DOI: 10.1111/cgf.70326

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Basis Networks: Learning basis functions for free-form triangulations

T. Djuren[1] and M. Alexa[1]

[1]TU Berlin, Germany

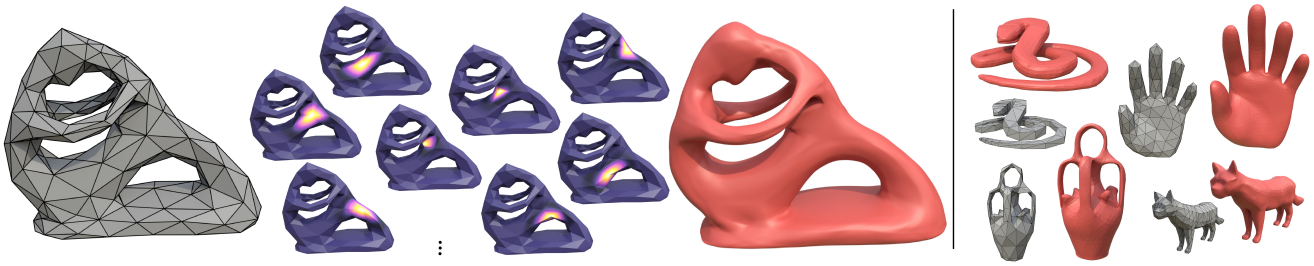


Figure 1: *Many parametric surfaces can be represented using a set of control points (gray), each equipped with a basis function (center of left). We propose to derive the basis function from a dataset using a small multilayer perceptron network. Without retraining, this allows to construct tangent continuous parametric surfaces (red) for various control meshes, regardless of genus. Snake by dercruz and cat by Vr-cvantorium, both under CC-BY.*

Abstract

We present a framework for learning compactly supported basis functions that define tangent continuous surfaces based on coarse irregular triangle meshes. The basis functions are represented as MLPs. Smoothness of the basis functions is achieved by using the values of Loop basis functions as the parameterization of the surface. Post-multiplying the value of the MLP with the Loop basis yields smooth compact support. We show that this approach works similar or better than Neural Subdivision in terms of recreating given geometry, while the runtime scales better with surface resolution and can be evaluated at arbitrary resolution.

CCS Concepts

• ***Computing methodologies*** → ***Parametric curve and surface models;*** *Machine learning approaches;*

1. Introduction

The classic way to represent a surface is through a net of *control points* that are used as weights for *basis functions*. Any point on the surface can be evaluated by evaluating the basis functions at a given parameter value and then multiplying these values with the associated control points. For example, the Bernstein basis leads to Bézier curves and, by using tensor products, to Bézier surface patches [FH00]. BSpline or NURBS surfaces are constructed in the same fashion, only using different basis functions.

Tensor products, however, severely restrict the topology of the control net. The possibility of evaluating spline surfaces by repeated subdivision has led to the extension of splines to irregular control nets, by defining appropriate subdivision rules also in the vicinity of irregular vertices [CC78]. The surface is then implicitly defined as the limit of infinitely repeated subdivision. It is clear that a *stationary* subdivision scheme that converges to a continuous surface effectively also corresponds to a set of basis functions, possibly different depending on the local combinatorics of the control mesh. Stam [Sta98b, Sta98a] was the first to demonstrate that, at least for linear subdivision rules, these basis functions can be efficiently evaluated, without actually performing the subdivision.

The connection between basis functions, whether defined by polynomials or by subdivision rules, and the resulting surface is fixed. In many applications, it would be desirable to *learn* the connection between the coarse control mesh and the surface defined by the control mesh from data. Liu et al. [LKC*20] demonstrated the possibility of learning subdivision rules that explain pairs of coarse control meshes and a corresponding detailed surface using a neural

net, aptly termed *Neural Subdivision*. The central idea of our work is to bypass the construction of a subdivision scheme and directly learn basis functions from the data. Using a particular construction, we can show that standard multi-layer perceptrons (MLP) using Gaussian Error Linear Unit (GELU) [HG16] activation functions to provide smoothness are sufficient for this task.

A potential problem with learning subdivision rules is that it is difficult to encode convergence or, assuming convergence, continuity of the resulting surface. In fact, the subdivision rules derived from the learning process are generally not converging, so there is effectively no surface being defined by the subdivision scheme [LKC*20]. In practice, this means subdivision has to be limited to roughly the level of detail that has been used for training. A major benefit of our approach is that *any* basis function defines a surface. One of our core contributions is a construction that leads to surfaces that are curvature continuous except for irregular vertices, where it is still tangent continuous. The 'trick' we use to achieve this is to use the basis functions of Loop subdivision [Loo87], evaluated using Stam's method [Sta98a], in two ways: (1) we use the values of the Loop basis as *input* for the MLP, and (2) we multiply the values generated by the MLP again with the value of the Loop basis. The first use of the Loop basis fixes the problem that common parameterizations of a mesh, such as barycentric coordinates, are not smooth across edges, i.e., are continuous but not continuous in derivatives. The second use of the Loop basis makes sure that the resulting basis functions are not only locally supported, but have zero derivatives at the boundary of the support. In addition to these *requirements* for defining a tangent continuous surface based on a coarse control mesh, the use of Loop basis functions also leads to natural symmetries of the basis functions for local neighborhoods that have combinatorial symmetries.

In the following, we first recall some basics of surface representations with bases for free-form triangulations and how to evaluate basis functions implicitly defined by linear subdivision (Sec. 2) as well as context for our work in terms of relevant subdivision techniques and approaches for learning parametric representations (Sec. 3). Based on this background, we develop the details for our approach in Sec. 4, including network design and generating input and output of the network. As we demonstrate, this construction allows for the enforcement of desirable properties for the predicted surface by appropriately constraining the basis functions (Sec. 5). We demonstrate the versatility of basis networks and, in particular, show that they generally outperform Neural Subdivision in Sec. 6. This includes the potential application as a surface scheme that more closely approximates input meshes than Loop subdivision, while producing fewer artifacts than Modified Butterfly subdivision. In this work, we focus on basis functions that depend only on the combinatorics of the control mesh, leaving conditioning of the basis on additional information for future work. We discuss this aspect and other limitations or possible extensions in Sec. 7. Source code and trained models will be made available along with the publication.

2. Background

One of the most popular methods to refine coarse triangle meshes

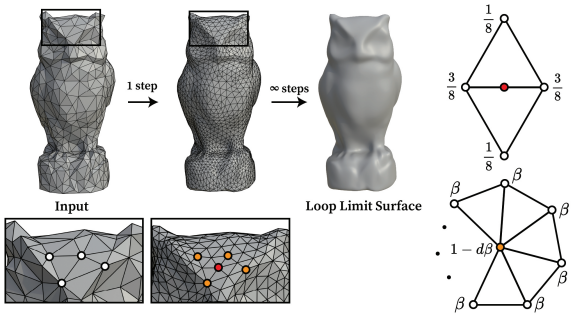


Figure 2: *In each step Loop subdivision [Loo87] inserts new vertices (red) along edges. All positions are computed as weighted averages of the vertices in the previous step using the stencils on the right. The stencil for updating previous vertices uses* $\beta = \frac{1}{d}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{d})^2)$ *where d is the vertex degree. Owl by Horton under CC-BY.*

is Loop subdivision [Loo87]. In each subdivision step, each triangle is split into four smaller triangles by inserting new vertices along the edges. All vertex positions are calculated as linear combinations of the vertices in the previous step. Repetition of the subdivision steps for an infinite number of times leads to a tangent continuous *limit surface* (Fig. 2).

If the vertex positions at each step are a linear combination of the previous vertices, then each point on the limit surface after an infinite number of subdivisions will also be a linear combination of the input vertices (if the scheme converges). The weights of the linear combination for the limit surface are the values of the *basis function*. Each vertex in the input mesh can be assigned a scalar basis function that describes how much that vertex contributes to the final surface. This is a slightly different perspective on subdivision: With basis functions, we can directly evaluate points on the limit surface, without having to go through many iterative subdivision steps. For each face j in the input mesh, we can evaluate the refined face patch $\mathbf{s}_j$ by summation over all vertices $\mathbf{c}_i$ multiplied by the basis value b_{ij} in that face. In practice, only vertices i that have that face j in their local support $\mathcal{N}(j)$ must be considered:

$$\mathbf{s}_j(\mathbf{u}) = \sum_{i \in \mathcal{N}(j)} \mathbf{c}_i b_{ij}(\mathbf{u}),\ \mathbf{u} = (u,v,w),\ u,v,w \geq 0,\ u+v+w = 1 \tag{1}$$

We will omit the barycentric coordinate $\mathbf{u}$ and index j when we are not referring to a specific refined point of a coarse face. In matrix notation, a set of m points $\mathbf{S} \in \mathbb{R}^{m\times 3}$ on the refined fine surface can then be computed by matrix multiplication with a sparse basis matrix $\mathbf{B}$ with the points of the input mesh $\mathbf{C} \in \mathbb{R}^{n\times 3}$:

$$\mathbf{S}^\top = \mathbf{C}^\top \mathbf{B} = [\mathbf{c}_0 \cdots \mathbf{c}_{n-1}]^\top [\mathbf{b}_0 \cdots \mathbf{b}_{m-1}] \tag{2}$$

Each sparse basis vector $\mathbf{b} \in \mathbb{R}^n$ contains all basis values to evaluate a point on the limit surface. The values of the basis function for each vertex are in the respective rows of $\mathbf{B}$.

