NESI: Shape Representation via Neural Explicit Surface Intersection

CONGYI ZHANG, University of British Columbia, Canada
JINFAN YANG, University of British Columbia, Canada
ERIC HEDLIN, University of British Columbia, Canada
SUZURAN TAKIKAWA, University of British Columbia, Canada
NICHOLAS VINING, NVIDIA, Canada and University of British Columbia, Canada
KWANG MOO YI, University of British Columbia, Canada
WENPING WANG, Texas A&M University, USA
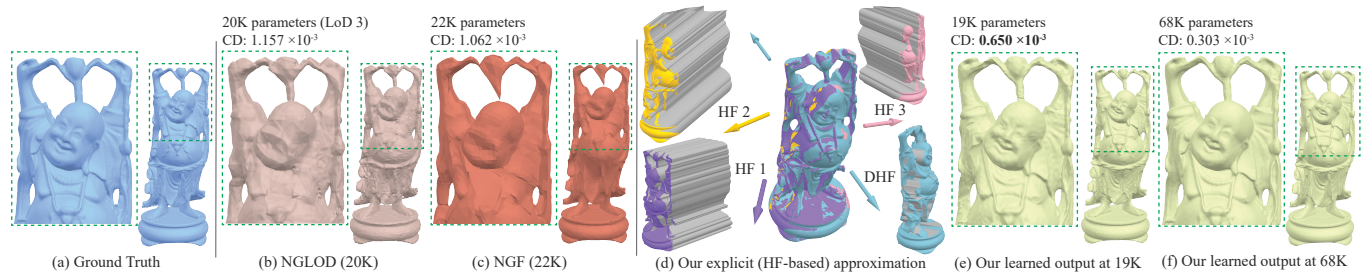ALLA SHEFFER, University of British Columbia, Canada

Fig. 1. NESI compactly represents detailed 3D shapes (a, 109K faces) as intersections of neural explict height-field surfaces (HFs) (d-f). It compactly encodes a set of differently oriented double (DHF) and single HFs (d) whose intersection well approximates the input. Given the same parameter budget, NESI provides more accurate approximations (e) of the inputs than state of the art alternatives: (b) NGLOD [Takikawa et al. 2021], (c) NGF [Sivaram et al. 2024]. At higher parameter counts (f) our results are visually indisitinguishable from the input.

Compressed representations of 3D shapes that are compact, accurate, and can be processed efficiently directly in compressed form, are extremely useful for digital media applications. Recent approaches in this space focus on learned *implicit* or *parametric* representations. While implicits are well suited for tasks such as in-out queries, they lack natural 2D parameterization, complicating tasks such as texture or normal mapping. Conversely, parametric representations support the latter tasks but are ill-suited for occupancy queries. We propose a novel learned alternative to these approaches, based on intersections of localized *explicit*, or *height-field*, surfaces. Since explicits can be trivially expressed both implicitly and parametrically, NESI directly supports a wider range of processing operations than implicit alternatives, including occupancy queries and parametric access. We represent input shapes using a collection of differently oriented height-field bounded half-spaces combined using volumetric Boolean intersections. We first tightly bound each input using a pair of oppositely oriented height-fields, forming a *Double Height-Field (DHF) Hull*. We refine this hull by intersecting it with additional

localized height-fields (HFs) that capture surface regions in its interior. We minimize the number of HFs necessary to accurately capture each input and compactly encode both the DHF hull and the local HFs as neural functions defined over subdomains of $\mathbb{R}^2$. This reduced dimensionality encoding delivers high-quality compact approximations. Given similar parameter count, or storage capacity, NESI significantly reduces approximation error compared to the state of the art, especially at lower parameter counts.

1 Introduction

Shape representations which support efficient geometry manipulation and processing, while also being accurate and compact, are of major interest for applications such as video games, 3D content streaming, and VR/AR [Karis et al. 2021]. Popular traditional representations include implicit surfaces, piecewise explicit representations, and piecewise parametric surfaces (B-Reps) [Botsch et al. 2010; Cohen-Or et al. 2015; Farin 2002; Marsh 2005], each with pros and cons. Implicits support in-out queries but cannot easily be parameterized, and thus do not directly support important geometry processing tasks such as texture mapping. Piecewise parametric or piecewise explicit surface representations, including meshes, can be effectively used for many geometry processing operations [Botsch et al. 2010; Cohen-Or et al. 2015], but do not directly support in-out queries. In general, traditional representations are far from compact,

Authors' Contact Information: Congyi Zhang, University of British Columbia, Canada; Jinfan Yang, University of British Columbia, Canada; Eric Hedlin, University of British Columbia, Canada; Suzuran Takikawa, University of British Columbia, Canada; Nicholas Vining, NVIDIA, Canada and University of British Columbia, Canada; Kwang Moo Yi, University of British Columbia, Canada; Wenping Wang, Texas A&M University, USA; Alla Sheffer, University of British Columbia, Canada.
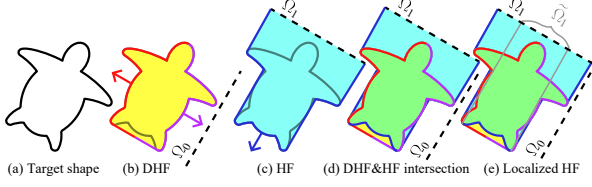
Fig. 2. A shape (a) represented (d) as the intersection (green) of a DHF hull (b) and an additional HF (c). Localizing the HF to a narrower parameter domain Ω (e), by implicitly assuming it to match the DHF elsewhere, reduces representation redundancy.

and require large numbers of parameters, or degrees of freedom, to capture detailed shapes; this has motivated the recent quest for more compact neural alternatives (Sec. 2). State-of-the-art neural implicit [Sitzmann et al. 2020b; Takikawa et al. 2022a, 2021, 2023] or parametric [Morreale et al. 2022, 2021a; Sivaram et al. 2024] shape representations provide a compact alternative to traditional representations, and can accurately encode highly detailed shapes using much fewer parameters. However, they inherit the processing limitations of their traditional counterparts: neural implicits do not support operations that require local or global surface parameterization, such as meshing and texture mapping, while neural parametric surfaces do not support occupancy queries. We propose a novel shape representation which is more compact than existing alternatives, and supports both fast in-out queries and processing tasks that leverage parameter domain information, such as texture or normal mapping.

We achieve this goal by leveraging the representational power of explicit, or *height-field* (HF), surfaces. We recall that an explicit or height-field (HF) surface is defined as the graph of a function $z = f(x, y)$ over a 2D domain $\Omega \in \mathbb{R}^2$ (see inset) and has an explicit parameterization relative to this domain (i.e. $P(x, y) = (x, y, f(x, y))$. Moreover, HF surfaces partition space into inside (light blue) and outside (white) *half-spaces*: a point $(x, y, z) \in \mathbb{R}^3$ is inside the half-space, or volume, $E(f)$, associated with the HF surface f if and only if

$(x, y) \in \Omega$ and $f(x, y) > z$.
As such, explicit HF-based representations combine the processing advantages of implicit and parametric ones. They are also inherently more compact than general parametric or implicit representations due to dimensionality reduction: all one must store are the $z = f(x, y)$ values over their parameter domain Ω. However, the range of shapes representable by a single HF surface is highly limited, as an HF can only represent a surface with a single z value for each (x, y). Piecewise explicit surface [Guskov et al. 2000; Maggiordomo et al. 2023] or volume [Muntoni et al. 2018, 2019; Yang et al. 2020] representations, which approximate surfaces or volumes using unions of HF surface patches or bounded half-spaces, are no longer compact and cannot robustly support in-out queries (Sec. 2).

We extend the advantages of explicit representations to generic shapes by observing that even extremely complex shapes can be accurately approximated using a Boolean **intersection** of just a few judiciously selected overlapping HF half-spaces; for instance,

the *happy buddha* (Fig. 1d) can be accurately approximated by intersecting just five such half-spaces. Moreover, using intersecting half-spaces as a shape representation allows for robust and efficient in-out queries *and* trivial surface parameterization. We refer to this HF intersection based representation as *Explicit Surface Intersection*, or *ESI*. We further note, importantly, that an intersection of HF half-spaces can be compactly encoded in neural form by taking advantage of the fact that each HF is simply defined by a function $z = f(x, y)$ over a 2D domain; encoding HFs in this manner produces a *Neural* Explicit Surface Intersection, or *NESI*, representations.

While early attempts at representing shapes using HF intersections [Richter and Roth 2018; Shade et al. 1998] use a large set of fixed, shape-independent, HF half-space orientations, or axis directions, they frequently fail to approximate large portions of the input surfaces (see Sec. 2, Fig. 5). In contrast, we compute a minimal set of best-approximating HF axes per input, achieving high approximation quality with just a handful of HFs (Sec 4, Fig. 5). This ability to accurately represent diverse geometries using a small handful of HFs is key to our shape representation.

Given an input shape defined via a mesh or other standard representation, we tightly bound it using a pair of oppositely oriented HF half-spaces that jointly define a *Double HF (DHF) hull* of the input shape (Fig. 2b, Fig. 1d - blue). We refine this hull by intersecting it with additional HF half-spaces that capture input surface regions lying inside the hull's interior (Fig. 2c, Fig. 1d - yellow, purple, pink). We optimize the choice of DHF and HF axis directions to minimize approximation error, while still keeping the number of HFs used as small as possible. Performing this optimization via brute-force search makes the problem intractable, as even the evaluation of the approximation quality of a single ESI is highly time consuming. We make the problem tractable via a combination of pre-computation and a branch-and-bound discrete optimization strategy that quickly rejects direction candidates to arrive at an optimal solution with minimal HF count. We avoid representational redundancy, where multiple explicits describe the same areas on the input shape (e.g. turtle arms in Fig. 2c), by only storing HF surface geometry in areas where it is not already adequately described by other explicits (Fig. 2e). This *localization* process reduces the geometric complexity of each HF, and thus the number of parameters required to encode it. Using our scheme, the vast majority of shapes in commonly used 3D shape databases [Koch et al. 2019; Zhou and Jacobson 2016] can be accurately represented using a DHF hull plus one to three additional HFs, with many shapes accurately represented using their DHF hull alone (31% of objects tested in our experiments; Sec. 7). We encode our DHFs and HFs using SIREN Multi-Layer Perceptron (MLP) architecture [Sitzmann et al. 2020b], as it does not require positional encoding [Tancik et al. 2020] and yet is able to encode both high- and low-frequency shape details. Our experiments (Sec. 6, 7) demonstrate that NESI allows for straightforward surface parameterization, enabling texturing (Fig. 9) and other similar tasks, and supports efficient and accurate in-out queries performed by following the sequence of local intersection operations; the latter are used to ray-trace all of our outputs throughout the paper.

We thoroughly validate the effectiveness of our method by evaluating ESI accuracy across 320 inputs, and by learning NESI representations of 100 diverse representative shapes using four different
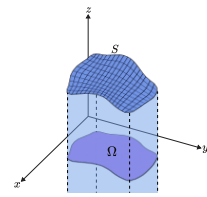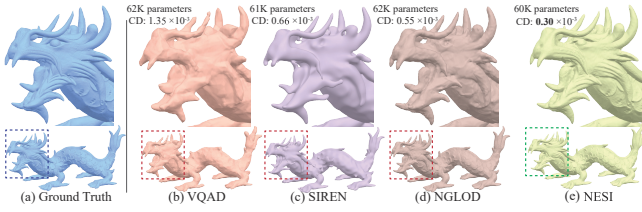
Fig. 3. NESI approximations (e) of complex inputs (a) are much more detailed and accurate than those generated by leading implicit alternatives: (b) [Takikawa et al. 2022a], (c) SIREN [Sitzmann et al. 2020b], and (d) [Takikawa et al. 2021], despite using fewer parameters.
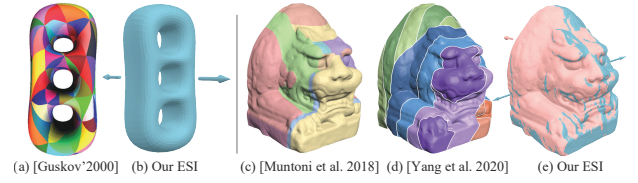

Fig. 4. Comparing ESIs vs prior explicit based representations: (a) [Guskov et al. 2000] use 98 explicit defined over a base mesh to represent the three-hole torus, (b) we use a single DHF. Shape partitions into single [Muntoni et al. 2018] height field explicits (c), and double height-field explicits [Yang et al. 2020] (d) use much larger numbers of explcits than ESI (e).

parameter counts for each shape. We compare our results to those generated by leading alternatives using same or higher parameter counts. On average our outputs are 30% more accurate than those produced by the best-performing alternative (NGF [Sivaram et al. 2024]) using the same or lower parameter counts, with improvement most pronounced at lower parameter counts. 86% of our learned outputs more accurately approximate the input ground truth shapes than those produced by this alternative, and our largest error across all inputs tested is only one third of theirs.

2 Background and Related Work

A vast body of previous work exists on 3D shape representations, each with their pros and cons. Here, we focus on representations closest to NESI in terms of goals or properties.

Traditional Shape Representations. *Parametric*, or boundary (B-Rep) representations, including polygonal meshes, define the bounding surfaces of closed 3D shapes using collections of parametric patches connected together along common boundary seams [Botsch et al. 2010; Farin 2002]. These representations were designed for surface-based tasks such as texturing, meshing, or (re)parameterization, and rasterization. To support computing in-out queries using these representations for tasks such as raytracing or collision detection, those representations require non-trivial intersection computations which are typically accelerated by building auxiliary data structures, increasing their memory footprint. Accurately approximating input shapes using either meshes or piecewise smooth parametric patches requires large patch and parameter counts [Botsch et al. 2010; Li et al. 2006; Litke et al. 2001; Luebke 2001]. Mesh compression schemes target compact mesh storage and transmission, and require decompressing the outputs prior to actual use [Alliez 2005; Maglo et al. 2015]. Geometry Images [Carr et al. 2006; Gu et al. 2002; Sander et al. 2003] compress meshes via 2D parameterization; they retain the inherent limitations of parametric representations and require large amounts of atlas space to obtain quality approximations. Our NESI representation combines the advantages of parametric representations with fast in-out queries, has a much smaller memory footprint, and can be processed directly in its compressed form. In Fig 6 we approximate the Buddha mesh with 120K triangles using just 48k parameters, producing a visually practically identical render (the chamfer distance between our model and the input mesh is 0.28).

Implicit representations (e.g. [Blinn 1982; Osher and Fedkiw 2005; Wyvill et al. 1998]) define a closed surface S as a level set of a function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$. Implicits support efficient in-out queries [Jones et al. 2006; Takikawa et al. 2022b] but are difficult to parameterize either globally or locally [Schmidt et al. 2006], making them challenging to texture, mesh, or normal map. Converting generic surfaces into analytic implicit form remains an open problem [Buonamici et al. 2018]; the commonly-used grid-based representations of implicits (e.g. [Museth 2013; Williams et al. 2024]) are highly memory consuming.

Explicit Surfaces. Classical *explicit*, or height-field (HF), surfaces are defined as height functions $z = F(x, y)$, $(x, y) \in \Omega$ over a 2D domain $\Omega \in \mathbb{R}^2$ [Farin 2002; Marsh 2005]. In the general case, the parameter domain can lie in any plane in $\mathbb{R}^3$, and the height z represents the offset or distance from this plane along the plane's normal, or *axis*. HFs can be viewed as a special case of parametric surfaces and trivially support parameterization-based tasks. Since few shapes can be described by a single explicit surface, numerous attempts had been made to describe shapes using combinations of multiple HFs.

Approximating existing shapes using piecewise HFs defined over polygonal domains [Guskov et al. 2000; Khodakovsky et al. 2000; Novák and Dachsbacher 2012] significantly reduces the memory footprint of a shape relative to a standard mesh representation, and facilitates efficient rendering using displacement maps [Maggiordomo et al. 2023; Thonat et al. 2021]. Accurate piecewise explicit approximation of complex shapes requires a large number of patches and is far from compact. For instance, [Guskov et al. 2000] uses 98 patches to approximate the three-holed torus (Fig 4a)., whereas we approximate it using a single DHF hull (Fig 4b); [Novák and Dachsbacher 2012] use hundreds of patches to represent the dragon in Fig. 3, which we approximate using one DHF and 3 HFs.

