university of groningen

University Medical Center Groningen

University of Groningen

Curvature-enhanced Neural Subdivision

Bruin, Sjoerd; Frey, Steffen; Kosinka, Jiri

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Final author's version (accepted by publisher, after peer review)

Publication date:
2023

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):
Bruin, S., Frey, S., & Kosinka, J. (2023). *Curvature-enhanced Neural Subdivision*. Paper presented at The 36th International Conference on Computer Animation and Social Agents, Limassol, Cyprus.

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 05-12-2023

Curvature-enhanced Neural Subdivision

Sjoerd Bruin Steffen D. Frey Jiří Kosinka
University of Groningen

Abstract

Subdivision is an important and widely used technique for obtaining dense meshes from coarse control (triangular) meshes for modelling and animation purposes. Most subdivision algorithms use engineered features (subdivision rules). Recently, neural subdivision successfully applied machine learning to the subdivision of a triangular mesh. It uses a simple neural network to learn an optimal vertex positioning during a subdivision step. We propose an extension to the neural subdivision algorithm that introduces explicit curvature information into the network. This makes a larger amount of relevant information accessible which allows the network to yield better results. We demonstrate that this modification yields significant improvement over the original algorithm, in terms of both Hausdorff distance and mean squared error.

Keywords: Subdivision, Machine Learning, Geometric Modelling

1 Introduction

One of the main areas of research in computer graphics and geometric modelling concerns the refinement of triangular meshes. Such approaches are used to create higher quality meshes from coarse ones, with applications in, among others, video games and animated films. Mesh refinement is used to produce smooth surfaces by recursively refining angular parts of the control mesh into smoother approximations, typically by introducing new triangles and by moving vertices into new positions that lead to better smoothness properties. The rules guiding this process are carefully designed to lead to well-behaved and smooth surfaces in the limit of the subdivision process, such as in the Loop subdivision scheme [1]. In a recent development, neural subdivision [2] was introduced to mimic Loop subdivision, but to improve on it using learned subdivision rules.

We expand the neural network used in neural subdivision to include explicit curvature information. This provides additional information about the shape of the mesh around a vertex, and has the potential to reintroduce specific detail into the subdivided mesh. We extract this curvature information from the intermediate meshes produced during the decimation process to create coarse mesh approximations of a ground truth mesh. We demonstrate that this addition leads to an improvement in the subdivision result both via objective metrics (Hausdorff distance (HD) and mean squared error (MSE)) and visual comparisons. Our main contributions are:

- We enhance neural subdivision by adding curvature information to the network.
- We conduct a detailed evaluation study (considering parameter settings, network sizes, and various meshes) and demonstrate that our extension is able to significantly improve results for challenging cases.

The remainder of the paper has the following structure. Section 2 gives an overview of the different approaches to mesh refinement and focuses on methods using machine learning. Section 3 discusses the neural subdivision algorithm in detail, and introduces several discrete curvature approximation schemes used in our method as well as a quality evaluation metric. Section 4 describes our curvature enhancement with respect to the original neural subdivision scheme and Section 5 covers the data used for training and evaluation. Section 6 discusses the results achieved with our curvature-enhanced neural subdivision algorithm. Section 7 concludes this paper and outlines directions for future work.

2 Related Work

Constructing smooth surfaces of arbitrary manifold topology is a difficult and well-studied problem. Such surfaces are typically controlled by a coarse mesh which guides the shape of the final surface. While methods tailored to control quad/hexagonal meshes exist [3, 4], sparse triangular meshes are among the most popular control structures. However, going from such a mesh to a smooth surface is non-trivial. One could use a single Bézier triangle [5] per mesh face, but it is difficult to achieve global G^1 continuity while keeping the patch degrees low or without resorting to rational patches (such as Gregory triangles [6–8]) or macro patches [9, 10]. When the control mesh is regular (all vertices have valency six), box-splines [11] can be used, for example the C^2 quartic box-spline. In the case of triangular meshes, Loop subdivision [1] extends the regular setting to provide globally smooth limit surfaces which are C^2 everywhere, except at points corresponding to extraordinary vertices where they are (only) G^1 (i.e., tangent-plane) continuous. Several methods based on Bézier triangles were later introduced to approximate Loop subdivision surfaces [12–14].

All the methods mentioned so far use carefully designed patches, or tailored and fixed subdivision rules. This has the obvious advantages of being simple and uniform, in the sense that the same patch type with fixed evaluation rules is used everywhere and the subdivision rules do not depend on the geometry of the control mesh. At the same time, this is a significantly limiting factor. To address this, one could look in the direction of non-linear subdivision schemes, such as that of [15]. While this offers greater freedom, complex rules need to be engineered by hand which introduces concerns regarding the effectiveness of the determined solution.

Machine learning-based techniques potentially address this issue, yet they have not been used extensively for subdivision so far. Most research has focused on point clouds [16–18], which—in contrast to the mesh data we work with—lacks connectivity information. Applying point cloud techniques to our scenario would therefore omit available information and make the problem more difficult. Additionally, the point cloud data would eventually need to be converted back to a triangular mesh (using for instance Poisson surface reconstruction [19]) for further processing. Other neural mesh generation techniques manipulate templates [20–23]. These template methods are either too general to produce good results or too restrictive to be generalisable to more complex shapes.