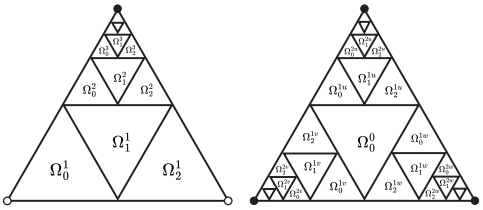


Figure 3: *For semi-regular tilings (left) the barycentric coordinates are reparameterize according to the subdivision level s and tile k to a new domain Ω_k^s [Sta98a]. We slightly extend this to irregular meshes (right) by reparameterize using multiple sets of infinite tilings $\Omega_0^0, \Omega_k^{su}, \Omega_k^{sv}, \Omega_k^{sw}$.*

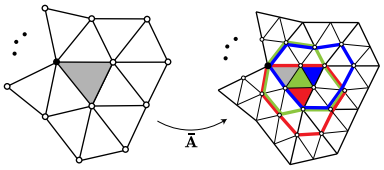
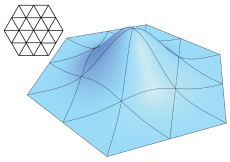


Figure 4: *Each subdivision step inserts new regular vertices (white) around the irregular vertex (black). Then, the limit surface for three of the four newly created triangles (red, green, and blue) can be evaluated because all of their vertices have become regular. The picking matrices $\mathbf{P}_k$ select the 12 points in the red, green or blue support to evaluate the box spline. Note, the subdivision matrix $\bar{\mathbf{A}}$ only needs generate points in these supports.*

For Loop's scheme, all basis functions are piecewise polynomial functions. They are locally supported and defined in patches over the barycentric coordinates of each face in the 2-ring face neighborhood around a vertex. For regular input meshes (all vertices have degree 6), the Loop basis function corresponds to a quartic box spline basis (see inset). Then there are exactly 12 input points that influence a point inside a refined face patch according to the box spline basis vector $\mathbf{b}^B \in \mathbb{R}^{12}$ given in Stam [Sta98a].



To compute basis functions for irregular meshes we briefly summarize the idea of Stam [Sta98a, Sta98b] and start with basis functions for semi-regular meshes. In a semi-regular mesh, each triangle has at most one irregular vertex (with degree $\neq 6$). By applying one step of Loop subdivision to the triangle, three new triangles are created that all have only regular vertices (Fig. 4). For these triangles, we can again evaluate the limit surface and the basis using the box spline basis. For the remaining triangle, we repeat Loop subdivision until the hole around the irregular vertex is small enough. Thus, we can reparameterize the barycentric coordinates for a semi-regular face into a set of infinite tiles (Fig. 3, left). Each row of tiles requires a different subdivision level. Since the Loop subdivision operator at subdivision level s can be expressed as product with subdivision matrices $\bar{\mathbf{A}}$ and $\mathbf{A}$ (specifically $\bar{\mathbf{A}}\mathbf{A}^{s-1}\mathbf{C}$ see Stam [Sta98a] for details), each point on the limit surface $\mathbf{s}$ can be written as sequence of matrix multiplications up to the desired precision that gives the control points for a box spline:

$$\mathbf{s} = \mathbf{C}^\top (\bar{\mathbf{A}}\mathbf{A}^{s-1}\mathbf{P}_k)^T \mathbf{b}^B = \mathbf{C}^\top \mathbf{b}^L \tag{3}$$

where $\mathbf{P}_k$ is a picking matrix consisting of zeros and ones that select the corresponding control points for the box spline depending on which of the three regular triangles the barycentric coordinate of $\mathbf{s}$ belongs to (Fig. 3, left). s and $k \in \{0,1,2\}$ are the number of subdivisions and tile according to Fig. 3 (left). Stam [Sta98a] gives explicit expressions for the subdivision and picking matrices and describes how the matrix product can be efficiently evaluated using its eigendecomposition.

Irregular meshes contain triangles with more than one irregular vertex. For these triangles an additional Loop subdivision $\mathbf{A}_0$ is applied that generates up to three semi-regular triangles and one or more regular triangles for which the bases and limit surface are known. In these cases, we suggest using another reparameterization with multiple sets of infinite tiles (Fig. 3, right). We then use another picking matrix $\bar{\mathbf{P}}_l, l \in \{0,1,2,3\}$ selecting the necessary points that are either the control points for a box spline (e.g. inner triangle) or control points processed with Stam's method [Sta98a] (e.g. triangles at the corners). This allows evaluating basis functions on faces in irregular meshes without first generating a semi-regular mesh by subdivision.

The number of entries in the Loop basis vector $\mathbf{b}^L$ depends on the topology. If there is only a single irregular vertex of degree d, as in [Sta98a], there are $d+6$ basis values in $\mathbf{b}^L$. In general, for a triangle with (multiple, possibly irregular) vertices of degrees d_0, d_1 and d_2, the number of bases is $d_0+d_1+d_2-6$ (for $d_i > 3$). We treat $\mathbf{b}^L$ as a sparse vector that contains the same number of elements n as there are vertices in total.

Stam's approach allows to evaluate basis values that are only close to irregular vertices. For Loop subdivision, the limit positions at the vertices can be computed by evaluating the same stencil (right bottom of Fig. 2) but with β replaced by $\chi = \frac{1}{3/8\beta+d}$ [Zor00].

3. Related Work

Subdivision surfaces. Probably one of the most popular surface representations that fit into the basis framework are linear subdivision surfaces. Starting on a coarse *control mesh*, successive iterations of topological rules on how faces are split into finer faces, and geometrical rules that describe how new vertices are inserted or old vertices are updated describe a sequence of increasingly finer meshes. Famous subdivision schemes are the Catmull-Clark subdivision [CC78] and the Loop subdivision scheme [Loo87]. Both produce C^2 continuous limit surfaces for regular meshes with the cubic BSpline and triangular box spline basis, respectively. On irregular meshes that contain vertices with degrees other than 4 or 6, the basis functions are given explicitly using the method of Stam [Sta98b, Sta98a]. Besides Catmull-Clark and Loop subdivision, there are numerous other surface subdivision schemes like $\sqrt{3}$-subdivision [Kob00], 4-8-subdivision [VZ01] or vertex

split schemes like Doo-Sabin subdivision [DS78] and midedge schemes [PR97, HW99]. If interpolation of the input vertices is desired, then there is (Modified) Butterfly subdivision [DLG90, ZSS96] for triangle meshes or the Kobbelt scheme for quadrilaterals [Kob96]. The basis functions for both schemes are not polynomial anymore. A known issue with these interpolating methods is the lower degree of smoothness and decreased surface quality, especially near extraordinary vertices [Zor00].

All the aforementioned schemes are *stationary*, as they use subdivision rules that do not depend on the subdivision level. There is also a broad area of *non-stationary* schemes which produce limit surfaces that also have basis representations. Examples for bi-variate non-stationary schemes that reproduce quadrics are in [NRY16, FMW14, JSD03, LY10].

Schemes that do not have a basis representation are the non-linear subdivision schemes. Examples are the Canonical Möbius subdivision that produces surfaces that are as spherical as possible [VMW18] or the Gaussian-Product subdivision surfaces that use covariances at vertices for improved modeling of (semi-)sharp features and concavities [PBW19].

For a more in-depth introduction to subdivision surfaces, we refer to overviews [Zor00, CD21] and detailed books on this topic [PR08, FHK02].

Learned parametric surfaces. The idea of deriving the surface refinement scheme from a dataset is not new. Most related to our approach is Neural Subdivision [LKC*20] which is a non-linear subdivision scheme. They optimize a set of three different neural networks that predict vertex offsets after each subdivision step. For multiple subdivisions, the networks are recursively applied. Since the networks are only trained for a few recursive subdivision steps, the resolution is limited, and the scheme generally does not converge to a limit surface. Learning the basis function avoids recursive evaluation and also ensures that a limit surface exists. Neural Subdivision was successfully used to progressively transmit mesh data over a network [CKAJ23].

