ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN
http://www.wscg.eu
WSCG 2025 Proceedings

2D B-spline Curve Reconstruction using Convolutional Auto-Encoders and Distance Fields

Alexander Komar
a.komar@tugraz.at

Saeedeh Barzegar Khalilsaraei
s.barzegarkhalilsaraei@tugraz.at

Ursula Augsdörfer
augsdoerfer@tugraz.at

Institute of Visual Computing, Graz University of Technology, Austria

ABSTRACT

Curve reconstruction is a crucial task in various research domains, including Computer-Aided Design (CAD) and Reverse Engineering. Recovering a B-spline control polygon from a given set of points representing a curve remains an active area of study. We introduce a novel approach that leverages a distance field representation of the curve as input to two neural networks to reconstruct a closed B-spline. One network predicts distance fields to the control points, while the other estimates distances to the control polygon. Using these outputs, we determine the connectivity between control points, enabling the reconstruction of the B-spline control polygon. Our method is evaluated against state-of-the-art machine learning techniques and traditional optimization-based approaches.

Keywords

Curve Reconstruction, Computer-Aided Design, Neural Networks

1 INTRODUCTION

The art of reconstructing an object from point data is an interesting and challenging problem. Curve reconstruction plays a fundamental role in many areas of geometric modeling, including Computer-Aided Design (CAD), Reverse Engineering, and Shape Analysis. In these applications, obtaining a smooth and accurate parametric representation of a curve from an unordered set of points is essential for downstream tasks such as surface reconstruction, manufacturing, and digital modeling. Among various curve representations, B-splines [2] are particularly desirable due to their flexibility, smoothness, and compact control point representation.

It is well known and widely supported simple yet elegant mathematical representation of a smooth curve. A B-spline curve is defined as follows:

$$C(t) = \sum_{i=0}^{n} N_{i,d}(t) P_i, \quad t \in [t_d, t_n], \tag{1}$$

Where $N_{i,d}$ are the B-spline basis functions of degree d that is defined on a knot sequence $U = \{t_1, \ldots, t_{n+d+1}\}$ which can be uniform or non-uniform and the control points are denoted via P_i. For a given knot sequence, the position and connectivity of the control points P_i fully describe the shape of the smooth B-spline curve.

However, recovering a B-spline control polygon from a given set of points remains a challenging task.

By hand this task can be time consuming and tedious, but automation may lead to unwanted results or low quality reconstructions. This is especially true in CAD/CAM where the quality of the representation of an object is of upmost importance to produce high-quality goods.

In this paper we propose a method for reconstructing closed uniform B-spline curves from planar unstructured curve data using neural networks. The input to our networks is a Distance Field (DF) of the curve. The reconstruction of the B-spline representation is split into two complementary data driven approaches, both employing convolutional auto-encoders: From the first one, we learn the DF representing the control point positions. The second one is used to reconstruct the connectivity of the control points.

The remainder of the paper is structured as follows: In Section 2 we summarise related work on data driven Signed Distance Fields (SDF) reconstruction and B-spline reconstruction. In Section 3 we explain how synthetic data is generated for training. In Section 4 we describe the two networks used to predict control points and their connectivity and present a number of results in section 5. Finally we conclude the paper.

2 RELATED WORK

In this section we describe the related work to our application, which represents itself two fold: Neural network SDF reconstruction and B-spline reconstruction.

2.1 Neural Network SDF Reconstruction

Since SDFs provide a fixed size input to neural networks that can represent almost any shape to a certain degree, they are commonly used when interfacing with them. Diverse research was conducted on reconstruction and processing of such SDFs.

Wang et al. [15] used SDFs and Voxel arrays as input to their convolutional auto encoder extreme learning machine (CAE-ELM) to learn features of 3D objects. They trained their network on a subset of ModelNet [16], namely ModelNet10 and ModelNet40 and compared their performance in classifying the models.

SDFs can also serve as intermediate results, as Zhang et al. [17] use a learned SDF as surface representation when reconstructing surfaces from multi-view images. The researchers use a learned light field to represent the texture of the reconstructed surface. Marching Cubes is used for extracting a triangle mesh from the SDF.