Neural subdivision [2] addresses this problem by defining a neural network that learns on the triangular mesh itself, which allows for direct and efficient triangle-for-triangle comparison between the subdivided mesh and a ground truth mesh. Neural subdivision can be considered as a form of non-linear subdivision, where the non-linear function is approximated using a neural network. We show that further improvements can be achieved by considering curvature information in the training process.

3 Background

In this section, we discuss the original neural subdivision network [2] along with the data generation method (Section 3.1), and the different discrete curvature approximations (Section 3.2) that we use to enrich neural subdivision.

3.1 Neural Subdivision

The neural subdivision algorithm uses machine learning to learn the optimal positions of vertex and edge points based on surrounding vertices [2], using similar stencil shapes as those used in Loop subdivision. The learning approach used in neural subdivision [2] necessitates the use of local coordinates to ensure orientation invariance, achieved by using half flaps. A half flap comprises a half edge, along with its associated face and the opposite face. The half flap has a canonical ordering of its vertices, allowing us to associate a unique half flap with each half edge.

Neural subdivision [2] learns subdivision rules which consider the local region to determine how the current vertex point should be moved, or where the new edge point should be positioned. In addition to positional information, the network also produces an intermediate high-level feature vector that later stages of the network use to guide the process of positioning the vertices. The network uses three modules: an initialisa-

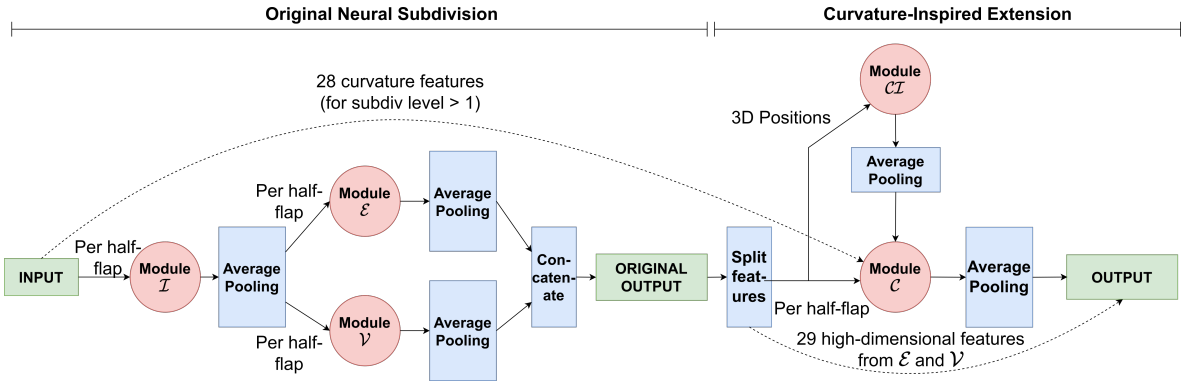

Figure 1: The structure of our curvature-enhanced neural subdivision network. Our network introduces two new modules ($\mathcal{C}$ and $\mathcal{C}I$) that produce position offsets that model the additional curvature information that the network is learning with.

tion module I, which is only applied at the first subdivision step; a vertex module $\mathcal{V}$ for manipulating vertex points; and an edge module $\mathcal{E}$ for edge points. Each module applies to a half flap associated with adjacent half edges, and average pooling is used to combine the updated features computed for each half flap into a single update for an existing vertex point or a new edge point. In addition to being applied to positions, average pooling is also applied to the high-level features; see Figure 1. The loss function is defined by mapping points from the coarse mesh into the ground truth mesh using a bijective mapping (see [2] for details) and subsequently calculating the mean squared error (MSE) loss between the subdivided mesh vertices and their corresponding ground-truth positions.

We need to produce a set of coarse meshes from a given fine mesh for training the network. We use a general scheme for generating coarse meshes by use of an edge decimation algorithm as in [2]. The decimation process removes an edge and reconnects the vertices both in 3D and in a UV-mapped space, as shown in Figure 2. Subsequently, [2] perform a number of checks to assess the quality of the resulting mesh. If the checks succeed, then the decimation step has succeeded. If not, then a different edge is chosen for decimation. We follow the procedure described in [24] for the selection of a suitable edge for the collapse. During the decimation process, we simultaneously construct the bijective mapping between the fine mesh and the coarse mesh.

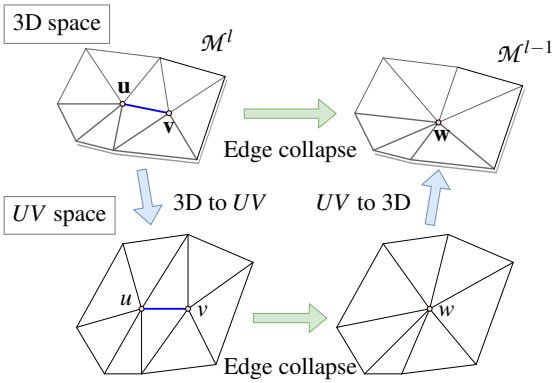

Figure 2: Edge decimation: the edge connecting **u** and **v** is collapsed into **w** (*from left to right*). Each edge collapse is applied both in 3D and in UV space (*top and bottom row*), and quality metrics are evaluated in each of them.