A high level of interest has developed over the last years in encoding parametric surfaces with network architectures. Pioneering, patch-based, learned surface representations include FoldingNet [YFST18], which learns a parametric mapping from a planar domain to a surface, and AtlasNet [GFK*18], which represents surfaces as collections of parametric surface elements. Numerous works further extended FoldingNet using differential quantities [BPG*20] and the AtlasNet approach by learning sets of primitives [DGF*19], by fully learnable parametric domains [LL22], to improve the quality by reducing the overlap of different patches of AtlasNet [GWM21] or for surface reconstruction from point clouds [WSS*19]. Another approach are the Neural surface maps [MAKM21]. They generate a parametric surface by learning a mapping from a given parametrization on the unit disk to a surface in 3D. Similarly, Williamson and Mitra [WM25] learn a mapping from a parametrization on the sphere to a surface of genus-0 to compute various differential quantities.

Rather than using disconnected patches in the plane for parameterization, an alternative approach is to use a feature complex given as a coarse mesh with feature vectors at the vertices. Neural parametric surfaces [YLL*23] learn a mapping from interpolated vertex features of a feature complex to fine vertex positions. Remarkably, this representation can also learn a morphable shape space for fixed connectivity. Recently, Sivaram et al. [SLR24] and Pentapati et al. [PPB25] demonstrated that a patch based network using this representation can achieve state-of-the-art results for mesh compression.

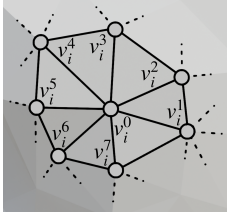
While these methods overfit a network to a single shape (or a shape with fixed connectivity), we discuss learning a tangent continuous parametric surface scheme for multiple meshes with different shape and topology.

4. Basis Network

Many subdivision methods like Loop or Catmull-Clark subdivision [Loo87, CC78] assume a fixed piecewise polynomial basis independently of the input geometry. Instead of polynomial functions we make use of a small multilayer perceptron network (MLP) that predicts a scalar value for every fine surface point w.r.t. all coarse input points in the respective local support (i.e. all non-sparse entries of $\mathbf{B}$). Since we expect the basis functions for the entire input to be similar in shape and style, we use the same network to encode all basis functions, using the same weights for all basis predictions. In the following, we consider triangular input meshes and, like most linear subdivision schemes, we also keep the basis functions independent of the geometry.

Local Support. Equivalent to Loop subdivision, we define the local support of a basis for a vertex as the 2-ring face neighborhood. This drastically reduces the number of basis functions that have to be predicted to compute a point on the fine surface. When refining a face of the input mesh, only basis functions of vertices that have that face in their local support must be considered. Given a barycentric coordinate and a face index, we first compute the Loop basis values $\mathbf{b}^L \in \mathbb{R}^n$. For each non-zero entry of $\mathbf{b}^L$, a new scalar basis value is then computed using a network.

Input features. Each scalar basis that need to be computed is associated to a vertex in the coarse input mesh. Intuitively, one could smoothly parameterize the 2-ring face neighborhood for this vertex and use the coordinate in the parameterization as input feature. However, parameterizing the face neighborhood for every vertex while also considering continuity conditions between adjacent patches in the parameterization for smooth bases can be expensive.



Instead, we suggest using the Loop basis values of this vertex and its adjacent vertices as input features. The values of the feature vector is a subset of $\mathbf{b}^L$ that is reordered into a vector $\mathbf{v}_i = (v_i^0, \cdots, v_i^{d_{max}})$ according to the basis i to be predicted and a predefined maximum vertex degree d_{max}. The ordering starts with the basis of the associated vertex, followed by the bases of the first ring of adjacent vertices in counterclockwise direction (beginning with one vertex with the highest degree, see inset). Because the input vector

must be of fixed length, all remaining entries are filled with zeros. This feature vector encodes the local connectivity and forms a smooth input for all possible 2-ring face neighborhoods without explicitly considering continuity conditions. In addition, the Loop basis can be computed efficiently [Sta98a]. This only amounts to computing a linear combination of the quartic box spline basis.

Note that for meshes with boundaries, the Loop basis is well defined, since there are well-known rules for Loop subdivision with boundary vertices [ZK02, HDD*94, BLZ00].

Network architecture. We use a single, fully connected network with a scalar output. Input of the network is a feature vector $\mathbf{v}_i \in \mathbb{R}^{d_{max}+1}$ of Loop basis values with maximum vertex degree d_{max}, i.e. the network is a function that maps $\mathbb{R}^{d_{max}+1} \to \mathbb{R}$. As non-linearity between network layers, we use Gaussian Error Linear Units (GELU) [HG16] as a smooth alternative to rectified linear units. This is because we require smooth basis functions to generate a tangent continuous surface.

5. Basis Constraints

The output of a basis network from Sec. 4 is already sufficient to compute points on a refined surface (Eq. 1). However, properties such as continuity of the refined surface are often highly desired. The prediction of basis values allows to enforce various attributes of the generated surface by using well known properties of basis functions.

Parametric Continuity & Loop Initialization. Most times it is required that all refined input triangles form a continuous surface. Given that the refined surface is a linear combination of local basis functions, we ask for n-times continuously differentiable basis functions and for vanishing n-th derivatives at the boundary of the support to archive C^n continuity.

When using linear layers with GELU activations and the Loop basis as input, then the basis functions are 2-times continuously differentiable (except at irregular vertices where only the first derivative is continuous). To ensure vanishing derivatives at the boundary for all bases, we multiply the network output with the Loop basis $b_i^* = b_i^L \,\mathrm{MLP}(\mathbf{v}_i)$. Note that the Loop basis is strictly positive inside the support. Then

$$D_{\mathbf{a}} b_i^* = \left(D_{\mathbf{a}} b_i^L\right) \mathrm{MLP}(\mathbf{v}_i) + b_i^L \left(D_{\mathbf{a}} \mathrm{MLP}(\mathbf{v}_i)\right)$$

where MLP is the multilayer perceptron and $D_{\mathbf{a}}$ is the directional derivative in barycentric direction $\mathbf{a}$, $D_{\mathbf{a}} f(\tau) = \sum_{i=1}^{3} \frac{\delta f}{\delta \tau_i} a_i$ [Far86]. At the boundary is $b_i^L = 0$ and $D_{\mathbf{a}} b_i^L = 0$, which also leads to vanishing derivatives of $D_{\mathbf{a}} b_i^* = 0$.

Learning a rescaling factor for the Loop basis also allows to initialize the network such that the initial surface prediction is identical to the limit surface of Loop subdivision. When initializing the weights of the last layer with zeros and the biases with one, then the initial prediction is constant one (since $GELU(0) = 0$). The Loop limit surface can be a good first guess for surface refinement to reduce convergence time and avoid local minima.

Affine Invariance. Translation, rotation and scaling of the input mesh should not affect the resulting surface. Affine invariance is equivalent to a partition of unity of the basis functions, i.e. the sum of basis values for a refined point should be one $\sum_i b_i = 1$.

In the n-dimensional space spanned by all basis functions, the basis vectors satisfying the partition of unity lie on a hyperplane defined by a point $\mathbf{t} = \frac{1}{n}\mathbb{1} \in \mathbb{R}^n$ and normal $\mathbf{n} = \frac{1}{\sqrt{n}}\mathbb{1} \in \mathbb{R}^n$. We suggest projecting the vector of basis predictions $\mathbf{b}^*$ onto this plane, after multiplication with the Loop basis. As projection direction we use the sparse Loop basis vector $\mathbf{b}^L \in \mathbb{R}^n$ (see Fig. 5 for a low dimensional example). Note that the projection direction is not orthogonal to the hyperplane, so the projection is generally *oblique*. The benefit is that this projection only changes the non-zero entries in the basis vector, preserving sparsity and local support. The projection can be written as follows:

$$\begin{aligned} &\mathbf{b} = \mathbf{T}(\mathbf{b}^* - \mathbf{t}) + \mathbf{t} \\ \text{and} \quad &\mathbf{T} = \mathbf{H}^\top \mathbf{Q}(\mathbf{H}\mathbf{b}^L)\mathbf{H} \end{aligned} \quad \text{with} \quad \mathbf{Q}(\mathbf{d}) = \begin{pmatrix} 0 & 0 & 0 & \\ -d_1/d_0 & 1 & 0 & \cdots \\ -d_2/d_0 & 0 & 1 & \\ & \vdots & & \ddots \end{pmatrix}$$

where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is the Householder reflection $\mathbf{H} = \mathbf{I} - 2/(\bar{\mathbf{n}}^\top \bar{\mathbf{n}})\bar{\mathbf{n}}\bar{\mathbf{n}}^\top$ with adjusted normal $\bar{\mathbf{n}} = \mathbf{n} - (1, 0, \cdots, 0)^\top$. $\mathbf{H}$ aligns the partition of unity hyperplane (shifted to the origin by $\mathbf{t}$) with the hyperplane spanned by all coordinate axes except the first. $\mathbf{Q}(\mathbf{d}) \in \mathbb{R}^{n\times n}$ is a projection onto this axis aligned hyperplane in direction $\mathbf{d} \in \mathbb{R}^n$. The alignment is reversed using the inverse $\mathbf{H}^{-1} = \mathbf{H}^\top$.

