



# SwarmCurves: Evolutionary Curve Reconstruction

Alexander Komar<sup>(✉)</sup>  and Ursula Augsdörfer 

Institute of Computer Graphics and Knowledge Visualisation,  
Graz University of Technology, Graz, Austria  
[a.komar@cgv.tugraz.at](mailto:a.komar@cgv.tugraz.at)

**Abstract.** The problem of recovering the shape of a curve given partial information about it is a fundamental problem in many applications in visual computing. Which types of curves are fitted to a given input data depends on the application and varies from piece-wise linear approximation to parametric splines. The choice of approximation method depends on the context of the problem, the nature of the data, and the desired level of accuracy and complexity.

In this paper we introduce SWARMCURVES, a curve reconstruction approach based on particle swarm optimization. For given input data SWARMCURVES offers a range of solutions, from linear polygons to rational B-Splines with various degrees of freedom. The algorithm works on dense, sparse or noisy, 2D or 3D input data. We demonstrate the performance of SWARMCURVES, on a number of examples.

**Keywords:** Curve Reconstruction · Evolutionary Optimization · B-Splines

## 1 Introduction

To recover the shape of a curve from data points is a challenging problem that appears frequently in a wide range of applications such as computer-aided design, virtual reality and computer graphics, data visualization and medical imaging. Depending on the available information about the curve and the application, qualitatively different approaches can be formulated in order to solve this problem.

*Curve fitting* aims to find a curve which best represents a given set of possibly noisy or sparse data points. In most applications of computer graphics and geometric modelling it refers to fitting low degree curves to input data. Curve fitting is also used in data analysis, to identify trends and patterns in data [23]. It is closely related to *curve approximation*, which aims to reduce the amount of data required to represent a curve, thereby reducing storage and processing requirements. It involves fitting a mathematical function to an input curve or data set. The goal of *curve reconstruction* is to derive a compact representation for (possibly partial) data at high accuracy, while ensuring that the resulting

curve is smooth and has no sharp corners or discontinuities. It aims at deriving a smooth and visually appealing curve which is highly relevant in fields such as geometric modelling and computer-aided design/manufacturing (CAD/CAM). In this context B-Splines are the widely preferred approximating functions due to their powerful mathematical properties and their wide support by CAD/CAM systems. Curve reconstruction has become a fundamental tool in reverse engineering, where dense data acquired from physical objects is converted to a digital model of such objects [29].

A combined approach for fitting, approximating and reconstructing arbitrary degree curves from ordered data is to our knowledge still an open problem.

In this paper we introduce SWARMCURVES, a curve reconstruction approach based on particle swarm optimisation, which is general and makes no assumption other than the initial input be a set of ordered data points, 2D or 3D. The output are a set of open or closed curves of arbitrary degree. SWARMCURVES may be applied in reverse engineering to reconstruct curves from very dense data typically obtained from scanner devices. However, SWARMCURVES can also handle sparse and noisy data, which may be collected through other means, for example in context of visual analytics, where curve fitting is used to identify trends. SWARMCURVES, does not require any additional pre- or post-processing and offers various types of fitting solutions: from simple polygons to rational B-Splines with various degrees of freedoms in form of control points. SWARMCURVES performs well regardless of whether the data comes from curves which are polynomial, rational or belong to any other family of functions and provides solutions even for sparse input data. In the reported examples, the mean error from the output curve to the input data points is usually below 0.35% of the bounding box diagonal.

## 2 Related Work

Reconstructing smooth curves from point data has been a research topic for decades and numerous approach have been put forward.

### 2.1 Curve Reconstruction

Curve fitting, approximation and reconstruction are closely related and solve common problems in many fields of research [1, 13, 17, 18].

To reconstruct smooth curves a common representation are NURBS [8] which are given by  $C(t) = \sum_{j=1}^n \frac{N_{j,d}(t)\omega_j P_j}{N_{j,d}(t)\omega_j}$ , where  $N_j$  are the B-Spline functions of degree  $d$  defined on a knot vector. The resulting B-Spline curve is smooth and approximates its  $n$  control points  $P_j$ . Although knots can be distributed non-uniformly over the parametric domain, they are often uniform in a modelling context except for multiplicities where interpolation of the control polygon is required.

Reconstructing B-Splines from input data was introduced by de Boor and Rice [5], first for a fixed knot vector and later also for variable knot vectors [4].

