Genetic Subdivision Curve and Surface Reconstruction

Alexander Komar
a.komar@cgv.tugraz.at

Marco Riegler
marco.riegler@student.tugraz.at

Reinhold Preiner
r.preiner@cgv.tugraz.at

Ursula Augsdörfer
u.augsdorfer@cgv.tugraz.at

Institute of Computer Graphics and Knowledge Visualisation, Graz University of Technology, Austria

ABSTRACT

In this paper we employ a new genetic algorithm approach for CAD shape reconstruction, where a mathematical shape representation is reconstructed from point data. We reconstruct planar subdivision curves and 3D subdivision meshes from ordered input point data by fitting the corresponding subdivision control polygon or control mesh respectively, from which the smooth subdivision limit surfaces can be derived. For the reconstruction of curves the system estimates the number and position of control points required to approximate the curve closely. To reconstruct subdivision surfaces from points, the system determines a sequence of CAD operations which is subject to mutation in the course of a genetic optimization. We discuss implementation details of the proposed genetic algorithms and demonstrate our approach on a number of example data.

Keywords

Genetic Algorithms, Curve Reconstruction, Surface Reconstruction, Shape Code, Procedural Modelling

1 INTRODUCTION

To recover the shape of a curve or surface from data points is a challenging problem that appears frequently in a wide range of applications such as Computer-Aided Design (CAD), virtual reality and computer graphics, data visualization and medical imaging. Shape reconstruction in CAD refers to the process of generating a smooth and accurate representation of a curve or surface from a set of data points. The goal is to transform discrete data into a smooth and continuous curve representation suitable for CAD modeling. Shape reconstruction has become a fundamental tool in reverse engineering, where dense data acquired from physical objects is converted to a digital representation [6].

In this context B-splines [2] are the preferred approximating functions due to their powerful mathematical properties and their wide support by CAD/CAM systems. The B-spline curve or surface approximates the control polygon given by linearly connected control points P_i. They are defined over a uniform or non-uniform knot sequence. The control points and the knots are the design freedoms to satisfy approximation requirements. If the degree is fixed and the knot vector is fixed and uniform an ordered set of control points fully define the smooth shape.

Smooth curves may be derived from control polygons either through the analytic expression of a B-spline curve or recursively through subdivision [12]. For arbitrary control meshes smooth subdivision surfaces are derived via subdivision refinement rules [1], where regular regions of the surface will correspond to B-spline surfaces.

The reconstruction of subdivision control meshes is a well researched problem [17, 23, 24, 10]. Traditional methods often use rather complex, multivariate optimization processes to reverse the subdivision algorithm. However, the main limitation of this class of algorithms is that the resulting control meshes usually do not exhibit a topology and distribution of extraordinary vertices that would emerge if the model was manually designed using a CAD system. This makes it more difficult to continue the design on the reconstructed shape as it enters a design pipeline.

In this paper, we approach the subdivision shape reconstruction problem by formulating it as a genetic optimization process. Genetic algorithms are a popular tool for a variety of search and optimization problems as they offer the possibility to search through multiple solutions for a given problem in parallel. They are based on various natural principles found in the real world, such as evolution, natural selection and reproduction. In this paper we introduce a genetic approach for B-spline or subdivision curve reconstruction and another genetic approach to reconstruct subdivision surfaces.

In our work we use a genetic optimisation approach to not only reconstruct the mathematical representation of the shape, but also a set of CAD operations that define a modelling process to construct said representation. We discuss the algorithm for 2D reconstruction in Section 3, the method for 3D reconstruction in Section 4. We present our results in Section 5 and propose future work in Section 6.

2 RELATED WORK

Our work relates to two major classes of methods: Genetic Algorithms and Evolutionary Optimization.

2.1 Subdivision Curves and Surfaces

The principle idea of subdivision is to define a smooth curve or surface as the limit of an iterative subdivision process in which an initial coarse control polygon or mesh is repeatedly refined by introducing new vertices in each subdivision step. Careful selection of the refinement rules ensures that the initial coarse control polygon or mesh converges to a smooth limit curve or surface, referred to as the subdivision limit curve or surface. That is, a subdivision curve or surface is fully defined by its control cage and a subdivision algorithm. The concept of subdivision curves was introduced by Chaikin [12]. Soon after, this approach was extended to surfaces. The concept of subdivision surfaces was introduced by Catmull and Clark [1] and Doo and Sabin [5] independently in 1978. Catmull-Clark subdivision is most commonly found in a wide range of CAD systems and is a standard representation in the entertainment industry. It is a generalization of uniform bi-cubic B-splines to arbitrary shapes and is based on quadrilateral meshes, like the Doo-Sabin algorithm. Generalizations to other domains were proposed by Loop [18] and Kobbelt [14] amongst others. Numerous different subdivision algorithms have emerged [19] based on a range of different types of control polyhedrons.

