The Visual Computer (2024) 40:4805–4815
https://doi.org/10.1007/s00371-024-03477-1

RESEARCH

InceptCurves: curve reconstruction using an inception network

Saeedeh Barzegar Khalilsaraei[1] · Alexander Komar[1] · Jianmin Zheng[2] · Ursula Augsdörfer[1]

Accepted: 9 May 2024 / Published online: 12 June 2024

Abstract
Curve reconstruction is a fundamental task in many visual computing applications. In this paper, a data-driven approach for curve reconstruction is proposed. We present an inception layered deep neural network structure, capable of learning simultaneously the number of control points and their positions in order to reconstruct the curve. To train the network, a large set of general synthetic data is generated. The reconstructed uniform B-spline closely approximates any arbitrary input curve, with or without intersections. Because the network predicts the number of control points required for the B-spline reconstruction, redundancy is reduced in the curve representation. We demonstrate our approach on various examples.

Keywords Machine learning · Curve reconstruction · Subdivision curves · B-splines

1 Introduction

In a typical industrial design modeling scenario, a real-world model is produced and subsequently scanned and stored digitally as point clouds. Techniques for reconstructing mathematical curves and surfaces from the scanned data are then applied to recover the geometric properties. What is referred to as *reverse engineering* is employed extensively in the design and manufacturing process and ensures a more efficient and faster design phase [1].

Curve reconstruction, the derivation of an analytic expression for a smooth curve which closely approximates given data points, is a key component in reverse engineering with numerous applications in computer-aided design (CAD), virtual reality and computer vision. A major concern in curve reconstruction is focused both on accuracy control and data reduction. Depending on applications, different types of curves such as parametric curves, implicit curves and subdivision curves are used for fitting. In this work, we reconstruct subdivision curves [2, 3] which in the limit are equivalent to uniform B-spline curves.

B-spline curves (1) are a standard curve representation in CAD due to their flexibility, with widespread usage in a variety of design fields. A designer modifies the B-spline curve by means of control points. Considering that the reconstructed curves enter a design pipeline, B-spline control points need to be reduced. The reduction not only enables a sensible modification of the curve by a designer but also speeds up most of downstream processes and leads to a decrease in storage requirements [4].

The aim of this work is to derive a coarse control polygon for a uniform B-spline which closely fits the data samples.

In this paper we explore a data-driven approach based on Deep Neural Networks (DNN). DNNs are able to learn features from input data without requiring any hand-crafted feature extraction or human intervention [5] and are used for a wide variety of applications to learn from data [6, 7]. By leveraging DNNs, our proposed method aims to capitalize on their ability to capture complex patterns and relationships within geometric datasets, thereby facilitating robust and accurate curve reconstruction.

The pipeline of the proposed method is depicted in Fig. 1: The input of the proposed network is an arbitrary ordered set of geometric data points. Note that our neural network directly operates on point data without the need for computing intermediate features. The output of the network is the number and positions of control points of a smooth B-spline or subdivision curve which closely approximates the input data.

The network was carefully crafted to include a number of convolutional layers. By employing an inception module, where convolutions are performed in parallel instead of in

✉ Saeedeh Barzegar Khalilsaraei
s.barzegar@cgv.tugraz.at

[1] Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Graz, Austria

[2] College of Computing and Data Science, Nanyang Technological University, Singapore, Singapore

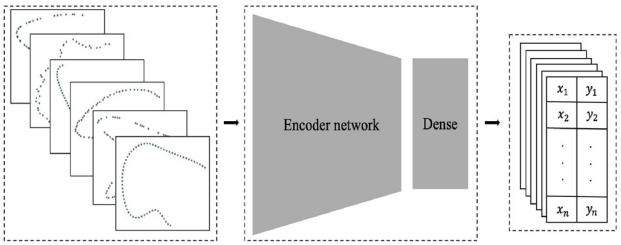


Fig. 1 Pipeline: The input of the neural network is point samples of a planar curve. Via the encoder part, features are extracted from input data and saved as a latent code. Then the features are aggregated by the dense layer and reshaped to predict the (x, y) coordinates of control points

sequence, we are able to achieve even higher accuracy in the reconstruction.

To enable the DNN to adjust the number of control points used in the curve reconstruction for each input data individually, we use a padding approach in the output layer. While padding is frequently employed to add flexibility to the input of a network [8], the authors are unaware of previous work using this approach being employed to the output of a network. Our work demonstrates that padding can be used to add flexibility to the output.

The contributions outlined in this paper are:

- A large set of high-quality synthetic data is derived for training. The data resemble real data and are general. It includes data samples from curves which may have intersections, are uniformly or nonuniformly sampled, or may have different types of noise.
- We introduce a carefully crafted DNN with layers of parallel convolutions, referred to as inception layers, which considerably improve the accuracy of the reconstruction.
- Because the network predicts not only the position, but also the appropriate number of control points to approximate any input curve, the number of B-spline control points is kept low.

The paper is structured as follows: Sect. 2 discusses works related to the approach presented in this paper. The proposed method, depicted in Fig. 1, is explained in detail in Sect. 3. Section 4 includes the evaluation of the proposed methodology and a comparison with other methods, followed by a conclusion in Sect. 5.

2 Background

B-spline curve fitting and reconstruction are closely related and solve common problems in many fields of research, and numerous approaches have been put forward.

Reconstruction aims at finding a parameterized curve $C(t)$ which approximates given a number of ordered data points d_i. The parametric curve is typically assumed to be a B-spline curve

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

where $N_{i,d}$ are the B-spline function of degree d defined on a knot sequence $U = \{t_1, \ldots, t_{n+d+1}\}$. The B-spline curve C approximates the control polygon given by n linearly connected control points P_i. The knot sequence may be uniform or nonuniform. The control points P_i and the knots t are the design freedoms to satisfy approximation requirements.

Reconstructing B-splines from input data was first introduced by de Boor and Rice [9], where given an initial set of knots their position is optimized by means of least squares approximation.