Diverting from the classical cube lattice Shen et al. [11] define a SDF on a tetrahedral grid. This representation is part of their shape synthesis pipeline and is predicted from a coarse input point cloud or voxel array. Then the low resolution SDF gets adaptively refined and deformed to more closely represent the surface. To extract a triangle mesh from the SDF the researchers propose a differentiable marching tetrahedra algorithm. Finally this triangle mesh gets subdivided to give the high resolution mesh.

Shim et al. [12] generate a high resolution SDF from Gaussian noise in a two step process. First a low resolution SDF is generated from a SDF of Gaussian noise using denoising diffusion models. Then conditioned on the output of the first model they generate a high resolution SDF. Finally a mesh is extracted using marching cubes.

2.2 B-spline Reconstruction

B-spline curves are a widely used curve representation based on a few connected user-placed points (control polygon) that control the shape of the curve. A point on the curve can be found in two ways: by subdividing the control polygon iteratively using Chaikin's Algorithm [6] or by directly evaluating the curve using the control point positions and basis functions. An overview of 2D curve reconstruction was given by Ohrhallinger et al. [9], where the researchers did a state-of-the-art report on curve reconstruction, where the result is a connected set of points.

In the following we describe reconstruction algorithms that output a B-spline representation of the curve.

Komar et al. [8] proposed a evolutionary algorithm, namely Particle Swarm Optimization (PSO) to reconstruct polygons and B-splines from a set of connected points. They show robustness against noise and sparse sampling. The evolutionary algorithm takes a long time to produce a result, our neural network takes a few milliseconds to execute.

A neural network reconstruction method was proposed by Barzegar et al. [1]. The researchers used an inception network to reconstruct a B-spline control polygon from a set of points, that describe a curve. The number of input points was fixed, since resampling a curve is not costly. The number of control points needed was chosen by the network and the output consists of the control point coordinates and a zero padding. In contrary to the researchers, we are not explicitly limited in the number of control points by the network architecture, further we require less training data and epochs to achieve convergence.

Gao et al. introduced DeepSpline [4], where they reconstructed curves from images. First, the features of the image are extracted using VGGNet [13]. Then the features are fed into a Recurrent Neural Network, that predicts a control point in each iteration along with a stopping probability. The researchers proposed a second step in the reconstruction in order to optimize the positions of the control points. Our proposed method works without this optimization step. The results come directly from the neural network.

In this paper we introduce a method for reconstructing closed B-spline curves from a DF. This DF can be directly computed from a point cloud or connected point set, circumventing the major research question on connecting a set of points resembling a curve [9]. We use two neural networks that predict control point positions and control points connectivity respectively. As outputs we used a DF for both networks. We show how we fair against state of the art methods and robustness against noise and unseen data.

3 DATASET GENERATION

To train the proposed model a large dataset of 2D curves and their corresponding distance fields was needed. This dataset is constrained to closed, non overlapping B-spline curves of degree 3.

To generate closed curves with n control points we start by using polar coordinates and split the unit circle into n parts and randomly offset the start angle by a random number in $[0, 2\pi/n]$. In each part we sample a random point using an angle and a distance in polar coordinates. An illustration can be seen in Figure 1.

After the control polygon generation and the B-spline evaluation the distance field is generated. We chose a resolution of 64x64 to represent the curve. This was chosen to be a trade-off between accuracy in representation and memory on disk and neural network size. Examples of DFs are shown in Figure 2. Left is the DF of the limit curve, middle a DF of the corresponding control points and right a DF of the control polygon.

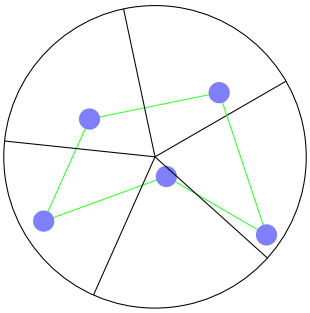


Figure 1: Illustration of the generation of a closed curve. The unit circle is split into 5 parts and a control point is sampled inside each part using polar coordinates.