The researchers proposed a least squares approximation method for approximating the control points for the spline curves and their knot vectors.

Park [25] presented an error-bound method for B-Spline curve approximation to a given planar input curve. He focused both on accuracy control and data reduction to derive a B-Spline curve with fewer control points while keeping the distance between the given curve and the B-Spline curve smaller than a specified tolerance. The fitting was achieved by first approximating a polygon to the data and then fitting a B-Spline curve to the polygon.

Park and Lee [24] fitted B-Spline curves to a set of ordered points by selecting points called dominant points to achieve a better curve approximation. They adaptively refined a B-Spline curve by selecting fewer dominant points at flat regions but more at complex regions. For the same number of control points, their approach could generate a B-Spline curve with less deviation than previous approaches.

In order to reconstruct cubic B-Spline curves Ebrahimi and Loghmani [6] used the Broyden-Fletcher-Goldfarb-Shanno algorithm [3] to improve the non-linear least squares optimization problem. As initialization for their optimization method, they use a uniformly spaced knot vector and then solve a linear system of equations for the initial control point positions. They optimised the fit by inserting additional control points until the approximation error does not decrease significantly in one iteration.

Recent approaches employ Neural Networks for curve reconstruction [15, 19, 20, 30, 32]. Different input modalities have been used for neural networks, e.g. Point clouds [15, 30], depth images [20] and 2D images [32].

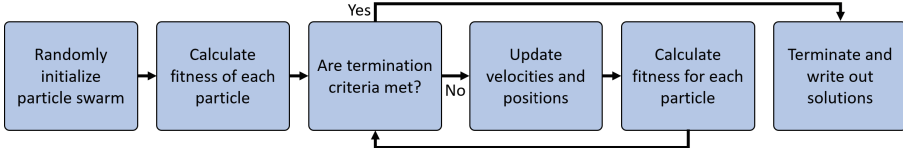
In the approach presented in this paper we assume the knot vector fixed and focus on optimising the number of control points, their positions  $P_j$  and their corresponding weights  $\omega_j$ . While for closed curves the knot vector is set to be uniform, open curves are reconstructed using a clamped knot vector, so that first and last control points are interpolated. We evaluate our algorithm by comparing it to a number of examples typically used to assess the quality of the reconstruction.

## 2.2 Evolutionary Algorithms for Curve and Surface Fitting

Evolutionary optimization has been used in many applications [2, 7, 11, 14, 21, 28, 31, 33], including curve fitting [9, 13, 27]. Various types of evolutionary optimisation methods exist.

Sun, Liu and Ge [28] are fitting B-Spline curves to a sampled pointset of the outline of a ship using an adapted Hungry Predation Optimization Algorithm. Galvez et al. [10] used a Genetic Algorithm for surface and curve reconstruction. They encoded the parameter values in a genome and used the mean squared error as an error measure for a found solution. They extended their approach and used particle swarm optimisation to fit Bézier surfaces to 3D point clouds [9].

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart [16] in 1995. It uses an evolutionary approach to optimization in order to eliminate the need for a derivative of the error function. With PSO a swarm



**Fig. 1.** Particle Swarm Optimization Pipeline

of particles is initially uniformly distributed over the  $n$ -dimensional space of possible solutions. Additionally every particle has a velocity with which it travels through this space. Every member of the swarm knows the best global solution and the best solution reached by its local neighbors. Using this information, every particle updates its position and its velocity. This is repeated until a stopping criterion is met or a limit on the number of iterations is reached. For a comprehensive overview on applications and current theory of PSO we refer the reader to two recent survey papers in this field given by Houssein et al. [12] and Shami et al. [26].