By defining the "inside" of a height-field $z = f(x, y)$ as the *volume* between the parameter domain and the surface, explicits can also be viewed as a special case of occupancy function implicits [Mescheder et al. 2019]: points (x, y, z) are inside the shape if and only if $(x, y) \in \Omega$ and $z \in [0, f(x, y)]$ (placing the parameter domain at $z = -\infty$ partitions $\mathbb{R}^3$ into inside and outside half-spaces). Several fabrication methods partition shapes into explicit volumes bounded by either a height-field and its parameter domain [Fekete and Mitchell 2001; Gao et al. 2015; Herholz et al. 2015; Hu et al. 2014;

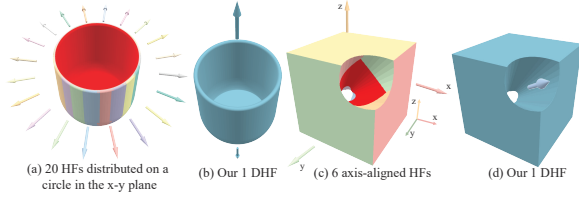(a) 20 HFs distributed on a circle in the x-y plane   (b) Our 1 DHF   (c) 6 axis-aligned HFs   (d) Our 1 DHF

Fig. 5. Depth fusion methods (a,c) use intersections of multiple depth maps with fixed, input independent axis directions (visualized by arrows) to approximate input shapes. These methods often fail to approximate large parts of the input surfaces (red in a and c) not visible along these axes. (a) 20 axis directions evenly distributed on a circle in the $x - y$ plane [Shade et al. 1998]; (c) 6 directions aligned with the +/- axes of the object's coordinate system [Richter and Roth 2018]. (b,d) ESI accurately captures both shapes using a single DHF with automatically computed optimal axis.

Muntoni et al. 2018, 2019] (Fig. 4c), or by pairs of oppositely oriented height-fields [Alderighi et al. 2021; Yang et al. 2020] (Fig. 4d). Both piecewise and volumetric explicit representations are far from compact, requiring high block counts for quality approximation; for the example Muntoni et al. [2018] require 13 explicit volumes and [Yang et al. 2020] requires 12 to approximate the lion statue in the inset; we accurately approximate this input with one DHF hull and one HF (Fig. 4d). More importantly, while partition-based representations are theoretically suitable for in-out queries, in practice assessing if a point is inside a union of non-overlapping blocks leads to false negatives for points next to boundaries between the different volumes, even deep inside the original shape. The likelihood of such catastrophic failures increases when the individual explicit volume geometries are compressed. By representing shapes as intersections of explicits, rather than unions, we drastically reduce the number of explicits required to accurately represent general shapes and sidestep the need to handle gaps and floating point issues along internal boundaries.

Our representation is inspired by depth fusion approaches for shape reconstruction [Curless and Levoy 1996; Richter and Roth 2018; Turk and Levoy 1994] and representation [Richter and Roth 2018; Shade et al. 1998] (Fig. 5ac). These methods define shapes as intersections of differently oriented depth maps or height-fields. The key difference between these approaches and ours is the choice of HF orientations, or axis directions. Depth fusion methods rely on large sets of *input independent* depth map axis directions. Reconstruction methods such as [Curless and Levoy 1996; Turk and Levoy 1994] use input camera views as directions. Others rely on a fixed set of axis directions: e.g. [Shade et al. 1998] uses 20 directions evenly distributed on a circle in the *xy* plane (Fig. 5a), while [Richter and Roth 2018] uses the positive and negative axes of the standard Euclidean coordinate system (Fig. 5c). As [Richter and Roth 2018] acknowledges, this approach often fails to capture large portions of input shape surfaces. To address this challenge, they use 5 layers of depth maps, placed one inside the other (forming a "matryoshka"); effectively, this means their method requires 30 (5x6) depth maps. The key distinction between these works and NESI is our use of HF axes that are *optimized per-input* so as to maximize approximation quality (Fig 5bd). Finding these optimal directions

efficiently requires solving a complex combinatorial optimization problem across a large potential solution space (Sec 4). This optimization based approach allows us to approximate 3D shapes with much higher accuracy, while using significantly fewer HFs overall. We require only 1 DHF for the examples in Fig 5; on average we use 1 DHF and fewer than 2 HFs to well approximate the 320 inputs tested (Sec 7).

Relief texture mapping [Baboud and Décoret 2006; Oliveira et al. 2000] is a rendering method that adds surface detail to low-resolution meshes during the rendering process without performing displacement mapping. It achieves this goal by applying heightmaps from fixed, orthogonal directions to a low resolution mesh at a per-pixel level, effectively augmenting standard texture maps with per-texel depth, as such they share some similarity with the reconstruction methods above, and specifically [Richter and Roth 2018] in their use of 6 axis-aligned heightmap directions. Relief texture mapping methods only attempt to simulate the illusion of detailed surface geometry during rendering, and do not attempt to represent it; as such, they solve a problem distinct from ours. We compare our computed axis selection against the use of orthogonal axis directions (similar to those used by relief texture mapping and [Richter and Roth 2018]) in Sec. 7.

Neural Shape Representations. Recent research efforts have attempted to encode many of the representations above using neural networks. Learning meshes or general explicit/parametric B-Rep/patch-based representations is known to be challenging due to their topological irregularity [Hanocka et al. 2019; Maron et al. 2017] and the need to ensure continuity across inter-patch boundaries [Groueix et al. 2018]. AtlasNet [Groueix et al. 2018] and its followups [Bednarik et al. 2020; Deng et al. 2020; Deprelle et al. 2022, 2019] represent surfaces using disconnected partially overlapping patches. Since this representation is not watertight, it cannot be reliably used for in-out queries. Yang et al. [2023] rely on largely manually created patch layouts to learn closed B-reps of input shapes. Both families of methods require megabytes of storage (20 for [Bednarik et al. 2020; Deng et al. 2020] and 5 for Yang et al.) to accurately represent input shapes. We achieve higher accuracy using filesizes under 280 kilobytes (Sec 7). Moreover, NESI is computed fully automatically and robustly supports in-out queries.

Neural Surface Maps [Morreale et al. 2021b] represents surfaces as learned geometry images; follow-up work [Morreale et al. 2022] learn geometry images and a series of patch-based displacements. Both methods suffer from the same issues as classical geometry images, most notably lack of support for in-out queries and a requirement that the input surface be cut so that it is homeomorphic to the unit disk. Neural displacement methods [Chen et al. 2023a; Sivaram et al. 2024] (Fig 1c) represent surfaces as a simplified coarse meshes overlaid with a neurally encoded displacement map; like other displacement map based representations, they do not directly support in-out queries, and require a sufficiently dense base mesh to capture topological details. Since extreme mesh simplification can be challenging for inputs with complex topology, these methods may fail or introduce severe visual artifacts at low parameter counts (Sec. 7). We outperform the state-of-the-art methods in this category [Morreale et al. 2022; Sivaram et al. 2024] by notable margins (Sec. 7).

NESI remains robust across all 400+ inputs and parameter count combinations tested.

Many methods address learning of compact neural *implicit* shape representations, including occupancy maps [Mescheder et al. 2019] and Signed Distance Functions (SDF) [Chen and Akleman 1999; Müller et al. 2022]. Recent efforts include learning compact neural implicit functions with a variety of internal representations [Chen et al. 2023b; Chen and Zhang 2019; Davies et al. 2021; Li et al. 2022; Mescheder et al. 2019; Park et al. 2019; Sitzmann et al. 2020a,b; Yifan et al. 2022], or focusing on adaptive multiresolution hierarchies, combining sparse hierarchical grids with neural networks [Takikawa et al. 2022a, 2021]. Yifan et al. [2022] represent shapes as a combination of an implicit SDF and a height, or displacement map. While drastically more efficient than naive storage, these methods inherit the limitations of traditional implicits when it comes to parameterization-driven processing tasks such as texturing or geodesic computation. Sec. 7 compares NESI to representative recent works in this category [Chen et al. 2023b; Li et al. 2022; Sitzmann et al. 2020b; Sivaram et al. 2024; Takikawa et al. 2022a, 2021; Yifan et al. 2022]; our method outperforms the best performing implicit-based alternative (SIREN w/o eikonal loss) on 93% of the input shape and parameter count combinations tested, and improves accuracy (measured using L_1 Chamfer distance) by a factor of 2.6.

[Richter and Roth 2018] propose a neural encoding of their depth map grid based shape representation. They use 30 (5x6) depth maps to encode each shape. Their reliance on grids (256^3 in their implementation) limits the accuracy of their outputs. NESI representation is based on precise HF/DHF intersection, and uses smooth 2D basis functions to encode the individual explicits, facilitating a much higher degree of accuracy. See Sec. 7 for additional comparisons.

Lastly, several methods focus on compact neural representation of specific classes of shapes, e.g. CAD models [Lin et al. 2022; Yu et al. 2023]. NESI is not class specific, and as demonstrated in Sec. 7 it compactly and accurately represents both organic and CAD shapes.

3 Overview

3.1 Definitions.

Our shape representation centers around two types of *volumetric explicits (VEs)*: closed *double height field (DHF) hulls* (Fig 2b) and open half-spaces defined by single *height-fields (HFs)* (Fig 2c). We define both DHFs and HFs with respect to their local *x-y-z* coordinate systems, as follows. Let Ω denote a 2D bounded domain in the *x-y* plane, and let $f_a(x, y)$ and $f_b(x, y)$ be two piecewise continuous functions defined over Ω, with $f_a(x, y) \geq f_b(x, y), \forall (x, y) \in \Omega$; then the DHF hull is defined by $DHF = \{(x, y, z) | f_a(x, y) \geq z \geq f_b(x, y) \wedge (x, y) \in \Omega\}$. Here $f_a(x, y)$ and $f_b(x, y)$ are called the *bounding functions* of the DHF. Similarly, let Ω denote a 2D bounded domain in the *x-y* plane, and let $f(x, y)$ be a piecewise continuous function defined over Ω. We define $HF = \{(x, y, z) | z \leq f(x, y) \wedge (x, y) \in \Omega\}$. Here $f(x, y)$ is the *bounding function* or *height function* of the HF.

When a DHF hull or an HF is assigned a general orientation, we associate it with its local coordinate system. In the notation below we use a 3D unit direction vector **d** to denote the *z*-axis of the local coordinate system and call **d** the *axis* of the corresponding VE. By definition, a DHF hull is always closed, or bounded. In contrast, a

HF is only half-bounded since it is unbounded in the direction of $-\mathbf{d}$. By these definitions, a DHF can be also viewed as the intersection of a pair of HFs with parallel but opposite axis directions.

3.2 ESIs and NESIs.

Our analytic *Explicit Surface Intersection (ESI)* and learned *Neural Explicit Surface Intersection (NESI)* representations use the intersection of one DHF hull $DHF(\mathbf{d}_0)$ and zero or more HFs $HF_k(\mathbf{d}_k)$ ($k = 1, 2, \ldots, m$) to approximate a given 3D object. More specifically, let S denote the set of points occupied by a given 3D shape; we then approximate S by the set $\tilde{S} = DHF(\mathbf{d}_0) \cap_{k=1}^{m} HF_k(\mathbf{d}_k)$. By construction, the DHF hull provides a closed and tight bounding volume of S. Intersecting the DHF hull with the HFs further tightens this bounding volume to achieve an accurate approximation of the input. The key to our method is the observation that the vast majority of shapes can be well approximated using the intersection of one DHF and a very small number of HFs, with judiciously selected coordinate system axes.

Computing ESIs. Acting on the observation above, we propose an effective and efficient algorithm for computing the combination of a DHF hull and an as-small-as-possible number of HFs whose intersection accurately approximates the input shape (Sec. 4). As our evaluation (Sec. 7) demonstrates, on average one DHF and two HFs are amply sufficient to well approximate typical geometric shapes, and many shapes (31% in our experiments) can be well approximated using a single DHF.

3.3 Computing and Utilizing NESIs.

We convert ESIs into a neural form by learning neural representations of the individual volumetric explicits (Sec. 5). We minimize the size of the learned DHF representations by leveraging the relation between their two bounding functions, and reduce the size of the individual HF encodings by only learning their surface shape in areas not well-represented by the DHF or other HFs. Finally, we propose efficient algorithms for performing common geometry processing tasks directly on the learned NESI representations (Sec. 6). NESI's support for real-time in-out query computation enables fast ray-tracing, collision detection, and other similar tasks; at the same time, explicit surface parameterization of the individual VEs enables other tasks such as texturing and meshing.

Extension to Occluded Surfaces. In our target applications, such as video games or VR/AR immersion, the viewer is typically located outside of rendered shapes and does not see content that is not well visible from outside; our core method targets this setting and implicitly prioritizes approximation of visible surface areas. We efficiently extend our method to approximate shapes containing fully or partially occluded surfaces by using unions of volumetric NESI explicits $\tilde{S}$ (see Fig. 22, Appendix C). Unless specifically indicated otherwise, all results and measurements reported and shown in the paper are generated *without* this extension.

4 ESI Computation

Given an input 3D shape S, represented using a triangular mesh, we seek to approximate it as an intersection of DHF hull and zero
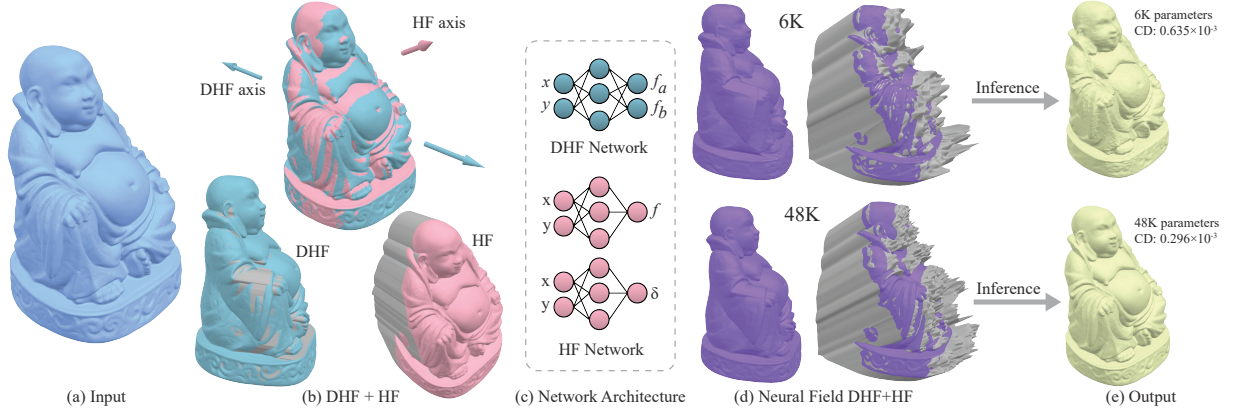
Fig. 6. Given an input 3D shape (a), we compute a DHF hull and HFs whose intersection accurately approximates the input (b). We then employ MLPs (c) to encode the DHF and HFs as $R^2 \rightarrow R$ functions (on the HF only purple areas are inside the $\tilde{\Omega}$ parameter domain) (d). At inference time we combine (intersect) the MLP outputs (e).

or more HFs. Since processing time and memory footprint both increase with the number of HFs, we aim to keep this number as small as possible, while maximizing approximation quality.

Our approximations must satisfy two properties: *volumetric approximation* and *surface coverage*. The former property requires the intersection of the volumetric explicits we use to closely overlap the input shape S, and is critical for reliable in-out queries. The surface coverage property requires the bounding functions of the participating volumetric explicits to jointly cover the surface. This property enables bijective piece-wise parametric representation of the outer surface, or shell, of S.

To satisfy these properties, for an HF or DHF with a known z-axis $\mathbf{d}_k$, we position its x-y plane just under the bounding sphere of S along the axis direction (at the sphere-axis intersection) and define the domain Ω of the VE in this plane as the 2D region bounded by the silhouette of the object viewed along the axis direction. For the DHF, denoted by $DHF(\mathbf{d}_0)$, we define its bounding functions $f_a(x,y)$ (inset red) and $f_b(x,y)$ (inset purple), over the domain Ω_0, as the two depth maps of the shell S when viewed along the $-\mathbf{d}_0$ and $\mathbf{d}_0$ directions, respectively. For any given HF, denoted by $HF_k(\mathbf{d}_k)$, we define its bounding function $f_k(x,y)$ (inset blue) over the domain Ω_k as the depth map of the shape S when viewed along the direction $\mathbf{d}_k$ (see Fig 2, 6b). This formulation ensures that the input shape S is entirely contained inside each VE, and that each of the VE's bounding functions overlaps with the depth map of S with respect to the corresponding axis.

With this definition in place, the problem of computing an optimal set of VEs can be recast as one of computing the optimal set of VE axis directions that best satisfy the two criteria above. We note that optimizing either volumetric approximation or surface coverage in isolation can produce outputs poorly suited to our needs.

In the inset, the top, horizontal, DHF axis choice produces a DHF that accurately approximates the input shape, but whose bounding functions only cover the
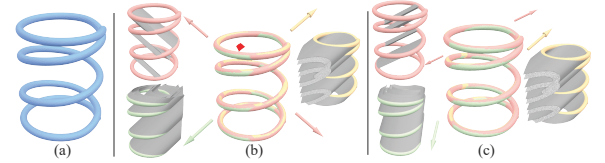

Fig. 7. Approximation criteria: given the inputs on the left (a), using only surface coverage as an axis selection criterion produces VE intersections that may contain extra undesirable connected components (b, highlighted in red). Optimizing for both volumetric approximation *and* surface coverage produces the outputs we seek (c).

square's sides, whereas using the diagonal axis on the bottom produces a DHF that satisfies both volumetric approximation and surface coverage. Perhaps less intuitively, as Fig 7 shows, surface coverage does *not* guarantee volumetric approximation: while the choice of axes in (b) results in a set of VEs whose bounding functions cover the input shape, the intersection of the corresponding volumes contains an additional undesirable connected component.