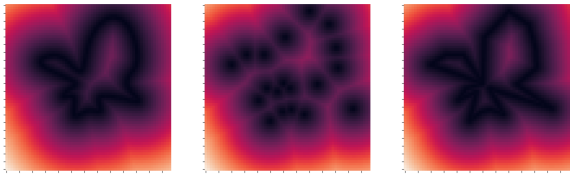


Figure 2: Visualization of the input (left) and ground truth labels for control point positions (middle) and control polygon connectivity (right) contained in our dataset.

Using this algorithm we generate 50000 samples as our dataset using 5 to 20 control points per curve. We use a 80/20 training/testing data split, meaning 40000 curves are used as training data.

4 NEURAL NETWORK B-SPLINE RECONSTRUCTION

The basic idea of the algorithm is that a curve can be well represented by a distance field. It can also be calculated straightforward from a point cloud or connected points of said curve. From this DF of the curve one neural network predicts a DF of the control points the other a DF to the control polygon. From the first neural network output we reconstruct the control point positions and using the second neural network output we finally reconstruct the connectivity between these points. We explain each step of the process with more details in the following.

4.1 Neural Network for DF reconstruction

A general visualization of the network structure can be seen in Figure 3. The basic architecture is based on U-Net [10], where skip layers are introduced before each MaxPool layer. The number of filters per layer also increases going left to right until the latent dimension is reached. Starting from 32 in the first convolution block to 64 in the second and finally 128 in the 3rd, 4th and 5th block. The number of filters in the decoder stays at 128 until the last convolution block at the right, where it

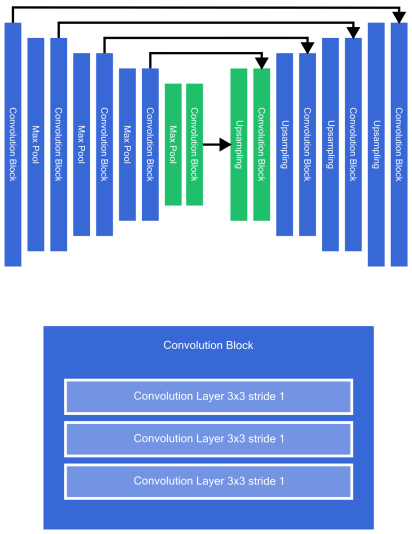


Figure 3: Overview of the network architecture. Top displays the network with the skip layers. Layers in green are only used for network B, which reconstructs a DF to the control polygon. Bottom is a detailed view of the convolution setup.

changes to 64 and finally 1 in the last layer. All activation functions are set to Rectified Linear Units (ReLU)

We use two specialized networks to reconstruct the B-spline curve. Input and output of these networks are 64x64 DFs. The input for both networks is a DF to the curve to reconstruct. Network **A** reconstructs a DF to the control points of the curve. To achieve this it uses all blue layers in Figure 3 and it omits the green layers and the skip connection to them. Network **B** reconstructs a DF to the control polygon of the curve. It uses all layers displayed in Figure 3 to achieve this goal. Both networks are trained on a 80/20 train/test split and for 400 epochs using the Adam [7] optimizer. The Mean Squared Error (MSE) was used as the loss function.

4.2 Control Polygon Reconstruction

Taking the output of network **A** we extract the control point positions from the DF. This is done using a 3x3 minimum filter and comparing the result to the original. The minimum filter works by replacing the current distance value by the minimum in the 3x3 neighborhood. If the minimum filtered DF matches the original DF a new minimum was found. Examples can be found in Figure 4.

To reconstruct the connectivity of the control points the output of network **B** is used in the following way. Starting from any control point reconstructed previously, we look for the edge to another control point with the lowest average error to the DF. This is calculated by marching along the edge and interpolating the DF to get the distance at precisely the current point. Finally the distances are averaged to give an error score to the edge.

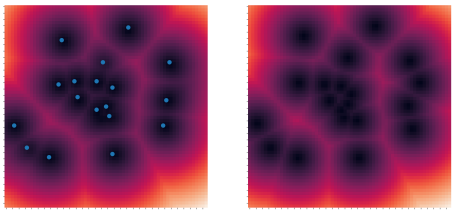


Figure 4: Approximated control points. Left is the predicted distance field to the control points with the estimated positions. Right is the true distance field.