3.2 Discrete Curvature

We use three types of discrete curvature to analyse our addition: area-adjusted angle deficit, vector mean curvature, and scalar mean curvature.

The area-adjusted angle deficit computes the sum of the angles of the triangles around a given vertex, computes its deviation from 2π radians, and weighs the measure by the summed area of the triangles around the vertex. The area-adjusted angle deficit is given by (see Figure 3, left)

$$K = \frac{4}{A}\left(2\pi - \sum_{j=1}^{\nu}\theta_j\right), \qquad (1)$$

where ν denotes the valency of **u** and A is the summed area of the triangles incident with **u**. It approximates (Gauss) curvature by the deviation

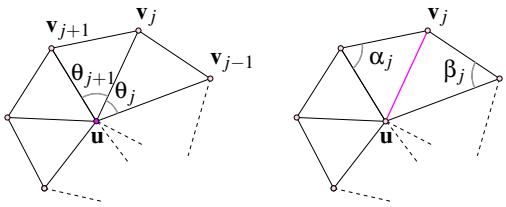

Figure 3: A graphical representation of the vertices and angles used in the discrete curvature metrics: angle deficit (left) and discrete mean curvature (right).

of the angle from a flat disk, and weights this by the size of the incident triangles, such that a given angle deficit implies higher curvature when the associated triangles are smaller.

The vector mean curvature [25] computes cotangent weights based on the angles that are located opposite a given edge for each edge connected to a given vertex. The vector mean curvature is given by (see Figure 3, right)

$$2H\mathbf{N} = \frac{1}{A}\sum_{j=1}^{\nu}\frac{\cot(\alpha_j)+\cot(\beta_j)}{2}(\mathbf{v}_j-\mathbf{u}). \quad (2)$$

The vector mean curvature specifies a direction and magnitude to the (mean) curvature. The scalar mean curvature value is given by H.

4 Curvature-Enhanced Neural Subdivision

Our extension of the basic approach (described in Section 3.1) explicitly represents the curvature of the input data. The architecture of our extended network is shown in Figure 1. The curvature extension has two modules. First, the curvature initialisation module $\mathcal{CI}$ sets up a curvature feature vector in an analogous fashion to the way module I initialises a feature vector. Module $\mathcal{CI}$ takes as input the positions of the vertices of the half flap and the curvature calculated at those vertices. The output is a 31-element feature vector, where the first three elements record the position update from the initial vertex position, and the remaining 28 elements encode additional features. Module $\mathcal{CI}$ is applied only during the first subdivision step in order to initialise the curvature feature vector.

The second module is the curvature module $\mathcal{C}$ itself. It takes as input a 31-element feature vector and the curvatures at the half flap vertices (potentially recomputed after applying the $\mathcal{CI}$ module). The output consists of a 31-element feature vector, of which the first three elements encode the position update of the vertices. Module $\mathcal{C}$ is applied at each subdivision step.

Similar to the I and $\mathcal{V}$ modules, $\mathcal{CI}$ and $\mathcal{C}$ use average pooling to combine the half flaps associated with a given vertex into a single update. The main novelty in these new modules is the use of curvature as input. A consequence of using curvature information across the half flaps is that instead of considering the one-ring neighbourhood around the central vertex or the central edge, a two-ring neighbourhood is used for the curvature information. The reason for this is that the curvature is evaluated at each vertex in the half flap, and evaluating the curvature at a vertex requires the one-ring neighbourhood of vertices there. Due to this, a two-ring neighbourhood is used for gathering information, and furthermore two different types of information are extracted. This leads to a larger amount and variety of information for use in the learning process.

In order to guide the learning process towards incorporating the curvature information, we use a loss function based on the MSE for the position loss and the L_1 loss for the curvature loss. We opted for the L_1 loss for the latter because local curvature can vary significantly, and therefore the robustness of the L_1 loss against outlier values helps to keep the update stable. With N the number of vertices in the mesh under consideration, the curvature loss is defined as

$$L_{\text{curv}} = \frac{1}{N}\sum_{i=1}^{N}\left|\kappa_i^{\text{sub}}-\kappa_i^{\text{target}}\right|, \quad (3)$$

where κ_i^{sub} is the curvature computed on the subdivided mesh, and κ_i^{target} is the target curvature at $\mathbf{v}_i$. The original position-based loss [2] reads

$$L_{\text{pos}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{v}_i^{\text{sub}}-\mathbf{v}_i^{\text{fine}}\right)^2, \quad (4)$$

where $\mathbf{v}_i^{\text{sub}}$ is the position of a vertex in the subdivided mesh, $\mathbf{v}_i^{\text{fine}}$ is the corresponding position in the fine mesh determined via the bijective mapping. Finally, the full loss function L for the curvature network is given by

$$L = L_{\text{pos}} + \gamma L_{\text{curv}}, \quad (5)$$

where γ is a hyperparameter that controls the impact of curvature loss. The value of γ determines how strongly the curvature loss defined in Equation 3 contributes to the repositioning of the vertices in the subdivision step. In Section 6.1.1 we perform a hyperparameter grid search to identify the best configuration of γ.

