Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Special Track on AI, the Arts and Creativity

Deep Interactive Surface Creation from 3D Sketch Strokes

Sukanya Bhattacharjee* , **Parag Chaudhuri**
Department of Computer Science and Engineering, IIT Bombay
{sukanyabhat, paragc}@cse.iitb.ac.in

Abstract

We present a deep neural framework that allows users to create surfaces from a stream of sparse 3D sketch strokes. Our network consists of a global surface estimation module followed by a local surface refinement. This facilitates in the incremental prediction of surfaces. Thus, our proposed method works with 3D sketch strokes and estimate a surface interactively in real time. We compare the proposed method with various state-of-the-art methods and show its efficacy for surface fitting. Further, we integrate our method into an existing Blender based 3D content creation pipeline to show its usefulness in 3D modeling.

1 Introduction

Artists have always sought ways to express the world around us in their creations. One of the most common ways of artistic expression has been sketching. Sketching on a 2D medium has been explored by generations of artists over centuries. New digital media has made expressing ideas in 3D possible. However, the interaction mechanisms for modelling in 3D are difficult to learn. In this work we present a method to translate sketch strokes to 3D surface patches. Our method allows an artist to sketch one or more strokes in 3D space. A deep neural network, which is designed to learn from a stream of sparse sketch strokes, predicts a surface patch from these strokes. Since several tools exist for modelling in 3D, we present a method which complements them by interactively refining existing 3D models of objects . We demonstrate this showing interactive *surface in-painting* during model creation, where an incomplete portion in a 3D model is completed by sketching. In classical surface in-painting, the scope of any creative or artistic input towards in-painting is severely restricted [Lui *et al.*, 2012; Verdera *et al.*, 2003; Sahay and Rajagopalan, 2015; Zhong and Qin, 2016; Bendels *et al.*, 2006; Harary *et al.*, 2014] and the desired surface is realized through the control handles post in-painting. Our method can be used to utilize the creative input from the user through the sketch strokes in creating a more desirable surface for in-painting.

*Contact Author

The problem of 3D sketch to surface generation is closely related to the problem of scattered data approximation, where the strokes can be perceived as a sparse set of samples from the surface which would pass through them. Classically, this is solved by regression methods like least squares optimization, fitting with radial basis functions or cubic spline interpolation. Another way to look at this problem is to treat the stroke points as point clouds and then use existing techniques to create a surface from it. However, we show in the paper, both these methods produce sub-optimal results, owing to the sparsity and irregularity of the strokes. We propose a deep neural network **StPNet**, which incrementally creates a surface from a sequence of strokes. We consider patch based 3D models as these are commonly encountered in modelling pipelines. With our method, both a missing or an existing patch can be in-painted using strokes, replacing the existing patch in the latter case. In both the cases, the core challenge is to produce a surface which approximates the strokes well and blends smoothly with the existing structure. We test our method in an existing interactive 3D content creation pipeline by implementing it as a plugin to Blender [Blender, 2017]. We show that we can quickly edit 3D surface models while using Blender's the Grease Pencil [Leung and Lara, 2015] 3D sketching tool

Our main contributions are:

- An end-to-end trained network for incrementally fitting a 3D surface patch to a stream of incoming 3D strokes.
- An interactive method for surface in-painting in a given patch based 3D model using sketch strokes as input.

We discuss related concepts and literature in Section 2. We describe our method with its various components in Section 3. The training approach including the dataset preparation is covered in Section 4. Section 5 is dedicated to the description of various experiments and their results.

2 Related Work

There is a lot of prior work that deals with 3D modeling from sketches. In [Nealen *et al.*, 2007], the authors present a 2D sketching interface inspired from [Igarashi *et al.*, 1999], in which a 3D model can be created and edited using control curves that define surface geometry. A 3D model guided sketching method to build a layered model on top of a base