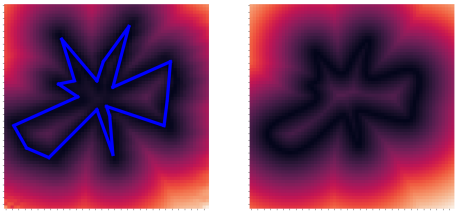


Figure 5: Reconstructed edges. Left is the reconstructed distance field and the reconstructed edges. Right is the ground truth distance field to the control polygon.

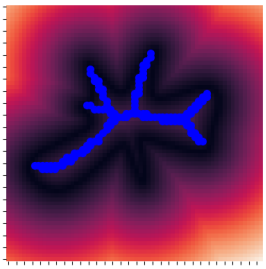


Figure 6: Visualization of the skeleton of the input distance field to the curve.

Having found the edge with the lowest error score, we insert the edge into the control polygon and continue with the point it ends at. A visualization of the reconstructed edges can be seen in Figure 5.

This straightforward algorithm works well, but tends to introduce edges that cross the limit curve because of inaccuracies in the DF. We counter steer this by adding an additional check. First, we calculate the skeleton of our input DF. This is done by using the metric by Gagvani and Silver [3], where the researchers check if each position is part of the skeleton by calculating the mean of the distances in the 8-neighborhood and checking if the distance at the current position is greater than the neighboring average plus some margin TP. More precisely: $MNT_p + TP < DT_p$ where MNT_p is the average distance in the 8-neighborhood of point p, DT_p is the distance at point p. We empirically set the value $TP = 0.002$. An example skeleton can be seen in Figure 6

After the calculation of the skeleton each candidate edge is checked if it intersects the skeleton. If an intersection is found the edge is skipped.

5 RESULTS

In this section we show reconstruction results of our method. First we show reconstructions of random samples of the dataset. Then we show the performance on curves with more than 20 control points and noisy curves. Finally we compare our results to state-of-the-art algorithms. All networks were trained on a system with a AMD Ryzen 9 8945H with 16GB of RAM and a Nvidia RTX 4060 with 8GB of VRAM. No augmentation of the training data was used. Training was split into batches of 5000 curves to not overshoot the memory limit of the GPU.

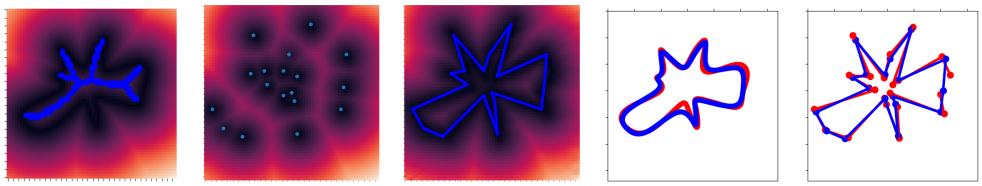
Figure 7 displays a set of reconstructed curves from the dataset. Looking at the examples, the reconstructed curves closely resemble the target curves. Example b) has a particular spike in the control polygon at the bottom right, which the networks could not reconstruct, since the two control points framing the spike are close together and could not be distinguished.

Figure 8 shows examples with a higher number of control points. Even though the examples contain some added complexity and the network has not seen such examples, it is still able to reconstruct the general shape of the curve is some cases. The example at the bottom displays a limitation of the reconstruction with more control points. With some examples the reconstruction fails to include a part of the curve.

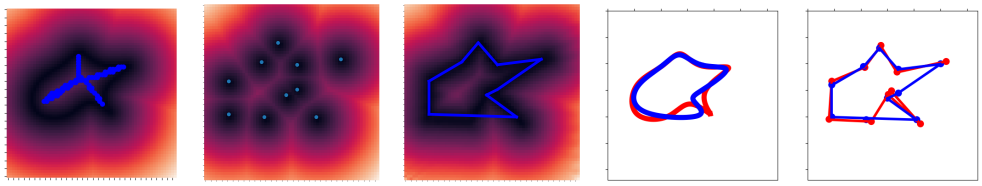
Figure 9 shows noisy examples fed into our pipeline. The input DF is directly calculated on the noisy limit curve data. As can be seen in the visualizations, where the noise is still low (standard deviation $\sigma = 0.2$, with a bounding box diagonal of 90.5) the reconstruction is still reasonable. Increasing the noise the network predicting control point positions estimates too many control points. An extreme case of this phenomenon is shown at the bottom of Figure 9.