We use target curvature information from the intermediate decimation steps (using linear interpolation after mapping a position into an intermediate step of the decimation process), such that the number of vertices in the subdivision step and the decimation step from which we extract curvature information match. During the learning process, the curvature updates are applied to a large number of meshes. This way, individual skews introduced by the intermediate decimation steps are averaged out. The consideration of intermediate meshes gives the network an additional chance to reintroduce curvature information into the subdivision process.

5 Evaluation Dataset

In order to compare with [2], we have made use of the same mesh dataset, namely the TOSCA dataset [26], which contains a number of high-resolution meshes, typically with several tens of thousands of vertices per mesh. The wolf mesh was employed for training the base neural network and the curvature-enhanced neural network. To this end, we have created 230 coarse meshes using the decimation process discussed in [2], of which 200 meshes are used for training and 30 are used for testing purposes. Figure 4 shows the original fine wolf mesh and a coarse version of the mesh. The decimated meshes from other meshes in the TOSCA dataset serve the role of holdout datasets, since they can be used as an extra test of the generalisability of the network, which has been fine-tuned for the set of 30 test wolf meshes.

Figure 5 shows the other meshes that we have used to analyse the behaviour of the network. The cat, centaur, and horse meshes are again from TOSCA. We have had to slightly manipulate the meshes in order to make them suitable for the decimation process. This includes reducing the number of initial vertices as well as changing the triangulation of the meshes to reduce the number

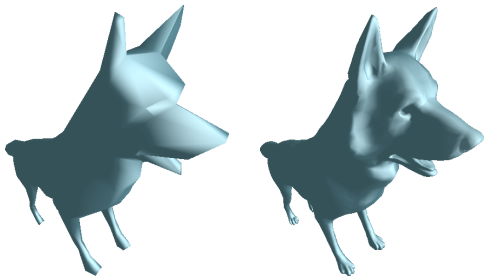

Figure 4: A coarse mesh (left) and the original mesh (right) from the TOSCA dataset.

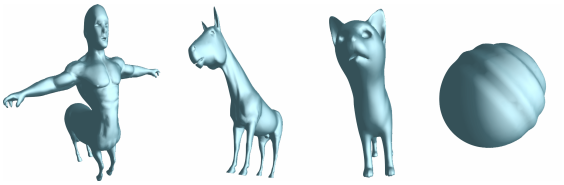

Figure 5: The fine meshes used for analysis in addition to the wolf mesh. The centaur, horse, and cat meshes are from the TOSCA dataset, whereas the sphere has been generated procedurally.

of triangles that have a poor aspect ratio. We have performed these changes using MeshLab [27]. The applied steps included the application of Loop subdivision on local patches surrounding triangles with poor aspect ratios, and applying MeshLab's isotropic explicit remeshing (based on [28]) in order to improve the connectivity of the mesh.

We have generated the sphere mesh in Figure 5 by constructing one hemisphere in a regular fashion and applying a sinusoidal displacement to the other hemisphere. The triangular mesh was created from a point cloud using the PyVista library [29] and post-processed using MeshLab.

6 Results & Discussion

In this section, we discuss the results of our curvature-enhanced extension and compare it to original neural subdivision. Section 6.1 describes our parameter and network size studies, Section 6.2 compares the visual characteristics of our extension against the baseline, and Section 6.3 demonstrates results achieved when the curvature network uses reference curvatures that were mapped to the ground truth rather than to an intermediate step of the mesh. In our experi-

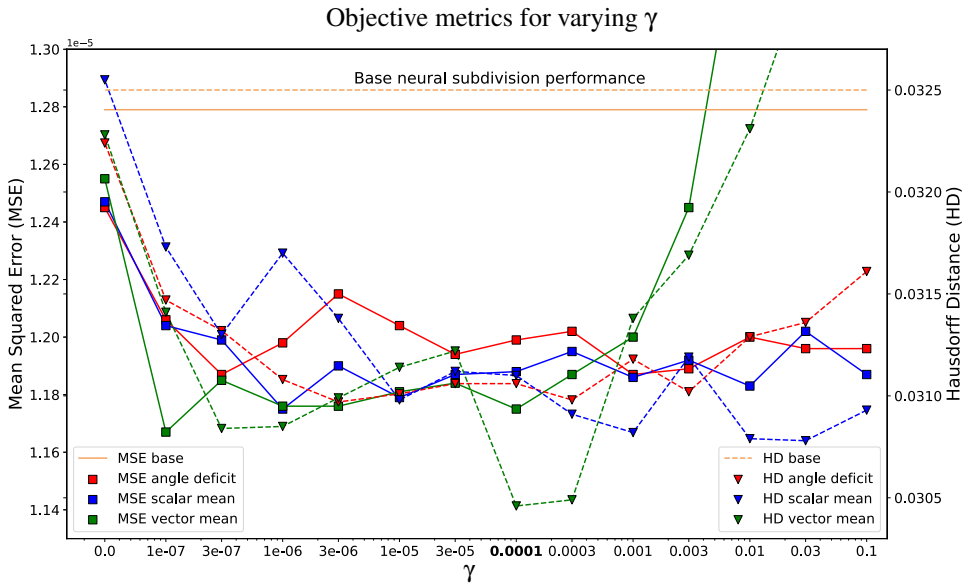

Figure 6: The MSE and HD test losses for the angle deficit, scalar mean, and vector mean curvature approximations for a range of different values of γ using 200 wolf meshes for training and 30 wolf meshes for testing. It becomes clear that each method can improve on the base network, but the vector mean curvature method has the best improvement for the HD.

ments, subdivision has been applied two times.