Based on these observations, we measure the quality of a given approximation $\tilde{S}$, defined in terms of the participating DHF and HF axes, as

$$E(\tilde{S}) = \text{Dist}(S, \tilde{S}) + \min(\mu(\tilde{S})(\text{Dist}(S, \tilde{S}) + \varepsilon_1), \varepsilon_2) \qquad (1)$$

Here $\text{Dist}(S, \tilde{S})$ measures the bidirectional L_1 Hausdorff, or closest, distance between S and $\tilde{S}$; and $\mu(\tilde{S})$ measures the quality of the surface coverage of S provided by the bounding functions of the participating VEs:
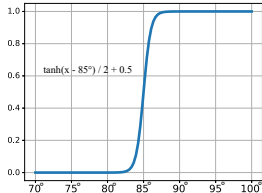
$$\mu(\tilde{S}) = \left(\frac{\text{Area}(\tilde{S}) - \text{Area}(\cup f_k)}{\text{Area}(\tilde{S})} \right) \qquad (2)$$

The combined loss function balances the two criteria while prioritising volumetric approximation. The set of variables we operate on is the number of HFs and the axis directions of the DHF and participating HFs; we seek to efficiently compute the combination

of axis directions that minimizes $E(\tilde{S})$. Unfortunately, even just computing $E(\tilde{S})$ for a given set of VE axes is highly time consuming, as it requires computing the geometry of the participating VEs, computing their Boolean intersection, and then finally computing the distance between this intersection and the input shape. To make this optimization tractable, we rely on a discrete optimization process that leverages the unique geometric properties of our DHF and HF shapes, and an effective branch-and-bound scheme that exploits our problem setup.

We first discretize $E(\tilde{S})$ by sampling both S and $\tilde{S}$ uniformly and densely, producing sets of points P and $\tilde{P}$. Dist is then evaluated point-to-point on these two sets (a.k.a. L_1 chamfer distance). We avoid explicit computation of the bounding functions f_k. Instead, for each point $\tilde{p} \in \tilde{P}$, we estimate its likelihood of being on f_k based on its visibility $v(\tilde{p}, d_k)$ along the axis d_k and the angle between $\tilde{p}$'s normal $\hat{n}_{\tilde{p}}$ and the axis direction. We set $v(\tilde{p}, d_k)$ to 1 if the point is visible (the ray from $\tilde{p}$ along $\mathbf{d}_k$ does not intersect the surface) and 0 otherwise. We set

$$\text{On}(\tilde{p}, d_k) = \max((1 - v(\tilde{p}, d_k)), T(\cos^{-1}(\hat{n}_{\tilde{p}} \cdot \mathbf{d}_k))) \qquad (3)$$

Here T is the shifted and scaled tanh function, $\tanh(x - 85°)/2 + 0.5$, shown in the inset, chosen so that T is 0 if $\hat{n}_{\tilde{p}}$ is well-aligned with $\mathbf{d}_k$ (the angle between them is significantly below 90°) and increasing to 1 as the angle approaches or exceeds 90°.

We then define μ as

$$\mu(\tilde{S}) = \frac{1}{|\tilde{P}|} \sum_{\tilde{p}=1}^{|\tilde{P}|} \min_k \text{On}(\tilde{p}, d_k) \qquad (4)$$

Even with this discretization in place, computing $E(\tilde{S})$ takes a non-trivial amount of time, as it requires sampling points on $\tilde{S}$ which is not explicitly defined. We therefore seek to minimize the number of $E(\tilde{S})$ evaluations.

To this end, rather than optimizing over an infinite set of possible axis directions, we use a finite set of well-distributed potential axis direction samples in our implementation (50 for DHF, and 80 for HFs). Since each direction vector corresponds to a point on the unit sphere, we evenly pick HF and DHF axis candidates by sampling these points on the unit sphere S^2 using spherical Fibonacci sampling [Keinert et al. 2015]. Since many objects encountered in practice are axis-aligned, we augment our sampled candidate directions with the three major axes (both directions).

As we expect approximation quality to improve as more HFs are added, we compute solutions incrementally for each possible HF count m, starting with $m = 0$ (i.e a DHF only); we terminate the process only once adding an extra HF fails to improve accuracy, or a maximal number of HFs is reached (we cap this number at 3; see Sec. 7 for validation of this choice.) Even with $m = 3$, however, the space of all possible axis combinations has 25.6 million combinations ($50 \cdot 80^3$), necessitating both a highly efficient strategy to evaluate $E(\tilde{S})$ and a robust search method that minimizes the number of evaluations required.

To achieve this speedup, we first recall that the term Dist is bidirectional, and can be written as the sum of two nonnegative, one-directional Chamfer distances (denoted CD): the Chamfer distance between the candidate ESI and the mesh, and the Chamfer distance between the mesh and the ESI:

$$\text{Dist}(S, \tilde{S}) = \text{CD}(S, \tilde{S}) + \text{CD}(\tilde{S}, S) \qquad (5)$$

We observe that we can sample points on our DHF and HFs by leveraging simple-to-evaluate in-out queries: a point is inside an HF (defined as above) if and only if a ray originating at the point and emanating along the HF direction axis intersects the input surface S, and a point is inside a DHF if and only if rays emanating along both positive and negative axis directions intersect S. We use this observation in a ray-casting framework to robustly compute points on the surface of all potential DHF and HF volumes. Notably, this computation is done as a pre-process, once for each potential DHF/HF axis. We further observe that an immediate consequence of this framework is that computing $\text{CD}(S, \tilde{S})$ is *much* faster than computing the inverse distance $\text{CD}(\tilde{S}, S)$, as it can be expressed in terms of distances between precomputed sample points on S and the participating VEs. Furthermore, as CD is strictly nonnegative, we can use $\text{CD}(S, \tilde{S})$ as a lower bound to quickly reject candidate VE combinations against the best known solution. We therefore search for VE candidates in parallel, and for each VE combination we only proceed to evaluate $E(\tilde{S})$ in full if $\text{CD}(S, \tilde{S})$ is lower than the best quality score encountered so far. This branch-and-bound strategy reduces the number of full evaluations for $m = 2, 3$ by 98% on average. For an additional speed up, before testing points and rays against S we reject sample points and ray directions that do not intersect the convex hull of S.

Overall our optimized ESI computation takes 3 minutes on average to determine the complete set of required VEs and axis directions on a 16-core Intel Xeon Gold 6130 CPU (an average of 2 minutes of preprocessing time, and 1 minute or less for axis selection), and up to 12 minutes on complex models like the *happy buddha* (Fig. 1); 9 minutes preprocessing time, 3 minutes axis selection).

5 Learning NESIs

Once ESI is computed, we encode the bounding functions f_a, f_b, and f_k of its participating VEs in a compact neural form as a set of MLPs. We now describe how to determine the domain Ω of each bounding function; the loss functions for training; the point sampling strategy in each domain used for evaluating the loss functions; and the overall neural network architecture.

To operate on a bounded numerical range, we define each explicit in its local axis-aligned coordinate system and restrict it to a $[-1, 1]^3$ bounding box. We normalize our 3D shapes S to be strictly inside this box by scaling them to be inside of $[-\frac{1}{1.1}, \frac{1}{1.1}]^3$. All our learned functions are thus defined over $D_k = [-1, 1]^2$.

5.1 DHF Hull Domain Sampling.

A fully-supervised training of a network that reproduces the DHF bounding functions f_a and f_b requires first obtaining sample points $(x_i, y_i) \in \Omega_0$, together with the ground-truth function values at these sample points (x_i, y_i) for supervision.

DHF bounding functions   Surface sampling & filtering   Projected sample points
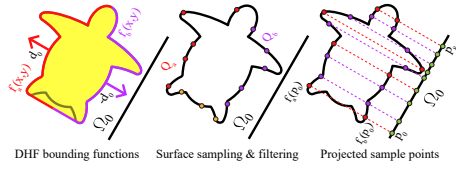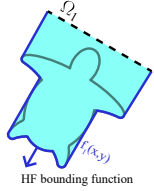
To generate sample points for learning $f_a(x, y)$ and $f_b(x, y)$ (inset, left), we first uniformly sample a dense set of points $Q = \{q_i\}$, where $q_i = (x_i, y_i, z_i)$, on the surface of the shape S (dots in the inset, middle). We then filter these points to find a set of samples Q_a that well approximates $f_a(x, y)$. We set $Q_a = Q$, then shoot rays from each point $q_i \in Q_a$ along $\mathbf{d}_0$, and remove q_i from Q_a if the ray intersects S (yellow dots in the inset, remaining points in red). We generate Q_b in a similar way, but using rays along $-\mathbf{d}_0$ (purple dot, inset middle). We project the sample points in $Q_a \bigcup Q_b$ onto the x-y plane to obtain $P_0 = \{p_i\}$, where $p_i = (x_i, y_i)$ (inset, right green). By construction, the projected sample points $p_i = (x_i, y_i)$ all lie within Ω_0 and provide a dense covering of the domain. For each $p \in P$, we compute the corresponding function values $f_a(p)$ and $f_b(p)$ (the surface intersections that are furthest apart along the DHF axis), as well as the surface normals $n_a(p), n_b(p)$ at these intersections (red and purple point pairs, inset, right).

In addition to the height functions themselves, we must also encode or store the domain Ω of the bounding functions for the DHF and HFs. For the DHF hull, we observe that we can encode this domain intrinsically by requiring that $f_b(x, y) < f_a(x, y)$ for all $(x, y) \in \Omega_0$, and $f_b(x, y) \geq f_a(x, y)$ for all $(x, y) \notin \Omega_0$. To obtain sample coverage for $D_0 \setminus \Omega_0$, we sample points in D_0, then shoot rays from each point along both $\mathbf{d}_0$ and $-\mathbf{d}_0$, discarding points if the ray intersects S. The remaining points form our $\overline{P_0}$ set.

5.2 HF Domain Sampling.

HF bounding function

We generate sample points for learning the height functions f_k of each HF volume $HF_k(\mathbf{d}_k)$ ($k = 1, 2 \ldots, m$, see inset) using a similar process, with one major difference (Fig 2de). We observe that each explicit provides two types of information about the approximated shape - the outline of its visual hull (Ω_k) and the shape geometry inside this outline. Our explicits often cover overlapping regions on the input shape (Fig 6b); encoding the geometry in these regions more than once introduces unnecessary redundancy, wasting network capacity. To avoid such redundancy we seek to restrict the parameter domain for which we store the geometry (height values) of each additional *learned* HF to only span those surface regions on the input shape S that have not been well-covered by the combination of the DHF and any previous HFs. We denote surface regions which are covered by $HF_k(\mathbf{d}_k)$, but *not well covered* by prior explicits, as S_k (in Fig 6c the purple regions on the HF correspond to areas not well covered by the DHF). A point q on S is in S_k if the ray from q along $\mathbf{d}_k$ does not intersect the surface *and if one* of the following three conditions holds:

Sampling in restricted domain
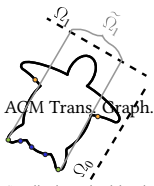
(1) A ray from q along any of the directions $\{\mathbf{d}_0, -\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{k-1}\}$ intersects S; in other words,

q is not on the surface of the DHF or any preceding HFs and hence is not represented by them (blue dots in the inset),

(2) q is on the surface of the *DHF* or any previous HF_i but is nearly a *grazing point* of S with respect to the axis of this previous HF or DHF; that is, the angle between the surface normal vector at q and the corresponding axis $\mathbf{d}_i$ is larger than a minimum threshold ($70°$ in our implementation). The rationale behind this condition is twofold. First, we seek an approximation which supports low distortion parameterization, thus we prefer surface regions to be approximated using HFs which better align with their normal, and to specifically avoid high parametric distortion. Moreover, small errors in height function approximation in grazing regions can result in large approximation error in $\mathbb{R}^3$ (green dots in the inset).

(3) There exists a point a on the ray from q along $\mathbf{d}_k$ that is invisible from all previous directions $\{\mathbf{d}_0, -\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{k-1}\}$. This condition is critical for approximating the interior of large voids or concavities which are not captured by previous explicits. see Fig. 21 for illustration of the importance of this criterion.

Formally, the domain Ω_k of a height field HF_k is defined the same way as for the *DHF*, delineating the outline of the HF's visual hull. However, for height learning purposes, we restrict HF_k to only cover the region S_k not properly covered by the *DHF* or any preceding *HF*s and focus on the subdomain of Ω_k given by the projection of S_k in the direction of $\mathbf{d}_k$ onto the x-y plane; we denote it as $\tilde{\Omega}_k$. We note that with these restricted domains $\tilde{\Omega}_k$ in place, one can recast the Boolean definition of NESI using subtraction instead of intersection (see Sec, 7.3).

With these criteria in place, we compute three sets of training samples. We first form Q_k, the subset of samples on S such that a ray from $q \in Q_k$ along $\mathbf{d}_k$ does not intersect S. We then compute the subset of points $Q'_k \in S^k$ by evaluating if they satisfy one of the criteria above. We project these points to the x-y plane of HF_k forming P_k and P'_k, and form $\overline{P_k}$ using the same process as for $\overline{P_0}$.

In addition to the height functions themselves, we must also encode the domains Ω_k and $\tilde{\Omega}_k$ of the bounding functions for each HF. The former defines the visual hull of the shape along the axis d_k while the later encodes the domain within which we want to encode the HF geometry. We encode $\tilde{\Omega}_k$ implicitly by forcing the height values at points inside $P_k \setminus P'_k$ to be above the surface. We use an explicit binary mask δ_k to specify the domain Ω_k and encode each HF as an MLP that returns both the bounding function f_k and the mask δ_k; we observe that this mask can be constructed to require significantly less network capacity than f_k itself.

5.3 Network Architecture.

DHF Network

Height Field Module

Mask Module
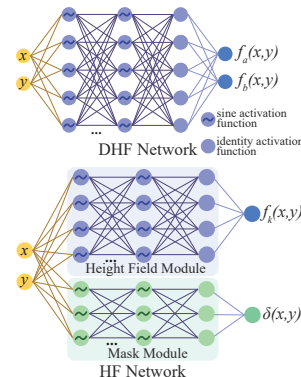
HF Network

We adopt the SIREN architecture [Sitzmann et al. 2020b] for our MLP network. Our network takes 2D locations (x, y) as input. For the DHF, the network outputs two values, each corresponding to the height values of the two bounding functions $f_a(x, y)$ and $f_b(x, y)$ of each

side of the DHF (inset, top). We en-
code the height and mask of each
HF as two separate MLP networks
(inset, bottom). The height func-
tion module outputs a single height
value $f_k(x, y)$ for all points in the
restricted domain $\tilde{\Omega}_k$; inside $\Omega \setminus \tilde{\Omega}_k$, the network is trained to out-
put a height value that is greater than ground truth (i.e. $f_k(x, y) >$
$f_{gt}(x, y), \forall (x, y) \in P'_k$). The domain of Ω_k is encoded as a binary in-
dicator function $\delta(x, y)$ by a separate compact MLP. Both networks
take 2D coordinates (x, y) as input, and pass them through n hidden
layers with sine activation functions between adjacent ones. For
DHFs, we output two height field values to bound the finite volume
of the target shape. For HFs and their masks, we use a smaller SIREN
network to infer height field values and a very compact network
to infer an indicator δ which indicates whether a point is inside
or outside the projected shape region Ω. The final encoded NESI
representation consists of $2m + 1$ MLPs: 1 MLP, the largest one, for
the DHF; m smaller ones for the HFs; and m tiny MLPs for the HF
masks.

5.4 Objective Functions.

To learn the neural encodings of the DHF hull and our k HF vol-
umes we minimize the following loss functions, defined in the local
coordinate frame of each explicit volume.