Since our proposed network is rather deep we show an ablation study in Figure 10. In our opinion the graphs support the deeper architecture as is displayed in Figure 3. Network **B** is a ore close call, but the larger network (Figure 10 (c)) has a lower loss value at the end of the training.

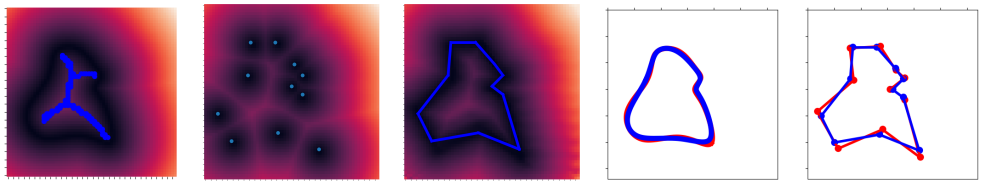
Comparing error metrics with state-of-the-art methods, we can say our reconstruction performance is similar to InceptCurves [1] and DeepSpline [4]. Although the researchers behind InceptCurves were also able to reconstruct open curves, their mean squared error generally is in the ballpark 10^{-2}, similar to our reconstructions. SwarmCurves performs better than our results, as can be seen in Figure 11. The execution times differ by multiple magnitues. Our method took 0.91s to execute, SwarmCurves took 873.98s. The researchers were also able to reconstruct open curves and non-parametric curves with their method.



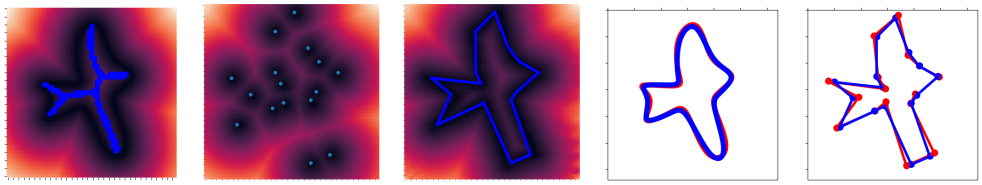
(a) predicted 18 control points, truth has 20, MAE limit curve: 0.04



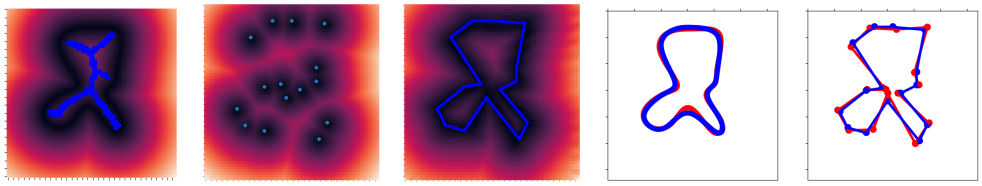
(b) predicted 10 control points, truth has 10, MAE limit curve: 0.0383



(c) predicted 11 control points, truth has 11, MAE limit curve: 0.087



(d) predicted 16 control points, truth has 16, MAE limit curve: 0.045



(e) predicted 15 control points, truth has 15, MAE limit curve: 0.0436

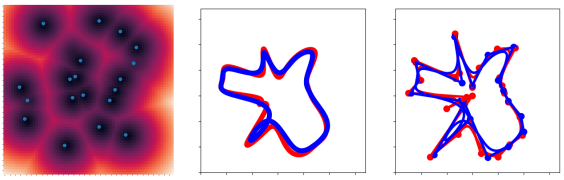
Figure 7: Visualization of results from the dataset. From left to right: input DF and extracted skeleton, predicted control points with predicted DF to control points, reconstructed edges and DF to control edges, comparison of reconstructed curve in blue and target curve in red and comparison of reconstructed control polygon in blue and target control polygon in red.