In this paper we propose a method able to automate the generation of control polygons and meshes for subdivision algorithms based on quadrilateral meshes which resemble manually created subdivision models. Throughout the paper we derive the corresponding limit curves using Chaikin [12] or uniform B-splines, and derive subdivision surfaces using the Catmull-Clark [1] subdivision.

2.2 Genetic Algorithms

There are many different types of evolutionary optimization, showing the potential and diversity in this area of optimization algorithms. In order to choose the best suited algorithm one has to look at different properties of the algorithms, for example their suitability for integer optimization or how they handle local optima. The fitness function, defined as a measure of the feasibility of a found solution, needs to be carefully designed as it guides the optimization to the global minimum.

Genetic Algorithms belong to the group of evolutionary algorithms, but differ in the reproduction process and gene encoding, e.g. genetic algorithms rely more on recombination than mutation for producing offspring, while evolutionary programming in contrast strongly relies on mutation. Evolutionary strategies use real-valued encoding for their genes [31].

Genetic algorithms were first introduced in 1970 by John Holland. They are employed to search and optimization problems where a large amount of variables are to be considered. They are based on evolution and natural genetics [27], modelling nature's principle of natural selection or "survival of the fittest". The measure of fitness varies with each application and is an important aspect in any evolutionary algorithm. While the weak are more prone to go extinct, the fitter individuals will be able to pass on their genetic information via reproduction [27]. As in nature, the offspring in genetic algorithms are also susceptible to mutation, which means that small parts of the inherited genes differ from their parents due to copying errors for example. We refer the reader to a comprehensive review on genetic algorithms and evolutionary optimization by Katoch et al. [13].

We chose genetic algorithms for our approach, since our goal was to not only be able to reconstruct the mathematical representation of the shape, but also define a set of CAD modelling operations which, applied to a template mesh, will yield the smooth shape which fits the data points.

2.3 Evolutionary Algorithms in Shape Reconstruction

Evolutionary approaches have been applied to curve and surface reconstruction using various types of algorithms.

Takeuchi et al. [28] proposed an algorithm able to reconstruct B-Spline surfaces from dense triangle meshes using Quadric Error Measures and elaborate ways of splitting triangle meshes into quad meshes. Xiyu et.al. [32] introduced a combination of genetic algorithms with Neural Networks to reconstruct 3D shapes. They used an encoding based on the principle genes, cells and organs. Combination operations were introduced to combine genes to cells and cells to organs or jelly. Galvez et al. [9] proposed genetic algorithms to reconstruct B-spline surfaces by first obtaining a parameterization for the surface and then determine the knot vectors and then calculating the control point positions according to the least squares approach. Galvez et al. improved their method to be able to execute in just one step [8]. They changed the optimization algorithm

from a genetic approach to a particle swarm approach. They combined their previous 2 algorithms into one and solved the surface fitting problem by means of Least Squares. Sabsch et.al. [22] proposed an implementation of NSGA-II [3] to approximate different datasets with open B-spline curves of degree 3 with a fixed, clamped and uniform knot vector. The researchers optimized the number of control point positions and their position. Robustness to noise was shown, but not to sparsity or intersections. Wang et al. [30] used the Loop subdivision scheme in order to reconstruct cavities of the human body from volumetric data, gathered from different imaging modalities (MRI and CT). Rahamathulla and Misro [21] used genetic algorithms to aid in reconstruction of a craniofacial fracture. They propose an algorithm adjoining two curves together with a degree 5 B-spline curve with different continuity constraints. The number of control points was fixed. The input curves were extracted from a scan. Moulaeifard et al. [20] used Particle Swarm Optimization in order to reconstruct a geological model, which is watertight and controllable by a control mesh. They targeted the specific use case of geological simulations, therefor a low number of control vertices is desirable. Komar and Augsdörfer [15] used a Particle Swarm Optimization to approximate data using B-splines and rational B-splines. Similar to the approach presented here, they optimised control point positions to fit a B-spline curve to input data using a evolutionary approach different from the one explored in this paper.