Our DHF loss is defined as

$$\mathcal{L}^D = \mathcal{L}^D_{Height} + \mathcal{L}^D_{Domain} + \alpha_{Normal} \mathcal{L}^D_{Normal} \qquad (6)$$

where the first terms encode the heights of the explicit surfaces;
the second delineates Ω_0; and the third term encodes the explicit
surface normals:

$$\mathcal{L}^D_{Height} = \frac{1}{|P_0|} \sum_{p \in P_0} \left(|f_a(p) - f_a^{gt}(p)| + |f_b(p) - f_b^{gt}(p)| \right) \qquad (7)$$

$$\mathcal{L}^D_{Domain} = \frac{1}{|\bar{P}_0|} \sum_{p \in \bar{P}_0} max(f_a(p) - f_b(p) + \epsilon, 0) \qquad (8)$$

$$\mathcal{L}^D_{Normal} = \sum_{i \in a,b} \frac{1}{|P_0|} \sum_{p \in P_0} (1 - \frac{n_i(p) \cdot n_i^{gt}(p)}{\max(\|n_i(p)\|_2 \cdot \|n_i^{gt}(p)\|_2, \epsilon)}) \qquad (9)$$

Here $f_i(p)$ and $f_i^{gt}(p)$ are the learned and input function values,
$n_i^{gt}(p)$ is the input surface normal at p, and $n_i(p)$ is the normal com-
puted by backpropagating the network that learns f_i. The weight
α_{Normal} is used to suppress the normal loss at the beginning of train-
ing, then gradually increase it after a certain number of iterations.
We set this weight as $\alpha = 0.05 \tanh((i - 4000)/10) + 0.05$, where i
is the current iteration.

The loss function for the kth HF is defined as

$$\mathcal{L}^H(k) = \mathcal{L}^H_{Height}(k) + \mathcal{L}^H_{Domain}(k) + \mathcal{L}_{Mask}(k) + \alpha_{Normal} \mathcal{L}^H_{Normal}(k) \qquad (10)$$

where $\mathcal{L}^H_{Domain}$ codifies the function behavior across $\Omega_k \setminus \tilde{\Omega}_k$, and
$\mathcal{L}_{Mask}$ defines the mask used to delineate Ω_k.

$$\mathcal{L}^H_{Height}(k) = \frac{1}{|P'_k|} \sum_{p \in P'_k} |f_k(p) - f_k^{gt}(p)| \qquad (11)$$

$$\mathcal{L}^H_{Domain}(k) = \frac{1}{|P_k \setminus P'_k|} \sum_{p \in P_k \setminus P'_k} max \left(\left(f_k^{gt}(p) - f_k(p) \right) + \epsilon, 0 \right) \qquad (12)$$

$$\mathcal{L}_{Mask} = \frac{1}{|\tilde{P}_k|} \sum_{p \in \tilde{P}_k} \text{BCE}(\delta_i(p), \delta_i^{gt}(p)) \qquad (13)$$

$$\mathcal{L}^H_{Normal} = \frac{1}{|P'_k|} \sum_{p \in P'_k} \left(1 - \frac{n_k(p) \cdot n_k^{gt}(p)}{\max(\|n_k(p)\|_2 \cdot \|n_k^{gt}(p)\|_2, \epsilon)} \right) \qquad (14)$$

Here BCE is the binary cross entropy (BCE) loss [Good 1952], and
$\tilde{P}_k = P_k \cup \bar{P}_k$ is the collection of all previously computed sample
points for this HF.

6 Using NESI Representations

Finally, we show how NESI can be used as either an implicit surface
or as a piecewise parametric representation.

NESI Occupancy Function. NESI trivially supports in-out occu-
pancy tests. For a given 3D point p, we simply check whether it
is inside or outside of the DHF and any HFs: we first convert p to
the local coordinates (x, y) of each DHF and HF; for HFs, we check
if the (x, y) in this coordinate system is inside the domain using
the mask; and finally we compare the associated height z to the
predicted height values from the network to determine if it is inside
or outside based on the definition of explicits. The point is inside
the NESI if and only if it is inside all its explicits.

NESI as a Parametric Representation. NESI's parameter domain,
or atlas, consists of Ω_0, used twice for the DHF hull top and bottom;
and $\tilde{\Omega}_k$ for each HF. To map a surface point p to the atlas, we locate
the explict whose surface it is closest to and then use the (x, y)
coordinates of the point in the coordinate system of this explicit as
its (u, v) parameters.

7 Results

We evaluate our analytic (ESI) and neural (NESI) representations
qualitatively, via visual inspection; and quantitatively, by measuring
the distance (chamfer L_1) between the inputs and our analytic and
learned outputs. We compare our results against an extensive list of
alternatives, and ablate different algorithmic choices as discussed
below. Finally, we showcase applications of using NESI for different
graphics applications, discuss its limitations, and propose extensions
addressing those. Throughout the paper we showcase 57 represen-
tative NESI outputs highlighting their high visual quality, which
remains high even at low parameter counts (e.g. Fig. 8, 13, 14, 18).
All renderings of both our and alternative results were generated
via our raytracing code and colored using a phong shading scheme.
Our raytracer uses NESI's (and ESI's) ability to instantaneously
and robustly evaluate in-out queries (Sec 6). See the appendix and
supplementary material for input sourcing details, implementation
details, galleries of input and output visuals, and additional details
of the evaluations below.

Evaluating ESI. To evaluate the premise behind our ESI and NESI representations, and their robustness, we seek to answer two questions: first, how accurately can the ESIs computed using our method (Sec 4) approximate typical content rendered by consumer facing graphics applications; and secondly, how many VEs are required to approximate such shapes to a desired accuracy using our method?

To answer these questions, we assembled a corpus of 320 diverse inputs representative of the types of geometries rendered by the applications we target. Our corpus included the Thingi32 [Takikawa et al. 2021; Zhou and Jacobson 2016] (32 shapes) and DHFSlicer [Yang et al. 2020](25 shapes) datasets representative of related prior work; non-trivial random shapes from the the ABC [Koch et al. 2019] dataset of CAD models (40 shapes); the dataset of Myles *et al.* [2014] (98 shapes commonly used in computer graphics); 122 additional random inputs from Thingi10K [Zhou and Jacobson 2016]; and complex, canonical, scanned shapes from the Stanford 3D Scanning Repository [Stanford 2024] (*david, dragon,* and *thai statue*). These inputs span both CAD and organic content, include highly complex shapes (*david, thai statue,* and *lucy*), as well as shapes with high genus, non-manifold geometry, and other artifacts.

To answer the first question, we used the method in Sec 4 to generate ESI approximations of these shapes using a fixed number of HFs (ranging from 0 to 4). We then measured the chamfer distance between the ground truth shapes and these ESI approximations (Table 1, all distances multiplied by 1000 and measured relative to the input bounding box diagonal). As the numbers show, even for a single DHF with no additional HFs, the approximation quality is often already very good. The chamfer distance decreases with additional HFs, but the amount of improvement tapers out, motivating our cutoff of using no more than 3 HFs. To provide some context to the numbers reported, we recall that while the Hausdorff distance between a surface and itself is zero, chamfer distance is an approximation of Hausdorff distance and is measured by computing the distance between point clouds sampled on the two surfaces. In particular, chamfer distance depends on sampling density - accuracy increases as point cloud size grows. chamfer distance between a surface and itself, measured using two different clouds sampled from the same surface, will never be strictly zero, but is expected to decrease as sampling density increases. To obtain high accuracy distances we use very large clouds (5M points). Even with this high density, the baseline chamfer from our inputs to themselves is 0.219. When using up to 3 HFs our ESI approximation quality (average chamfer 0.241) is within 0.022 of these values (in short, our chamfer distance is extremely close to the chamfer distance between our input meshes and *themselves.* Using 4 HFs instead of 3 HFs provides only a marginal improvement (0.233 versus 0.241), and only 14% of inputs show *any* non-trivial improvement (chamfer distance reduction of under 10^{-5}). These numbers confirm the main insight behind our method: *typical 3D geometries used in computer graphics applications can be well approximated by intersecting a small number of judiciously selected volumetric explicits.* While as per above a small percentage of shapes can benefit from using more than 3 HFs, we cap the number of HFs used in all experiments at 3 for runtime efficiency. As the experiments below show even with this cap our approximations are dramatically better than the alternatives. The ESI to input distances can be thought of as a lower bound on the

Dataset	DHF only	DH + 1 HF	DHF + 2 HFs	DHF + 3 HFs	DHF + 4 HFs	# VEs used
Myles'14	0.442(1.101)	0.244(0.491)	0.218(0.284)	0.216(0.241)	0.213(0.226)	3(2.82)
Thingi10k	0.249(0.734)	0.234(0.332)	0.229(0.261)	0.227(0.239)	0.227(0.232)	1(2.03)
Thingi32	0.447(0.687)	0.257(0.301)	0.234(0.238)	0.226(0.231)	0.225(0.229)	3(3.16)
ABC	0.359(1.636)	0.278(0.669)	0.263(0.305)	0.259(0.269)	0.259(0.266)	2(2.15)
Other	0.398(0.622)	0.256(0.328)	0.218(0.241)	0.217(0.223)	0.216(0.216)	3(2.89)
Overall	0.328(0.945)	0.249(0.419)	0.228(0.269)	0.225(0.241)	0.223(0.233)	3(2.48)

Table 1. Median and average (in brackets) Chamfer distances between input shapes and analytic intersections of volumetric explicits best approximating them, for different numbers of HFs across the different data sources and overall. The average lower bound on chamfer distances between input shapes and themselves is 0.219, representing the maximum accuracy achievable with this metric; our average and median ESI chamfer distances meet or are close to this lower bound in many cases, representing extreme accuracy compared to ground truth.

Dataset	1 DHF	1 DHF + 1 HF	1 DHF + 2 HFs	1 DHF + 3 HFs
ABC	16	8	9	7
Thingi32	0	2	18	12
Others	5	4	8	11
All	**21**	**14**	**35**	**30**

Table 2. DHF/HF counts used for each of the 100 input shapes in our NESI experiments, separated by originating dataset.

approximation quality provided by the corresponding learned NESI approximations.

To answer the second question, we computed the ESIs of these shapes, this time letting our method determine the output number of HFs used automatically. In our experiments, we did not use an error tolerance to determine the number of HFs needed, and added additional HFs if doing so reduced the Chamfer distance by *any* amount. In practice we expect users to specify a tolerance for the accuracy they need, limiting the number of HFs generated further. We then measured the number of HFs generated for each input (Table 1, last column). Across the set of 320 inputs tested, 98 inputs only required a single DHF (31%); 59 used 1 HF (18%); 75 used 2HFs (23%); and 88 used three HFs in addition to a DHF (28%). This experiment highlights the efficacy of our approach - the median input requires just 1 DHF and 2 HFs to produce accurate approximations.

Evaluating NESI. We evaluate our neural NESI representation by learning 400 NESI models using diverse parameter counts and ground truth input shapes. Specifically, we use a subset of the dataset above containing the Thingi32 [Takikawa et al. 2021; Zhou and Jacobson 2016] (32 shapes) and DHFSlicer [Yang et al. 2020] (25 shapes) datasets, the subset of the ABC [Koch et al. 2019] dataset (40 shapes), and the 3 models from [Stanford 2024] for a corpus of 100 input shapes total. These inputs are representative of the type of content we target, as well as shapes processed by state-of-the-art neural representation and compression methods. Tab 2 shows the number of volumetric explicits used by our method across this dataset. Predictably many CAD shapes can be well approximated using a single DHF, while organic content often requires including one or more HFs. Roughly half (49%) of the approximations use one to two HFs, and only 30% use our maximal HF number - three.

We approximate each input using NESI neural encoding using 4 levels of detail, or DHF/HF network parameter counts, resulting in a total of 400 encoded NESIs, see Appendix for details. Since we
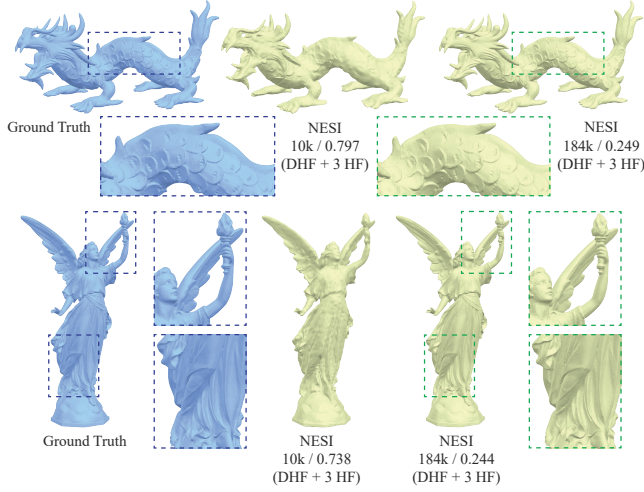
Fig. 8. NESI approximations of the *lucy* and *xyzdragon models* at low (10k) and extra high (180k) parameter counts. At low parameter counts, NESI captures the core properties of the inputs; at high parameter counts, even very fine details such as *lucy*'s torch and the scales of the dragon are accurately represented.

Dataset	LOD 1	LOD 2	LOD 3	LOD 4
ABC	0.444	0.350	0.314	0.303
Thingi32	0.538	0.374	0.302	0.279
Others	0.542	0.375	0.293	0.265
All	0.501	0.365	0.304	0.285

Table 3. Average NESI chamfer distances across different (ascending) levels of detail (parameter counts, see sup for exact settings). As desired NESI accuracy improves as parameter count increases.

LOD	Dataset	1 DHF	1 DHF + 1 HF	1 DHF + 2 HFs	1 DHF + 3 HFs
ESI	ABC	0.215	0.259	0.261	0.426
ESI	Thingi32	N/A	0.255	0.213	0.267
ESI	Others	0.196	0.234	0.224	0.236
ESI	**All**	**0.210**	**0.251**	**0.228**	**0.293**
LOD 1	ABC	0.325	0.404	0.493	0.696
LOD 1	Thingi32	N/A	0.437	0.449	0.689
LOD 1	Others	0.246	0.490	0.444	0.765
LOD 1	**All**	**0.306**	**0.433**	**0.459**	**0.718**
LOD 2	ABC	0.264	0.346	0.357	0.543
LOD 2	Thingi32	N/A	0.324	0.318	0.466
LOD 2	Others	0.216	0.362	0.316	0.496
LOD 2	**All**	**0.252**	**0.348**	**0.328**	**0.495**
LOD 3	ABC	0.243	0.313	0.302	0.491
LOD 3	Thingi32	N/A	0.280	0.257	0.372
LOD 3	Others	0.204	0.298	0.259	0.356
LOD 3	**All**	**0.234**	**0.304**	**0.269**	**0.394**
LOD 4	ABC	0.235	0.312	0.285	0.474
LOD 4	Thingi32	N/A	0.269	0.238	0.343
LOD 4	Others	0.200	0.275	0.242	0.309
LOD 4	**All**	**0.227**	**0.295**	**0.251**	**0.361**

Table 4. (top 4 rows) Average ESI chamfer distances across the 100 shapes in our NESI experiments broken down by source and number of VEs used. (rows 5 and on) Average NESI chamfer distance for all shapes and levels of details, broken down by source, the number of VEs used, and level of detail (DHF/HF parameter count). For each subset of shapes, Chamfer distance decreases as the level of detail increases, approaching ESI accuracy at LOD 4. Both ESI and NESI average chamfer distances are slightly larger at maximal number of VEs due to presence of shapes with nearly invisible regions. The overall average NESI distance at LOD 4 is 0.28, the average ESI distance is 0.247, and the chamfer distance from the 100 shapes to themselves is 0.228.

target consumer facing applications, such as streaming/rendering on low memory devices, we focus most experiments on learning models using low to medium parameter counts. At the lowest level we encode each input using under 10K parameters; at the finest level, we use 50K to 70K parameters to encode each input. We note that final per-shape parameter count depends both on the specified DHF/HF network parameter counts and the number of HFs used to encode the shape. We inspect the results both visually and quantitatively. Tab. 3 reports the average Chamfer L_1 distances between NESI approximation and input shapes for different levels of detail (all distances multiplied by 1000, measured relative to the input bounding box diagonal). As the table and visuals (e.g. Fig 8 (middle)) show, even at very low parameter counts NESI captures the core features of the input shapes. As desired, distance decreases as parameter count increases. We note that the average distance between NESI models trained with the largest parameter count settings and corresponding inputs (0.285) is very close to the distance between these inputs and their respective ESI approximations (0.247). This highlights the effectiveness of our neural encoding. We further break down (Table 4) the distance metrics the for each dataset in our corpus by both the number of HFs used and the explicit 'level of detail' used. We also report the ESI distances for each subset of shapes. As expected, for each subset of shapes the overall chamfer error decreases as level of detail increases, getting closer to the ESI distance for that subset. We note that higher EDI distance for the ABC 3HF subset is reflective of the fat that some of the shapes in this subset have areas barely visible or entirely invisible from outside (Fig 22). As somewhat expected, the gap between ESI and NESI distances is smaller for for simpler shapes (zero to two HFs).

To evaluate NESI's performance at higher parameter counts we additionally trained it on the *lucy* and *xyzdragon* inputs with 120K and 180K parameters each. Fig. 8 shows the results at 180K. As

the figure shows we accurately capture fine details such as the dragon's scales or the fine geometry on the dress and torch of *lucy*. The Chamfer distances for these experiments were: for *lucy*, 0.272 and 0.244 respectively; for *xyzdragon*, 0.263 and 0.249. For comparison, for Lucy the chamfer distance between its ESI and the input is 0.217, and between two point clouds sampled on the *lucy* input it was 0.186; for *xyzdragon*, these numbers were 0.193 and 0.170 respectively. These measurements confirm that as parameter count increases, our accuracy approaches that of ESI, which in turn accurately approximates the original input.