6 DISCUSSION AND FUTURE WORK

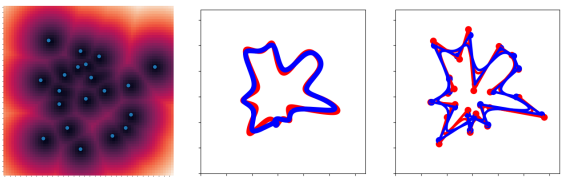
We proposed a pipeline for reconstructing closed B-spline curves from a distance field. Our pipeline consists of two convolutional auto-encoders each specialized to learn a specific task in the reconstruction. Network **A** learns to predict control point positions. The output is a DF to the control points. Network **B** learns to predict the connectivity of the control points. the output is again a DF to the control polygon.

Using the information in the output of network **A** we are able to approximate the control point positions by using a minimum filter. Using the information from network **B** we can greedily connect our approximated control points to reconstruct the control polygon of the curve.

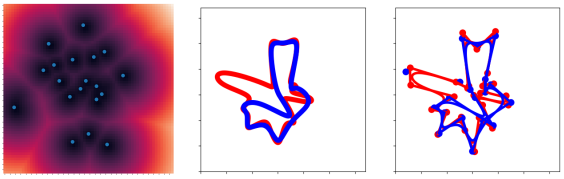
We showed that our pipeline is robust to some noise, even though our training data did not contain noise. More extreme noise starts to reduce the quality of reconstruction drastically. Especially network **A** seems to struggle with noise. This would require a de-noising preprocessing. In future work we will further increase the robustness against noise, by looking into denoising autoencoders [14].



(a) predicted 19 cps, truth has 25, MAE: 0.034



(b) predicted 20 cps, truth has 23, MAE: 0.0217



(c) predicted 19 cps, truth has 25, MAE: 0.0405

Figure 8: Examples with a higher number of control points, which the network has not seen during training. Left is the predicted DF to the control points, middle is the reconstructed curve in blue and the target curve in red and right is the reconstructed control polygon in blue and the target control polygon in red. The caption states the predicted and true number of control points and the Mean Absolute Error (MAE).

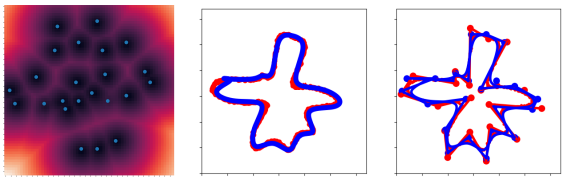
Further, we evaluated the performance of our pipeline on more complex data. We fed the network a curve, constructed with more control points than our training data. The results show that the network can still reconstruct more complex data, although it will try to use less control points.

For future work we would like to extend this method to 3-dimensional curves and shapes. Being able to reconstruct the DF to the control mesh edges would greatly reduce the work required to reconstruct the connectivity of the control mesh. For this task an extension to the network architecture might be needed, for example using more sophisticated skip connections like the ones proposed by Huang et al. [5].

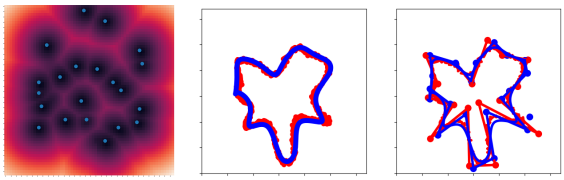
REFERENCES

[1] Barzegar Khalilsaraei, S., Komar, A., Zheng, J., Augsdörfer, U., 2024. Inceptcurves: curve reconstruction using an inception network. The Visual Computer , 1–11.

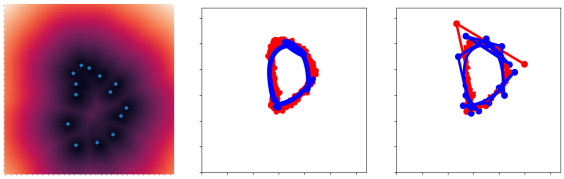
[2] De Boor, C., Rice, J.R., 1968. Least squares cubic spline approximation i-fixed knots. International Mathematical and Statistical Libraries .