The projection $\mathbf{Q}(\mathbf{d})$ with $\mathbf{d} = (d_0, \cdots, d_{n-1})^\top$ is not stable for $d_0 = 0$. Geometrically, this means that the projection direction $\mathbf{d}$ must not be parallel to the projected plane (and $\mathbf{d} \neq \mathbf{0}$). All vectors parallel to the partition of unity plane fulfill $\sum_i b_i = 0$. Since $\sum_i b_i = 1$ for all $\mathbf{b}^L$ and even $b_i \geq 0$, the projection is always stable and properly defined.

Also note that when $\mathbf{b}^*$ already fulfills the partition of unity property, then the oblique projection is equivalent to the identity. With this, the initial prediction remains the Loop limit surface.

An alternative and naive way to enforce the partition of unity is to normalize the basis vector $\mathbf{b} = \mathbf{b}^* / \sum_i b_i^*$. Although this leads to the desired surface properties, we observed poor convergence behavior during gradient descent. Division by the sum of the basis functions leads to unbounded gradients for small basis sums and is undefined when $\sum_i b_i^* = 0$.

Convex Hull. Many applications benefit from the definition of a convex hull that is guaranteed to contain a surface. With bases functions a convex hull is directly available by asking for only positive basis functions, e.g. by mapping the network output through a softplus function (with slightly adapted initialization as $\mathrm{softplus}(0) \neq 0$). When all bases are positive, the refined surface is guaranteed to be contained in the convex hull of the input mesh.

However, when a smooth surface that interpolates the input points is desired, then the convex hull property should not be enforced. There is no surface that can fulfill the convex hull property, interpolates the input points and is smooth at the same time [Pet95].

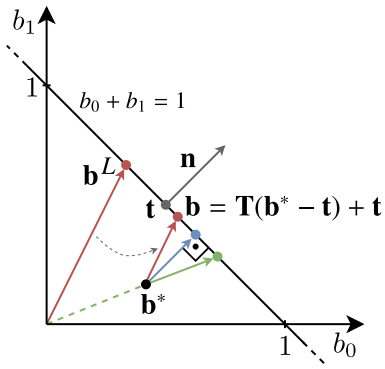


Figure 5: *For $n = 2$, the hyperplane that contains all bases forming a partition of unity, i.e. $\sum_i b_i = 1$, is a line. The Loop basis $\mathbf{b}^L$ is a point on this hyperplane. The network output $\mathbf{b}^*$ is projected onto the hyperplane treating $\mathbf{b}^L$ as projection direction (red). The other transformations shown are the orthographic projection (blue) and normalization (green).*

Algorithm 1: Basis prediction and constraints

Input: $\mathbf{b}^L \in \mathbb{R}^n$ (Loop basis)
Output: $\mathbf{b} \in \mathbb{R}^n$ (new basis)
Gather and reorder values from $\mathbf{b}^L$ into feature vectors $\mathbf{v}_i$;
for every vertex i in support **do**
  $b_i^* \leftarrow MLP(\mathbf{v}_i)$;
  $b_i^* \leftarrow softplus(b_i^*)$; ▷ `Optional: Convex Hull`
  $b_i^* \leftarrow b_i^L b_i^*$; ▷ `Parametric Continuity`
$\bar{\mathbf{n}} \leftarrow \frac{1}{\sqrt{n}}\mathbb{1} - (1, 0, \cdots, 0)^\top$;
$\mathbf{H} \leftarrow \mathbf{I} - 2/(\bar{\mathbf{n}}^\top \bar{\mathbf{n}})\bar{\mathbf{n}}\bar{\mathbf{n}}^\top$;
$\mathbf{T} \leftarrow \mathbf{H}^\top \mathbf{Q}(\mathbf{H}\mathbf{b}^L)\mathbf{H}$;
$\mathbf{b} \leftarrow \mathbf{T}(\mathbf{b}^* - \frac{1}{n}\mathbb{1}) + \frac{1}{n}\mathbb{1}$; ▷ `Affine Invariance`

6. Optimization & Evaluation

In all our experiments, we use a small multilayer perceptron with two hidden layers of sizes 256 and 128. We draw 16 barycentric coordinates from each triangle of the input mesh. Each sample is drawn uniformly from 16 non-overlapping triangular tiles in barycentric coordinates to prevent triangle fold overs. The number of 16 samples is equal to the number of faces after two steps of Loop subdivision. The network weights are optimized using the ADAM optimizer [KB17] with a learning rate of 10^{-3} ($\beta = (0.9, 0.999)$). We summarize the basis computation from Sec. 4 and Sec. 5 in Algorithm 1.

We found that using a simple chamfer loss between predicted surface samples and a fine target mesh produced accurate results (Fig. 6). Although the Chamfer loss could lead to incorrect correspondences between spatially close points, we did not observe these artifacts in our experiments. The basis functions are reused for multiple spatially different vertices during surface computation, possibly regularizing the solution. Also note that the initial guess is identical to the Loop limit surface, which might also avoid mismatches. Other methods, such as neural subdivision [LKC*20], use a quasi-bijective mapping between the surfaces in all subdivision levels [LSS*98,LZBCJ21]. This mapping is not smooth and can be expensive to generate.

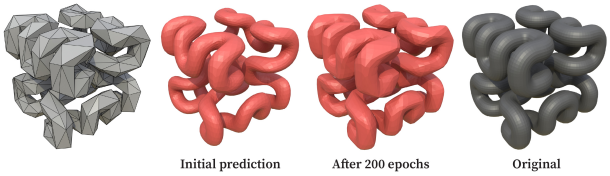


Figure 6: *The initial prediction of the network is identical to the Loop limit surface. This allows to learn complicated structures while avoiding poor local minima. In this example the network is trained to reproduce the Hilbert cube only by the chamfer loss. Hilbert Cube by tbuser under CC-BY-SA.*

6.1. Surface Refinement

To evaluate the surface refinement capabilities, we optimize a scheme that approximates the input mesh more closely than Loop subdivision. Loop Subdivision often gives acceptable results, but suffers from strong smoothing and shrinkage of the input. The goal is to preserve more detail of the input mesh by using a scheme that approximates the input more closely.

We use the TOSCA dataset [BBK08] to generate a dataset of smooth surfaces that serves as train data. The TOSCA dataset contains a set of 80 different models classified into 9 categories. First, all holes in the TOSCA meshes were closed using the hole-filling functionality in Blender [Ble25]. We simplified all models using QSLIM [GH97] to create coarse meshes with roughly 400 vertices. We then generated the Loop limit surface for the coarse TOSCA meshes with roughly 50 samples for each face. These surfaces become target surfaces because Loop subdivision [Loo87] produces visually pleasing results in many cases. As the Loop limit surface does not interpolate the input vertices, we replace all vertices of the simplified meshes with their corresponding positions on the Loop limit surface. We then train a basis network for 200 epochs to reproduce the Loop limit surface using the interpolating input meshes. Because the task is to generate an interpolating scheme, we do not enforce the convex-hull property.

We compare the basis network with Neural Subdivision [LKC*20] – probably the most similar previous method that learns a subdivision refinement scheme from a dataset. We train the Neural Subdivision method on the same smooth dataset using the default setup [LKC*20]. The only difference is that we used their algorithm for simplification [LZBCJ21] of the smooth dataset to generate coarse meshes with roughly 400 vertices, since the method requires a quasi-bijective map generated during simplification. As Neural Subdivision is usually trained for a low number of subdivision steps, we also evaluate the basis networks on at most two subdivision steps for the comparison. Both methods are compared for meshes that were not part of the training data in Fig. 7 (top).