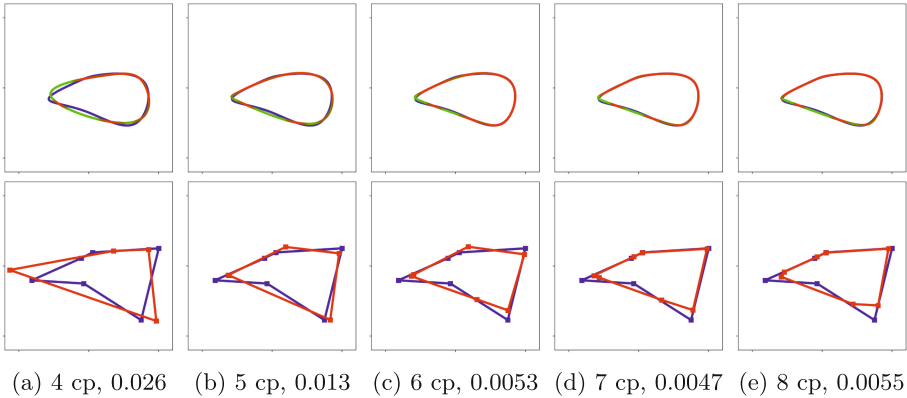
An application of PSO to the problem of curve fitting was proposed by Song et al. [27]. The authors adapted the classical PSO algorithm in order to better escape local minima. With this improved algorithm the researchers found paths through partially obstructed workspaces using Bézier curves.

Iglesias et al. [13] introduced a method for B-Spline curve reconstruction using a multilayer embedded bat algorithm (ME-BAT). This algorithm is modelled after the behaviour and communication of bats and, similar to PSO tries to find a globally optimal solution to a given problem. Contrary to our approach the researchers fixed the number of control points before executing the algorithm and do not state a method on how to find the optimal number of control points.

In this paper we propose SWARMCURVES, a novel approach to fit polygons, B-Splines, or rational B-Splines to an ordered set of points. SWARMCURVES is based on PSO, which is better able to solve floating point problems, compared to genetic algorithms for example. SWARMCURVES is able to reconstruct parametric curves of arbitrary degree and various number of control points. We can handle noisy and sparse data.

### 3 Methodology

We employ the PSO to find the control points of a B-Spline curve which best fits the input data. A single particle of the swarm consists of a list of ordered control point positions in 3D,  $P_j = (x_j, y_j, z_j)$ . Each position is associated with a velocity, which determines the distance and direction this control point will move in the next iteration. A key element of any evolutionary approach is the fitness of a solution. In context of curve reconstruction the fitness is related to the distance between a given set of data points and the reconstructing B-Spline curve  $C$ , computed with the set of control points derived by a single particle. If the data points are sampled from a curve  $T$ , then the distance between the target curve

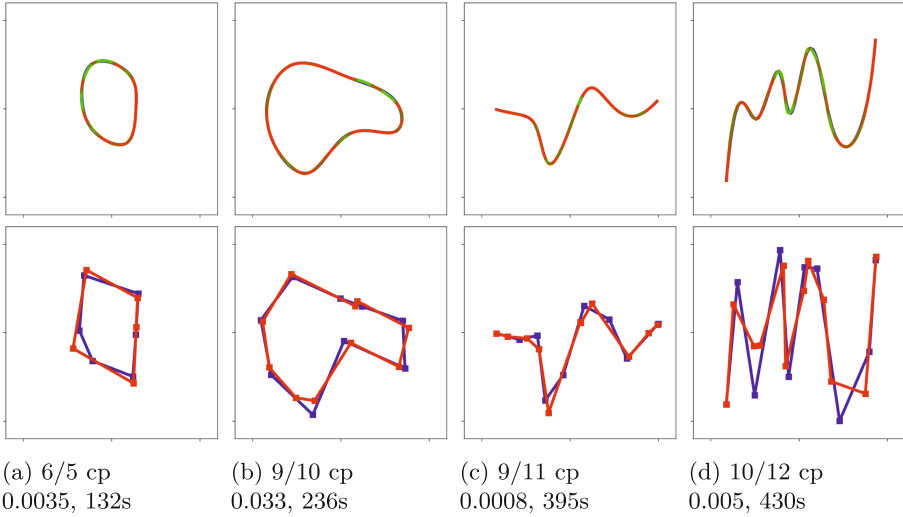


**Fig. 2.** An example of a single output provided by SWARMCURVES. Top: Input data points (blue) are shown together with the reconstructed curve colored in the Euclidean distance, where red indicates a close approximation. Below: The reconstructed control polygon (red) is shown together with the control polygon of the original curve (blue). From left to right the number of control points of the solution of the curve reconstruction is increasing, from 4 to 8 control points. The captions also list the error of the solution. (Color figure online)

$T$  that we want to reconstruct, and the B-Spline curve  $C$ , defined by a particle's set of control points,  $P_j$ , is defined by the error measure  $err(c) = d_{avg}(C, T)$ , where  $d_{avg}()$  is the average Euclidean distance between the sampled curve  $C$  to the sampled target curve  $T$ .