6.1 Study with Objective Measures

We now discuss the impact on objective measures when varying the parameter γ from Equation 5 (Section 6.1.1), the network size (Section 6.1.2), as well as the mesh (Section 6.1.3). During the training of the network, we saw similar convergence properties as those reported in [2].

6.1.1 Parameter Study

First, we apply a parameter search on the value of γ in Equation 5 for all three discrete curvature approximation methods mentioned in Section 3.2. Figure 6 shows the objective measure scores produced by different values of γ. All of the curvature methods produce an improvement over the base model. More interestingly, each curvature method also produces an improvement over the curvature network model trained with $\gamma = 0.0$, that is, without the curvature loss. This makes sense since the introduction of curvature adds new information to the network, and the network is able to learn based on the additional information. When $\gamma = 0.0$, this advantage disappears since the curvature loss is not used.

Each curvature method performs quite well, outperforming the base network. The angle deficit networks produce the worst upgrade over the base network, bottoming out at a Hausdorff distance of 0.03097. The scalar mean curvature network produces the lowest Hausdorff distance of 0.03078. The vector mean curvature network improves on this, reaching the lowest Hausdorff distance of 0.03046. This represents a Hausdorff distance improvement of 6.3% over the base network. The corresponding mean squared error of $1.175 \cdot 10^{-5}$ represents an improvement of 8.1% over the base network, which is a relatively modest improvement compared to the 38.1% decrease in Hausdorff distance and an 88.8% decrease in MSE from Loop subdivision to neural subdivision. This makes sense since our aim has been to improve details rather than comprehensively improve the subdivision scheme.

It should be noted that both metrics represent a different deviation from the ground truth mesh: the mean squared error represents how close a vertex-to-vertex correspondence between two sets of vertices is. This correspondence fixes which vertex should be compared to which other vertex, and therefore it does not take into account that the vertices might represent the ground truth better collectively than as a one-to-one corre-

spondence. The Hausdorff distance accounts for this by computing the maximum distance that one needs to travel from any vertex to its nearest vertex in the other set, yielding an upper bound on the deviation from the ground truth across the entire collection of vertices. Therefore, we have chosen to use Hausdorff distance as the primary metric for comparing the objective measures. Based on this, the vector mean curvature network with $\gamma = 0.0001$ produces the best result and we use it in the remainder of this work.

6.1.2 Network Size Study

We now demonstrate that increasing the network size from the standard setting has only negligible impact on performance for both the original as well as our extended model (using the best performing model from Section 6.1.1). To this end, we have increased the size of the base (original) network both in terms of hidden layer size and feature vector size (from 32 first to 64 and then to 128). For the network size of 64, the network produced a test result with MSE = $1.281 \cdot 10^{-5}$ and HD = 0.03248, and for the network size equal to 128, the test result was MSE = $1.275 \cdot 10^{-5}$ and HD = 0.03241. For the curvature-enhanced network, this also did not produce a significant change in performance: for $\gamma = 0.0001$, the new performance result for network size equal to 64 was MSE = $1.182 \cdot 10^{-5}$ and HD = 0.03051, and for network size equal to 128 the result was MSE = $1.186 \cdot 10^{-5}$ and HD = 0.03053.

These results suggest that the improvements seen in Section 6.1.1 cannot be achieved by simply increasing the size of the base network, at least not without fundamentally changing the architecture (which is outside of the scope of this research). Similarly, the curvature enhancement does not appear to improve in any significant manner when the network size is increased.

6.1.3 Mesh Study

Table 1 shows the test error of the other meshes discussed in Section 5 across 30 coarse meshes for the curvature network and the base network. From the table, it becomes clear that some meshes are significantly improved by the curvature application, whereas others see less of an impact or the metrics might even be slightly worse according to the metric. In general, note that our approach particularly improves the accuracy in challenging parts of the mesh and might exhibit similar or even slightly worse performance in other parts. Below, we now have a closer look by means of visual inspection and discuss the results in detail (Section 6.2).

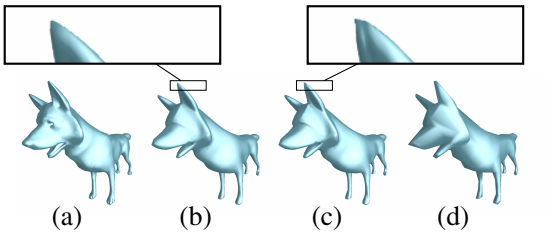

Figure 7: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The results of the curvature and base networks are similar, but differences can be seen in the ears and in the cheek.

6.2 Visual Comparison

We now discuss the visual quality of the curvature network results for different meshes.