As it is usually not known in advance how many control points are at least required to achieve a good reconstruction, they extended their approach to variable knot vectors using an iterative approach, where systematically new knots were introduced in each iteration. They found that their approach always led to more knots than required [10]. Since their initial work, a wide range of similar least squares approaches have been put forward which typically suffer similar problems.

An important concern in B-spline curve fitting is to reduce the number of knots and, correspondingly, the number of B-spline control points while reconstructing at high accuracy. Therefore, the data reduction often requires some iterative process, where new control points are introduced in each iteration [11–14]. This typically leads to B-spline curves with more control points than required. Various solutions to reducing the number of redundant control points have been proposed [4, 11, 12]. Instead of locally optimizing each control point individually, a global approach to optimizing the approximation has shown better results.

More recently various intelligent approaches have been put forward in order to reconstruct a curve or surface, i.e., evolutionary algorithms [15], the particle swarm optimization method [16–19] and machine learning [20–23].

A variety of neural networks have been devised to reconstruct curves or surfaces from input data.

Laube et al. [23] proposed a data-driven approach to approximate open B-spline curves. Two interdependent networks are used to learn parametric values and knot vectors. Their approach requires segmenting the input curve into smaller pieces.

A B-spline-based reconstruction technique on binary images and point clouds was presented by Gao et al. [22]. They used a hierarchical recurrent NN for curve and control point prediction and were able to detect multiple curves in the image. To achieve the necessary accuracy, their learned

predictions are post-processed using classical optimization methods. In this paper, we aim to achieve high reconstruction accuracy without any post-processing. Scholz and Jüttler [24] introduce a deep residual network as a means to acquire a parameterization for approximating the input data points with a polynomial curve. Tong et al. [25] introduce a novel function approximation method wherein the Taylor series serves as the activation function within the NN for polynomial fitting. Mandal and Uhlmann [26] propose a convolutional neural network to fit a parametric curve onto a biological object contour. The input of the network is a bioimage and the network should learn the distribution of control points to capture the boundary of the existing object in input data. While their approach reconstructs the simple curves from input data, we found that it may fail to accurately reconstruct more complex curves.

Approaches often aim at finding optimal knots placements for a given B-spline, where support vector machine (SVM) and DNNs are trained to predict knots for given discrete points [20, 22, 23]. Again, an iterative approach is chosen, where new points are introduced in each step.

More recently an unsupervised DNN has been proposed to optimize the knots of a fixed-size knot vector [21]. While their approach is promising as it liberates from having to generate large amounts of synthetic data, their DNN-Solvers can give knot positions only for a fixed number of knots and thus, for a given number of control points to improve the fit of the B-spline curve.

In literature, B-spline reconstruction focuses on improving the approximation by adjusting knot placement. However, in a design context, nonuniformity is often only applied in specific cases, e.g., where the control polygon should be interpolated. For a given knot sequence and degree, the curve is fully defined by the position of its control points P_i.

If a uniform parameterization is chosen, the curve corresponds to a subdivision curve [3]. Chaikin [2] first introduced a fast recursive algorithm, referred to as *subdivision*, for generating quadratic B-spline curves on uniform knot vectors. Subdivision curves have since been generalized to various degrees.

If the knot vector is fixed and uniform, the derivation of the curve using the control points may be through the analytic expression for a B-spline curve (1) or may be recursively derived through the subdivision.

Most approaches optimize either a fixed set of knots [21] and control points or employ an iterative approach, where new control points are introduced in each iteration [20, 22] to derive the necessary control points. Both may lead to redundancy in the curve representation by employing more than the required number of control points in the reconstruction.

Instead, we introduce a data-driven approach where the knot vector is fixed. Because we aim to reconstruct subdivision curves, we assumed the knot vector to be uniform. The number of design freedoms, in form of control points, required for a good approximation is derived by the system together with the positions of these control points.

The solution is a sparse reconstruction of a single curve which can be represented using 4–10 control points. We demonstrate our algorithm by applying it to several examples of open curves, which may intersect.

Although examples are presented for quadratic and cubic 2D curves, our method can be extended in a straightforward manner to fit data points by a B-spline curve in higher dimensions at arbitrary degrees.

3 Methodology

We expect that reconstructed curves enter a design pipeline. For the designer to effectively manipulate the reconstructed B-spline curve, the number of control points P_i should be low. This requires careful positioning of the few available degrees of freedom in order to achieve an accurate reconstruction.

Our DNN is trained to learn the relationship between the input point samples and the uniform B-spline control point positions P_i. The network learns not only the order and position, but also the number of control points P_i required to define a B-spline curve that fits a smooth curve to a given set of ordered data points.

Using a data-driven approach to reconstruct curves involves some challenges which have been addressed in this paper:

- Training the DNN requires a large training dataset of points sampled from a curve together with the control points. Real data are highly dependent of the application and are generally unavailable.
- The output dimension of a DNN is fixed, whereas for reconstruction, we seek to avoid superfluous control points which are not required for a good reconstruction.

In this paper, all the above-mentioned challenges have been addressed.

Because large amounts of real data are typically unavailable for training, synthetic data were derived. Any network is very sensitive to its training data. Creating good general synthetic data which shares characteristics with real data is challenging. We will explain the careful derivation of the synthetic dataset in Sect. 3.1.

We address the fixed size output problem using a padding approach [27]. Using the proposed method, our network provides 4-10 control points, depending on the number of control points required to reconstruct a smooth curve from a given input data. The DNN adapts the size of the output vector automatically, providing a solution at a dimension required to ensure an accurate reconstruction. Avoiding redundant con-

trol points ensures that a designer can manipulate the curve after reconstruction sensibly and in a controlled manner.

To increase the accuracy of the data-driven reconstruction, we carefully designed a suitable DNN. By utilizing the inception module [28] within its architecture, the network is able to perform convolutional operations using filters of varying sizes concurrently, as opposed to the traditional sequential approach. This enables the network to capture multi-scale features within a single layer. While including the inception module increases the number of parameters in our NN architecture, this architecture represents a significant breakthrough in terms of efficiency. We believe that the inception module's ability to capture details at different scales as well as its role in reusing features effectively is key to this efficiency.