We restrict the reconstruction in our approach to a uniform knot distribution for closed curves and place knot multiplicities at end points of open curves in order to interpolate the first and last control point positions of the reconstructed B-Splines curve.

A visualization of the PSO Pipeline can be seen in Fig. 1. The swarm is initialized randomly with the number of particles specified by the user. Each particle is initialized with a  $3n$ -dimensional random position inside a bounding box of normalized data points  $[-1, 1]$ . Then the local (one particle) and global (over the whole swarm) best position and error are determined. The position of each particle is denoted  $q_{local}$ , the optimal position over the whole swarm as  $q_{global}$ . Then the iteration loop starts. With each iteration the velocity of each particle is updated according to  $v_{i+1} = v_i\psi + (q_{local} - q_i)r_q\phi_q + (q_{global} - q_i)r_g\phi_g$  where  $v_i$  is the current velocity,  $\psi$  is the inertia weight,  $q_{local}$  is the best found solution by the particle,  $q_i$  is the current position of the particle,  $r_q$  is a random number in the interval  $[0, 1]$  and  $\phi_q$  is the weight of influence for the local best position. Further,  $q_{global}$  is the globally best position of the swarm,  $r_g$  is another random number in the interval  $[0, 1]$  and  $\phi_g$  is the weight of influence for the global best position. The position is updated according to  $q_{i+1} = q_i + v_{i+1}$ . Finally, the error of each particle is updated and the local and global best position



**Fig. 3.** Reconstruction of randomly generated closed and open B-Spline curves of degree 3. The top row is the target data points in blue and the reconstructed B-Spline curve colored according to the Euclidean distance. Bottom row is the target control polygon in blue and the output control polygon in red. Each caption lists the number of control points of the target B-Spline curve/the number of control points for the reconstructed B-Spline curve/the mean error/the execution time. (Color figure online)

are re-evaluated. The iteration terminates if either the approximation error lies below a predefined threshold or a predefined limit of iteration is reached.

With the start of the algorithm multiple swarms of particles are launched independently, each swarm with a fixed number of control points. The user can choose either the most accurate reconstruction or choose the reconstruction which provides the desired number of control points. Additionally, SWARM-CURVES can provide a number of different degree B-Splines simultaneously. Which degree of smoothness is required for the reconstruction will depend on the context of the application.

## 4 Results

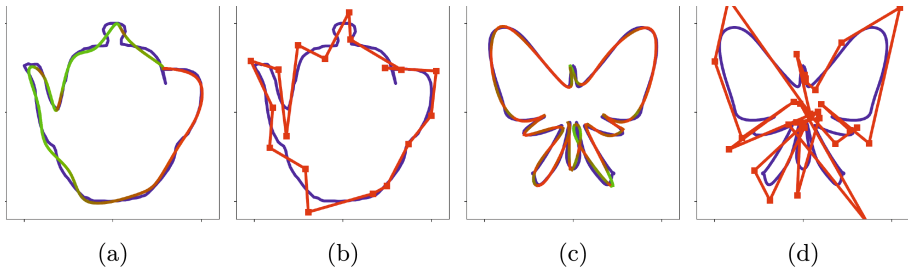
In this section we present the results of the SWARM-CURVES reconstruction algorithm. Although in the previous section the algorithm was described for a setting in 3D, results are shown for 2D for easy visibility. We show results using different benchmarks from related work that demonstrate different abilities of our algorithm. All curves were normalized to fit inside the range of  $[-1, 1]$ . Multiple swarms of particles are launched for each example and each swarm has an iteration limit (2000). Each swarm provides a range of solutions. The final choice is either made by the system or by the user based on preferences. Out

of the range of solutions, the user may be interested in e.g. the sparsest control polygon or the closest approximation.

#### 4.1 Reconstructing B-Splines from Point Data Sampled from Arbitrary B-Spline Curves

Data sampled of closed and open B-Splines and rational B-Splines and non-rational curves may be reconstructed using SWARMCURVES. While for closed curves the knot vector is set to be uniform, open curves are reconstructed using a clamped knot vector, so that the first and last control points are interpolated. All optimizations were run for 2000 iterations.