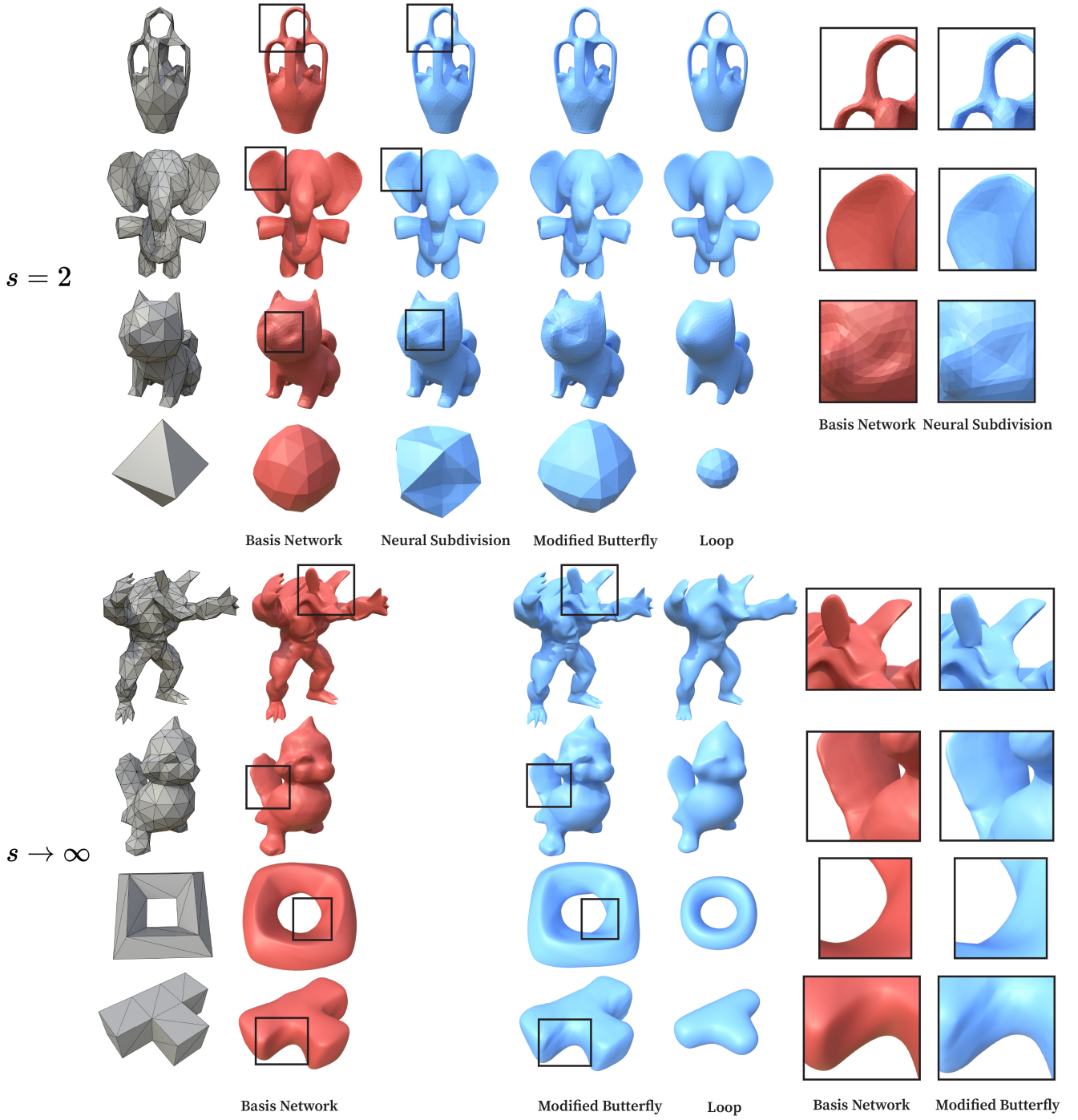


Figure 7: *We apply the basis network, Neural Subdivision [LKC*20], Modified Butterfly Subdivision [ZSS96] and Loop Subdivision [Loo87] to input meshes (gray) of different complexity using two steps of subdivision (top) and $s \to \infty$ (bottom). The basis functions of the network can be sampled at high resolutions and result in a C^1 continuous surface everywhere. Results for higher resolutions using Neural Subdivision are missing since the method is limited to a low number of subdivisions. The basis network avoids artifacts that Modified Butterfly suffers from (Sec. 6.1). This is particularly visible for coarse inputs (bottom, last rows). Loop subdivision leads to both shrinking and strong smoothing of the input. Shiba by zixisun02 under CC-BY.*

Table 1: *We refine simplified meshes from the TOSCA dataset [BBK08] using a basis network, Neural Subdivision (N.S.) [LKC*20], Loop subdivision [Loo87] and Modified Butterfly subdivision (M.B.) [ZSS96]. The two-sided Hausdorff distances and mean surface distances are relative to the bounding box diagonal ($\times 10^2$) and where computed using METRO [CRS98].*

	$\mathbb{H}$ - Hausdorff distance				$\mathbb{M}$ - Mean surface distance			
Category	$\mathbb{H}_{Basis}$	$\mathbb{H}_{N.S}$	$\mathbb{H}_{Loop}$	$\mathbb{H}_{M.B.}$	$\mathbb{M}_{Basis}$	$\mathbb{M}_{N.S.}$	$\mathbb{M}_{Loop}$	$\mathbb{M}_{M.B.}$
Cat	1.9634	3.5864	3.1807	**1.7307**	**0.2936**	1.9278	0.7083	0.3400
Dog	3.4268	**2.8602**	4.5753	3.2575	**0.3515**	1.5356	0.7687	0.4061
Gorilla	**2.7770**	12.0532	4.0697	2.8054	**0.4966**	11.0032	1.1104	0.5909
Michael	**2.9905**	7.9652	4.3429	3.0852	**0.4290**	6.7131	0.9423	0.4723
Victoria	**2.4997**	3.6491	3.6424	2.6194	**0.3175**	1.6411	0.7243	0.3614
Wolf	**1.6750**	4.5455	2.9337	1.6758	**0.2746**	3.4848	0.6155	0.3256

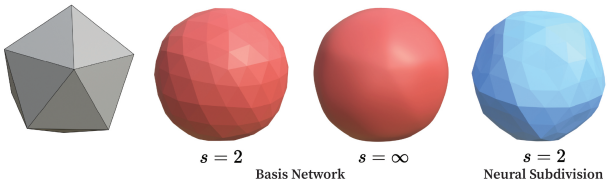


Figure 8: *Most subdivision algorithms ensure tangent (and curvature) continuity. If the refined patches do not form a parametrically continuous surface, as in Neural Subdivision [LKC*20], then the triangulation of the input may be clearly visible on the fine surface. The basis network is C^2 continuous except at vertices in irregular regions. s is the number of subdivisions.*

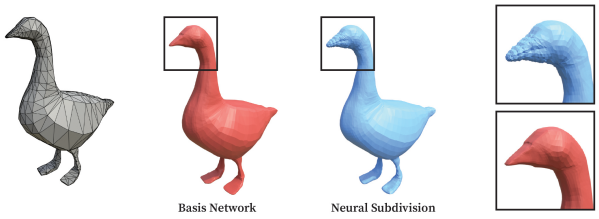


Figure 9: *Neural Subdivision [LKC*20] operates directly on triangle flaps and stars and depends on the sizes of the refined triangles. This becomes apparent after a few subdivisions, or when finer triangles are used for the more detailed parts of the models (such as the beak of the goose). For this comparison we used the publicly available network weights of Neural Subdivision. Basis networks are invariant to affine transformations. Goose by Atlas under CC-BY.*

We observed that the basis network can produce smooth, plausible surfaces even for unseen two-ring face regions. Although neural subdivision often generates a similar, acceptable surface, sharp artifacts are sometimes visible. Neural Subdivision generally does not produce a tangent continuous surface, and the triangulation of the input may be clearly visible on the refined surface (Fig. 8). When learning basis functions, smoothness is explicitly considered in the construction. Differences to Neural Subdivision are also visible for meshes with unseen edge angles such as the coarse inputs (Fig. 7, last row of top) or if the input mesh is not uniformly tessellated (Fig. 9). One reason for this is that neural subdivision is not scale-invariant. For finely triangulated regions, or if the triangles become small enough after a few subdivisions, neural subdivision might generate a noisy surface. This also prevents the existence of a limit surface.

Higher resolution. In Fig. 7 (bottom) we compare a set of fine surfaces generated with the basis network with limit surfaces of two popular subdivision methods for triangle meshes - Loop subdivision [Loo87] and Modified Butterfly subdivision [ZSS96]. Modified Butterfly subdivision has a lower degree of continuity (C^1), but interpolates the input vertices. For coarse inputs, Loop Subdivision often leads to strong smoothing of the mesh and a shrinking of the surface. While Modified Butterfly and the basis network generate a similar surface for larger inputs, the basis network produces a smoother looking result on the fine scale. To isolate the effect of a single vertex in all three schemes, we plotted various basis functions in Fig. 10. Basis functions are the impulse responses when subdividing a grid in $z = 0$ with one vertex at $z = 1$. This shows that the learned basis functions suffer from significantly fewer artifacts than the basis of Modified Butterfly subdivision. Note that the Modified Butterfly subdivision's support is larger than that of the learned basis functions, which have equal support as in Loop's scheme. Artifacts in the basis functions can also be seen in reflections on the surface. In Fig. 11 we plot reflection lines [Kla80] for Modified Butterfly and the basis network. For a G^n continuous surface, reflection lines are G^{n-1} continuous, so G^2 continuity is visible by smooth reflection lines. Modified Butterfly is C^1 continuous and the surfaces of the basis networks are C^2 continuous (except at irregular vertices where the surface is C^1).