Two important components in any deep learning-based approach are an appropriate network architecture and a sufficiently large representative dataset. In the following sections, the above-mentioned components will be explained in detail.

3.1 Dataset generation

Since there is no publicly available dataset for this task, we generate a synthetic dataset. To derive good general synthetic data is challenging because care has to be taken to avoid a *clever Hans* effect [29], where correspondences between synthetic data and the mathematical expression are learned which are not in fact present in real-world data.

To create general B-spline curves, which may be observed in a wide range of applications in visual computing, we used two different methods for control points generation. In the first method, a set of random control points within a circle of radius R is generated. To generate n control points, P_i, the circle is divided into n sectors, S_j, where $j \in 1, n$, each a random angle α_j wide. For each sector, we derive two random numbers: a random radius $r_j \in (0, R)$ and a random angle $\beta_j \in (0, \alpha_j)$ to derive the Cartesian coordinates (x, y) of a control point. Using a circle for the derivation of the synthetic data is ideal to derive closed curves, but here it is also employed to derive open curves. In the second method, the 2D plane is divided into uniform sections based on the number of control points vertically and in each section, a random point is generated.

Figure 2 depicts two examples of control point generation with two different approaches.

Using the derived control points, we generate open or closed control polygons used to derive uniform B-splines. To derive closed curves we assume the first and last control points coincide. For open curves, we do not interpolate end points of the control polygon, consistent with a B-spline with a uniform knot vector. The control polygons may be intersecting. The connectivity between control points is implicit

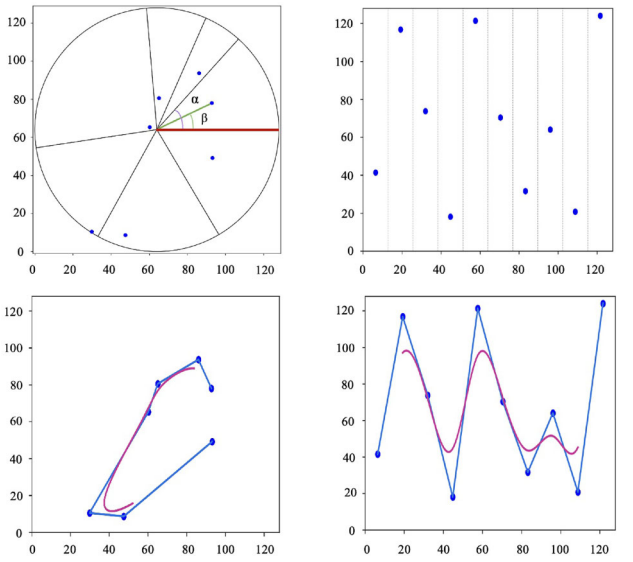


Fig. 2 Generating synthetic data. Top left: The circle is divided into seven nonuniform sectors S_j, where $j \in (1, 7)$ starts from angle zero (red line). α_j is the angle of each circle sector S_j. In each sector, a random angle (β_j) and radius are generated to determine the position of control points (blue). The top right shows dividing the 2D plane into uniform sections vertically and generating random control points in each section. The derived control polygons (blue) and their corresponding open cubic B-spline curves (pink) are shown below

by the order of derivation. To allow for intersection in the curve, we may randomly exchange the order of derivation.

We generate examples of cubic planar curves. However, the approach presented in this paper may be adapted to generate and also learn the reconstruction of arbitrary degree B-spline curves of higher dimensions by adjusting the derivation of the synthetic training data accordingly.

Input data are derived by sampling of the generated B-spline curves to provide the (x, y) coordinates of 161 data points, such that the input to the network is a vector of dimension (161×2). To create a general dataset, we apply equidistant and nonuniform sampling for some curves in the dataset. In addition, some sample data are perturbed by uniform or Gaussian noise.

To train the network presented in this paper, we generated a synthetic dataset of 490000 planar uniform cubic B-spline curves with different number of control points (from 4 to 10). Seventy percent of the dataset is used as training data and the rest for evaluation.

3.2 Neural network model

The structure of the NN architecture is crucial for achieving a good performance in a data-driven approach. We carefully examined various architectures for our task.

Using a fully connected network, where each neuron in one layer is connected to every neuron in the next layer, resulted in a high reconstruction loss. Increasing the num-

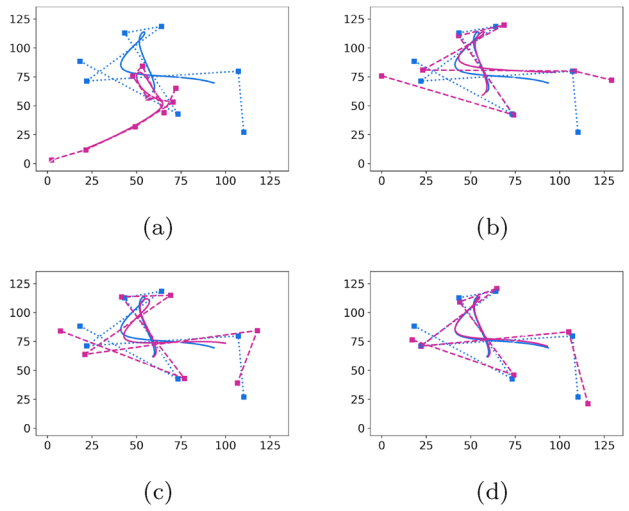


Fig. 3 Different networks reconstruct a 2D B-spline curve from input data. **a** Fully connected network, **b** CNN, **c** CNN with two inception modules, **d** CNN with four inception modules. Details to these networks are found in the text. The original curve from which the sample points have been taken and the corresponding control polygon are depicted in blue and the network's prediction is shown in pink

ber of layers and neurons in each layer does not improve the reconstruction. While employing this network in the decoding part works well, see e.g., by Park et al. [30], these networks are not practically suited to extracting relevant features from spatially correlated data such as curves. Figure 3a shows the reconstruction obtained employing a fully connected network.