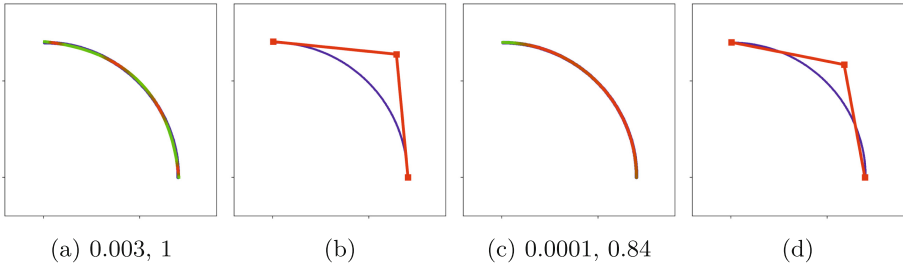
An example output of five SWARMCURVES can be seen in Fig. 2. The result shows the reconstruction of a B-Spline from data points sampled from a cubic B-Spline originally derived using six control points. The best approximation is given by SWARMCURVES solution shown in Fig. 2(d).



**Fig. 4.** Results from reconstructing a benchmark curve (blue) [22] (a) and [13] (c), with its corresponding control polygon (red) in (b) and (d). (Color figure online)

As is clear from the example shown in Fig. 2, the closest approximating SWARMCURVES do not necessarily have the same number of control points as the target curve. This is also evident in Fig. 3, where we show further examples of SWARMCURVES reconstructions of arbitrary closed and open B-Spline curves (top) and their corresponding control polygons. Each reconstruction has been selected for best accuracy from a selection of seven solutions using different number of control points. Calculating a large number of SWARMCURVES does not slow down the reconstructions significantly, since all solutions are derived in parallel.

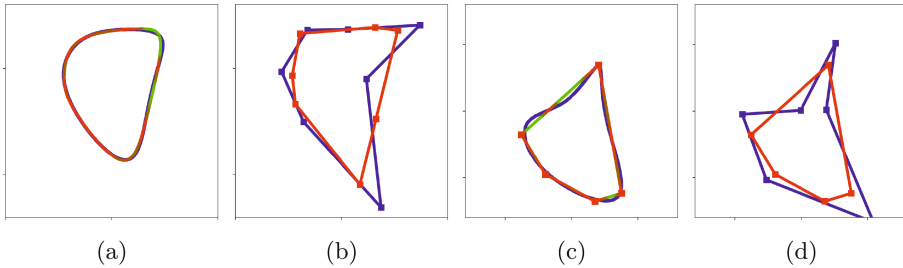
Even more complex examples are shown in Fig. 4. Figure 4(a) shows the SWARMCURVES fitting of cubic B-Spline to a data point example taken from a benchmark dataset proposed by Ohrhallinger et al. [22]. Although the overall shape is approximated well, small details like the handle on the top lacks almost entirely. This means that it is difficult for the algorithm to reconstruct rather small features. Figure 4(b) shows a SWARMCURVES fitting to the data sampled



**Fig. 5.** Approximated circle segment. (a) and (c) show the curve segment (target curve, blue) together with the SWARMCURVES coloured wrt to Euclidean distance. (c) and (d) are the target curve (blue) and control polygon of a fitted rational B-Spline. Captions state the error of the reconstruction and the weight of the center vertex. (Color figure online)

from the closed NURBS curve “Butterfly” presented as a benchmark by Iglesias et al. [13]. While the butterfly has been reconstructed closely, the control polygon provided by SWARMCURVES is not useful for further manipulation of its shape by a designer. Both examples in Fig. 4 clearly show the limitations of SWARMCURVES.