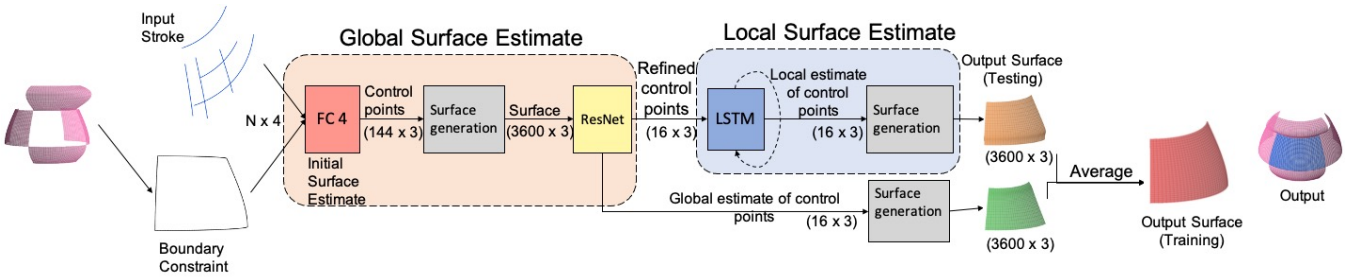


Figure 1: **Overview of StPNet architecture:** (1) *Global surface estimate* takes the 3D sketch strokes (with optional neighboring patch boundaries for blending) and produces its 3D surface approximation. (2) *Local surface estimate* refines the predicted 3D surface to incorporate the changes in shape with each incoming stroke. (3) An additional *post processing* step stitches the predicted 3D surface with the neighboring patches.

model is presented in [De Paoli and Singh, 2015]. The generated curve primitives are represented using a graph where loops indicate a surface modelled as a three or four sided Coons patch [Coons, 1967]. Smirnov et al. [Smirnov *et al.*, 2019] construct 3D models for some categories of man made objects by using a CNN to infer a series of parametric surfaces that fit onto a template model for the given 2D sketch. Xu et al. [Xu *et al.*, 2018] devise a different approach for 3D sketching that is guided by a reference 3D model. The reference model is used to determine the sketching plane for the 3D strokes. Rosales et al. [Rosales *et al.*, 2019] compute manifold surface strips from the sketches drawn in a VR platform. A survey by Bhattacharjee and Chaudhuri [Bhattacharjee and Chaudhuri, 2020] presents more details on various aspects of sketch based content creation including 3D modelling. In more recent work, Wang et al. [Wang *et al.*, 2022] develop an interface using a PDE based geometric modeling to obtain C^0 continuous patches from vertex frames modified in real time.

Geometric deep learning based techniques also have found favour for surface fitting in recent times. Yang et al. propose an auto-encoder that trains on point cloud data [Yang *et al.*, 2018]. A graph-based layer is added to PointNet [Qi *et al.*, 2017] to emphasize local structures. A folding-based decoder then deforms a 2D grid onto the underlying 3D object surface representing the point cloud. Sharma et al. [Sharma *et al.*, 2020] device an end-to-end neural network that decomposes a 3D point cloud into geometric primitives like spheres, cones, cylinders and B-spline surfaces by segmenting the point cloud using DGCNN [Wang *et al.*, 2019] and then predicting the parameter values corresponding to the primitive type for that segment. In [Mehr *et al.*, 2019], a 3D shape editing tool is proposed that uses both sketch strokes and control handles. It uses a series of CNN based autoencoders that learn each connected component of a disconnected manifold and then infer the 3D surface by deforming a pre-learned 3D template specific to each connected component. A detailed survey on geometric deep learning for point clouds can be found in [Guo *et al.*, 2020].

However, these methods either lack in processing an incoming stream of 3D free-form strokes and create plausible 3D surfaces, or approximate viable 3D surfaces from a sparse and irregular set of points as shown in our experiments.

3 Method

The core challenge in our problem is handling sparse and irregular set of 3D points. As opposed to this, the existing methods are designed to handle dense and regular(uniformly spaced) set of 3D points. We need a method that can translate an irregular and sparse structure to a more regular and dense one such that significant shape features and changes in them with the incoming strokes can be extracted. We propose an end-to-end neural network model **StPNet**, which can fit a Bézier patch to an incoming stream of 3D sketch strokes such that it can support interactive in-painting while overcoming these challenges. We represent the inferred 3D patches as bicubic Bézier surface as they offer a good trade-off between ease of processing and versatility in representing various surface shapes.

The input to the model is a sequence of strokes $S = \{s_k\}_{k=1}^{N}$ such that $s_k = \{p_i\}_{i=1}^{n}$ where $p_i = (x, y, z)$ is a point in 3D space. The model outputs a patch B that is represented by its control points $\{c_k\}$.

3.1 StPNet: Patch Fitting on 3D Strokes

The network aims at fitting a 3D bicubic Bézier surface to a sequence of incoming sketch strokes. As noted in [Sharma *et al.*, 2020], predicting the control grid corresponding to a parametric surface from a network is more robust as compared to predicting the surface directly. Thus, we choose to predict the control grid corresponding to the output surface from our model and then compute the surface from it.