In the following two sections we introduce a genetic algorithm which outputs a 2D vector of control point positions that define a clamped, uniform B-spline curve optimised to represent the input data. We also propose a genetic algorithm for surface reconstruction that takes a point cloud as an input and outputs a Catmull-Clark subdivision control mesh. In both cases, the genetic algorithms output a sequence of CAD operations to an initial template control polygon or mesh in order to create a CAD model of the control polygon or control mesh that approximates the input.

3 2D CURVE RECONSTRUCTION

To reconstruct subdivision curves the algorithm receives as input a two-dimensional array, containing x and y values of points along a planar curve. Each individual member of the population describes a control polygon. The control polygon is an ordered array of 2D control points, which defines a B-spline. The proposed algorithm also provides a list of operations, the chromosomes, that describe which operations were applied to an initial template polygon to arrive at the control polygon of the curve which optimally approximates the input data. In case of curves, the operations are limited to adding or deleting control points or translating existing control points.

The general structure of the genetic algorithm is based upon the algorithm proposed by Holland [11]. A genetic algorithm maintains a list of individuals, called *population*. The characteristics of each member of the population is defined by an individual *chromosome*, that corresponds to a solution to the given problem. Each individual is then ranked based on an objective value, its *fitness*, describing how well the proposed solution performs. In each generation, pairs of individuals are selected according to their fitness to produce offspring, whose properties are defined by a combination of its parents chromosomes, for the next round of evolution until a predefined termination criterion is met [29].

In our work each individual is represented as a sequence of CAD operations, which applied to the starting control polygon in order, gives the solution proposed by the individual. The initial population in genetic algorithms hugely influences its performance, that is a well conditioned starting population comes with a higher chance of finding a satisfying solution. One way to influence the quality of the first generation of individuals is the population size. In general, using too few individuals leads to worse solutions, while having a large population leads to higher computation time [4]. In our work we employ a mixed strategy: The algorithm is given a large initial population, thus increasing the initial diversity. During selection the population is then reduced to only 1/10th of the initial population, encouraging more diversity in the starting solutions. As a result, the first evolution round takes more time. To create the population, the algorithm creates 1000 individuals, all containing four to eight random CAD operations as genes in their chromosome. Each individual therefore represents a polygon which evolves to a control polygon of a uniform B-spline curve approximating the input data.

Each individual created is initialized with four control points. The first and last control point are placed on the first and last points of the input curve, as the curve generated by this algorithm is clamped. The two remaining control points then consist of the swapped x and y coordinates of the endpoints. The bounding box of the input curve is used to restrict the area in which control points can be moved or added. The quality of each individual is assessed by its fitness function.

The chromosome of each individual is a list of genes, which are the set of CAD operations applied to an initial template polygon. These operations can either be the addition of a new point to the current polygon or the translation of a point. If a new point is introduced, the index of the new point is added in the control point array together with its position. If the point is translated, its position is updated. The first and last control points are barred from being translated. Since they lie on the end points of the input point set.

The fitness function determines how well a candidate solution performs and is an important part in the selection mechanism. In the proposed algorithm this function takes an individual as input, derives the B-spline or subdivision curve using its control point positions and calculates the Euclidean distance between the samples lying on the reconstructed curve and the samples on the target curve, calculated index wise. We sample the curve to have the same number of points as the target curve, therefore the distance calculation can be carried out index-wise. The fitness value is then the sum over all euclidean distances. After calculating the fitness value for each member of the population, the algorithm can begin with the reproduction of the individuals for the next generation.

We use the roulette wheel selection to select parents for reproduction [13]. Each individual is given a weight based on its fitness value. After two parents have been selected, the algorithm computes a crossover which produces offspring by combining their genes. In our work single-point crossover [26] is used to combine two individuals to their offspring. Different length parents can also be combined, by choosing the crossover point according to the shorter length.

In the mutation operation, the algorithm loops over the chromosome and, for each gene, draws a random mutation probability value. After comparing different mutation probability values, a constant probability of 5% has been shown to perform well. If a gene is chosen to be mutated, the algorithm chooses with a probability of 1% to delete the current gene. Otherwise, the gene is mutated by replacing it with a randomly generated operation. After looping over all genes of the individual, another check is performed if a new operation should be appended to the list. This is also done, if a chromosome is left with less than three genes after too many deletions, as this would result in errors during a crossover operation.