Wolf In general, Figure 7 indicates that the curvature network and the base network produce similar results, which is in line with the results in Section 6.1. Despite the high structural similarity, Figure 7 shows a few noticeable differences. One difference can be seen in the ears: the left ear (on the right side for the viewer) in the subdivided mesh from the curvature network does not have the fold that is visible in the base network. This fold does not correspond to any structure in the ground truth mesh, but is present in the coarse mesh. Additionally, the ear of the curvature network mesh is more pointed like the ear of the ground truth mesh, whereas the base network mesh has the flat top that the coarse mesh has.

Further differences between the two networks can be observed. First, the shading on the cheek near the mane is lighter in the curvature network mesh and darker in the base network mesh, which is more in line with the ground truth mesh. Secondly, the base network has reproduced a fold at the bottom of the brow more strongly than the curvature network. We believe that this fold is

Mesh	Base MSE	Base HD	Curv. MSE	Curv. HD
Centaur	$3.897 \cdot 10^{-5}$	0.06224	$\mathbf{2.920 \cdot 10^{-5}}$	**0.04327**
Horse	$1.633 \cdot 10^{-5}$	**0.03868**	$\mathbf{1.577 \cdot 10^{-5}}$	0.03893
Cat	$\mathbf{7.282 \cdot 10^{-6}}$	**0.02072**	$7.742 \cdot 10^{-6}$	0.02557
Sphere	$7.158 \cdot 10^{-6}$	0.02337	$\mathbf{6.151 \cdot 10^{-6}}$	**0.02118**

Table 1: Test errors for the meshes discussed in Section 5. The results for the centaur and sphere are better, the result for the horse is roughly the same and the result for the cat is slightly worse.

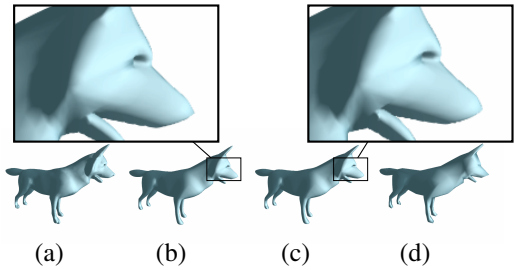

Figure 8: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). Differences are visible in the shape of the nose and the eye socket.

supposed to represent the remainder of an eye in the coarse mesh, but the reconstruction of the fold is in the wrong location. Due to this, the base network is very partially reconstructing an erroneously positioned feature, which is an error that the curvature network has managed to avoid.

Figure 8 shows a side view of the meshes. A difference can be seen in that the nose of the curvature network mesh points slightly more upward, which is more in line with the ground truth mesh. Another noticeable difference is the eye socket, since the base network mesh has an eye socket that stretches slightly further backward than the eye socket of the curvature network. The results of the curvature network matches more closely with the ground truth for this feature.

Figure 9 shows another small difference. The tongue of the curvature network has a slightly clearer separation from the jaw than the base network and the tongue tip is positioned slightly higher. The curvature network results are closer to those of the ground truth mesh.

Centaur Figure 10 shows the centaur mesh. The overall shape between the two networks is similar, but some parts are significantly different. Most importantly, the back of the horse part

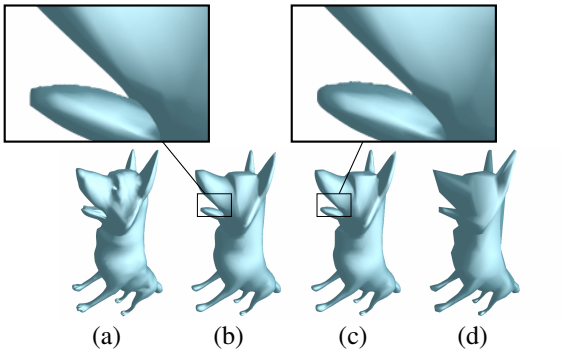

Figure 9: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The tongue shape is different between (b) and (c).

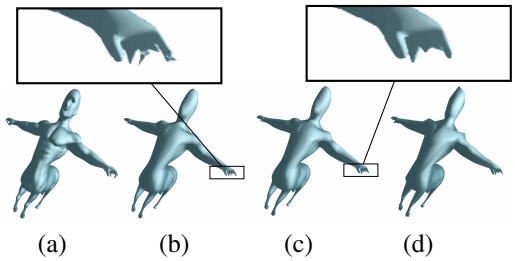

Figure 10: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The shape of the centaur mesh along the back of the horse and human part is more accurately reconstructed, but there are some artifacts at the fingers.

of the centaur is positioned lower in the base network than in the curvature network. The curvature network is closer to the ground truth in this region. The difference in elevation is quite noticeable, and will have a significant impact on the objective measures. This could be one of the explanations why the objective measures are significantly better for the curvature network. The hooves of the curvature network mesh are also larger, which matches the hooves of the ground

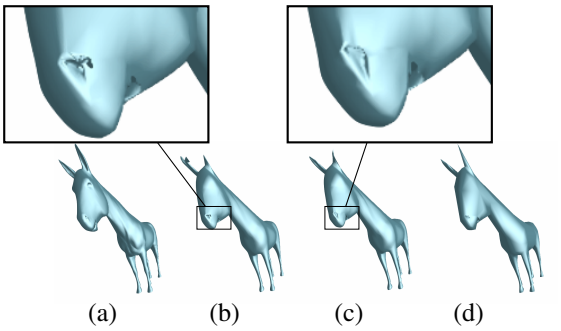

Figure 11: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The curvature network produces slightly better results for the left ear than the base network, but there are small artifacts on the right ear.

truth mesh more closely. The contour of the latissimus dorsi muscle is also wider in the curvature network than in the base network, and is closer to that of the ground truth mesh. The curvature network appears more able to reconstruct the contour of the centaur mesh.