Utilizing several convolutional layers in the network, essential patterns and features are extracted from the input data and the extracted features are passed to the next layers as feature maps. To accelerate the learning process, batch normalization [31] is added to adjust the feature maps followed by a pooling layer to reduce the dimensions of the feature maps. We use max pooling which takes the maximum value of patches in a feature map. Figure 3b shows the reconstruction obtained employing a convolutional neural network with three convolutional layers followed by batch normalization and max pooling layers. Mandal and Uhlman [26] applied CNN on biological images. While CNN is suitable for their type of data which only contained simple shapes, it did not successfully reconstruct complex curves in our experiments.

Although convolutional layers improve the performance of the network, the error of the reconstruction remains high. Tuning hyper-parameters such as the number and size of kernels in convolutional layers, as well as increasing the number of epochs, testing different batch sizes and learning rates, did not significantly impact the network's performance.

Adding an inception module [28] improves the performance of the network considerably. This module defines multiple convolutional filters of different sizes in parallel to enable the network to learn spatial features at different scales. The extracted multi-level features are then concatenated and fed to the next layer.

We experimented with different numbers of inception modules. An example of a curve reconstruction using a NN with two inception modules is shown in Fig. 3c. The final architecture shown in Fig. 4 gave consistently best results obtained using four inception modules, see e.g., Fig. 3d.

In Table 1, the average reconstruction error using the above-mentioned networks on 2000 test data sampled from B-spline curves is illustrated.

One fully connected (FC) layer with 20 neurons is defined at the end of the network to aggregate the extracted features and do the prediction. To get the (x, y) coordinates of control points, the output of the FC layer is reshaped to 10×2, thus providing a maximum of ten control points for a given input data.

Since we want to predict a flexible number of control points, the output size of the network needs to be variable. In order to learn variable control points, padding is added to the output of the network. Padding is a special form of masking where the masked steps are at the start or the end of a sequence to maintain the dimension [27]. The initial part of this array corresponds to the coordinates of the control points required to derive a curve which fits the input data closely, while the remaining entries in the output vector remain zero. In order to generate the predicted curve based on the output of our neural network, we ignore zero values and generate the curve with predicted control points.

ReLU is added as an activation function to the convolutional layers. The number of convolutional kernels is 64 in the first inception module and 256 in the rest which are determined via trial and error. Adam [32] with a 0.001 learning rate is used as an optimizer. This network is trained for 4000 epochs. This many epochs are required for learning, because we have a complex NN system and much data, but are low enough to prevent overfitting.

The implementation is based on TensorFlow and all experiments are run on a Linux machine with a GeForce RTX 4090 GPU. The number of trainable parameters of InceptCurves is about 6.5 million; thus, it needs approximately 14 GB memory for a batch size of 512 in our system.

3.3 Loss function

During the training phase, the weights are adjusted through gradient descent on a loss function that evaluates the DNN's performance.

We have experimented with different types of loss functions to train the network. Examples of the reconstruction for networks trained with different loss functions are shown in Fig. 5, with the complexity of the loss function increasing from left to right: the average maximum Euclidean distance between the corresponding control points of the target and

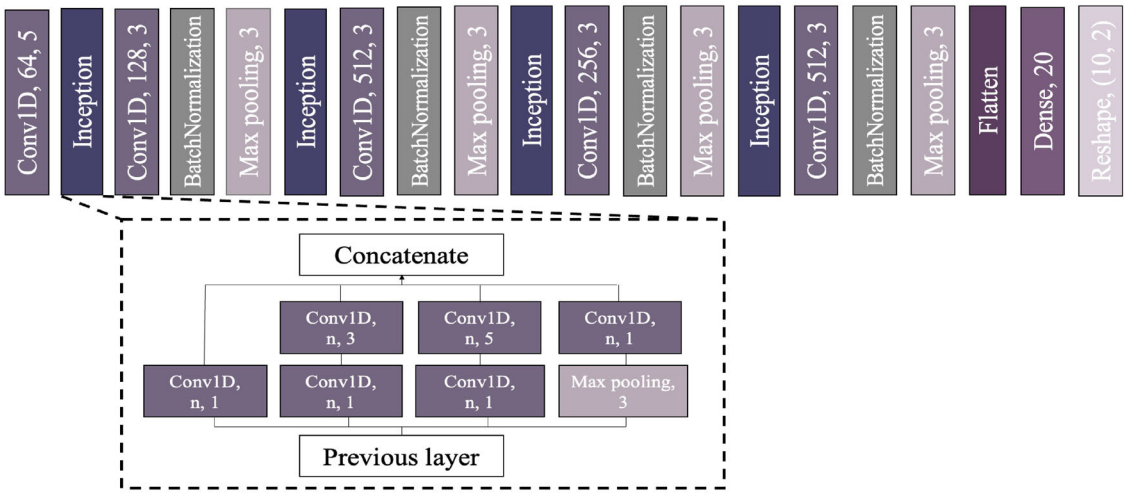


Fig. 4 The architecture of the network employed in this paper. The first number in each convolutional layer shows the number of filters and the second one is the size of the filter. We used four inception modules and in each, the number of filters is specified via n since we use different numbers in each module. The output of the network is reshaped to (10×2) to predict the coordinates of control points

Table 1 Table shows a comparison of the reconstruction error, measured as Chamfer distance and mean squared error between the original input curve and its learned reconstruction, using four different networks as described in the paper. The best results are achieved using four inception modules within the network

Network	Average CD	Average MSE
Fully connected network	4.456	6.268
Convolutional network	1.181	3.231
With 2 inception modules	0.329	1.165
InceptCurves	0.103	0.382

predicted curves, the mean squared error between control points of the target and the predicted B-spline, and the mean squared error between point samples along the target and the predicted curve. The network took considerably longer to be trained when comparing point samples along the curve (Fig. 5 on the right) and we did not observe a significant improvement in the accuracy achieved when comparing the B-spline curves instead of the control points. Therefore, mean squared error is selected as a loss function to compare the target and predicted control points. The loss is computed as follows:

$$Loss(P_i, \hat{P}_i) = \frac{1}{n}\sum_{i=1}^{n}(P_i - \hat{P}_i)^2 \tag{2}$$

where P_i are the target control points, $\hat{P}_i$ are the predicted control points, and n is the number of control points.