Applications. We demonstrate the versatility of NESI representations by leveraging the manipulation modes they support (implicit and parametric) for different classical geometry processing tasks. We use the implicit access mode for fast in-out queries for raytracing (used throughout the paper), and use the parametric access for texture mapping (Fig. 9) and meshing (Fig. 10); see Appendix for implementation details.

7.1 Comparative Evaluations

We compare our learned NESI outputs against an extensive list of representative alternatives whose authors provide either outputs or code to compare against. For fairness and consistency we use
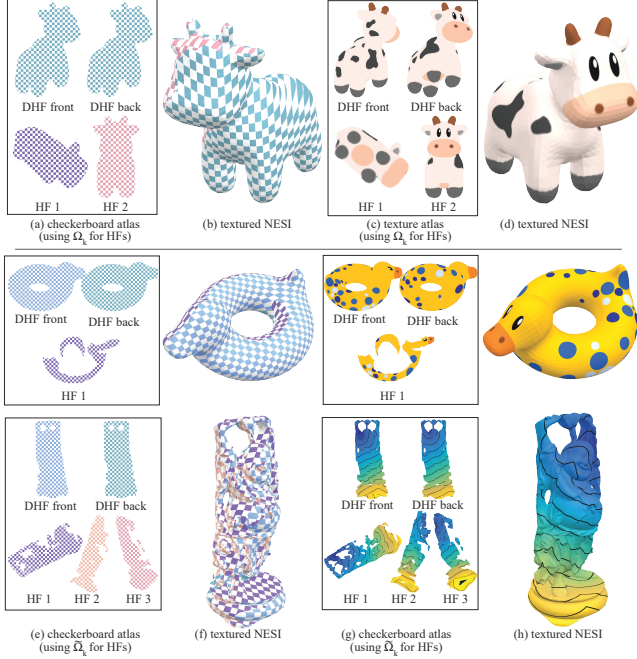
Fig. 9. NESI supports straightforward texturing of the output models (left) as well as texture transfer from the input meshes to these models (right): atlases with checkerboard and original texture (a,c,e,g), and NESI models (textured with a standard checkerboard texture (a,b,e,f) and texture transferred from original mesh (c,d,g,h). In the top row we use Ω_i as HF atlases, in the two bottom ones we restrict the atlases to $\tilde{\Omega}_i$, reducing memory footprint.
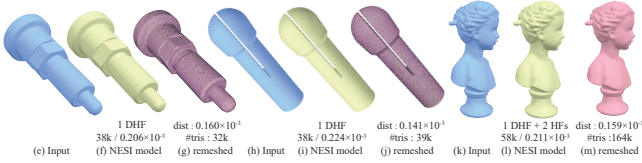

Fig. 10. NESI models (green) learned from input meshes (blue) and then meshed using our parameterization-based method (purple). As reflected by the Hausdorff distances listed, the output meshes well approximate the inputs.

raytracing to render all outputs, and use the same point cloud sampling strategy to compute L_1 chamfer distance for all outputs. (Prior works may use different sample counts and metrics in their reporting (e.g. [Li et al. 2022] and Sivarim *et al.* [2024] report the square of L_2 chamfer distances.)

We compare against the state of the art parametric neural representation method of [Morreale et al. 2022] by training our method on 4 example inputs they show and provide outputs for, and compare our outputs to theirs (Fig 11) (the remaining 3 outputs they show are open surfaces not amenable to volumetric representation). While their respective outputs use 100K parameters, ours use only 48K to 68K parameters. Despite the lower parameter count, all of our outputs more accurately approximate the input shapes (average chamfer distance 0.335 across their outputs; 0.237 for ours): the

distance for armadillo [Morreale et al. 2022] was 0.430, ours was 0.230 (68K); for dragon their distance is 0.334, ours is 0.296 (68K); for bimba theirs is 0.356 and ours 0.241 (58K); and for seahorse their distance is 0.220 and ours 0.180 (48K) (notably the numbers reported for their results in their paper are even larger as they used smaller point clouds to sample the input and outputs). As Fig. 11 shows, our results retain significantly more visual details.

We compare our method against multiple representative state of the art neural implicit methods: SIREN [Sitzmann et al. 2020b], both with and without Eikonal constraints; NeuRBF [Chen et al. 2023b]; implicit displacement fields (IDF) [Yifan et al. 2022]; NGLOD [Takikawa et al. 2021]; VQAD [Takikawa et al. 2022a]; and the key spheres method of Li et al. [2022]. For all the methods, except Li et al. [2022] we ran the code provided by the authors on the complete dataset detailed above with parameter counts comparable to ours. We were unable to run the code of [Li et al. 2022]; we therefore qualitatively and quantitatively compare our results to the output models provided by the authors for the Thingi32 dataset.

Fig. 12 shows representative comparisons of NESI against SIREN with Eikonal constraints [Sitzmann et al. 2020b], NeuRBF [Chen et al. 2023b], and IDF [Yifan et al. 2022]. All three methods generate implicit SDFs as their output; [Chen et al. 2023b; Yifan et al. 2022] target much larger models than us (700K and 800K parameters respectively) but can be modified to use lower parameter counts. On the examples shown (Fig 12), as well as on many additional inputs, these methods produce unstable outputs with additional spurious zero level-set surfaces. In our experiments, such spurious surfaces appeared in over 50% of the experiments for NeuRBF and over 30% for SIREN. For IDF [Yifan et al. 2022], the failure rate was highly parameter-count dependent: e.g. for 14K, 18K and 100K parameter counts, the method introduced spurious surfaces on over 30% of the inputs, but performed well for other counts (e.g. 25K). By using explicit rather than implicit representations, NESI robustly and consistently generates outlier-free approximations. The average and maximal errors (chamfer distance between an output and input shapes) across all input and parameter count combination tested for these methods are 16.507 and 61.538 for NeuRBF, 6.170 and 80.891 for SIREN w/eikonal, and 4.469 and 62.535 for [Yifan et al. 2022]. Our respective average and maximum errors are 0.364 and 1.494, demonstrating NESI's consistency/robustness.

We compare NESI both qualitatively (Fig 13-17 and quantitatively (Tab 5, 6) against four recent methods that use neural implicits: SIREN [Sitzmann et al. 2020b] *without* Eikonal constraints, VQAD [Takikawa et al. 2022a], NGLOD [Takikawa et al. 2022a], and [Li et al. 2022]. SIREN provides an important baseline, since we use the SIREN network (i.e., an MLP with sinusoidal activation functions) as the backbone architecture for neural encoding. The difference is that we use the SIREN network for learning the simple, explicit height functions for different pieces of our NESI representation, while the original SIREN method uses the same network for computing a single zero-level set in $\mathbb{R}^3$ to approximate the entire surface of an input shape, which can potentially be very complex. VQAD, NGLOD, and [Li et al. 2022] specifically target low parameter count compression as an application, and thus serve as a natural baseline to our method. These methods were shown to outperform earlier alternatives such as [Müller et al. 2022] and [Davies et al. 2021] in
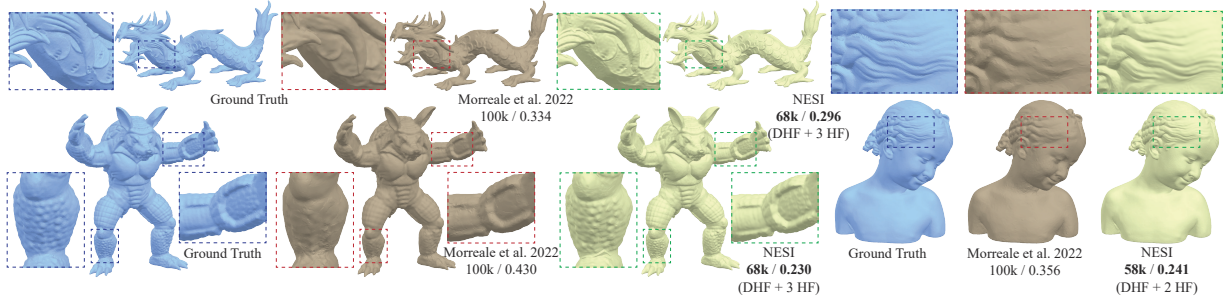
Fig. 11. NESI outputs trained using 58K to 68K parameters provide a better, more detailed, approximation of the inputs than the neural models of [Morreale et al. 2022] which use much higher parameter counts (100K).
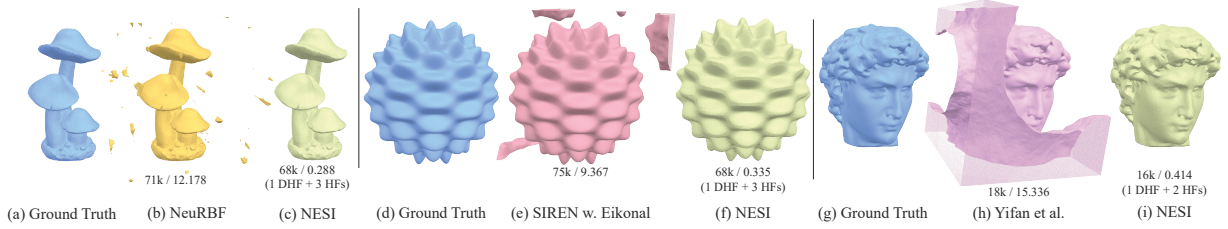

Fig. 12. Comparisons of NESI outputs to these of NeuRBF [Chen et al. 2023b], SIREN with Eikonal constraints [Sitzmann et al. 2020b], and the method of Yifan et al. [2022]. While NESI results are consistently stable, these prior methods generate spurious extra surfaces on a significant percentage of inputs.
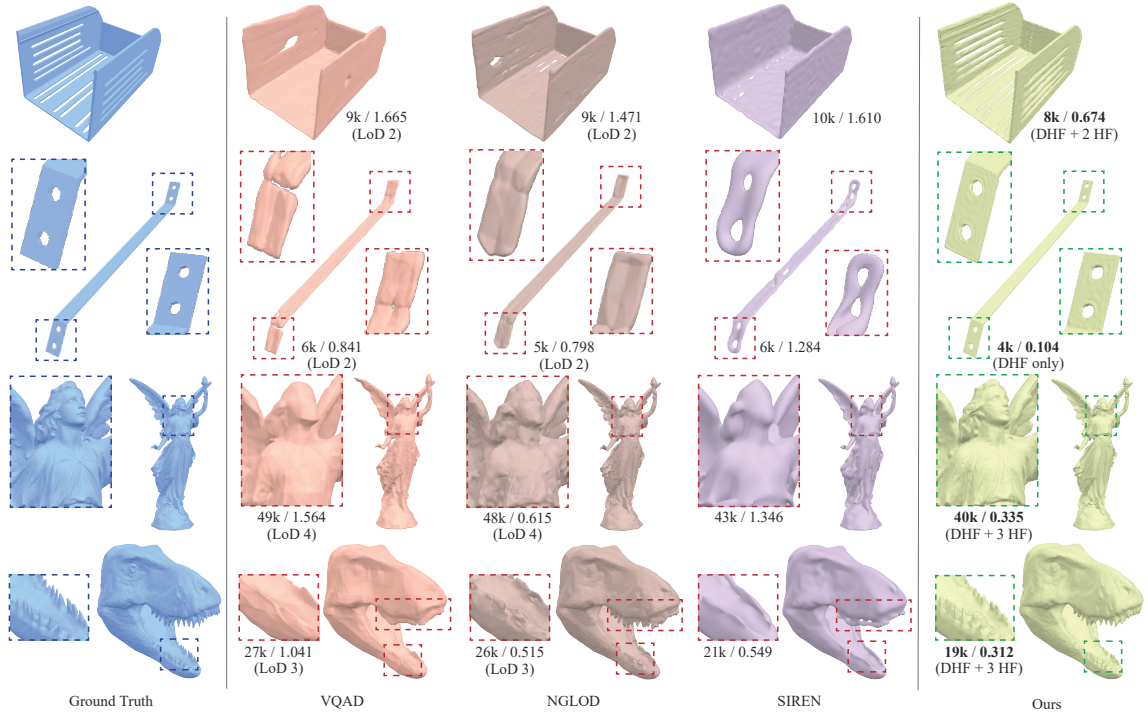

Fig. 13. Representative comparisons of NESI outputs to these of SIREN [Sitzmann et al. 2020b] (no Eikonal constraints), VQAD [Takikawa et al. 2022a], and NGLOD [Takikawa et al. 2021]. For similar parameter counts NESI consistently better captures fine details of the input shapes.
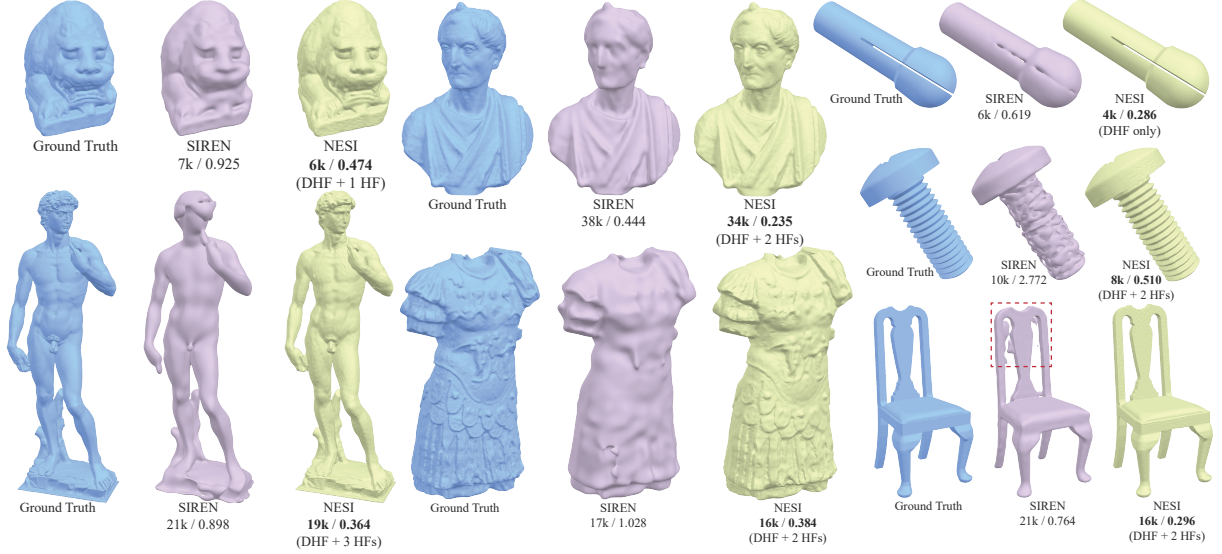
Fig. 14. Additional comparisons with SIREN (no Eikonal constraints). Even with smaller parameter counts NESI consistently captures more geometric details.
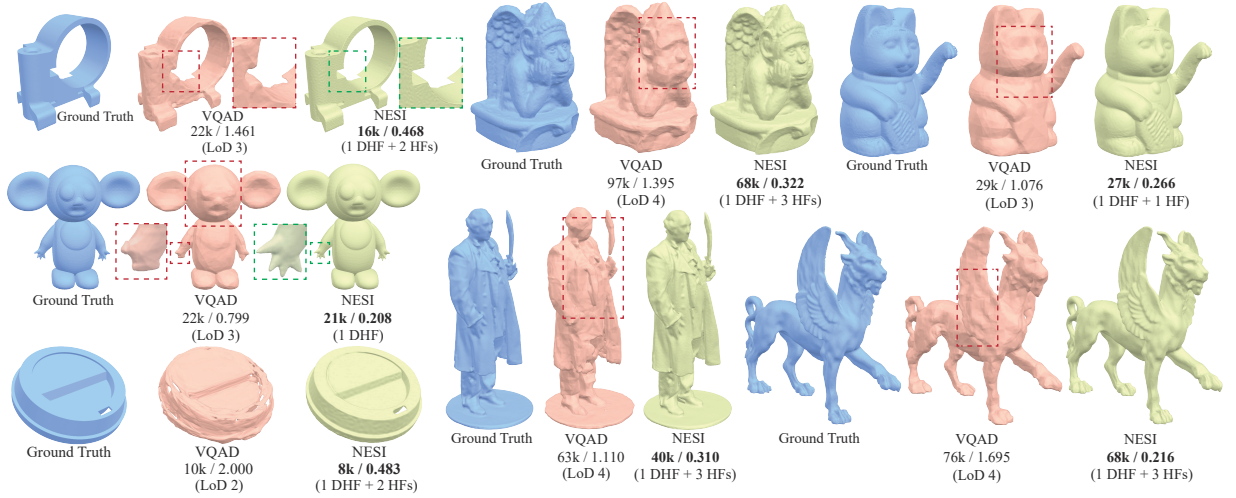