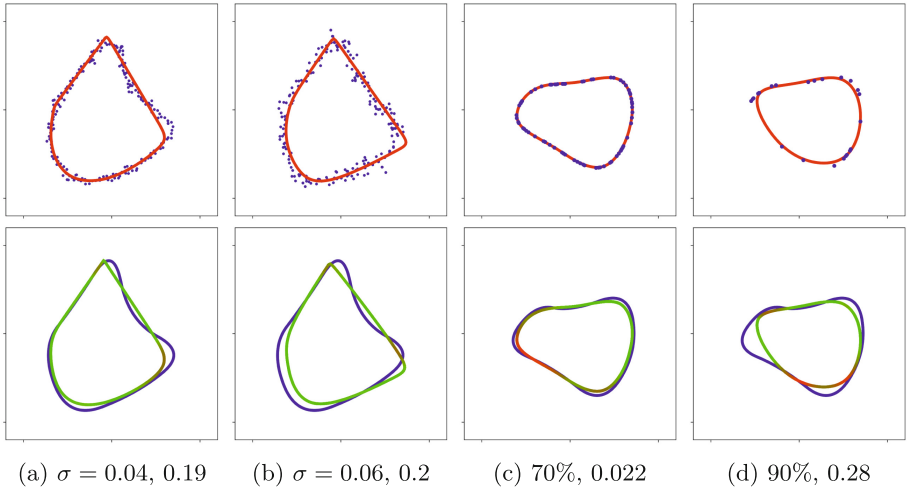
SWARMCURVES can also reconstruct conic sections as can be seen in Fig. 5, where a curve segment has been approximated with a rational B-Spline. The weights,  $\omega_j$ , of the rational B-Spline are optimised to fit a circle segment.



**Fig. 6.** (a) A degree five B-Spline curve in blue is reconstructed by a cubic B-Spline curve in red. (b) The control polygon of the target curve (blue) is shown together with the control polygon of the reconstruction. (c) A cubic B-Spline curve in blue is approximated by a polygon in red. (d) The polygon approximation is compared to the control polygon of the target curve. (Color figure online)

SWARMCURVES also offer the possibility to choose the degree of the reconstruct curve from a set of solutions. Figure 6(a) shows the reconstruction of a cubic B-Spline from points sampled from a degree 5 B-Spline. Figure 6(b) shows





**Fig. 7.** Examples show the ability of our algorithm to fit curves (red) to noisy (a),(b) and sparse data (c),(d). Standard deviation ( $\sigma$ ) and reduction (in percentage) are stated together with the mean error to the original curve (blue) without noise from which the samples are taken. (Color figure online)

the result of fitting a polygon (linear B-Spline) to data sampled from a cubic B-Spline.

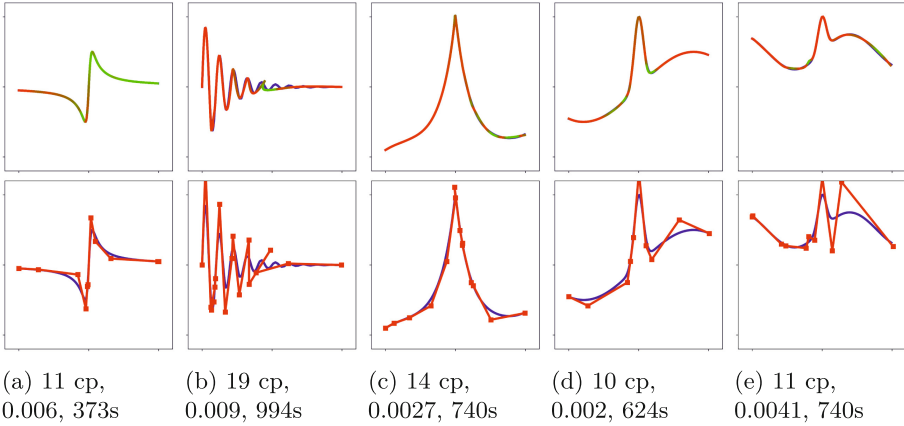
## 4.2 Reconstructing B-Splines from Noisy or Sparse Data Points Sampled from B-Splines

The robustness against noisy and sparse data is demonstrated in the Fig. 7. The noisy data was generated by sampling from a B-Spline curve and adding a Gaussian noise offset to the curve data in the normal direction. The Gaussian distribution had a mean  $\mu = 0$  and standard deviation  $\sigma \in \{0.04, 0.06\}$ . The cubic B-Spline curve from which the noisy data was created was reconstructed reliably, although more control points were required for a good approximation.

The sparse data was created by randomly removing points from a densely sampled curve until the desired reduction percentage is reached. As can be seen in Fig. 7(c)-(d), the original cubic B-Spline curve was reconstructed even for sparse data samples which was non-uniformly distributed along the original curve.