(a) $\sigma = 0.2$, predicted 23 cps, truth has 21, MAE: 0.04

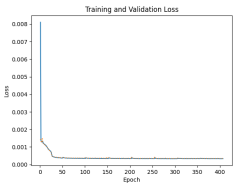


(b) $\sigma = 0.4$, predicted 21 cps, truth has 17, MAE: 0.03

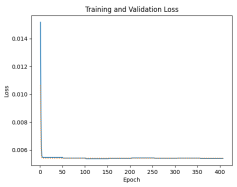


(c) $\sigma = 0.6$, predicted 14 cps, truth has 5, MAE: 0.04

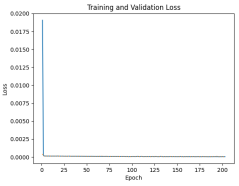
Figure 9: Noisy data as input into our pipeline. Gaussian noise levels increase going down the figure from standard deviation $\sigma = 0.2$ at the top up to $\sigma = 0.6$ at the bottom. Reconstruction quality decreases with increasing noise. The caption states the predicted and true number of control points and the Mean Absolute Error (MAE).



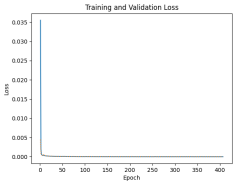
(a) Loss curve for the proposed network **A**.



(b) Loss curve for a slimmer network, where 2 inner convolution blocks were removed.

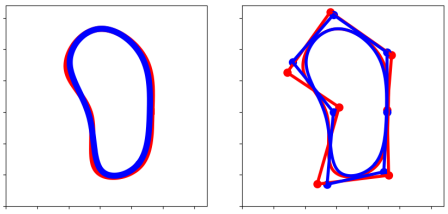


(c) Loss curve for the proposed network **B**.

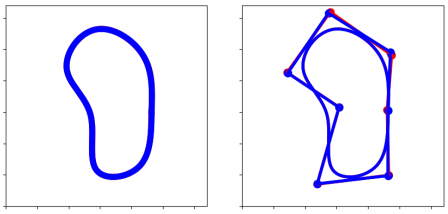


(d) Loss curve for the proposed network **A**, trying to do the work of network **B**.

Figure 10: Blue is training loss, orange is testing loss.



(a) Performance of our method, MAE 0.0045, 0.91s



(b) Performance of SwarmCurves [8], MAE 0.00022, 873.98s

Figure 11: Comparison between our method (a) and SwarmCurves (b) in reconstructing a closed B-Spline with 7 control points. In addition to the visual comparison we list the Mean Absolute Error (MSE) and the execution time in seconds.

[3] Gagvani, N., Silver, D., 1999. Parameter-controlled volume thinning. Graphical Models and Image Processing 61, 149–164.

[4] Gao, J., Tang, C., Ganapathi-Subramanian, V., Huang, J., Su, H., Guibas, L.J., 2019. Deepspline: Data-driven reconstruction of parametric curves and surfaces. arXiv preprint arXiv:1901.03781 .

[5] Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., Chen, Y.W., Wu, J., 2020. Unet 3+: A full-scale connected unet for medical image segmentation, in: ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE. pp. 1055–1059.

[6] Joy, K.I., 1999. Chaikin's algorithms for curves. Visualization and Graphics Re .

[7] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980, doi:10.48550/ARXIV.1412.6980.

[8] Komar, A., Augsdörfer, U., 2023. Swarmcurves: Evolutionary curve reconstruction, in: International Symposium on Visual Computing, Springer. pp. 343–354.

[9] Ohrhallinger, S., Peethambaran, J., Parakkat, A.D., Dey, T.K., Muthuganapathy, R., 2021. 2D points curve reconstruction survey and benchmark, in: Computer Graphics Forum, Wiley Online Library. pp. 611–632.

[10] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham. pp. 234–241. doi:10.1007/978-3-319-24574-4_28.

[11] Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S., 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis, in: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 6087–6101.

[12] Shim, J., Kang, C., Joo, K., 2023. Diffusion-based signed distance fields for 3d shape generation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 20887–20897.

[13] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

[14] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A., 2008. Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103.

[15] Wang, Y., Xie, Z., Xu, K., Dou, Y., Lei, Y., 2016. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. Neurocomputing 174, 988–998. doi:10.1016/j.neucom.2015.10.035.

[16] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1912–1920.

[17] Zhang, J., Yao, Y., Quan, L., 2021. Learning signed distance field for multi-view surface reconstruction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6525–6534.