Fig. 15. Additional comparisons with VQAD. Even with smaller parameter counts NESI has higher accuracy and captures more details.
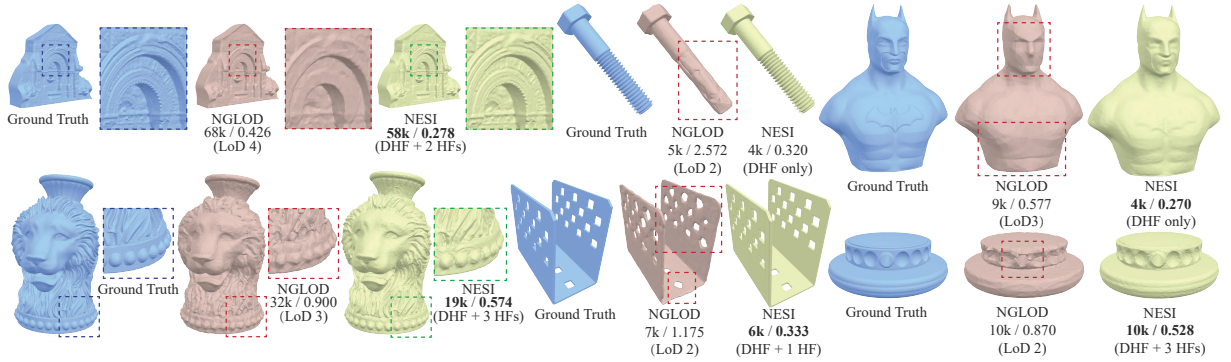

Fig. 16. Additional comparisons with NGLOD. Even with smaller parameter counts NESI consistently captures more geometric details.
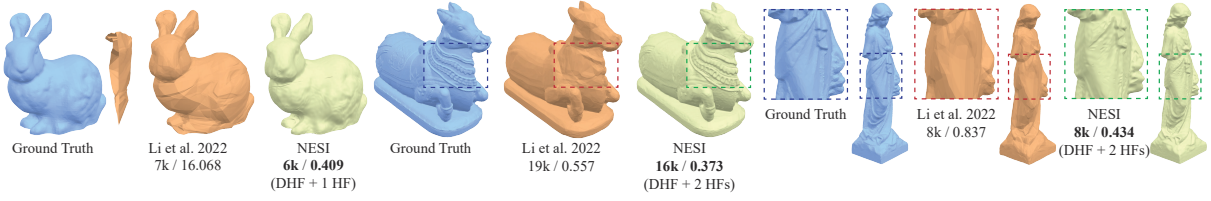
Fig. 17. Comparisons with [Li et al. 2022]. Even with smaller parameter counts NESI consistently captures more geometric details; improvement is most noticeable at lower parameter counts. As the bunny (left) illustrates the method of [Li et al. 2022] exhibits occasional instabilities while NESI remains robust.

Dataset	#Params	VQAD	Li et all	NGLOD	SIREN (wo. Eikonal)	NGF	NESI
ABC	<10k	2.112		1.879	1.524	0.774	**0.392**
Thingi32	<10k	2.965	2.932	3.091	1.539	0.635	**0.538**
Others	<10k	2.532		2.103	1.508	0.639	**0.492**
All	**<10k**	2.516		2.345	1.525	0.698	**0.458**
ABC	10k-20k	1.104		0.597	0.808	0.634	**0.408**
Thingi32	10k-20k	1.555	0.668	0.779	0.820	0.546	**0.374**
Others	10k-20k	1.690		0.852	0.829	0.606	**0.410**
All	**10k-20k**	1.384		0.722	0.817	0.598	**0.395**
ABC	20k-40k	1.054		0.509	0.591	0.460	**0.291**
Thingi32	20k-40k	1.381	0.521	0.598	0.609	0.357	**0.302**
Others	20k-40k	1.211		0.599	0.601	0.435	**0.279**
All	**20k-40k**	1.205		0.561	0.600	0.420	**0.291**
ABC	>40k	0.729		0.470	0.450	0.405	**0.349**
Thingi32	>40k	1.010	0.498	0.493	0.478	0.327	**0.279**
Others	>40k	1.010		0.476	0.472	0.362	**0.280**
All	**>40k**	0.905		0.480	0.465	0.368	**0.300**
Overall		1.747		1.398	0.932	0.505	**0.364**

Table 5. Quantitative comparison of NESI (right) against prior art [Li et al. 2022; Sitzmann et al. 2020b; Sivaram et al. 2024; Takikawa et al. 2022a, 2021] across different parameter counts and datasets (Chamfer-L_1 distance between input and predicted surfaces). NESI consistently outperforms all baselines, with improvement most pronounced for low parameter counts.

Dataset	#Params	NESI vs VQAD	NESI vs Li et al.	NESI vs NGLOD	NESI vs SIREN (wo.Eikonal)	NESI vs NGF
ABC	<10k	100%		100%	97%	83%
Thingi32	<10k	100%	100%	100%	100%	95%
Others	<10k	100%		100%	100%	100%
All	**<10k**	**100%**		**100%**	**99%**	**91%**
ABC	10k-20k	96%		95%	93%	90%
Thingi32	10k-20k	100%	88%	100%	97%	91%
Others	10k-20k	100%		93%	93%	85%
All	**10k-20k**	**98%**		**96%**	**94%**	**89%**
ABC	20k-40k	100%		97%	83%	88%
Thingi32	20k-40k	100%	97%	95%	97%	81%
Others	20k-40k	100%		100%	93%	82%
All	**20k-40k**	**100%**		**97%**	**90%**	**84%**
ABC	>40k	100%		83%	80%	78%
Thingi32	>40k	100%	100%	100%	100%	88%
Others	>40k	100%		100%	93%	82%
All	**40k**	**100%**		**94%**	**90%**	**82%**
Overall		**100%**	**96%**	**96%**	**93%**	**86%**

Table 6. Percentage of shapes that NESI outperforms each baseline [Li et al. 2022; Sitzmann et al. 2020b; Sivaram et al. 2024; Takikawa et al. 2022a, 2021] on across different datasets and parameter counts. Each individual comparison evaluates a NESI output against an alternative method's output produced using same or higher parameter count.

terms of quality/compactness tradeoff. Both visual inspection and quantitative comparisons confirm that NESI significantly outperforms all four alternatives, both on average (Tab. 5) and in head to head comparisons (Tab 6). Our error is 3.8 times smaller than that of NGLOD, 4.9 times smaller than that of VQAD, and 2.6 times smaller than that of the SIREN baseline. It is 1.9 times smaller than that of [Li et al. 2022] (on Thingi32).

For the head-to-head comparisons, we encode each input using NESI and an alternative method where NESI encoding uses same or smaller parameter count than the alternative (for each output of an alternative method, we locate our result with the closest parameter

count smaller than the alternative). As Tab. 6 shows, NESI outperforms NGLOD on 96% of inputs tested, VQAD on 100% of inputs, SIREN on 93%, and [Li et al. 2022] on 96%. Lastly and critically, NESI is notably more stable than these alternatives: our maximal deviation from ground truth across all inputs and parameter counts is 1.494, versus 6.127 for SIREN, 16.068 for [Li et al. 2022], 29.431 for VQAD, and 35.687 for NGLOD.

We note that our parameter count is a function of both MLP size and number of HFs used. Accordingly, comparisons of our method's aggregate performance to *itself* across parameter ranges reflects both MLP size and HF count within each range (Tab. 5). Notably while the <10K and 20K-40K ranges contain 32 models of ABC shapes represented using a single DHF (Tab. 8) the 10K-20K and >40K ranges contain zero models of such shapes. This split explains the inversion in the distance metric across these ranges.

Finally, we compare our method to the state-of-the-art Neural Geometry Fields (NGF) method [Sivaram et al. 2024], that encodes shapes as a combination of a QSLIM simplified mesh and a learned displacement (Fig 18). Their method fails to produce an output on 21 of the input model/parameter combinations we tested, with most failures happening on models with non-trivial topology (*fertility, happy buddha, david, xyzdragon*) at lower parameter counts (QSLIM mesh of 100 to 250 faces). This seems to be due to failures during their triangle pairing step. Visual inspection and quantitative analysis of their successful outputs confirm that NESI outperforms this alternative, both on average (Tab. 5) and in head-to-head comparisons (Tab 6). The improvement is most pronounced at lower parameter counts: when using under 10K parameters our average distance is 0.458 versus 0.842 for NGF. A qualitative advantage of NESI over NGFs is its support for straightforward in-out queries and 2D parameterization - neither of which are directly supported by the NGF representation, as it is based on displacement maps.

Please see supplemental material and appendix for additional visual comparisons against the above methods; these comparisons demonstrate that NESI encodings are consistently more accurate and more detailed compared to the alternatives across different parameter counts, with improvement being most noticeable at lower parameter counts.

7.2 Ablation Studies

We ablate several key algorithmic choices made in ESI and NESI computation.

Fixed vs Computed VE axes We ablate our strategy for computing DHF/HF axes by comparing it to using a fixed, input independent,
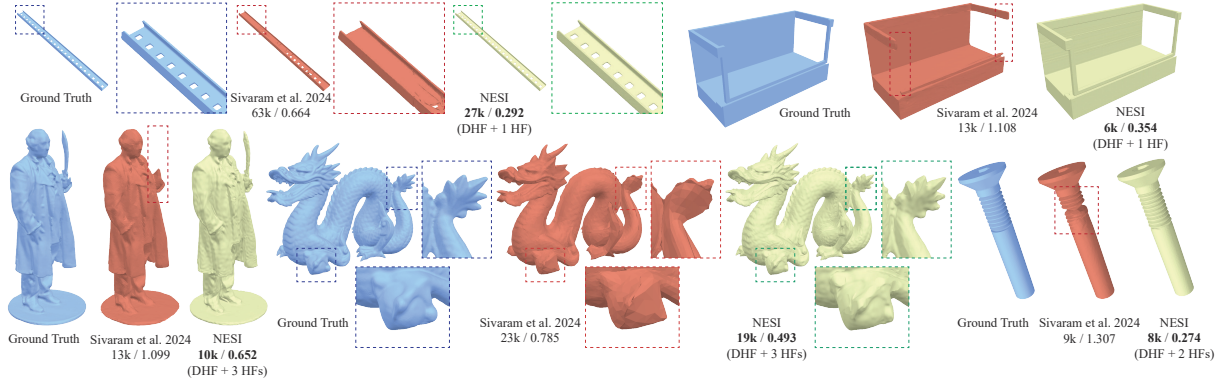
Fig. 18. Comparisons with [Sivaram et al. 2024]. While at lower parameter counts, the alternative method frequently fails to capture prominent topological and geometric features, NESI fatefully approximates these.
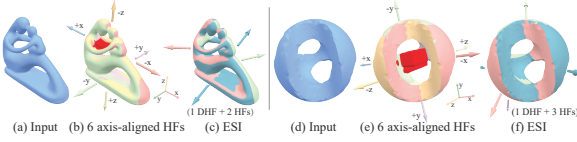

Fig. 19. Using coordinate system axes as VE axes (6 HFs) (b,e) produces catastrophically poor approximations that dramatically deviate (deviating part highlighted in red) from the input (a,d). Our ESIs use notably fewer bounding height fields to accurately approximate these inputs (c,f).
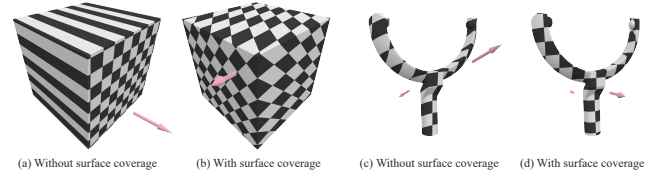

Fig. 20. Absent an explicit surface coverage term ESI axis selection may produce volumetric explicits with large surface areas (near-)orthogonal to all VE axes (a, c). While the resulting ESIs are still suitable for in-out queries, parameterizing these areas via projection leads to extreme parametric distortion. (b, d) Our loss function (Eq 1) which accounts for coverage, results in outputs that can be parameterized with much lower distortion.

Dataset	#Params	axes views wo mask	axes views with mask	NESI
ABC	<10k	0.790	0.857	**0.392**
Thingi32	<10k	1.475	1.060	**0.538**
Others	<10k	1.554	1.033	**0.492**
All	**<10k**	1.223	0.971	**0.458**
ABC	10k-20k	0.442	0.414	**0.408**
Thingi32	10k-20k	0.670	0.538	**0.374**
Others	10k-20k	0.685	0.534	**0.410**
All	**10k-20k**	0.583	0.487	**0.395**
ABC	20k-40k	0.403	0.361	**0.291**
Thingi32	20k-40k	0.468	0.400	**0.302**
Others	20k-40k	0.467	0.397	**0.279**
All	**20k-40k**	0.442	0.384	**0.291**
ABC	>40k	0.363	0.354	**0.349**
Thingi32	>40k	0.409	0.366	**0.279**
Others	>40k	0.403	0.359	**0.280**
All	**>40k**	0.389	0.359	**0.300**
Overall		0.610	0.514	**0.364**

Table 7. Quantitative comparison of NESI (right) against six axes views across different parameter counts and datasets (Chamfer-L_1 distance between input and predicted surfaces).

set of axes, mimicking depth fusion [Richter and Roth 2018; Shade et al. 1998] and relief texture mapping [Baboud and Décoret 2006; Oliveira et al. 2000](Fig 5, 19). As shown in Fig. 5a-c, having axes in the horizontal plane only [Shade et al. 1998] is clearly insufficient to achieve reasonable approximation even for simple shapes. Similarly, examples in Fig 5, 19 demonstrate that using 6 coordinate system axes can lead to catastrophically poor approximation quality. Our method has no such catastrophic failures, despite using notably fewer HFs (maximum 1 DHF + 3HF, average 1 DHF + 1.5 HFs).

Table 7 compares our learned NESI outputs to a baseline using the six coordinate-system axes as DHF and HF directions (one DHF and 4 HFs) across different parameter counts and datasets. We report

results for this six-axis baseline both as simple SIREN-encoded HFs (i.e. learning each HF across its complete parameter domain Ω_i), and using our combined HF plus mask based MLP network architecture (i.e. utilizing the restricted domains $\tilde{\Omega}_i$, Sec. 5). Overall we have 40% improvement compared to the first baseline, and 29% improvement compared to the second. This experiment demonstrate both the superiority of our axis selection scheme and the importance of our use of the restricted domains $\tilde{\Omega}_i$.

ESI Axis Selection. Our ESI optimization explicitly accounts for surface coverage (Eq. 1) and promotes outputs that can be projectively parameterized without excessive distortion or degeneracies. Fig. 20 demonstrates the impact of eliminating this penalty: while the resulting DHF approximation (Fig. 20b) precisely captures the input geometry, its projection along the axis is not bijective. Our output (Fig. 20c) is parameterized bijectively, facilitating texturing. Fig. 7 validates another key choice of our method: the use of symmetric rather than one sided distance between the input and the ESI approximation. While distance from ESI-to-input is much more computationally expensive to compute than the inverse, using it as part of the optimized loss function is critical for obtaining suitable approximations.
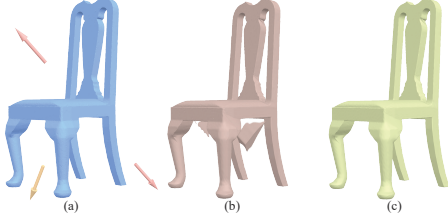
Fig. 21. Given the input (a, arrows illustrate DHF and HF axis directions), using the method in Sec. 5 to define $\tilde{\Omega}_k$ for $k = 1, 2$ produces the desired outcome (c); removing the ray visibility condition for HFs (#3) results in an output that does not accurately capture the void between the chair legs (b).

NESI Computation. Figure 21 demonstrates the impact of our set of tests used to define $\tilde{\Omega}$ (Sec 5). Using a more restrictive definition of $\tilde{\Omega}$ (one that omits the ray visibility test) can produce outputs that may not well approximate interior voids (Fig 21b). This criterion can be thought of as mirroring the need to measure the distance from ESI to input in the axis computation above.

Localized HFs. We reduce NESI memory footprint and avoiding learning duplicate surface geometry by learning HF surface geometry only in areas where it is not already covered by the DHF/other HFs (Fig 2de, Sec 5). We ablate the importance of this step by comparing the accuracy of our results to that of results generated using same parameter counts, but without such localization (using $\tilde{\Omega} = \Omega$) on inputs that require at least one HF (localization is not performed when only a single DHF is used). Using our approach improves approximation quality by 15% across all parameter counts (0.393 NESI vs 0.453 ablation), with the improvement being more pronounced for small parameter counts - at parameter counts under 10K, our choice leads to a 25% improvement over the ablation alternative (0.553 NESI, vs. 0.689 ablation).