## 4.3 Reconstructing B-Splines from Point Data Sampled from Non-parametric Curves

Sun, Liu and Ge [28] proposed a set of analytic curves for testing their B-Spline fitting algorithm. The formulas can be seen in Eqs. 1 through 5, all curves were normalized in the range of  $[-1, 1]$ . The results of fitting B-Splines to those curves using SWARMCURVES can be seen in Fig. 8. The error distribution leads to the



**Fig. 8.** The fitted curves to the benchmark curves from [28]. Top: the fitted curves colored according to the Euclidean distance and the input point cloud in blue. Bottom the fitted control polygon (red) and the input point cloud (blue). The captions list the number of control points used in the fitting, the mean error of the result and the execution time. All fitted curves are B-Splines. (Color figure online)

conclusion that the algorithm has more difficulty fitting a curve to segments with higher curvature.

$$f_1(2x) = \frac{10x}{(1 + 100x^2)} \tag{1}$$

$$f_2(2(x + 1)\pi) = e^{-0.5x} \sin(5x) \tag{2}$$

$$f_3\left(\frac{x + 1}{2}\right) = \frac{100}{55e^{|10x-5|}} + \frac{(10x - 5)^5}{27500} - 0.8 \tag{3}$$

$$f_4(2x) = 0.5 \sin(x) + e^{-30x^2} \tag{4}$$

$$f_5(2x) = 0.5 \sin(2x) + e^{-36x^2} + 1 \tag{5}$$

## 5 Conclusion and Future Work

We presented SWARMCURVES, an algorithm based on Particle Swarm Optimization that is able to reconstruct a range of B-Spline curves, from linear polygons to rational B-Splines with various degrees of freedom, from ordered point sets. The user may select a B-Spline curve from a list of different types and numbers of control points and an error value of the reconstruction. We evaluated the reconstruction capabilities by comparing the results to common benchmarks. The algorithm works on dense, sparse or noisy, 2D or 3D input data. We demonstrate the performance of SWARMCURVES, on a number of examples.

**Table 1.** A table showing the number of control points (cp) and error between the original curve and the B-Spline reconstruction. Note, that for Equation (4) and (5) SWARMCURVES is able to derive a more accurate reconstruction despite using considerable fewer control points as used by Sun et al. [28].

	Eq	(1)	(2)	(3)	(4)	(5)
Sun et al	cp	57	81	44	50	41
	err	$6.32 \cdot 10^{-7}$	$2.31 \cdot 10^{-6}$	$7.5 \cdot 10^{-5}$	$1,38 \cdot 10^{-4}$	$4.18 \cdot 10^{-4}$
SWARMCURVES	cp	11	19	14	10	11
	err	$5.2 \cdot 10^{-3}$	$4.5 \cdot 10^{-4}$	$9.7 \cdot 10^{-2}$	$1.98 \cdot 10^{-5}$	$1.3 \cdot 10^{-4}$

For future work SWARMCURVES may be extended to support the optimization of non-uniform knot vectors during the reconstruction. This means that the algorithm would be able to reconstruct even more features and sharper features. Further, to reduce the time of reconstruction future version of SWARMCURVES will be implement partially on the GPU.

## References

1. Chen, L., Ghosh, S.K.: Uncertainty quantification and estimation of closed curves based on noisy data. *Comput. Stat.* **36**, 2161–2176 (2021)
2. Chiong, R., Weise, T., Michalewicz, Z.: *Variants of Evolutionary Algorithms for Real-World Applications*. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-23424-8>
3. Cipolla, S., Di Fiore, C., Tudisco, F., Zellini, P.: Adaptive matrix algebras in unconstrained minimization. *Linear Algebra Appl.* **471**, 544–568 (2015)
4. De Boor, C., Rice, J.R.: Least squares cubic spline approximation i-fixed knots. *International Mathematical and Statistical Libraries* (1968)
5. De Boor, C., Rice, J.R.: Least squares cubic spline approximation, ii-variable knots. *International Mathematical and Statistical Libraries* (1968)
6. 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)
7. Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H.: Marine predators algorithm: a nature-inspired metaheuristic. *Expert Syst. Appl.* **152**, 113377 (2021)
8. Farin, G.: *Curves and Surfaces for CAGD*. Morgan Kaufmann, San Fransisco (2002)
9. Gálvez, A., Cobo, A., Puig-Pey, J., Iglesias, A.: Particle swarm optimization for Bézier surface reconstruction. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) *ICCS 2008*. LNCS, vol. 5102, pp. 116–125. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-69387-1\\_13](https://doi.org/10.1007/978-3-540-69387-1_13)
10. Gálvez, A., Iglesias, A., Cobo, A., Puig-Pey, J., Espinola, J.: Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation. In: Gervasi, O., Gavrilova, M.L. (eds.) *ICCSA 2007*. LNCS, vol. 4706, pp. 680–693. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74477-1\\_62](https://doi.org/10.1007/978-3-540-74477-1_62)
11. Gandomi, A.H., Alavi, A.H.: Krill herd: a new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **17**(12), 4831–4845 (2012)