One issue with the centaur mesh can be seen in the fingers where small self-intersections occur. This appears to happen for long and narrow features such as fingers. An explanation for this could be that the curvature varies very rapidly at the tip of these constructs, leading to instabilities in the curvature network.

Horse Figure 11 shows an interesting result for the horse mesh: the height of the left ear has been reduced by the subdivision results, both for the base network and for the curvature network. Also, the ear looks somewhat similar to the ear of the wolf mesh. The style transfer of the neural subdivision, which was reported by [2], might explain why the ear takes on a rounder and shorter shape. Comparing the base and curvature networks for this ear, the curvature network retains the shape of the horse ear a little bit better.

The right ear, which is also a long and narrow feature, shows some of the same artifacts that the centaur mesh did for the fingers. A similar phenomenon appears at the nostrils.

Cat The results for the cat mesh in Figure 12 look quite similar overall, but one clear advantage that the curvature network mesh exhibits is that the left ear has a more rounded appearance than the base network mesh. This is more in line with the ground truth mesh, which has rounder ears than either subdivision result. Another advantage is seen at the jawline: the base network mesh has a highlight that stretches further back than the curvature network mesh, which in turn stretches further back than the ground truth mesh.

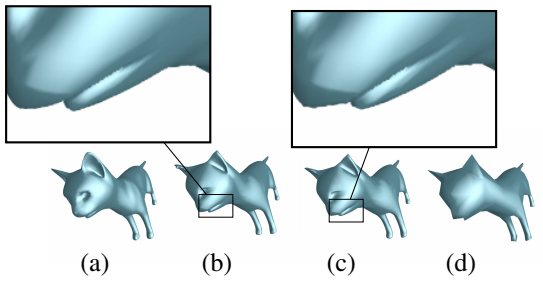

Figure 12: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The left ear has a better rounding for the curvature network mesh than for the base network mesh, but some artifacts appear on the right ear.

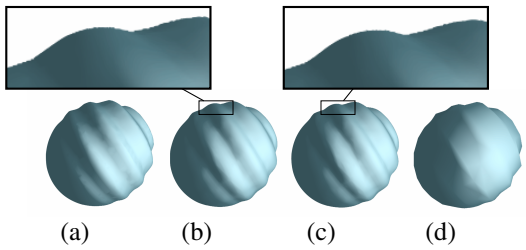

Figure 13: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The results in (b) and (c) are similar; only a few subtle lighting changes can be seen, showing a slightly better reconstruction from the curvature network.

It can further be seen that the right ear shows spikes. The right ear is more pointed than the left ear of the mesh, which could explain why the right ear suffers from these spikes while the left ear does not.

Sphere We use the sphere mesh (Figure 13) to check whether the curvature network hallucinates inappropriate detail in the hemisphere that does

not have the sinusoidal displacement applied to it. The curvature network result shows that the non-sinusoidal hemisphere does not introduce any undue curvature. This implies that the network is able to correctly distinguish between areas with and without sinusoidal patterns.

6.3 Using Ground Truth Curvature

In addition to assessing the network when training with curvatures extracted from the decimation process, we have also experimented with curvatures mapped back to the ground truth. When considering this change, it would make sense that the network is able to extract extra information from the curvature, since the curvature encodes explicit information that is more difficult to encode when only considering half flaps. This way ground truth curvature should also introduce new information to the learning process. Figure 14 shows that new detail has been introduced in the features of the mesh, for example on the nose and the toes of the wolf. However, the mesh also displays noticeable artifacts, especially around the ear of the wolf.

One explanation for these results is that the learning process has overfitted on the wolf mesh. Curvature can vary significantly across the mesh, which makes it sensitive to the exact mapping that is being used. When the reference curvatures are located in a single fixed ground truth mesh, then the mapping from coarse objects allows for overfitting on these exact curvatures, and small deviations from these curvatures will lead to the introduction of artifacts. On the other hand, when the network uses curvatures from the decimation process, then a level of stochastic variation in the curvature has been introduced. This prevents the mesh from overfitting curvature while still learning additional information.

7 Conclusion

In this paper, we propose an extension to neural subdivision [2] that yields significantly improved mesh quality. In particular, we introduce curvature information into the learning process via two curvature modules. We found intermediate curvature value representations using the same mapping procedure as in [2] to find corresponding

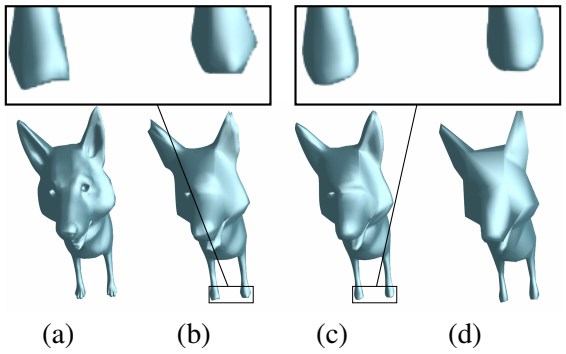

Figure 14: The meshes of the ground truth (a), curvature network (b), base network (c), and the coarse mesh (d). The curvature network introduces more detail into the tip of the nose and into the toes for the wolf mesh than the base network does, but also shows artifacts in the reconstruction of the ear.

curvatures. We subsequently added a curvature loss term to the loss function and used the new loss function to train the network.