Our network consists of the following modules: (i) an initial global surface estimation and a subsequent (ii) local feature based surface estimation (Figure 1). The initial surface estimator consists of a series of fully connected layers to predict a starting estimate of the patch corresponding to the strokes input till then. This is done to translate the given input, which is essentially non-uniformly spaced over the 3D space, into a more uniformly spaced structure to improve the performance of the convolution layers that follow next. The output of this block is fed to the global feature based surface estimator which comprises of ResNet [He *et al.*, 2016] based layers. The ResNet block refines the initial surface such that it fits to all the strokes given till that point of time while trying to capture the overall shape that the strokes represent. This forms the global solution which fits all the input strokes from

scratch. Since the sketch strokes are added in an incremental manner, there is a local characteristic associated with the estimation as the sketch progresses. We leverage this property by passing the output of the global surface estimator to the local feature based surface estimator, which consists of a LSTM layer. This takes into account the surface fitted to the previous strokes and predicts the current surface. The average of these two surfaces is taken as the final output from the network while training. This is done to give equal weightage to the features learned by both the estimators. The surface estimated finally at the end of the entire network, which includes both the global and local surface estimators, is considered as output at test time.

3.2 Blending StPNet: Smooth Joining of Patches

A surface patch has to join smoothly with adjoining patches, so that it can be used as part of a more complex object models. In order to achieve this, we find the common boundaries from the adjacent patches. For this, we maintain a global graph based data structure of all the patches from the object. As the strokes are sketched, we find its four closest neighbors. We maintain a local octree based structure to determine the relative orientation of these neighbors to the input strokes. These boundaries along with the input strokes are given to our network to produce a patch which nearly blends to the rest of the object. To ensure smooth connectivity, we stitch the boundaries of the predicted patch to the corresponding ones from the adjacent patches.

4 Training

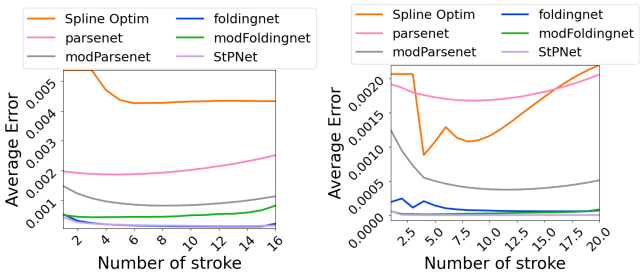
4.1 Dataset

Estimating a surface from a set of strokes is an under-constrained problem, with a wide range of mathematically valid but unrealistic solutions. We would want our model to predict meaningful surfaces from strokes, for which an obvious choice would be to subject the model to a supervised training on a wide range of realistic surfaces.

We use the SplineDataset given in [Sharma *et al.*, 2020] derived from the ABC dataset [Koch *et al.*, 2019]. We extract the B-spline surfaces from the dataset, convert them to bicubic Bézier surfaces and sample random strokes from these surfaces. We call this dataset as ABCSpline dataset. (Train-Valid-Test: $25K$-$1K$-$6K$) We also generate a dataset with the boundary strokes (4) for the surfaces from the SplineDataset, which we refer to as the boundABCSpline dataset. We use Geomdl NURBS [Bingol and Krishnamurthy, 2019] library to create the bicubic Bézier surfaces from the predicted control points. We randomly sample strokes of varied lengths on a surface and on its boundaries. We empirically fix the number of strokes sampled from the interior of a surface patch to 16.

Preprocessing

We downsample the number of points from each stroke to 20 for simpler processing by the model. We fit a cubic 3D Bézier curve to each stroke in the datapoint for smoothing out jitter in the input strokes. Empirically, we found that smoothing the curves improves the performance of the network. We randomly shuffle the order of the interior strokes to make the model stroke order agnostic. Our network is by design limited to take atmost 16 (or 20, in case boundaries are added) strokes. The network takes a fixed input size long enough to accommodate 16 (or 20) strokes. However, in the incremental sketching process, when all 16 (or 20) strokes are not present, we need to signal the network the same. Towards this, apart from 3 dimensions for stroke coordinates, we add another channel of a multi-hot vector which has 1s for the entries which have a sampled stroke point.



(a) ABCSpline (b) boundABCSpline

Figure 2: Plots to show variation of error of various baselines and our method in the predicted surface with ground truth with given number of strokes on two datasets.