12. Houssein, E.H., Gad, A.G., Hussain, K., Suganthan, P.N.: Major advances in particle swarm optimization: theory, analysis, and application. *Swarm Evol. Comput.* **63**, 100868 (2021)
13. Iglesias, A., Gálvez, A., Collantes, M.: Multilayer embedded bat algorithm for B-spline curve reconstruction. *Integr. Comput.-Aided Eng.* **24**(4), 385–399 (2017)
14. Kaveh, A., Dadras Eslamlou, A.: Water strider algorithm: a new metaheuristic and applications. *Structures* **25**, 520–541 (2020)
15. Kavita, K., Navin, R., Madan, A.S.: Piecewise feature extraction and artificial neural networks: an approach towards curve reconstruction. *Indian J. Sci. Technol.* **9**(28), 121–134 (2016)
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948. IEEE (1995)
17. Lee, I.K.: Curve reconstruction from unorganized points. *Comput. Aided Geom. Des.* **17**(2), 161–177 (2000)
18. Lin, H., Chen, W., Wang, G.: Curve reconstruction based on an interval B-spline curve. *Vis. Comput.* **21**, 418–427 (2005)
19. Liu, Y., D'Aronco, S., Schindler, K., Wegner, J.D.: PC2WF: 3D wireframe reconstruction from raw point clouds. *arXiv preprint [arXiv:2103.02766](https://arxiv.org/abs/2103.02766)* (2021)
20. Matveev, A., et al.: DEF: Deep estimation of sharp geometric features in 3D shapes. *ACM Trans. Graph.* **41**(4) (2022)
21. Mirjalili, S., Lewis, A.: The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016)
22. Ohrhallinger, S., Peethambaran, J., Parakkat, A.D., Dey, T.K., Muthuganapathy, R.: 2D points curve reconstruction survey and benchmark. In: *Computer Graphics Forum*, vol. 40, pp. 611–632. Wiley Online Library (2021)
23. Ostertagová, E.: Modelling using polynomial regression. *Procedia Eng.* **48**, 500–506 (2012). *Modelling of Mechanical and Mechatronics Systems*
24. Park, H., Lee, J.: B-spline curve fitting based on adaptive curve refinement using dominant points. *Comput. Aided Des.* **39**, 439–451 (2007)
25. Park, H.: An error-bounded approximate method for representing planar curves in B-splines. *Comput. Aided Geom. Des.* **21**, 479–497 (2004)
26. Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A., Mirjalili, S.: Particle swarm optimization: a comprehensive survey. *IEEE Access* **10**, 10031–10061 (2022)
27. 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)
28. Sun, C., Liu, M., Ge, S.: B-spline curve fitting of hungry predation optimization on ship line design. *Appl. Sci.* **12**(19), 9465 (2022)
29. Varady, T., Martin, R., Cox, J.: Reverse engineering of geometric models - an introduction. *Comput. Aided Des.* **29**(4), 255–268 (1997)
30. Wang, X., et al.: PIE-NET: parametric inference of point cloud edges. In: *Advances in Neural Information Processing Systems*, vol. 33, pp. 20167–20178. Curran Associates, Inc. (2020)
31. Yazdani, M., Jolai, F.: Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm. *J. Comput. Des. Eng.* **3**(1), 24–36 (2015)
32. Ye, Y., Yi, R., Gao, Z., Zhu, C., Cai, Z., Xu, K.: NEF: neural edge fields for 3D parametric curve reconstruction from multi-view images (2023)
33. Yilmaz, S., Sen, S.: Electric fish optimization: a new heuristic algorithm inspired by electrolocation. *Neural Comput. Appl.* **32**(15), 11543–11578 (2020)