Normal Preservation. We measured the difference in approximation quality between results obtained using our loss functions $\mathcal{L}^D$, $\mathcal{L}^H(k)$ (Eqs 6, 10) and ones which do not include the normal terms $\mathcal{L}^H_{Normal}(k)$, $\mathcal{L}^D_{Normal}$ (Eqs 9, 14). While in theory one can expect Chamfer distance to improve when normals are not optimized for, we found that including the normal terms marginally reduces approximation error on average (CD 0.364 with normal preservation and 0.367 without; adding the term leads to better approximation on 288 out of 400 inputs). Visual assessment confirms that our results look better than ones without the normal preservation term.

Image Compression. A naive alternative to our learned neural representation of ESI, is to convert the HFs and the two DHF height-fields into depth images and store those in compressed form. To ablate this alternative, we stored the 'batman' (Fig. 16) DHF and HFs as 500×500 depth-images (JPEG; 300K filesize). Decompressing and combining these HFs produces an ESI model with chamfer distance of 0.358 to input; a NESI output with 1/10 the filesize (24K, 6K parameters) has chamfer distance of 0.270. This experiment demonstrates that this alternative provides a much worse size/accuracy tradeoff. Moreover, while NESI can be processed *as-is*, compressed images need to be decompressed before processing with decompression

significantly increasing their memory footprint. Our ability to support as-is processing distinguishes NESI from all traditional image and geometry compression methods which require decompression before processing.

File Size. The file size of a NESI model is determined by two factors: the number of explicits used and the number of parameters allocated to each. Our file sizes range from 24K for a single 6K parameter DHF, to 273K for a 38K parameter DHF and three 10K parameter HFs.

Runtimes. All models in our experiments were trained on an NVIDIA Tesla V100 16GB for 10,000 iterations, which takes between 5 and 30 minutes depending on the number of explicits used and network size. Determining the optimal number of explicits and their axes takes on average 3 minutes, and up to 12 minutes for our worst performing model (*hundepaar* from the DHFSlicer dataset).

7.3 Discussion and Limitations.

Subtractive Formulation of NESI. It is instructive to take an alternative view of the Boolean definition of the NESI representation based around the restricted domains $\tilde{\Omega}_k$ for each HF. Recall that the NESI representation is defined as the approximating volume

$$\tilde{S} = DHF(\mathbf{d_0}) \cap (\cap_{k=1}^m HF_k(\mathbf{d_k})). \tag{15}$$

Clearly, it can be re-written as

$$\tilde{S} = DHF(\mathbf{d_0}) \setminus (\cup_{k=1}^m \overline{HF}_k(\mathbf{d_k})), \tag{16}$$

where $\overline{HF}_k(\mathbf{d_k}) = \{(x, y, z) | z \geq f_k(x, y) \wedge (x, y) \in \Omega_k\}$.

We now observe that if we adopt the Boolean definition of $\tilde{S}$ in Eqn. 16, each $\overline{HF}_k$ can be viewed as a subtractive volumetric primitive. It then follows that this Boolean expression of $\tilde{S}$ in Eqn. 16 holds true when we replace the domain Ω_k of $\overline{HF}_k$ by the restricted domain $\tilde{\Omega}_k$, by our construction.

NESI Scope/Limitations. NESI representation is targeted at consumer facing graphics applications, such as video games, online shopping, AR/VR, or remote communication. As such we aim to create representations of typical everyday shapes that have very low memory footprint, and that can be effectively rendered on consumer devices.

Consequently, like other implicit representations [Takikawa et al. 2022a, 2021; Yifan et al. 2022], our method is, by construction, designed for objects that can be well represented in implicit form. Specifically, we expect our inputs to have non-zero thickness; this limitation means that NESI is well-suited for representing most human-made and organic content, but less suited for zero-thickness objects for such as leafy trees or clothing. We support shapes with multiple connected components, boundaries, and non-manifold mesh connectivity - three of our inputs have more than one component (*fisherman* from the *Thing32* dataset, and *75* and *395* from the *abc* dataset), and two (fisherman and mask1 from the Thingi32 dataset) have open boundaries (see supplemental material). At the same time, our focus is on single-shape representation. Thus, it is likely more memory efficient to represent scenes consisting of multiple distinct shapes using a separate NESI for each shape.
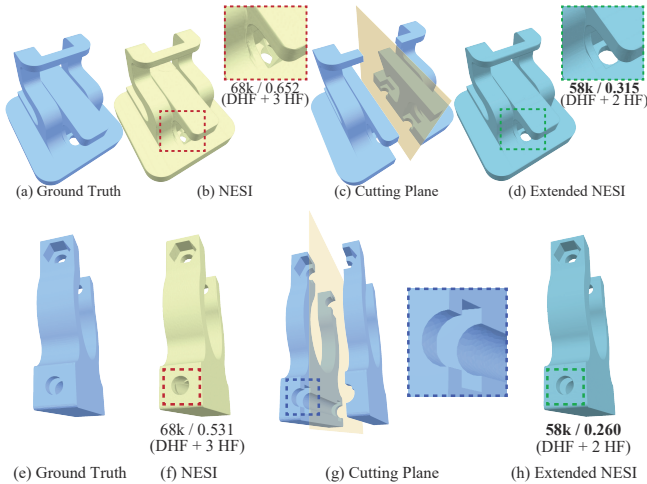
Fig. 22. Shape with severe self-occlusions (a,e) may require a very large number of volumetric explicits to obtain a numerically accurate approximation (interior voids entirely invisible from outside simply cannot be captured). However, for practical purposes, using a DHF hull and 3 HFs already provides a good approximation of the shape for rendering from any external viewpoints (b,f). Using our extended method (c,d,g,h) improves numerical accuracy for such inputs. Notably while the numerical accuracy improves - visually both the default (b,f) and the extended method outputs (d,h) look very similar to the inputs (a,e).

NESI is based on capturing surface geometry visible from outside the processed shapes. This assumption is consistent with the typical rendering setup in the applications we target. Given shapes with surface regions that are heavily or entirely occluded (Fig 22ad), NESI approximations will still produce visually high quality results when rendered from outside (Fig 22be) but will exhibit high numerical error. The approximation quality can be further improved by using our extended method (Sec. C, Fig. 22dh, 24). Since such surface regions almost never need to be rendered, this limitation is rarely relevant for practical settings for viewing purposes. We note that all numerical results reported in the paper as well as all other visuals do *not* include this extension.

8 Conclusions

We presented NESI, a novel compact neural representation for 3D shapes. Our representation combines the processing advantages of implicit and parametric representations, making it exceptionally suitable for a wide range of geometry processing applications. Our experiments convincingly demonstrate that NESI approximates diverse, complex 3D shapes much more accurately than state-of the art alternatives, when using the same parameter count, or memory footprint. This improvement is most pronounced at lower parameter counts, where our average error is 35% smaller than that of closest alternative (0.46 NESI vs 0.7 [Sivaram et al. 2024]). Across the entire set of inputs and parameter counts NESI outperforms this closest alternative 86% of the time. This improvement is made possible by our reduction of the 3D approximation problem to a combination of two sub-problems: (1) locating optimal DHF and HF axes such that

the intersection of these volumetric explicits tightly approximates the input; and (2) compactly representing each DHF and HF as a 2D neural function such that the intersection of these functions well approximates the input.

Future Work. One interesting future NESI direction is to explore the use of explicit surfaces defined over non-Euclidean spaces, where the mapping between the domains and surfaces is not necessarily orthographic but is perhaps perspective or fish-eye based. Using such surfaces has the potential to further reduce the number of explicits necessary to accurately represent complex shapes. Another is to explore additional geometry processing applications that can benefit from our ESI and NESI representations. Given that ESIs are guaranteed to always contain the input shape, one potential, immediate, application is collision detection. In this context an important question to consider is the overlap between surface regions covered by the different VEs, and in particular the choice of optimal parameter domain to use for each surface point in the context of different downstream applications.

References

Thomas Alderighi, Luigi Malomo, Bernd Bickel, Paolo Cignoni, and Nico Pietroni. 2021. Volume decomposition for two-piece rigid casting. *ACM Trans. Graph.* 40, 6 (2021), 272–1.

Pierre Alliez. 2005. Recent advances in compression of 3D meshes. In *Proc. 13th European Signal Processing Conference.* IEEE, 1–4.

Lionel Baboud and Xavier Décoret. 2006. Rendering geometry with relief textures. In *Graphics Interface'06.*

Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. 2020. Shape reconstruction by learning differentiable surface representations. In *Proc. IEEE/CVF CVPR.* 4716–4725.

James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.* 1, 3 (1982), 235–256.

Mario Botsch and Leif Kobbelt. 2004. A remeshing approach to multiresolution modeling. In *Proc. Eurographics SGPR.* 185–192.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. 2010. *Polygon Mesh Processing.* A K Peters.

Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. 2018. Reverse engineering modeling methods and tools: a survey. *Computer Aided Design and Applications* 15, 3 (2018), 443–464.

Nathan A. Carr, Jared Hoberock, Keenan Crane, and John C. Hart. 2006. Rectangular multi-chart geometry images. In *Proc. SGP.* 181–190.

Jianer Chen and Ergun Akleman. 1999. Generalized Distance Functions. In *Proc. SMI.*

Yun-Chun Chen, Vladimir Kim, Noam Aigerman, and Alec Jacobson. 2023a. Neural Progressive Meshes. In *ACM SIGGRAPH 2023 Conference Proceedings.* 1–9.

Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. 2023b. NeuRBF: A neural fields representation with adaptive radial basis functions. In *Proc. IEEE/CVR CVPR.* 4182–4194.

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *Proc. IEEE/CVF CVPR.* 5939–5948.

Daniel Cohen-Or, Chen Greif, Tao Ju, Niloy J. Mitra, Ariel Shamir, Olga Sorkine-Hornung, and Hao (Richard) Zhang. 2015. *A Sampler of Useful Computational Tools for Applied Geometry, Computer Graphics, and Image Processing.* A. K. Peters.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph* 32, 4 (2013), 1–10.

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proc. Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96).* 303–312.

Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. 2021. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. arXiv:2009.09808 [cs.GR]

Zhantao Deng, Jan Bednařík, Mathieu Salzmann, and Pascal Fua. 2020. Better patch stitching for parametric surface reconstruction. In *International Conference on 3D Vision (3DV).* IEEE, 593–602.

Theo Deprelle, Thibault Groueix, Noam Aigerman, Vladimir G. Kim, and Mathieu Aubry. 2022. Learning Joint Surface Atlases. *ECCV Workshop, Learning to Generate 3D Shapes and Scenes 2022* (2022).

Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. 2019. Learning elementary structures for 3D shape generation and matching. In *Advances in Neural Information Processing Systems.* 7433–7443.

Gerald E Farin. 2002. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann.

Sándor Fekete and Joseph Mitchell. 2001. Terrain decomposition and layered manufacturing. *Computational Geometry and Applications* 11, 06 (2001).

Wei Gao, Yunbo Zhang, Diogo Nazzetta, Karthik Ramani, and Raymond Cipra. 2015. RevoMaker: Enabling multi-directional and functionally-embedded 3D printing using a rotational cuboidal platform. In *Proc. UIST.* ACM.

Irving John Good. 1952. Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological)* 14, 1 (1952), 107–114.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proc. IEEE/CVF CVPR.*

Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. 2002. Geometry images. In *Proc. SIGGRAPH 2002.* 355–361.

Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. 2000. Normal meshes. In *Proc. CGIT.* 95–102.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Trans. Graph.* 38, 4 (2019).

Philipp Herholz, Wojciech Matusik, and Marc Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum* 34, 2 (2015).

Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition. *ACM Trans. Graph.* 33, 6, Article 213 (2014).

Mark W Jones, J Andreas Baerentzen, and Milos Sramek. 2006. 3D distance fields: a survey of techniques and applications. *IEEE Trans. Vision and Computer Graphics (TVCG)* 12, 4 (2006), 581–599.

Brian Karis, Rune Stubbe, and Graham Wihlidal. 2021. A Deep Dive into Nanite Virtualized Geometry. In *ACM SIGGRAPH Talks.*

Benjamin Keinert, Matthias Innmann, Michael Sänger, and Marc Stamminger. 2015. Spherical Fibonacci Mapping. *ACM Trans. Graph.* 34, 6, Article 193 (2015).

Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. 2000. Progressive geometry compression. In *Proc. CGIT.* 271–278.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *Proc. IEEE/CVF CVPR.*

Wan-Chiu Li, Nicolas Ray, and Bruno Lévy. 2006. Automatic and interactive mesh to T-spline conversion. In *Proc. SGP (SGP '06).* 191–200.

Yuanzhan Li, Yuqi Liu, Yujie Lu, Siyu Zhang, Shen Cai, and Yanting Zhang. 2022. High-fidelity 3D Model Compression based on Key Spheres. In *Data Compression Conference.* 292–301.

Connor Lin, Niloy Mitra, Gordon Wetzstein, Leonidas J Guibas, and Paul Guerrero. 2022. NeuForm: Adaptive Overfitting for Neural Shape Editing. In *Advances in Neural Information Processing Systems,* S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 15217–15229. https://proceedings.neurips.cc/paper_files/paper/2022/file/623e5a86fcedca573d33390dd1173e6b-Paper-Conference.pdf

Nathan Litke, Adi Levin, and Peter Schröder. 2001. Fitting subdivision surfaces. In *Proc. VIS'01 (VIS '01).* 319–324.

David P Luebke. 2001. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications* 21, 3 (2001), 24–35.

Andrea Maggiordomo, Henry Moreton, and Marco Tarini. 2023. Micro-mesh construction. *ACM Trans. Graph.* 42, 4 (2023), 1–18.

Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 2015. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–41.

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4 (2017), 71–1.

Duncan Marsh. 2005. *Applied geometry for computer graphics and CAD.* Springer Science & Business Media.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE/CVF CVPR.* 4460–4470.

Luca Morreale, Noam Aigerman, Paul Guerrero, Vladimir G. Kim, and Niloy J. Mitra. 2022. Neural Convolutional Surfaces. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 19333–19342.

Luca Morreale, Noam Aigerman, Vladimir G. Kim, and Niloy J. Mitra. 2021a. Neural Surface Maps. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 4639–4648.

Luca Morreale, Noam Aigerman, Vladimir G. Kim, and Niloy J. Mitra. 2021b. Neural Surface Maps. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 4639–4648.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 1–15.

Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Trans. Graph.* 37, 5 (2018), 169:1–169:15.

Alessandro Muntoni, Lucio Davide Spano, and Riccardo Scateni. 2019. Split and Mill: User Assisted Height-field Block Decomposition for Fabrication. In *Smart Tools and Apps for Graphics (STAG).*

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph* 32, 3, Article 27 (2013).

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-aligned Global Parametrization. *ACM Trans. Graph.* 33, 4 (2014), Article No. 135. http://vcg.isti.cnr.it/Publications/2014/MPZ14

Jan Novák and Carsten Dachsbacher. 2012. Rasterized bounding volume hierarchies. In *Computer Graphics Forum,* Vol. 31. Wiley Online Library, 403–412.

Manuel M Oliveira, Gary Bishop, and David McAllister. 2000. Relief texture mapping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* 359–368.

Stanley Osher and Ronald P Fedkiw. 2005. *Level set methods and dynamic implicit surfaces.* Springer-Verlag.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE/CVF CVPR.* 165–174.

Stephan R. Richter and Stefan Roth. 2018. Matryoshka Networks: Predicting 3D Geometry via Nested Shape Layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Pedro V Sander, Zoë J Wood, Steven Gortler, John Snyder, and Hugues Hoppe. 2003. Multi-chart geometry images. In *Proc. SGP 2003.*

Ryan Schmidt, Cindy Grimm, and Brian Wyvill. 2006. Interactive decal compositing with discrete exponential maps. In *ACM SIGGRAPH 2006 Papers.* 605–613.

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* 231–242.

Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. 2020a. MetaSDF: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems* 33, 10136–10147.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020b. Implicit Neural Representations with Periodic Activation Functions. 7462–7473 pages.

Venkataram Sivaram, Tzu-Mao Li, and Ravi Ramamoorthi. 2024. Neural Geometry Fields For Meshes. In *ACM SIGGRAPH 2024 Conference Papers.* 1–11.

Stanford. 2024. The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/.

Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022a. Variable bitrate neural fields. In *SIGGRAPH 2023 Conference Papers.* 1–9.

Towaki Takikawa, Andrew Glassner, and Morgan McGuire. 2022b. A Dataset and Explorer for 3D Signed Distance Functions. *Comput. Graph. Tools* 11, 2 (2022), 1–29.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proc. IEEE/CVF CVPR.* 11358–11367.

Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. 2023. Compact Neural Graphics Primitives with Learned Hash Probing. In *SIGGRAPH Asia 2023 Conference Papers.* 1–10.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020), 7537–7547.

Theo Thonat, Francois Beaune, Xin Sun, Nathan Carr, and Tamy Boubekeur. 2021. Tessellation-free displacement mapping for ray tracing. *ACM Trans. Graph.* 40, 6 (2021), 1–16.