4.2 Loss Functions

The loss function needs to measure the difference between the ground truth surface and the two predicted surfaces, one from each global and local feature based surface estimations. We give equal weights to both of them to enforce learning from both feature sets. We use a modified one sided Chamfer distance [Barrow *et al.*, 1977] for this purpose, as used in [Sharma *et al.*, 2020]. The L_2 norm is replaced with a variant of L_α-norm inspired from [Barron, 2019].

$$\mathcal{L}_{\text{surf}} = \sum_{S \in D} \frac{1}{|S|} \sum_{x \in P} ||x - \hat{x}||_\alpha \tag{1}$$

Here, S denotes a surface in dataset D, and x is a point sampled from the predicted surface P and $\hat{x}$ is a point sampled from ground truth surface S such that they correspond to the same grid point in UV space. $||.||_\alpha$ denotes the α norm between two 3D points and $|.|$ denotes the size of a set. We use $\alpha = 1$ which represents a smoothed L_1 norm. The α norm helps account for lack of information about regions on the surface patch from which no input strokes are present (due to sparsity of input).

5 Experiments and Results

We describe various experiments performed to compare our method to the baselines along with the analysis of the results.

We use both sided Chamfer distance [Yang *et al.*, 2018] as given below to measure the accuracy of our method, as compared to various baselines, using the test datasets mentioned in Section 4.1.

$$d = \max\{\frac{1}{|S|} \sum_{x \in S} \min_{\hat{x} \in \hat{S}} ||x - \hat{x}||_2, \frac{1}{|\hat{S}|} \sum_{\hat{x} \in \hat{S}} \min_{x \in S} ||\hat{x} - x||_2\} \tag{2}$$

We compute this distance for 3.6K points sampled from the predicted surface $\hat{S}$ and ground truth surface S, and report the average values in our results. The hyperparameters used are as follows - learning rate: 0.0005, weight decay (Adam): 5e-06, batch size: 32, epochs: 100. The details of the network architecture is given in the following format - *Block name: [(Input dim-Output dim), (Activation)](other info)*. The global surface estimation module- FC4: [(n*20*4, 512), lrelu-0.2], [(512, 432), lrelu-0.2], [(432, 432), lrelu-0.2], [(432, 12*12*3), lrelu-0.2], followed by Resnet: [((3600,4), (1000)), -](Resnet18), [(1000,4*4*3),-](FC). The local surface estimation module- LSTM: [(48,48), -](number of layers=1), [(48,48),-](FC). Here, n is number of strokes, lrelu-0.2 is leaky relu with negative slope 0.2. The surface generator computes the Bézier surface using the control points and knot vectors.

5.1 Patch Fitting Evaluation

We use the methods given in [Sharma *et al.*, 2020] (PsN), [Yang *et al.*, 2018] (FdN) and cubic spline surface interpolation (Spline Optim (SOp)) from Geomdl library [Bingol and Krishnamurthy, 2019] as baselines. We train the first two baselines with the same datasets and set-ups as ours. For a better comparison between our method which is specifically designed for streaming input of strokes, we equip these methods with our loss function and the initial surface estimation networks too (see Section 3.1). We call this variant of Parsenet as modParsenet (mPsN) and that of Foldingnet as modFoldingnet (mFdN). We report the quantitative results in Table 1 (Top half) and qualitative comparisons in Figures 3 and 4.

Also, to show the improvement in approximation of the surface with each added stroke, we compare the error of the surfaces predicted from fewer strokes with ground truth surface. Figure 2 shows the error plots of stroke by stroke output surfaces, and Figure 5 shows how the patches evolve with additional strokes.

Method	Boundary strokes	Initial surface	Loss α	Error
Spline Optim	✗	-	-	0.2639
Parsenet	✗	✗	2	0.1510
modParsenet	✗	✓	1	0.0683
Foldingnet	✗	✗	2	0.0141
modFoldingnet	✗	✓	1	0.0495
StPNet	✗	✓	1	**0.0096**
Spline Optim	✓	-	-	0.1317
Parsenet	✓	✗	2	0.1235
modParsenet	✓	✓	1	0.0306
Foldingnet	✓	✗	2	0.0051
modFoldingnet	✓	✓	1	0.0041
StPNet	✓	✓	1	**0.00035**

Table 1: Comparison between various methods and our method on ABCSpline and boundABCSpline datasets for a Bézier surface patch fitting. Loss: α=2:original Chamfer loss, α=1:modified Chamfer loss. The percentage improvement of our method over the best performing baselines: ABCSpline: **31.91%**, boundABCSpline: **91.46%**

