computational Mechanics 60 (2017) 101–116.

[163] G. Moutsanidis, D. Kamensky, J. S. Chen, Y. Bazilevs, Hyperbolic phase field modeling of brittle fracture: Part II–immersed IGA–RKPM coupling for air-blast–structure interaction, Journal of the Mechanics and Physics of Solids 121 (2018) 114–132.

[164] M. Behzadinasab, G. Moutsanidis, N. Trask, J. T. Foster, Y. Bazilevs, Coupling of IGA and peridynamics for air-blast fluid–structure interaction using an immersed approach, Forces in Mechanics 4

(2021) 100045.

[165] M. Behzadinasab, M. Hillman, Y. Bazilevs, IGA–PD penalty-based coupling for immersed air-blast fluid–structure interaction: a simple and effective solution for fracture and fragmentation, Journal of Mechanics 37 (2021) 680–692.

[166] M. D'Elia, X. Li, P. Seleson, X. Tian, Y. Yu, A review of local-to-nonlocal coupling methods in nonlocal diffusion and nonlocal mechanics, Journal of Peridynamics and Nonlocal Modeling 4 (2022) 1–50.

[167] M. N. Rahimi, G. Moutsanidis, An SPH-based FSI framework for phase-field modeling of brittle fracture under extreme hydrodynamic events, Engineering with Computers 39 (2023) 2365–2399.

[168] M. N. Rahimi, G. Moutsanidis, IGA–SPH: coupling isogeometric analysis with smoothed particle hydrodynamics for air-blast–structure interaction, Engineering with Computers (2024). https://doi.org/10.1007/s00366-024-01978-0.

[169] F. Xu, E. L. Johnson, C. Wang, A. Jafari, C.-H. Yang, M. S. Sacks, A. Krishnamurthy, M.-C. Hsu, Computational investigation of left ventricular hemodynamics following bioprosthetic aortic and mitral valve replacement, Mechanics Research Communications 112 (2021) 103604.

[170] E. L. Johnson, M. C. H. Wu, F. Xu, N. M. Wiese, M. R. Rajanna, A. J. Herrema, B. Ganapathysubramanian, T. J. R. Hughes, M. S. Sacks, M.-C. Hsu, Thinner biological tissues induce leaflet flutter in aortic heart valve replacements, Proceedings of the National Academy of Sciences 117 (2020) 19007–19016.

[171] M. R. Rajanna, M. Jaiswal, E. L. Johnson, N. Liu, A. Korobenko, Y. Bazilevs, J. Lua, N. Phan, M.-C. Hsu, Fluid–structure interaction modeling with nonmatching interface discretizations for compressible flow problems: simulating aircraft tail buffeting, Computational Mechanics (2024). https://doi.org/10.1007/s00466-023-02436-2.

[172] X. G. Pan, A. M. Corpuz, M. R. Rajanna, E. L. Johnson, Parameterization, algorithmic modeling, and fluid–structure interaction analysis for generative design of transcatheter aortic valves, Engineering with Computers (2024). https://doi.org/10.1007/s00366-024-01973-5.

[173] H. Casquero, L. Liu, C. Bona-Casas, Y. Zhang, H. Gomez, A hybrid variational-collocation immersed method for fluid–structure interaction using unstructured T-splines, International Journal for Numerical Methods in Engineering 105 (2016) 855–880.

[174] S. C. Divi, C. V. Verhoosel, F. Auricchio, A. Reali, E. H. van Brummelen, Topology-preserving scan-based immersed isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 392 (2022) 114648.

[175] S. C. Divi, P. H. van Zuijlen, T. Hoang, F. de Prenter, F. Auricchio, A. Reali, E. H. van Brummelen, C. V. Verhoosel, Residual-based error estimation and adaptivity for stabilized immersed isogeometric analysis using truncated hierarchical B-splines, Journal of Mechanics 38 (2022) 204–237.

[176] A. Moola, A. Balu, A. Krishnamurthy, A. Pawar, THB-Diff: a GPU-accelerated differentiable programming framework for THB-splines, Engineering with Computers (2024). https://doi.org/10.1007/

s00366-023-01929-1.

[177] T. Rüberg, F. Cirak, A fixed-grid b-spline finite element technique for fluid–structure interaction, International Journal for Numerical Methods in Fluids 74 (2014) 623–660.

[178] T. Rüberg, F. Cirak, J. M. García Aznar, An unstructured immersed finite element method for non-linear solid mechanics., Advanced Modeling and Simulation in Engineering Sciences 3 (2016) 22.

[179] E. Febrianto, J. Šístek, P. Kůs, M. Kecman, F. Cirak, A three-grid high-order immersed finite element method for the analysis of CAD models, Computer-Aided Design 173 (2024) 103730.

[180] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, R. Ng, NeRF: Representing scenes as neural radiance fields for view synthesis, Communications of the ACM 65 (2022) 99–106.

[181] B. Kerbl, G. Kopanas, T. Leimkuehler, G. Drettakis, 3D Gaussian splatting for real-time radiance field rendering, ACM Transactions on Graphics 42 (2023) 139.