4 Results

To test the network's performance, a test set of 2000 open cubic curves is generated. We apply different sampling (uniform and nonuniform) and add noise to this set separately to analyze the performance of our inception network. Gaussian

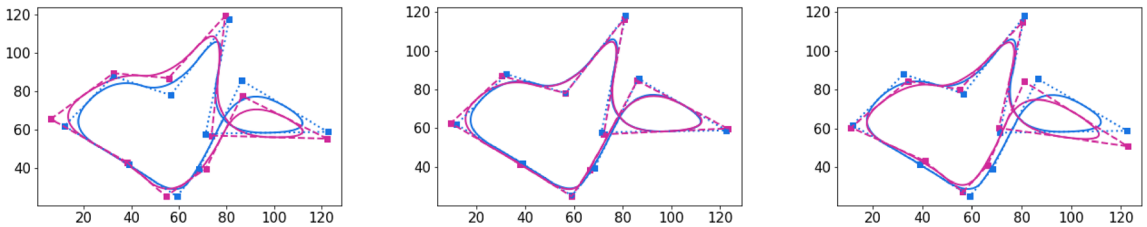


Fig. 5 The data-driven curve reconstruction employed on the same point samples predicted by NN using with three different loss functions. The target control points and corresponding quadratic B-spline curve are visualized in blue, while the predicted control points and corresponding curve are depicted in pink. Left: The network with maximum Euclidean distance loss function. Middle: the evaluation of the network with control points comparison using mean squared error is displayed. Right: the evaluation of the network with curve comparison loss function is shown

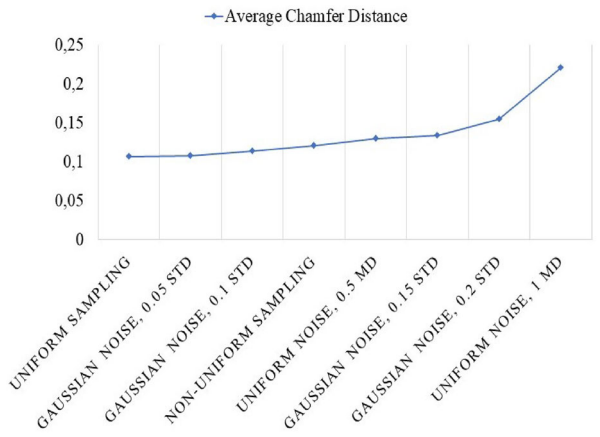


Fig. 6 The chart shows the average Chamfer distance between the target and predicted curves while feeding different noise or sampling methods to input curves

noise with four distinct standard deviations of 0.1, 0.5, 1.0 and 2.0 and Uniform noise with 0.5 and 1 maximum deviation are added to this test set separately. To compare the difference between target and predicted curve, we use the Chamfer distance [33] and MSE, since these are the error measures most typically used in curve reconstruction. As shown in Fig. 6, the network has the lowest error when fitting curves to equidistant samples. One reason is that there are more equidistributed samples in the training set. If the data are subject to noise or changing the sampling method, the accuracy of the reconstruction decreases. As expected, the more noise the larger the error of the reconstruction. However, even in this case, the overall shape of the curve was always captured.

Examples of predictions made by the inception network are depicted in Fig. 7. The last example (bottom right corner) is a nonparametric curve, $f_2(t) = \frac{1}{3}e\left(-\frac{81}{4}(t-0.5)^2\right)$. Our network reconstructs this curve with a B-spline curve with 8 control points. The same example was used by Park et al. [4]. In their approach, they used 30 control points. In this work we are aiming at curve reconstruction which is a lot sparser than previous works.

Note that no post-processes have been applied to optimize the output of our neural network as in [22]. The network can handle curves with and without intersections. It can reconstruct the overall shape and recover high curvature features well (Fig. 8).

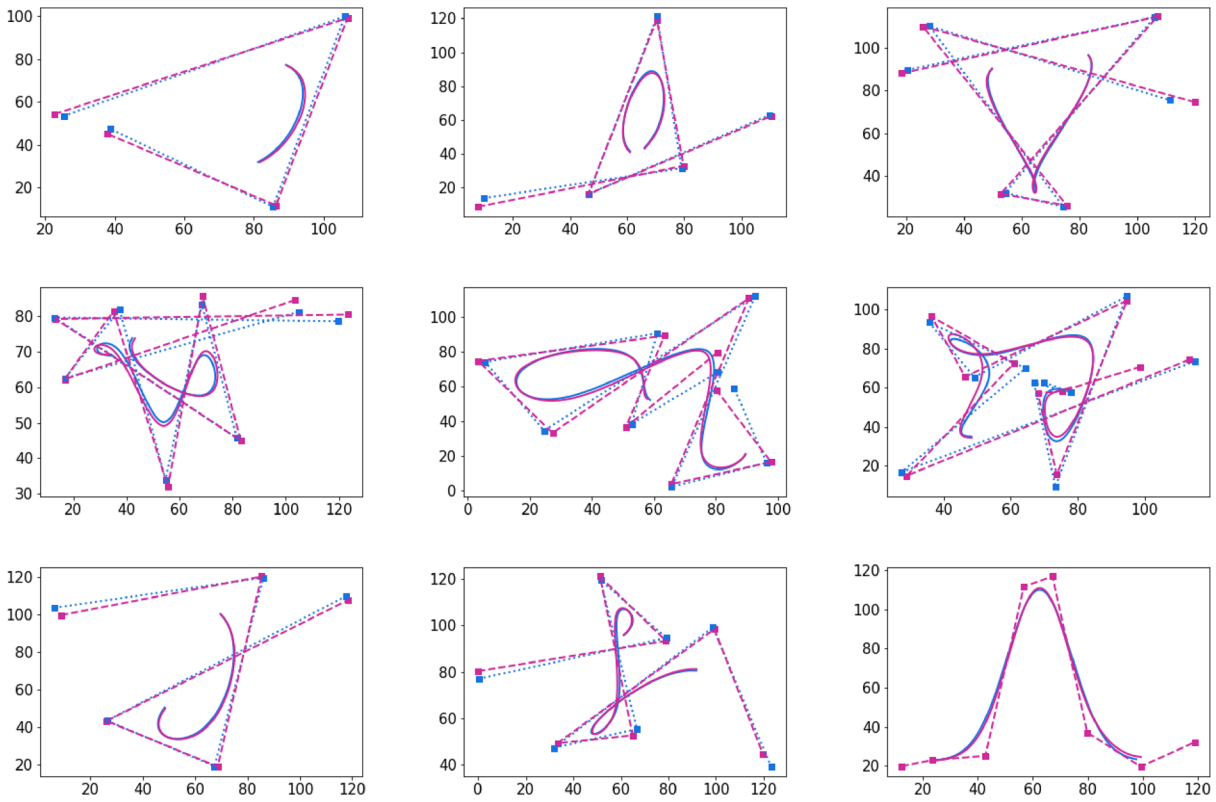
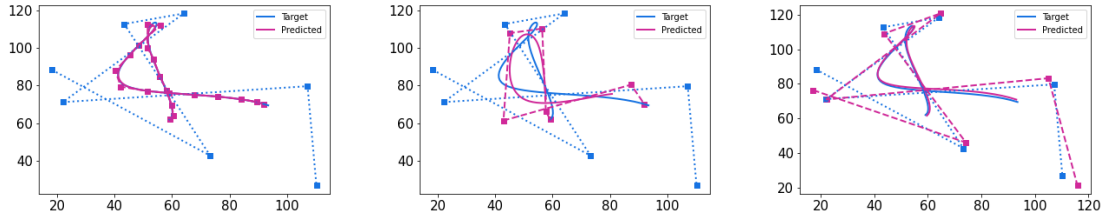