We use the Loop basis as input for the network, which depends on the local connectivity of the input mesh. Some connectivity configurations, such as those with high vertex degrees, are not well represented in the training data. Despite this, the network seems to handle such configurations sufficiently (Fig. 12), which may be due to the initialization with the Loop basis.

Quantitative results. For quantitative comparison, we train another basis network and Neural Subdivision on three categories of the original TOSCA dataset (Centaur, David, Horse) and compute the two-sided Hausdorff distances and the Mean surface distances of the refined to the original surfaces using METRO [CRS98] for

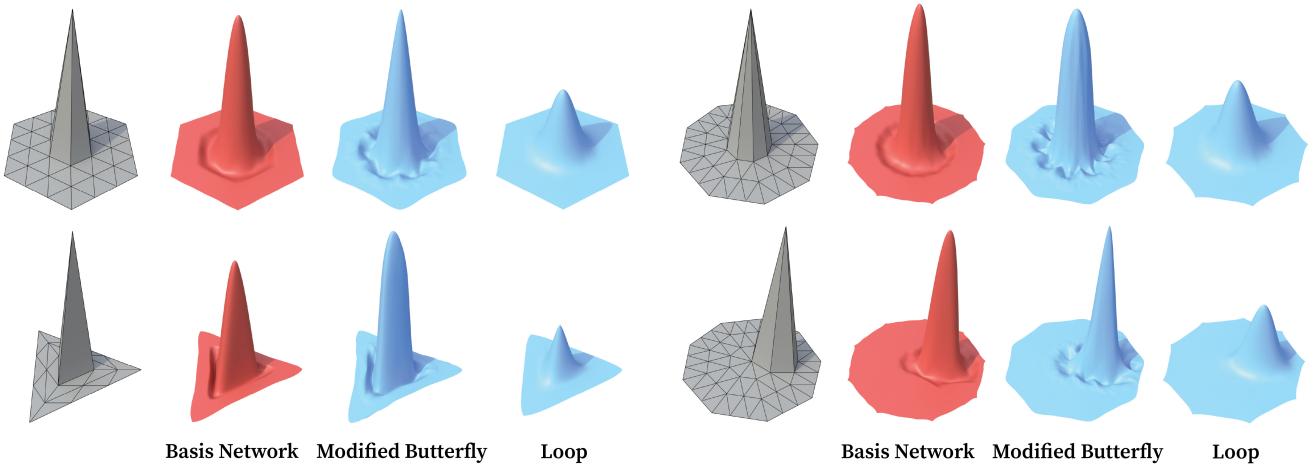


Figure 10: *Basis functions for linear subdivision schemes can be visualized by subdividing a planar grid $z = 0$ with one vertex at $z = 1$. The learned basis functions have fewer artifacts than the bases of Modified Butterfly subdivision [ZSS96]. The bases of Loop subdivision [Loo87] are far from interpolating the input vertices. Top left: regular basis; bottom left: basis of a degree 3 vertex; top right: basis of a degree 9 vertex; bottom right: basis of a degree 6 vertex adjacent to a degree 9 vertex.*

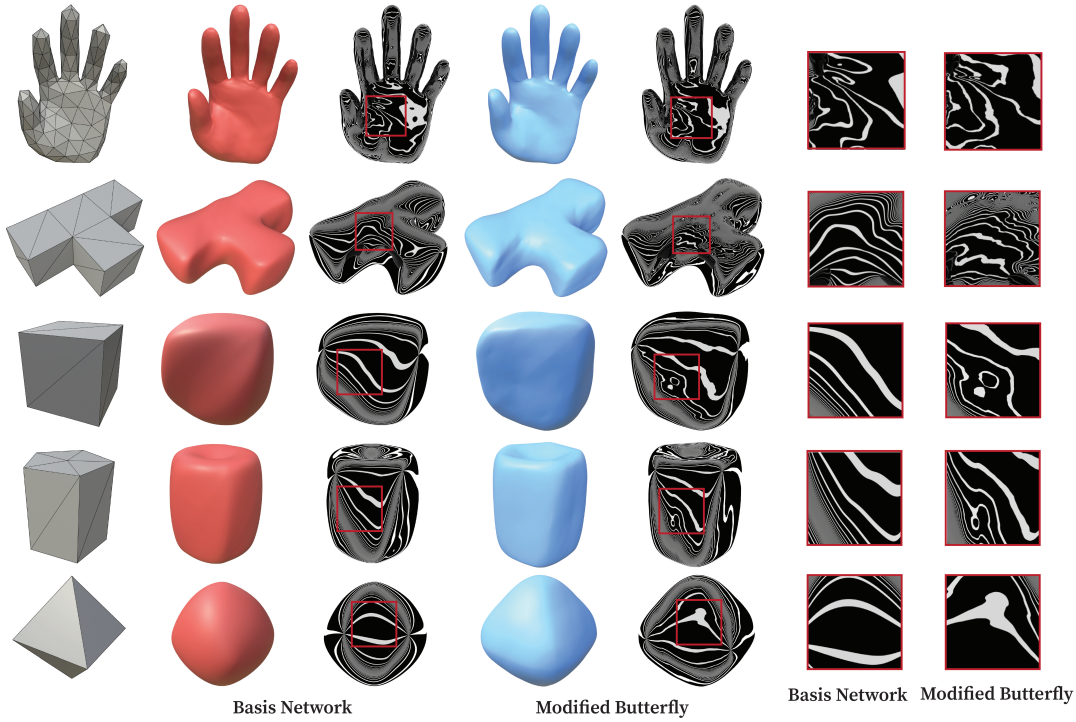


Figure 11: *Each ('black') surface has a purely reflective material that reflects stripes from the environment map. These reflection lines visualize the change of normal directions more clearly. With learned basis functions we are able to generate surfaces with smoother looking reflections than Modified Butterfly [ZSS96].*

all other categories. The same training parameters were used as in the previous sections. Further, we also provide the results for Loop subdivision and Modified Butterfly subdivision. In all categories, the learned basis functions generated surfaces that had the closest mean surface distance to the ground truth surfaces (Tab. 1).

The Hausdorff distance is more sensitive to outliers. When we repeated the training with different network sizes for the basis network (see next paragraph), we observed that the modified butterfly subdivision and the neural subdivision performed better in some

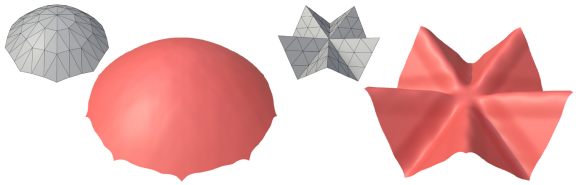


Figure 12: *Surface generated by the basis network around a vertex of degree 10 for different control meshes. Vertices with high degrees are underrepresented in the data set.*

Table 2: *We compute the mean surface distance for multiple basis networks with various features and different network sizes (numbers in parenthesis are sizes of the hidden layers). All networks are trained on three categories of the TOSCA dataset (Centaur, David, Horse) and tested on the other categories. All values are relative to the bounding box diagonal ($\times 10^2$).*

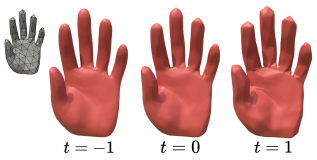
Features	$\mathbb{M}_{Basis}$
Only central basis	0.4711
Two rings of bases	0.3800

Network	$\mathbb{M}_{Basis}$
(256, 256, 128)	0.3485
(512, 256)	0.3521
(256, 128)	0.3639
(128, 64)	0.3701

categories. However, the basis network achieved the best Hausdorff distance in most categories.

Ablation. The performance of the basis network depends on the size of the network and on the choice of input features. We repeated the quantitative comparison with the TOSCA dataset with different network sizes and vary the number of basis functions that are the input of the network. For this comparison, we used the mean surface distance because it is less prone to outliers than the Hausdorff distance (Tab. 2). The network fails to predict meaningful surfaces when the only input is the basis of the respective vertex. This is intuitive, as a single basis value encodes few relations regarding how the weights are distributed in the local neighborhood. Conversely, providing all basis functions that could influence a given point (i.e. the bases of two rings of adjacent vertices) does not improve surface quality and increases the run time quadratically with d_{max}. Larger networks also lead to slightly improved mean surface distances. We chose a small network with hidden layers of depths 256 and 128 to balance size with mean surface distance.