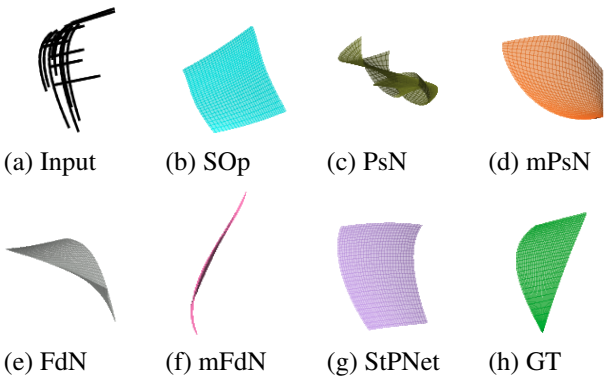


Figure 3: Qualitative results for patch fitting on ABCSpline dataset. (a) input strokes (b)-(g) represent predicted patch from the mentioned methods. (b) cubic interpolation [Bingol and Krishnamurthy, 2019] (c) parsenet [Sharma *et al.*, 2020] (d) modParsenet (e) foldingnet [Yang *et al.*, 2018] (f) modFoldingnet (g) StPNet (our method) (h) Ground truth patch.

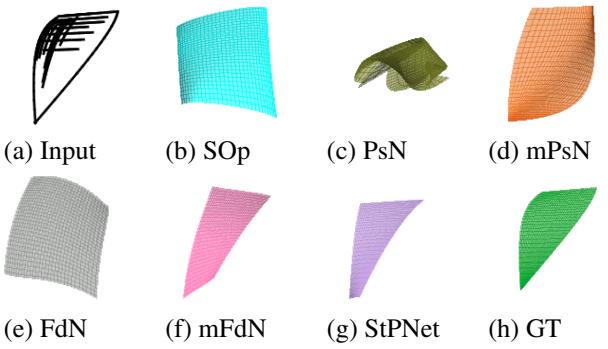


Figure 4: Qualitative results for patch fitting on boundABCSpline dataset. (a) input strokes (b)-(g) represent predicted patch from the mentioned methods. (b) cubic interpolation [Bingol and Krishnamurthy, 2019] (c) parsenet [Sharma *et al.*, 2020] (d) modParsenet (e) foldingnet [Yang *et al.*, 2018] (f) modFoldingnet (g) StPNet (our method) (h) Ground truth patch.

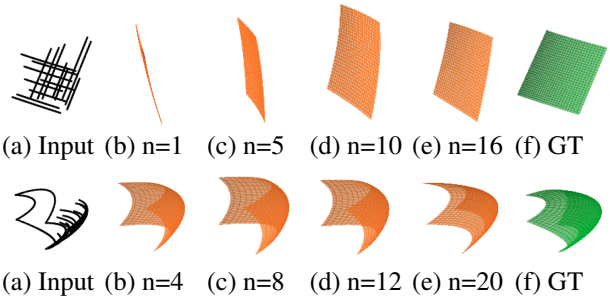


Figure 5: Evolution of the predicted surface from StPNet with increasing number of input strokes (n is the the number strokes). Top row: ABCSpline, Bottom row: boundABCSpline

Boundary strokes	Initial surface est	Global feature est	Local feature est	Error
✗	✓	✗	✗	0.0150
✗	✓	✓	✗	0.0116
✗	✓	✓	✓	**0.0096**
✓	✓	✗	✗	0.00095
✓	✓	✓	✗	0.00038
✓	✓	✓	✓	**0.00035**

Table 2: Ablation study of various components of StPNet on ABC-Spline and boundABCSpline datasets for single Bezier surface patch fitting.

Method	Boundary strokes	Initial surface	Loss α	Error
Spline Optim	✗	-	-	0.1732
Parsenet	✗	✗	2	0.0867
modParsenet	✗	✓	1	0.0428
Foldingnet	✗	✗	2	0.0135
modFoldingnet	✗	✓	1	0.0313
StPNet	✗	✓	1	**0.0110**
Spline Optim	✓	-	-	0.1050
Parsenet	✓	✗	2	0.0722
modParsenet	✓	✓	1	0.0263
Foldingnet	✓	✗	2	0.0100
modFoldingnet	✓	✓	1	0.0098
StPNet	✓	✓	1	**0.0069**

Table 3: Ablation study on ABCSplineClosed and boundABC-SplineClosed datasets for single Bézier surface patch fitting. Loss: α=2:original Chamfer loss, α=1:modified Chamfer loss. The percentage improvement of our method over the best performing baselines: ABCSplineClosed: **18.52%**, boundABCSplineClosed: **29.59%**

We compare the improvement in our approximation when the boundary strokes are explicitly sketched at the beginning and the inner stroke details are filled subsequently. Table 1 (bottom half) and Figure 4 show these results. As part of the ablation study, we compare the performance of various components of our network separately as well in Table 2.