Our extended network in many cases produces a better result than the original version both visually and in performance metrics, e.g. with a 6.3% improvement in the Hausdorff distance and an 8.1% improvement in the mean squared error for the wolf mesh. Note that we aimed to manipulate the detail representation rather than significantly change the overall positioning of the vertices. We also demonstrated that our performance improvement is due to the introduction of the curvature information, rather than the increase in size of the network itself.

While our extension succeeds in providing better mesh quality overall, our evaluation also showed that there are still open challenges to address in future work. In particular, our approach could be improved to address the instabilities occurring at long and narrow features. For this, we plan to consider other interpolation schemes for interpolating the curvature across the triangles, improved curvature approximations such as by employing the full curvature tensor, and other more sophisticated approaches to extracting curvature information from the ground truth mesh. Using other network types could also be a promising way of improving the performance of the network, especially sequential network architectures such as recurrent neural networks.

References

[1] C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.

[2] H.-T. D. Liu et al. Neural Subdivision. *ACM Transactions on Graphics*, 39(4.124):1–16, 2020.

[3] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[4] Pieter Barendrecht, Malcolm Sabin, and Jiří Kosinka. A bivariate C^1 subdivision scheme based on cubic half-box splines. *Computer Aided Geometric Design*, 71:77–89, 2019.

[5] Gerald E. Farin, Josef Hoschek, and Myung-Soo Kim. *Handbook of Computer Aided Geometric Design*. North-Holland/Elsevier, Amsterdam, Boston, 2002.

[6] John A Gregory. Smooth interpolation without twist constraints. In *Computer aided geometric design*, pages 71–87. Elsevier, 1974.

[7] Lucia L. Longhi. Interpolating patches between cubic boundaries. Technical Report UCB/CSD-87-313, EECS Department, University of California, Berkeley, Dec 1985.

[8] Gerben J. Hettinga and Jiří Kosinka. Multisided generalisations of Gregory patches. *Computer Aided Geometric Design*, 62:166–180, 2018.

[9] R. W. Clough and J. L. Tocher. Finite element stiffness matrices for analysis of plates in bending. In *Conference on Matrix Methods in Structural Mechanics*, pages 515–545. Wright Patterson Air Force Base, Ohio, 1965.

[10] M. J. D. Powell and M. A. Sabin. Piecewise quadratic approximations on triangles. *ACM Trans. Math. Softw.*, 3(4):316–325, December 1977.

[11] C De Boor, K Höllig, and S Riemenschneider. *Box splines, volume 98 of Applied Mathematical Sciences*. Springer-Verlag, New York, 1993.

[12] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN triangles. In *Proceedings of the 2001 symposium on interactive 3D graphics*, pages 159–166. ACM, 2001.

[13] T Boubekeur and C. Schlick. QAS: Real-time quadratic approximation of subdivision surfaces. In *15th Pacific Conference on Computer Graphics and Applications*, pages 453–456. IEEE, 2007.

[14] T. Boubekeur and M. Alexa. Phong tessellation. In *ACM Transactions on Graphics*, volume 27, 2008.

[15] S. Schaefer, Vouga. E., and R. Goldman. Nonlinear subdivision through nonlinear averaging. *Computer Aided Geometric Design*, 25(3):162–180, 2008.

[16] L. Yu et al. PU-Net: Point cloud upsampling network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799. IEEE Computer Society, 2018.

[17] Y. Wang et al. Patch-based progressive 3D point set upsampling. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[18] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-GAN: A point cloud upsampling adversarial network. In *IEEE/CVF International Conference on Computer Vision*, pages 7207–7211. IEEE, 2019.

[19] M. M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, 2013.

[20] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the 15th European Conference on Computer Vision Part III: Lecture notes in Computer Science*, volume 11207, pages 725–741, 2018.

[21] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3D mesh models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5841–5850. IEEE Computer Society, 2018.

[22] N. Wang et al. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proceedings of the 14th European Conference on Computer Vision Part XI: Lecture Notes in Computer Science*, volume 11215, pages 55–71, 2018.

[23] C. Wen et al. Pixel2Mesh++: Multi-view 3D mesh generation via deformation. In *IEEE/CVF International Conference on Computer Vision*, pages 1042–1051. IEEE, 2019.

[24] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209–216. Association for Computing Machinery (ACM), 1997.

[25] E. Vouga. Lectures in discrete differential geometry 3 – discrete surfaces. `https://www.cs.utexas.edu/users/evouga/uploads/4/5/6/8/45689883/notes3.pdf`, 2014. Accessed: 17-03-2022.

[26] M. Bronstein et al. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[27] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[28] H. Hoppe et al. Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 10–26. Association for Computer Machinery, 1993.

[29] C. Bane Sullivan and Alexander A. Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, 2019.