Greg Turk and Marc Levoy. 1994. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques.* 311–318.

Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, et al. 2024. fvdb: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–15.

Brian Wyvill, Eric Galin, and Andrew Guy. 1998. The Blob Tree, Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Implicit Surfaces* 3 (1998).

Jinfan Yang, Chrystiano Araújo, Nicholas Vining, Zachary Ferguson, Enrique Rosales, Daniele Panozzo, Sylvain Lefebvre, Paolo Cignoni, and Alla Sheffer. 2020. DHFSlicer: Double Height-Field Slicing for Milling Fixed-Height Materials. *ACM Trans. Graph.* 39, 6 (2020).

Lei Yang, Yongqing Liang, Xin Li, Congyi Zhang, Guying Lin, Alla Sheffer, Scott Schaefer, John Keyser, and Wenping Wang. 2023. Neural Parametric Surfaces for Shape

Modeling. arXiv:2309.09911 [cs.GR]

Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. 2022. Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields. In *Proc. ICLR*.

Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. 2023. D^{2}CSG: Unsupervised Learning of Compact CSG Trees with Dual Complements and Dropouts. In *Advances in Neural Information Processing Systems*, Vol. 36. 22807–22819.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. (2016).

A   Method Implementation Details.

Network Architecture. For our experiments, we represent DHFs as MLPs with 5 hidden layers; all activation functions are sinusoidal except for the last layer, which has no activation functions. We allow the width of each layer to change depending on the desired tradeoff between compactness and accuracy; in our experiments, we used four different settings for layer width (32, 48, 72, 96) depending on the desired parameter count. We encode HFs as two MLPs; one MLP for the height function, and one smaller MLP for the mask. The HF uses 5 hidden layers, with four different settings for layer widths (16, 24, 36, 48) depending on the target parameter count. We note that, for a given compression setting, the DHF layer width is exactly double the HF layer width. The mask MLP returns a binary function {0, 1} which indicates whether the input point is inside or outside the domain Ω; internally, the MLP returns a value clamped by a sigmoid function to lie between [0, 1], and we say that a point is inside the mask if the MLP returns 0.5 or greater. The mask MLP has 3 hidden layers, with 16 as its width for all experiments.

All parameters in our experiments are represented as 32-bit floats. All modules are trained with an ADAM optimizer, using a learning rate that starts at 10^{-3} and decays to 0 using a cosine annealing scheduler. In each iteration we randomly select 500k points from the set of sampled points, with 250k in the surface region Ω and the other half outside it. For HF networks, we further split the surface training samples between 125k in $\tilde{\Omega}$ and 125k in $\Omega \setminus \tilde{\Omega}$.
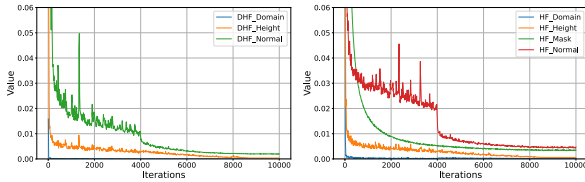

Fig. 23. Changes in DHF and HF losses during the training process.

For the Max Planck model in our dataset, which uses one DHF and one HF module, we plot the changes of losses during training process in Fig. 23.

ESI Computation Parameters. We set $\varepsilon_1 = 10^{-4}$ and $\varepsilon_2 = 10^{-3}$ empirically to prioritize volumetric approximation over surface coverage. We use 10000 sample points for input and VE discretizations.

B   Application Implementation Details.

We provide implementation details of our geometry processing applications below.

Raytracing. Raytracing requires two operations: computing ray-surface intersections; and computing normals at the points of intersection. While *any* ray-surface intersection algorithm can be used, we use a simple strategy that provides guaranteed error bounds for our evaluation, based on subdivisions. We evenly sample n points on a given ray and check their occupancy values. We then find the first segment whose occupancy changes, starting from the source point. We refine this segment by evenly subdividing it into n points and repeat this process. In our quantitative evaluations, we set n=200 and repeat this process three times such that the final segment is sufficiently small.

Texture Atlassing. We can use the parametric representation underpinning NESI by exploiting the mapping from the parameter domains Ω_i to store input surface signals in parameter space (Fig. 9). Given a textured object, we first use the explicit mapping from points on the surface to the explicits that contain that point to save the surface signal in an atlas. Users can either store the atlases for the entire HF surface parameter domain Ω_i, see cow atlas (row one), or store only the texture for the restricted domains $\tilde{\Omega}_i$, see duck and Buddha (rows two and three). The later approach allows for additional memory savings. At render time, our renderer identifies the explicit(s) that cover the given point and fetches the signal from the atlas. In the examples in Fig. 9 we use this approach to transfer texture from a mesh to the NESI representation of this mesh, and subsequently render it by raytracing the NESI.

Meshing. Many downstream applications for neural surface representations require converting them back to a mesh-based format, with the most common approach being an isosurface extraction step (e.g. marching cubes). Because NESIs are defined by explicit surfaces, we can mesh them by exploiting their parametric domain representation, producing superior outputs.

We start by generating a watertight mesh of the DHF hull by creating a grid of points covering Ω_0, and directly generate a mesh M_0 from the DHF hull by assigning DHF hull height values f_a to points on the grid and connecting points with triangles in the obvious manner. We repeat this process for f_b, then stitch the two sides together. To ensure that the DHF hull mesh is high quality, we remesh grazing regions (triangles whose normal with respect to the DHF axis is greater than 70°) that join the two sides of the DHF using an isotropic remesher [Botsch and Kobbelt 2004]. Then for each HF_k, we generate a mesh proxy M_k that represents the bounded volume between $z = 0$ and the height values within the HF's surface domain. We then create a final mesh by incrementally computing the CSG difference operation of $M_0 \setminus M_1 \cdots \setminus M_k$. To avoid discretization issues during subtraction in regions where the surface is represented by the DHF and one or more HFs, before performing the subtraction task we find any vertex on M_k that is also covered by M_0 and offset it slightly inwards along its normal. As shown in the paper our meshes closely approximate the input geometry.

C   Extension: Handling (Nearly)-Occluded Surfaces

By default, NESI targets approximation of surfaces that are visible from some points outside the input shapes. This is consistent with the consumer facing applications we target, where users see content
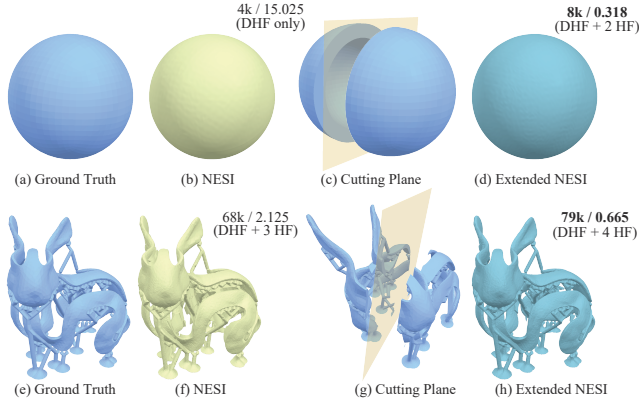
Fig. 24. Our extended method combines the DHF of the input with HFs computed on intersections of the input with half-spaces (c,g) to obtain numerically more accurate results (d,h) than those produced by the standard method (b,f). At the same time, from outside the rendered NESI (b,f) and extended NESI (d,h) results appear practically identical to one another, and to the input (a,e).

rendered from outside. As such, NESI is not designed for capturing interior voids or surfaces which are only revealed when a shape is cut open (Fig 24c).

In cases where capturing such surfaces is important for application purposes (e.g. renderings from inside such voids) one can use the following BSP-like strategy to capture them. Specifically, the proposed strategy subdivides the original shape by cutting planes separating pairs of occluded/occluder elements. Starting from the input surface, we test visibility along our candidate DHF directions by sampling multiple points on each face, shooting a ray along the DHF axis direction, and checking for intersections. A face is said to be *occluded* if it is not visible from any direction; that is, no matter what candidate direction a ray follows from points on the face, it always hits some other part of the surface geometry. We then find the weighted average of all occluded face vertices, weighted by face area, to find a centroid for a cutting plane; then select a normal direction for the cutting plane from our candidate DHF directions which eliminates the most occlusions.

We then add two new HFs to the NESI whose directional axis is the normal of the cutting plane but in opposing orientations, and that are situated such that the $x-y$ plane of the HF (i.e. where $z = 0$) lies on the cutting plane. Finding sample points for the HF and training then proceeds as usual. When using these HFs for in/out tests, we allow the intersection test to pass if one of the following two conditions are met: either $z \le f(x, y)$ (the standard criterion); or $z \le 0$, in which case the point being tested is on the other side of the cutting plane, and so this HF should not be considered when testing this point. Additional HFs are added to each half using the same process as for the standard method. This process can be repeated if multiple cuts are necessary. In practice, we offset the plane locations by a slight offset ε along the cutting plane directional axis so that the two HF regions overlap. Inserting HF planes in this manner and retaining a common DHF hull, rather than subdividing the NESI

explicitly into two, eliminates the risk of false negatives without introducing false positives when performing in-out queries.

Figs. 24 and 22 compare the results quantitatively with and without the extension for shapes with occluded surfaces. This extension allows NESIs to capture even highly complicated geometry such as the scaffolded bunny (Fig. 22) which has many regions that are invisible or barely visible from outside. At the same time, the results with and without the extension look practically identical when rendered from outside. We note that all numerical results reported in the paper, as well as all other visuals, do *not* include this extension.

D Algorithm and Comparison Settings

Data Sourcing. We evaluate NESI on 100 diverse shapes taken from four publicly available datasets, chosen to represent a broad range of shapes commonly used in graphics applications and related prior art. We include the Thingi32 subset of Thingi10k [Zhou and Jacobson 2016] (32 shapes), used by several prior methods for neural shape representation (e.g. [Li et al. 2022; Takikawa et al. 2021]), to provide comparison with shapes chosen by the baseline methods. We include 40 semi-randomly selected objects from the ABC [Koch et al. 2019] dataset of CAD models; as this dataset contains many basic (e.g. cubes or cylinders) and repetitive shapes, we curate the subset to keep only relatively complicated shapes and avoid repetition. We also use 25 objects from the DHFSlicer [Yang et al. 2020] paper which first introduced the notion of DHF surfaces and experimented with decomposing diverse shapes into DHF blocks. Finally, we include 3 canonical, complex scanned shapes from the Stanford 3D Scanning Repository [Stanford 2024]: the *Buddha*, *Thai statue*, and *David* models. To evaluate performance of the ESI representation without neural encoding, we augment this data set with 122 additional models from the Thingi10k dataset, chosen at random; and the data set of Myles et al. [2014], consisting of 98 shapes commonly used in computer graphics.

Two additional models are used to illustrate different aspects of the method: the 'spot' cow [Crane et al. 2013] (Fig. 10 in our paper) and the scaffolded bunny (Fig 24, from the *thingiverse* model repository under Creative Commons Attribution).

Parameter Counts: NESI and Alternatives. We test our outputs with four network size settings for DHF and HF networks: (1) DHF network with 4390 parameters and respective HF network with 1766 parameters; DHF network with 9654 parameters and respective HF network with 3110 parameters; DHF network with 21390 parameters and respective HF network with 6088 parameters; and DHF network with 37734 parameters and respective HF network with 10214 parameters. Our overall parameter counts depend on the number of HFs used; at our lowest DHF model size, the overall model sizes vary from 4390 (zero HFs) to 9688 (3 HFs), and at the highest model size from 37734 (zero HFs) to 68376 (3 HFs).

For NGLOD and VQAD we generated models at levels of detail 1 through 4, where their method determines the parameter counts per model based on level of detail. For SIREN we used parameter counts of: 4801, 6841, 9241, 12001, 16241, 21121, 26641, 32801, 39601, 47041, 53041, 61601, and 75641. For [Yifan et al. 2022] we used parameter counts of 6K, 14K, 18K, 25K, 39K, 56K, and 100K. For NeuRBF we use hash map sizes from 6 to 11. To compare against Sivaram *et al.*[2024]
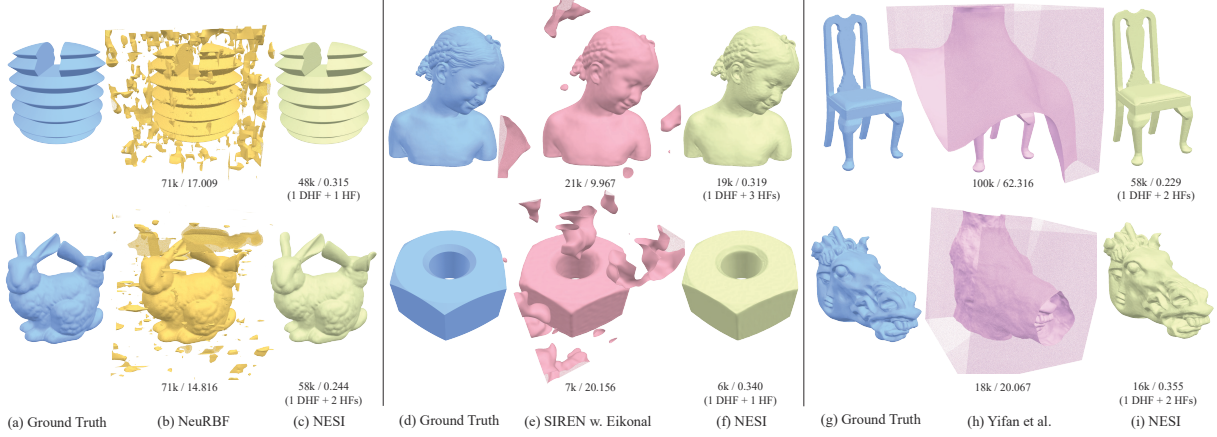
Fig. 25. Additional comparisons of NESI outputs to NeuRBF [Chen et al. 2023b], SIREN with Eikonal constraints [Sitzmann et al. 2020b], and Yifan *et al.* [2022]. While NESI results are consistently stable, these prior methods generate spurious extra surfaces on a significant percentage of inputs.

Dataset	#Params	1 DHF	1 DHF + 1 HF	1 DHF + 2 HFs	1DHF + 3 HFs
ABC	<10k	32	8	9	7
Thingi32	<10k	0	2	18	12
Others	<10k	10	4	8	11
All	**<10k**	**42**	**14**	**35**	**30**
ABC	10k-20k	0	8	9	7
Thingi32	10k-20k	0	2	18	12
Others	10k-20k	0	4	8	11
All	**10k-20k**	**0**	**14**	**35**	**30**
ABC	20k-40k	32	8	9	7
Thingi32	20k-40k	0	2	18	12
Others	20k-40k	10	4	8	11
All	**20k-40k**	**42**	**14**	**35**	**30**
ABC	>40k	0	8	9	7
Thingi32	>40k	0	2	18	12
Others	>40k	0	4	8	11
All	**>40k**	**0**	**14**	**35**	**30**
Overall		*84*	*56*	*140*	*120*

Table 8. DHF/HF counts for each of the 400 encoded NESI models in our experiments, broken down by parameter counts and data set. The parameter counts of NESI shapes are a function of both HF count and MLP size, leading to different HF count distributions across the four parameter ranges (notably only the <10K and 20K-40K ranges contain shapes represented using single DHFs).

(NGF), we adjust their "mesh LOD" parameter, which in turn adjusts the target number of faces during their QSLIM operation. Numbers used for mesh LOD are taken from their paper, except for 500 and 2000, which we added to have parameter counts that are closer for comparison. We only change NGF's MLP width when comparing on less than 12k parameters.

Comparison Setting All renderings of both our and alternative results were generated via our raytracing code and colored using a flat shading scheme. We quantitatively evaluate our method through the standard protocol [Takikawa et al. 2021] of measuring Chamfer-L_1 distance, measured using 5 million well-sampled points on the input mesh and the output models. We opt for a high sampling rate in order to capture surface-to-surface distances as accurately as possible. Direct comparison across methods with different parameter counts is challenging; we therefore report two sets of numbers in the main paper: the accuracy of each method for models within a fixed parameter range (e.g. under 10K, or 10k to 20K); and pairwise comparisons where, for each output of our method, we algorithmically select the

alternative method output with the smallest parameter count that is larger than, or equal to, ours. The latter strategy evaluates the percentage of inputs on which our method outperforms the alternative, while the former quantifies the magnitude of improvement. Both approaches show that that our method significantly outperforms all baselines given the same parameter count. Table 8 breaks down the number of HFs used for each input source and each of the parameter ranges in our comparison tables (Tab 5, 6).

E   Additional Visual Comparisons

Fig. 25 shows additional comparisons of NESI outputs to those of NeuRBF [Chen et al. 2023b], SIREN with eikonal constraints [Sitzmann et al. 2020b], and Yifan *et al.* [2022], demonstrating these approaches' lack of robustness.