6.2. Additional basis features



We can assign to each vertex of the input mesh an additional feature vector that serves as a descriptor of a basis or the respective vertex comparable to [SLR24, YLL*23]. As a simple example, we assign each vertex a scalar value t. Then we trained the network with half of the TOSCA dataset upsampled with midpoint subdivision and $t = 1$. For the other half, we use train data that has been upsampled with Loop subdivision (as in Sec. 6.1) and $t = -1$. We can then use the additional feature t to blend between a smooth variant of midpoint subdivision and the scheme of Sec. 6.1 (see inset).

Note that there are no constraints regarding parametric continuity to this feature vector as long as this feature remains constant for all samples of a basis function (i.e. each row of **B**).

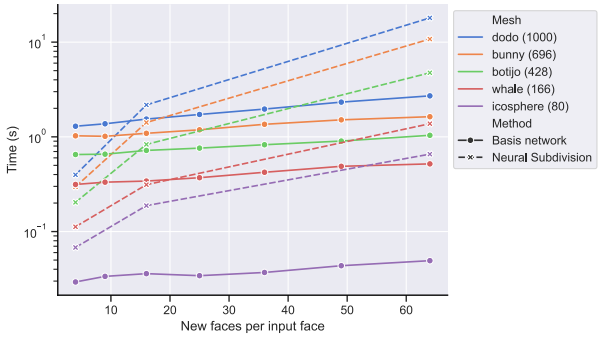


Figure 13: *Runtime performance of the basis network and Neural Subdivision [LKC*20]. The number after the mesh name indicates the number of faces in the mesh. During refinement, each face is split into a number of new faces (x-axis). Each data point is averaged over 10 runs. We used an Intel i9-12900K processor with an Nvidia RTX A5000 GPU.*

6.3. Runtime

We compare the runtime performance of the basis network and Neural Subdivision [LKC*20] for a set of input meshes of different sizes in Fig. 13. Both methods are implemented in Python using PyTorch [PGM*19]. They extract the vertex and face adjacencies in Python, a process that could probably be greatly accelerated. Nevertheless, Neural Subdivision requires recursive evaluation of multiple networks, whereas the basis approach uses a single network that directly generates points on the surface. Due to local support, every refined point needs at most $3*(d_{max}-3)$ (with $d_{max} > 4$) predicted scalar basis functions that can be evaluated in parallel, which scales better with surface resolution (Fig. 13). Another difference is that the basis network only requires adjacency information in the coarse input mesh, whereas Neural Subdivision performs average pooling operations over adjacent triangle flaps between subdivision steps. With basis functions, we can evaluate the refined surface at arbitrary parameter values which gives more control on the resolution of the refined surface.

Note that basis functions are independent of the geometry of the input mesh. This allows the basis functions to be reused when the connectivity and resolution do not change, which is often the case for animation. Table 3 provides a more detailed breakdown of the surface computation time.

7. Discussion & Future Work

Learning basis functions is a versatile approach to exploring the

Table 3: *Runtime to compute a surface (equivalent to two subdivisions) using the basis network (Basis) and Neural Subdivision (N.S) [LKC*20]. The time for the basis network consists mostly of computing the Loop basis $\mathbf{b}^L$ (Loop). Extracting the vertex adjacencies and faces in support of each vertex to build the $\mathbf{v}_i$ (Feats.), as well as the actual surface computation in Algorithm 1 and Eq. 1 (Surface) only requires a small fraction of the total time. Results are in seconds and averaged over 10 runs.*

Mesh (faces)	**Basis**	Loop	Feats.	Surface	**N.S.**
dodo (1000)	1.744	1.181	0.223	0.227	2.515
bunny (696)	1.215	0.898	0.111	0.122	1.623
botijo (428)	0.775	0.624	0.057	0.059	0.970
whale (166)	0.374	0.293	0.033	0.018	0.366

space of basis functions for surfaces. Rather than optimizing vertex positions after each subdivision step [LKC*20], basis functions avoid the recursive application of networks by directly generating points on a 'limit surface'. This surface can be evaluated at arbitrary parameter values and forms a C^2 continuous surface (except at irregular vertices where this scheme is C^1). Using the predicted basis functions, various properties of the generated surface can be achieved, such as continuity, the convex hull property or affine invariance. Linear precision is obtained when all vertices in the support are coplanar, since the surface is computed as a convex combination of the vertices.

Such a basis network can be trained to avoid artifacts that appear with Modified Butterfly subdivision [Zor00] by deriving bases that suffer from fewer artifacts. This shows in more pleasant reflection lines for coarse control meshes. Predicting bases instead of vertex positions is more restrictive since this can only generate points in the space spanned by the input points. The lower Hausdorff and mean surface distances compared to Neural subdivision (as well as Modified Butterfly subdivision or Loop subdivision) for reproducing given geometric data indicates that the increased flexibility of a non-linear scheme might not be necessary for many tasks. In contrast to Neural subdivision, the basis network is limited to a maximum vertex degree, which must be fixed before training.

That being said, we believe a promising direction is the use of additional features for learning non-stationary bases that vary for different vertices in the input. Our example using Loop and midpoint subdivision is clearly only a first indicator in this direction, and various other modifications, for example based on curvature or annotations, are possible. We also did not experiment with sharp features. An approach to implementing sharp creases could be to use the Loop basis computed with the well known crease rules as input for the network.

We see several directions for follow-up work based on learning basis functions. The idea of modifying a well-known set of basis functions is largely independent of our choice for the Loop scheme. It is straightforward to replace the Loop basis with another bases from a different subdivision scheme. For example, using the Catmull-Clark basis, which is based on cubic BSplines, might lead to a simple extension for quadrilateral (and polygonal) meshes. Basis functions of bi-quartic schemes [Qu90, Zor00] could increase the order of continuity, or the improved Loop subdivision rules could further improve the initialization [Loo02, PX04]. Interpolation of the input vertices can be enforced using the basis of an interpolating subdivision scheme [DLG90, ZSS96]. Evaluating Modified Butterfly subdivision at arbitrary parameter values is challenging (see [LLL*11] for evaluation at rational points). The basis for an interpolating scheme is one for only a single vertex at the parameter position. Multiplying the network output by this basis results in a constrained basis that is also interpolating. However, since a smooth interpolating basis must contain negative values, other parameter positions at which the basis is zero will also be unintentionally constrained. Learning basis functions may be valuable in other domains. For subdivision of tetrahedral meshes, there are classes of trivariate box splines [CMQ02].

Acknowledgments

The authors would like to express their sincere gratitude to Hendrik Meyer for fruitful discussions, Koray Akalin for helping with Modified Butterfly and all the reviewers for their thoughtful feedback. The authors also would like to thank everyone who contributed to the discussions on the CSCG 2025 on this topic, especially Philipp Erler and Diana Marin. This work was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 101055448, ERC Advanced Grand EMERGE). Open Access funding enabled and organized by Projekt DEAL.

References

[BBK08] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008. 6, 8

[Ble25] BLENDER ONLINE COMMUNITY: *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2025. URL: http://www.blender.org. 6

[BLZ00] BIERMANN H., LEVIN A., ZORIN D.: Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 113–120. doi:10.1145/344779.344841. 5

[BPG*20] BEDNARIK J., PARASHAR S., GUNDOGDU E., SALZMANN M., FUA P.: Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 4

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. doi:https://doi.org/10.1016/0010-4485(78)90110-0. 1, 3, 4

[CD21] CONTI C., DYN N.: Non-stationary subdivision schemes: State of the art and perspectives. In *Approximation Theory XVI* (Cham, 2021), Fasshauer G. E., Neamtu M., Schumaker L. L., (Eds.), Springer International Publishing, pp. 39–71. 4

[CKAJ23] CHEN Y.-C., KIM V., AIGERMAN N., JACOBSON A.: Neural progressive meshes. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. doi:10.1145/3588432.3591531. 4

[CMQ02] CHANG Y.-S., MCDONNELL K. T., QIN H.: A new solid subdivision scheme based on box splines. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2002), SMA '02, Association for Computing Machinery, p. 226–233. doi:10.1145/566282.566316. 11

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. doi:https://doi.org/10.1111/1467-8659.00236. 8

[DGF*19] DEPRELLE T., GROUEIX T., FISHER M., KIM V., RUSSELL B., AUBRY M.: Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems* (2019), Wallach H., Larochelle H., Beygelzimer A., d'Alché-Buc F., Fox E., Garnett R., (Eds.), vol. 32, Curran Associates, Inc. 4

[DLG90] DYN N., LEVINE D., GREGORY J. A.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph. 9*, 2 (apr 1990), 160–169. doi:10.1145/78956.78958. 4, 11

[DS78] DOO D., SABIN M.: Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design 10*, 6 (1978), 356–360. 4

[Far86] FARIN G.: Triangular bernstein-bézier patches. *Computer Aided Geometric Design 3*, 2 (1986), 83–127. doi:https://doi.org/10.1016/0167-8396(86)90016-6. 5