Although we train our network with open surfaces, we also check performance of our method on closed surfaces. For this, we use a dataset of randomly chosen 200 closed surfaces from the same SplineDataset, and call it as closedABCSpline (boundclosedABCSpline) dataset. We split the closed surface into open surfaces and join appropriately. We test our method and the baselines on both the ABCSplineClosed and bound-ABCSplineClosed datasets (see Table 3 and Figures 6 and 7).

We use a synthetic dataset of randomly generated 6K 3D bicubic Bézier surfaces to test the efficacy of our model to generalize to other datasets. We call this dataset as RBST (boundRBST) dataset. We also check the improvement in result when finetuning is employed. See Figure 8 for results.

5.2 Interactive Surface Creation

Synthetically Sampled Sketch Strokes

To evaluate the performance of our network for the surface inpainting task on patch based 3D models, we use the models from the Utah tea set.

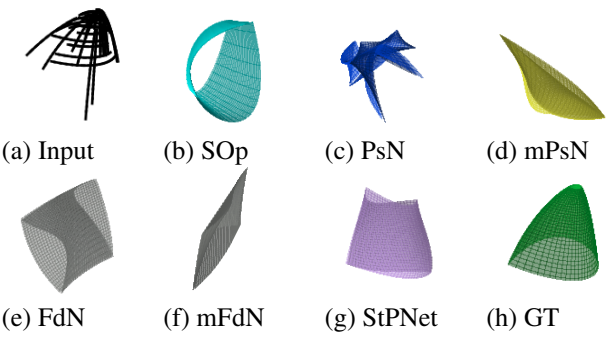


Figure 6: Qualitative results for patch fitting on ABCSplineClosed dataset. (a) input strokes (b)-(g) represent predicted patch from the mentioned methods. (b) cubic interpolation [Bingol and Krishnamurthy, 2019] (c) parsenet [Sharma *et al.*, 2020] (d) modParsenet (e) foldingnet [Yang *et al.*, 2018] (f) modFoldingnet (g) StPNet (our method) (h) Ground truth patch.

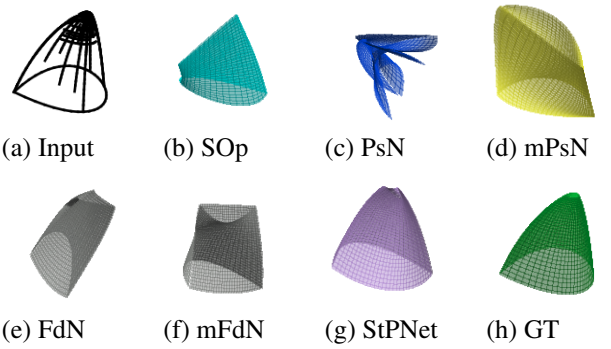


Figure 7: Qualitative results for patch fitting on boundABC-SplineClosed dataset. (a) input strokes (b)-(g) represent predicted patch from the mentioned methods. (b) cubic interpolation [Bingol and Krishnamurthy, 2019] (c) parsenet [Sharma *et al.*, 2020] (d) modParsenet (e) foldingnet [Yang *et al.*, 2018] (f) modFoldingnet (g) StPNet (our method) (h) Ground truth patch.

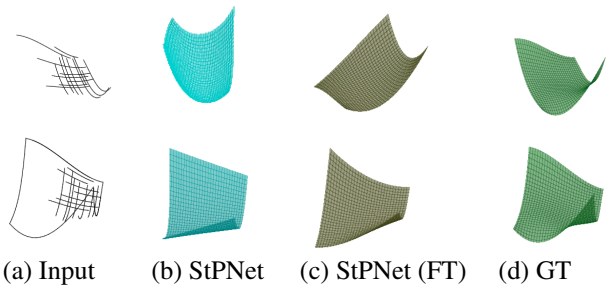
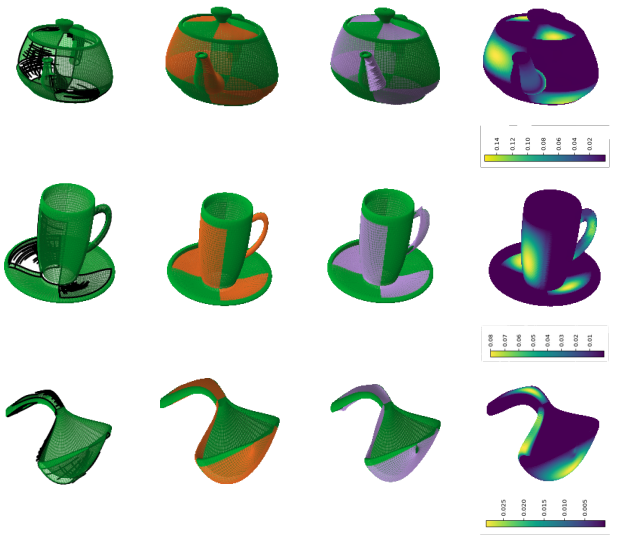