Fig. 7 Nine examples of InceptCurves reconstructions are shown. The target curves and their control polygon are shown in blue color, and the predicted control points and the corresponding smooth B-spline curve are in pink. The last example (bottom right corner) is a nonparametric curve which has been reconstructed as a B-spline curve using 8 control points, much sparser than previous work, e.g., [4]



(a) Least squares fitting with 20 CPs. (b) Least squares fitting with 7 CPs. (c) InceptCurves with 7 CPs.

Fig. 8 The left figure shows the least square curve fitting with 20 control points. Fitting a curve with seven control points using the least square method is depicted in the middle picture. Fitting automatically a curve with seven control points using InceptCurves is shown in the right picture

The InceptCurves can not only reconstruct smooth curves from data sampled nonuniformly from curves with and without intersections but also noisy data. Examples of uniform B-splines fitted to noisy and nonuniform data are shown in Fig. 9.

We find that in general, open curves are reconstructed at higher accuracy than closed curves. We expect this since in a closed curve the start and end control points are not readily defined as in open curves.

One of the standard techniques for curve fitting is based on the least squares approach. In Fig. 8, we provide a comparison between InceptCurves and the least squares approach on an example curve.

Note that least squares can also achieve a high accuracy, but only when many more control points are fitted. To show this, compare InceptCurves to least squares in Fig. 8, where we see that InceptCurves approximates the curve with 7 control points, which is almost as good as the approximation of least squares with 20 control points. A comparison between our approach and the least squares method on several nonparametric curves using 10 control points is provided in Table 2. InceptCurves achieves a more accurate reconstruction of these nonparametric curves. It is able to capture the overall shape and reconstruct the regions with high curvature more precisely. The reconstruction of these nonparametric curves using InceptCurves is depicted in Fig. 10.

There are more advanced techniques of curve fitting based on stochastic optimization approaches such as the genetic algorithms and particle swarm [18, 34]. However, these methods suffer from several drawbacks compared to our approach. First, they typically need a proper initialization for the method to converge to a global optimum. Otherwise, the solution can reach only a local optimum which is undesirable. Such an initialization would need a knowledge of the shape of the curve which is not needed in our method. Second, the number of control points in these methods is often not easily adjustable. In contrast, our method achieves a low approximation error with a very few number of control points as presented in Table 2. Finally, stochastic optimization methods are slow in producing the approximated curve

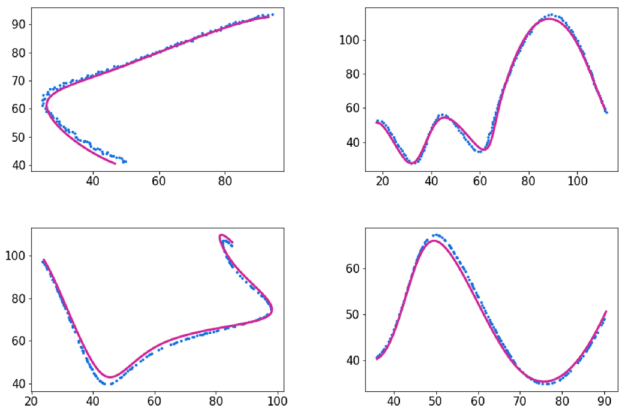


Fig. 9 In the first row, two examples of noisy curves are fitted with cubic B-spline curves with 6 and 10 number of control points. The noisy curves are depicted in blue and the fitted B-spline curve is shown in pink. In the second row, two nonuniform curves are fitted with cubic B-splines with 7 and 6 number of control points. The nonuniform curve is depicted in blue and the fitted B-spline curve is in pink

Table 2 The mean squared error between target and predicted curves using least squares and InceptCurves. All curves are approximated by 10 control points. The errors are normalized individually via dividing it by the diagonal length of the bounding box covering all data points of target and predicted curves. Examples of InceptCurves reconstruction of these curves are shown in Fig. 10

Curve Formula	Least squares	InceptCurves
$f_1(x) = \frac{10x}{1+100x^2}$	9.7×10^{-2}	6.09×10^{-2}
$f_2(x) = \frac{100}{e\|10x-5\|} + \frac{(10x-5)^5}{500}$	1.5×10^{-1}	9.9×10^{-2}
$f_3(x) = \sin x + 2e^{-30x^2}$	9.7×10^{-2}	1.9×10^{-2}
$f_4(x) = \sin 2x + 2e^{-36x^2} + 2$	1.2×10^{-1}	4.9×10^{-2}

(in the order of hundreds of seconds) due to their iterative nature, whereas our method produces the curve very fast at run time (in the order of a few seconds) since the forward pass of the neural network only uses basic operations.