[FH00] FARIN G., HANSFORD D.: *The essentials of CAGD*. AK Peters/CRC Press, 2000. 1

[FHK02] FARIN G., HOSCHEK J., KIM M. S.: *Handbook of Computer Aided Geometric Design*, 1 ed. San Diego, 2002. 4

[FMW14] FANG M., MA W., WANG G.: A generalized surface subdivision scheme of arbitrary order with a tension parameter. *Computer-Aided Design 49* (2014), 8–17. doi:https://doi.org/10.1016/j.cad.2013.12.003. 4

[GFK*18] GROUEIX T., FISHER M., KIM V. G., RUSSELL B. C., AUBRY M.: A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 4

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., p. 209–216. doi:10.1145/258734.258849. 6

[GWM21] GADELHA M., WANG R., MAJI S.: Deep manifold prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops* (October 2021), pp. 1107–1116. 4

[HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., MCDONALD J., SCHWEITZER J., STUETZLE W.: Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, Association for Computing Machinery, p. 295–302. doi:10.1145/192161.192233. 5

[HG16] HENDRYCKS D., GIMPEL K.: Gaussian error linear units (gelus), 2016. arXiv:1606.08415. 2, 5

[HW99] HABIB A., WARREN J.: Edge and vertex insertion for a class of c1 subdivision surfaces. *Computer Aided Geometric Design 16*, 4 (1999), 223–247. doi:https://doi.org/10.1016/S0167-8396(98)00045-4. 4

[JSD03] JENA M., SHUNMUGARAJ P., DAS P.: A non-stationary subdivision scheme for generalizing trigonometric spline surfaces to arbitrary meshes. *Computer Aided Geometric Design 20*, 2 (2003), 61–77. doi:https://doi.org/10.1016/S0167-8396(03)00008-6. 4

[KB17] KINGMA D. P., BA J.: Adam: A method for stochastic optimization, 2017. arXiv:1412.6980. 6

[Kla80] KLASS R.: Correction of local surface irregularities using reflection lines. *Computer-Aided Design 12*, 2 (1980), 73–77. doi:https://doi.org/10.1016/0010-4485(80)90447-9. 8

[Kob96] KOBBELT L.: Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum 15*, 3 (1996), 409–420. doi:https://doi.org/10.1111/1467-8659.1530409. 4

[Kob00] KOBBELT L.: $\sqrt{3}$-subdivision. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 103–112. doi:10.1145/344779.344835. 3

[LKC*20] LIU H.-T. D., KIM V. G., CHAUDHURI S., AIGERMAN N., JACOBSON A.: Neural subdivision. doi:10.1145/3386569.3392418. 1, 2, 4, 6, 7, 8, 10, 11

[LL22] LOW W. F., LEE G. H.: Minimal neural atlas: Parameterizing complex surfaces with minimal charts and distortion. In *European Conference on Computer Vision* (2022), Springer, pp. 465–481. 4

[LLL*11] LI B., LI B., LIU X., SU Z., YU B.: Exact evaluation of limits and tangents for interpolatory subdivision surfaces at rational points. *Journal of Computational and Applied Mathematics 236*, 5 (2011), 906–915. The 7th International Conference on Scientific Computing and Applications, June 13–16, 2010, Dalian, China. URL: https://www.sciencedirect.com/science/article/pii/S0377042711002615, doi:https://doi.org/10.1016/j.cam.2011.05.014. 11

[Loo87] LOOP C. T.: Smooth subdivision surfaces based on triangles, 1987. 2, 3, 4, 6, 7, 8, 9

[Loo02] LOOP C.: Bounded curvature triangle mesh subdivision with the convex hull property. *The visual computer : international journal of computer graphics ; official journal of the Computer Graphics Society, CGS. 18*, 5-6 (2002), 316. 11

[LSS*98] LEE A. W. F., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, pp. 95–104. doi:10.1145/280814.280828. 6

[LY10] LEE Y. J., YOON J.: Non-stationary subdivision schemes for surface interpolation based on exponential polynomials. *Applied Numerical Mathematics 60*, 1 (2010), 130–141. doi:https://doi.org/10.1016/j.apnum.2009.10.005. 4

[LZBCJ21] LIU H.-T. D., ZHANG J. E., BEN-CHEN M., JACOBSON A.: Surface multigrid via intrinsic prolongation. *ACM Trans. Graph. 40*, 4 (July 2021). doi:10.1145/3450626.3459768. 6

[MAKM21] MORREALE L., AIGERMAN N., KIM V. G., MITRA N. J.: Neural surface maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 4639–4648. 4

[NRY16] NOVARA P., ROMANI L., YOON J.: Improving smoothness and accuracy of modified butterfly subdivision scheme. *Applied Mathematics and Computation 272* (2016), 64–79. Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability. doi:https://doi.org/10.1016/j.amc.2015.07.065. 4

[PBW19] PREINER R., BOUBEKEUR T., WIMMER M.: Gaussian-product subdivision surfaces. *ACM Trans. Graph. 38*, 4 (July 2019). doi:10.1145/3306346.3323026. 4

[Pet95] PETERS J.: Biquartic c1-surface splines over irregular meshes. *Computer-Aided Design 27*, 12 (1995), 895–903. doi:https://doi.org/10.1016/0010-4485(95)00010-0. 5

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. 10

[PPB25] PENTAPATI S. K., PHILLIPS G., BOVIK A. C.: Mesh compression with quantized neural displacement fields. *Computer Graphics Forum n/a*, n/a (2025), e70074. doi:https://doi.org/10.1111/cgf.70074. 4

[PR97] PETERS J., REIF U.: The simplest subdivision scheme for

smoothing polyhedra. *ACM Trans. Graph. 16*, 4 (Oct. 1997), 420–431. doi:10.1145/263834.263851. 4

[PR08] PETERS J., REIF U.: *Subdivision Surfaces*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 57–81. doi:10.1007/978-3-540-76406-9_4. 4

[PX04] PAN Q., XU G.: Fast evaluation of the improved loop's subdivision surfaces. In *Geometric Modeling and Processing, 2004. Proceedings* (2004), pp. 205–214. doi:10.1109/GMAP.2004.1290042. 11

[Qu90] QU R.: *Recursive subdivision algorithms for curve and surface design*. PhD thesis, Brunel University, School of Information Systems, Computing and Mathematics, 1990. 11

[SLR24] SIVARAM V. E., LI T.-M., RAMAMOORTHI R.: Neural geometry fields for meshes. In *ACM SIGGRAPH 2024 Conference Papers* (New York, NY, USA, 2024), SIGGRAPH '24, Association for Computing Machinery. doi:10.1145/3641519.3657399. 4, 10

[Sta98a] STAM J.: Evaluation of loop subdivision surfaces, 1998. SIGGRAPH'98 CDROM Proceedings. 1, 2, 3, 5

[Sta98b] STAM J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, p. 395–404. doi:10.1145/280814.280945. 1, 3

[VMW18] VAXMAN A., MÜLLER C., WEBER O.: Canonical möbius subdivision. *ACM Trans. Graph. 37*, 6 (Dec. 2018). doi:10.1145/3272127.3275007. 4

[VZ01] VELHO L., ZORIN D.: 4–8 subdivision. *Computer Aided Geometric Design 18*, 5 (2001), 397–427. Subdivision Algorithms. doi:https://doi.org/10.1016/S0167-8396(01)00039-5. 3

[WM25] WILLIAMSON R., MITRA N. J.: Neural geometry processing via spherical neural surfaces. *Computer Graphics Forum n/a*, n/a (2025), e70021. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70021, doi:https://doi.org/10.1111/cgf.70021. 4

[WSS*19] WILLIAMS F., SCHNEIDER T., SILVA C., ZORIN D., BRUNA J., PANOZZO D.: Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 4

[YFST18] YANG Y., FENG C., SHEN Y., TIAN D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 4

[YLL*23] YANG L., LIANG Y., LI X., ZHANG C., LIN G., SHEFFER A., SCHAEFER S., KEYSER J., WANG W.: Neural parametric surfaces for shape modeling, 2023. URL: https://arxiv.org/abs/2309.09911, arXiv:2309.09911. 4, 10

[ZK02] ZORIN D., KRISTJANSSON D.: Evaluation of piecewise smooth subdivision surfaces. *The Visual Computer 18*, 5-6 (2002), 299–315. 5

[Zor00] ZORIN D.: Subdivision zoo. *Subdivision for modeling and animation, Schröder, Peter and Zorin, Denis* (2000), 65–104. 3, 4, 11

[ZSS96] ZORIN D., SCHRÖDER P., SWELDENS W.: *Interpolating Subdivision for meshes with arbitrary topology*. ACM, Aug 1996, p. 189–192. doi:10.1145/237170.237254. 4, 7, 8, 9, 11