Figure 8: Qualitative results for finetuning to fit patches on RBST and boundRBST datasets. (a) input sketch strokes (b) predicted patch from StPNet (c) predicted patch from fine tuned StPNet (d) Ground truth patch. Top row: RBST, Bottom row: boundRBST



(a) Input (b) GT (c) StPNet (d) Error map

Figure 9: Qualitative results for editing patch fitting on utah teaset models. (a) Incomplete model with input strokes (b) Patches in orange are ground truth for missing patches (c) Patches in lavender are predicted patches for the missing patches (d) Point-wise error between predicted and ground truth missing patches. Top row: Teapot, Middle row: Teacup, Bottom row: Teaspoon.

Model	Teapot	Teacup	Teaspoon
Average error	0.0350	0.0094	0.0035

Table 4: Average error between the predicted and ground truth patches for the Utah tea set models. Average bounding box dimension: teapot: [1.391, 1.154, 0.627], teacup: [0.659, 0.312, 0.635], teaspoon: [0.123, 0.303, 0.056]]

We extract the bicubic patches from the extended set of classic Utah teapot models [Torrence, 2006] and sample strokes on these patches. We randomly remove patches from these models and validate our results by predicting these missing patches in the models. See Figure 9 for the results and Table 4 for average error between the ground truth and predicted patches.

Interactive (Real-time) Sketch Strokes

We implement a Blender plug-in to validate our method for in-painting 3D patch based models interactively. We show examples of adding new patches with different stroke input in Figure 10. See the supplementary video for the full process.

5.3 Results and Analysis

We see from the various experiments in previous subsections that our method significantly improves the surface approximation accuracy as compared to existing methods. The initial surface estimation facilitates our method as well as the baselines to perform better. The global feature extraction captures the overall structure of the surface and features such as inner surface curvatures from given strokes. The local feature extraction helps in capturing how the surface changes

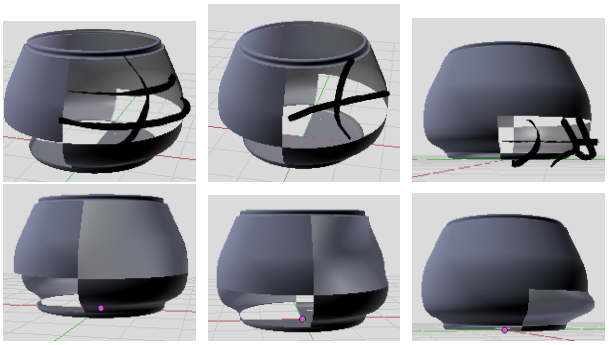


Figure 10: Results of in-painting in real time. Top row: Input stroke on incomplete object, Bottom row: Output surface with the object

with incoming strokes, compensating the ambiguity caused by sparsity of input. As we can see from Table 2, StPNet with local feature extraction performs better than one with the global feature extraction network only. Without the local feature extraction module, the set-up resembles a non-streaming case where there is no notion of previous output from the network. The major advantage of our method is that it can work well with very few strokes and produce viable solutions. We also observe that when boundary strokes are added, the performance of all the methods improve. This shows that these strokes hold significantly important features both for approximating surfaces and blending the patch with other adjoining patches. Our method is able to approximate the closed surfaces without training separately on these surfaces, which reduces the dependency on having a dataset of closed surfaces. We also see that our method is able to capture the shape of the surfaces for newer datasets and the accuracy further improves when finetuning is employed. For interactive surface in-painting, we see that in case of both synthetic and real sketch input blending StPNet is able to produce accurate and visually consistent results. The estimated patches in case of real strokes follow the stroke shapes, which can reduce the effort by the user in further modifying the patch. This can also help in completing patch based models which have a rough but sparsely filled structure in place.

6 Conclusion

We propose a deep neural network based framework that allows for interactive surface estimation from 3D sketch strokes. We demonstrate how this can be used in an existing 3D content creation pipeline in Blender for in-painting of patch based 3D models. We outperform various existing neural and non-neural methods for scattered data interpolation, showing the efficacy of our method. The stability of our method with a sparse set of strokes can prove to be handy when an artist wants to quickly repair a 3D model. We intend to further improve our network to reduce dependency on explicitly determining the boundaries of adjacent patches by enabling the network to adapt to its neighborhood automatically. Also, extending the method to other classes of 3D objects like organic shapes (e.g., human faces) is a direction for improvement. In future work, we also want to explore the use of our network in creation of 3D models from scratch.