Note that InceptCurves can predict the number and position of control points by training on samples from arbitrary curves, which may be parametric and nonparametric curves, noisy, and with different sampling densities. This shows that

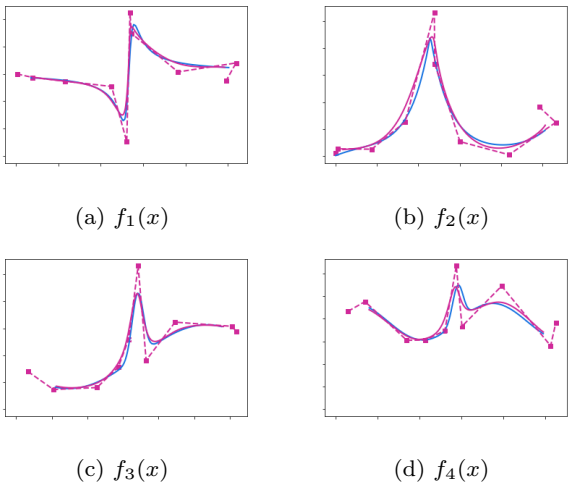


Fig. 10 B-spline open curve reconstruction using InceptCurves on some analytic curves, as listed in Table 2

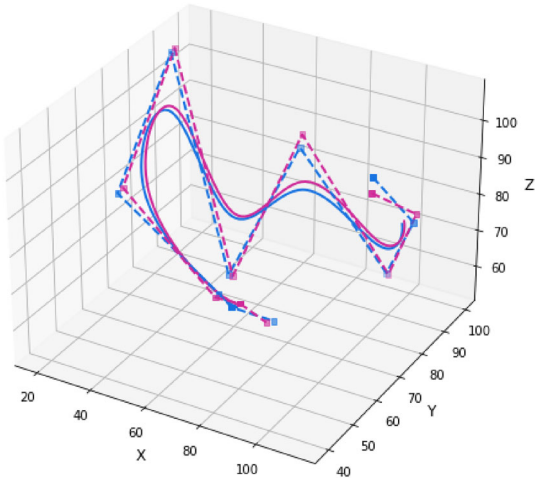


Fig. 12 3D curve reconstruction using InceptCurves. The original curve and control points are depicted in blue. Control points prediction and its corresponding B-spline curve are shown in pink

it can be applied to various tasks without a restriction on these modalities.

A limitation of our network is that, like other NNs [20], it requires the input samples to be of a fixed size. InceptCurves was applied to the reconstruction of planar cubic B-spline curves with a 161×2-sized input. To handle this restriction, any arbitrary curve should be up/down-sampled in order to enter our pipeline. An example of uniform downs-sampling by selecting points at regular intervals along the curve is illustrated in Fig. 11. Feature-preserving down-sampling will lead to nonuniform sample distributions which can also be reconstructed. It will depend on the application which approach leads to better results.

The InceptCurves is easily extendable to handle 3D curves of different degrees and larger or lower numbers of samples. An example of an InceptCurves reconstruction of 3D cubic B-spline curves is depicted in Fig. 12.

5 Conclusion

In this work, we introduced a data-driven approach to find the positions of a coarse set of control points for a uniform B-spline curve or subdivision curve which closely fits the input data. The approach, called InceptCurves, is based on a neural network with inception modules that is able to reconstruct such curves from input sample data with a very few number of control points. The coarseness of the control polygon enables the reconstructed curve to directly enter a design pipeline.

To train the network we generate a large set of synthetic data which is generic and resembles the data observed in a wide range of visual computing applications. By adding flexibility to the output, the inception network is able to learn the number and position of control points concurrently to derive a smooth curve of the given input, thus avoiding redundancy in the reconstructed curve.

This method is fast. Once the network is trained, a reconstruction takes few seconds. It requires no initialization or pre-processing and none of the results shown have been post-processed. Post-processing, as employed in previous work [22] to achieve even higher accuracy is always an option. We compared our method to the classic method based on least squares and showed that it can achieve much lower fitting mean squared errors for fewer control points. Although results have been presented for the reconstruction of cubic 2D B-spline or subdivision curves, InceptCurves can be extended

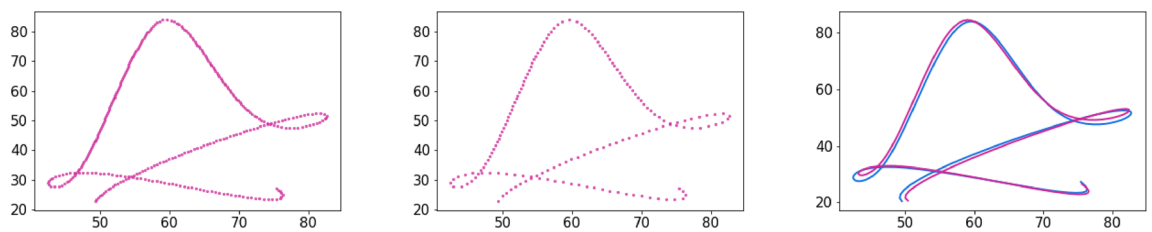


Fig. 11 The left figure illustrates a 2D curve of size 300×2. The middle picture shows the uniform down-sampled version of the original curve. The prediction of InceptCurves is depicted in pink, while the original curve is depicted in blue in the right figure

in a straightforward manner to fit data points by any B-spline or subdivision curves.

Author Contributions Initiating the idea of InceptCurves as well as implementation and evaluation of the method and the composition of the paper has been done by SB. AK has contributed to the generation of dataset. JZ and UA have actively contributed in discussions on developing the method and have helped in revising the paper several times.

Funding Open access funding provided by Graz University of Technology.

Data availability We will provide the dataset upon request.

Declaration

Conflict of interest The authors declare no conflict of interest.