References

[Barron, 2019] Jonathan T Barron. A general and adaptive robust loss function. In *Proc. of CVPR*, pages 4331–4339, 2019.

[Barrow *et al.*, 1977] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, Sri International AI Center, 1977.

[Bendels *et al.*, 2006] Gerhard H Bendels, Michael Guthe, and Reinhard Klein. Free-form modelling for surface inpainting. In *Proc. of the 4th International Conference on Computer graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 49–58, 2006.

[Bhattacharjee and Chaudhuri, 2020] Sukanya Bhattacharjee and Parag Chaudhuri. A survey on sketch based content creation: from the desktop to virtual and augmented reality. In *Computer Graphics Forum*, volume 39, pages 757–780. Wiley Online Library, 2020.

[Bingol and Krishnamurthy, 2019] Onur Rauf Bingol and Adarsh Krishnamurthy. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9:85–94, 2019.

[Coons, 1967] Steven A Coons. Surfaces for computer-aided design of space forms. Technical report, MIT Cambridge Project MAC, 1967.

[De Paoli and Singh, 2015] Chris De Paoli and Karan Singh. Secondskin: sketch-based construction of layered 3d models. *ACM Trans. on Graphics*, 34(4):126, 2015.

[Guo *et al.*, 2020] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, 2020.

[Harary *et al.*, 2014] Gur Harary, Ayellet Tal, and Eitan Grinspun. Context-based coherent surface completion. *ACM Trans. on Graphics*, 33(1):1–12, 2014.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.

[Igarashi *et al.*, 1999] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH*, pages 409–416. ACM, 1999.

[Koch *et al.*, 2019] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proc. of CVPR*, pages 9601–9611, 2019.

[Lui *et al.*, 2012] Lok Ming Lui, Chengfeng Wen, and Xianfeng Gu. A conformal approach for surface inpainting. *arXiv preprint arXiv:1212.0981*, 2012.

[Mehr *et al.*, 2019] Eloi Mehr, Ariane Jourdan, Nicolas Thome, Matthieu Cord, and Vincent Guitteny. Disconet: Shapes learning on disconnected manifolds for 3d editing. In *Proc. of CVPR*, pages 3474–3483, 2019.

[Nealen *et al.*, 2007] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 papers*, pages 41–50. ACM, 2007.

[Qi *et al.*, 2017] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of CVPR*, pages 652–660, 2017.

[Rosales *et al.*, 2019] Enrique Rosales, Jafet Rodriguez, and Alla Sheffer. Surfacebrush: from virtual reality drawings to manifold surfaces. *ACM Trans. on Graphics*, 38(4):96:1–96:15, 2019.

[Sahay and Rajagopalan, 2015] Pratyush Sahay and AN Rajagopalan. Geometric inpainting of 3d structures. In *Proc. of CVPR Workshops*, pages 1–7, 2015.

[Sharma *et al.*, 2020] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *Proc. of ECCV*, pages 261–276. Springer, 2020.

[Smirnov *et al.*, 2019] Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. Deep sketch-based modeling of man-made shapes. *arXiv preprint arXiv:1906.12337*, 2019.

[Torrence, 2006] Ann Torrence. Martin newell's original teapot. In *ACM SIGGRAPH 2006 Teapot*, pages 29–es. 2006.

[Verdera *et al.*, 2003] Joan Verdera, Vicent Caselles, Marcelo Bertalmio, and Guillermo Sapiro. Inpainting surface holes. In *Proc. of ICIP*, volume 2, pages II–903. IEEE, 2003.

[Wang *et al.*, 2019] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. On Graphics*, 38(5):1–12, 2019.

[Wang *et al.*, 2022] Shuangbu Wang, Yu Xia, Lihua You, Hassan Ugail, Alfonso Carriazo, Andres Iglesias, and Jianjun Zhang. Interactive pde patch-based surface modeling from vertex-frames. *Engineering with Computers*, pages 1–19, 2022.

[Xu *et al.*, 2018] Pengfei Xu, Hongbo Fu, Youyi Zheng, Karan Singh, Hui Huang, and Chiew-Lan Tai. Model-guided 3d sketching. *IEEE TVCG*, pages 2927–2939, 2018.

[Yang *et al.*, 2018] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. of CVPR*, pages 206–215, 2018.

[Zhong and Qin, 2016] Ming Zhong and Hong Qin. Surface inpainting with sparsity constraints. *CAGD*, 41:23–35, 2016.