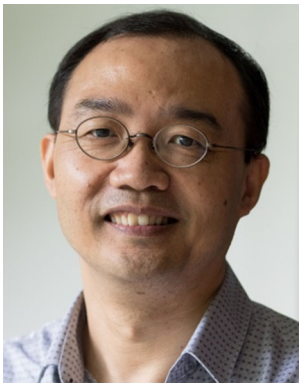
References

1. Varady, T., Martin, R.: Reverse Engineering. Handbook of Computer Aided Geometric Design, pp. 651–681 (2002)
2. Chaikin, G.M.: An algorithm for high-speed curve generation. Comput. Graph. Image Process. **3**(4), 346–349 (1974)
3. Sabin, M.: Analysis and Design of Univariate Subdivision Schemes. Springer, Berlin (2010)
4. Park, H.: An error-bounded approximate method for representing planar curves in B-splines. Comput. Aided Geom. Des. **21**(5), 479–497 (2004)
5. Masci, J., Rodolà, E., Boscaini, D., Bronstein, M.M., Li, H.: Geometric deep learning. SIGGRAPH ASIA 2016 Courses, pp. 1–50 (2016)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
7. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (2017)
8. Wang, H., Zhang, J.: A survey of deep learning-based mesh processing. Commun. Math. Stat. **10**(1), 163–194 (2022)
9. De Boor, C., Rice, J.R.: Least squares cubic spline approximation, II-variable knots. (1968)
10. De Boor, C., Rice, J.R.: Least squares cubic spline approximation I-Fixed knots. (1968)
11. Deng, C., Lin, H.: Progressive and iterative approximation for least squares B-spline curve and surface fitting. Comput. Aided Des. **47**, 32–44 (2014)
12. Ebrahimi, A., Loghmani, G.B.: B-spline curve fitting by diagonal approximation BFGS methods. Iran. J. Sci. Technol. Trans. A Sci. **43**, 947–958 (2019)
13. Hoschek, J., Lasser, D., Schumaker, L.L.: Fundamentals of Computer Aided Geometric Design. A. K. Peters, Ltd. (1993)
14. Piegl, L., Tiller, W.: The NURBS Book. Springer, Berlin (1995)
15. Valenzuela, O., Delgado-Marquez, B., Pasadas, M.: Evolutionary computation for optimal knots allocation in smoothing splines. Appl. Math. Model. **37**(8), 5851–5863 (2013)
16. Gálvez, A., Cobo, A., Puig-Pey, J., Iglesias, A.: Particle.: Swarm optimization for Bézier surface reconstruction. In: Proceedings of 8th International Conference Computational Science-ICCS. Kraków, Poland, June 23–25, 2008, Part II 8, pp. 116–125. Springer, Berlin (2008)
17. Song, B., Wang, Z., Zou, L.: An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. Appl. Soft Comput. **100**, 106960 (2021)
18. Komar, A., Augsdörfer, U.: SwarmCurves: Evolutionary Curve Reconstruction. In International Symposium on Visual Computing, pp. 343–354. Springer Nature, Cham (2023)
19. Gálvez, A., Iglesias, A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. Computer-Aided Des. **43**(12), 1683–1692 (2011)
20. Laube, P., Franz, M.O., Umlauf, G.: Learnt knot placement in B-spline curve approximation using support vector machines. Comput. Aided Geom. Des. **62**, 104–116 (2018)
21. Wen, Z., Luo, J., Kang, H.: The deep neural network solver for B-spline approximation. Comput.-Aided Des. **169**, 103668 (2024)
22. Gao, J., Tang, C., Ganapathi-Subramanian, V., Huang, J., Su, H., Guibas, L.J.: Deepspline: data-driven reconstruction of parametric curves and surfaces. arXiv:1901.03781 (2019)
23. Laube, P., Franz, M.O., Umlauf, G.: Deep learning parametrization for B-spline curve approximation. In: International Conference on 3D Vision (3DV). IEEE (2018)
24. Scholz, F., Jüttler, B.: Parameterization for polynomial curve approximation via residual deep neural networks. Comput. Aided Geom. Des. **85**, 101977 (2021)
25. Tong, Y., Lina, Yu., Li, S., Liu, J., Qin, H., Li, W.: Polynomial fitting algorithm based on neural network. ASP Trans. Pattern Recogn. Intell. Syst. **1**(1), 32–39 (2021)
26. Mandal, S., Uhlmann, V.: A learning-based formulation of parametric curve fitting for bioimage analysis. In: Numerical Mathematics and Advanced Applications ENUMATH 2019: European Conference, Egmond aan Zee, The Netherlands, September 30-October 4. pp. 1031–1038 (2021)
27. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: International Conference on Engineering and Technology (ICET), pp. 1–6 (2017)
28. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
29. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.-R.: Unmasking Clever Hans predictors and assessing what machines really learn. Nat. Commun. **10**(1), 1096 (2019)
30. Park, J.J., et al. Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
31. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
33. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In:

International Conference on Machine Learning, pp. 40–49. PMLR (2018)
34. Sun, C., Liu, M., Ge, S.: B-spline curve fitting of hungry predation optimization on ship line design. Appl. Sci. **12**(19), 9465 (2022)

Saeedeh Barzegar Khalilsaraei is a PhD student at the Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Austria. Her research topics include Machine Learning, Computer Vision, Geometric Deep Learning, and Computer Graphics.

Alexander Komar is a PhD student at the Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Austria. His research topics include 3D Object Retrieval and 3D Surface Representation.



Jianmin Zheng is a Professor in the College of Computing and Data Science at Nanyang Technological University, Singapore. He received his PhD degree from Zhejiang University. His research interests include Computer-Aided Geometric Design, Computer Graphics, Geometric Modeling, CAD, Visualization, Interactive Digital Media, AI for design, and 3D printing. He is an SMA fellow.

Ursula Augsdörfer is an Associate Professor at the Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Austria. She received her PhD degree from De Montfort University of Leicester, United Kingdom. Her main research topics include Computer-Aided Geometric Design, Geometry Processing, and Isogeometric Simulation and Analysis.