The last step of the mutation function is to check whether all operations in the mutated individual are still valid. The validity might not be given if an operation that adds a control point is removed, as a move operation might access the control point later in the chromosome. The algorithm iterates through the chromosome, and checks whether each operation can be executed. If not, the operation is removed from the chromosome. Lastly, it also checks the length of the chromosome again, to ensure the required minimum length is still given, otherwise it would add as many random operations as needed to make sure crossover can be executed.

In this work the *steady-state* technique was used [25]. After the population is sorted according to the fitness value of each individual, the top 20% are moved into the next generation. The remaining 80% open spots are filled with offspring, created using the reproduction steps described above.

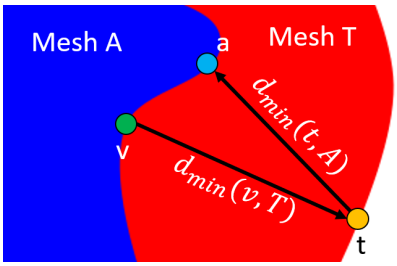


Figure 1: Illustration of the distance function from a single vertex a to the target mesh T. $d_{min}(x, M)$ represents the minimum distance from a vertex x to any vertex on a mesh M.

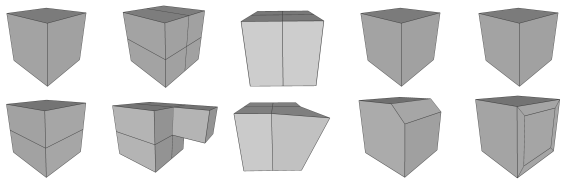
The steps above constitute the evolution of one generation. This process is repeated for as many generations as desired, or until a certain termination criterion is met. One typical termination criteria used is that one candidate solution satisfies a specified minimum threshold of the fitness value [16], which we employed in the optimisation presented in this paper.

4 3D SHAPE RECONSTRUCTION

Surface reconstruction closely follows the approach employed in genetic curve reconstruction. However, the design of 3D shapes on a CAD system involves numerous operations. We demonstrate our approach considering seven CAD operations defined in Blender [7], five of which are shown in Figure 2. However, our approach may be extended to more types of operations. The algorithm then derives control meshes by combining a set of modelling operations in a genetic approach. The set of CAD modelling operations, when applied in sequence to an initial template mesh, yield a subdivision control mesh which defines a smooth surface, the reconstruction solution. The resulting meshes, see Figures 5 and 6, exhibit features which are typically found subdivision surface which has been designed on a CAD system.

A good fitness function defines the quality criteria of the surface reconstruction. We employ the mixed mean distance between the limit surface corresponding to the found control mesh and the input sample point which belong to a target surface.

The first step in the evaluation of the fitness is the calculation of the mixed mean distance between the meshes, which is illustrated in Figure 1. For every vertex v on the mesh computed by the algorithm (Mesh A) the following is calculated. First, the closest vertex t by Euclidean distance on the target mesh (Mesh T) is found. Then, starting from the vertex t the closest vertex a by Euclidean distance on mesh A is found. The relevant part for the mixed mean distance is the Euclidean distance d_{min} from vertex v to vertex t. Finally, if vertex



(a) [L] (b) [*E*] (c) [V] (d) [B] (e) [I]

Figure 2: Five of the seven operations used in our method. (a) Loop insertion, which introduces a new edge in the middle of an edge which is continued through the model, (b) face extrusion along the face normal, (c) vertex translation in a random direction, (d) bevelling of an edge and (e) face inset.

v and vertex a are not the same, the distance is set to the bounding box diagonal. The mean of this distance is calculated over all vertices of mesh A to mesh T and vice versa. The mixed mean distance is then the maximum of the two and the fitness is the inverse of the mixed mean distance.

The quality of the fitness function is essential. A simple mean distance measure proved insufficient to capture the error between meshes well and did not converge. One of the problems was overlapping regions in one of the shapes, which caused the simple mean distance to be very small. Various other distance measures could be used here, but need to be evaluated carefully. We observed that a Hausdorff distance, for example, is not expressive enough for this kind of optimization. It only captures the maximum distance and therefore, does not consider all parts of the mesh. This leads to small adjustments being discarded.

Our genome is a design code, comprised of a set of modelling operations applied to an initial template mesh. This template mesh may be any quadrilateral mesh and can be chosen by the user. We demonstrate our approach using a unit cube as an initial template mesh.

Evolving the sequence of modelling operations rather than the explicit shape representation itself leads to a procedural genome that reflects closely the variety of results produced by a design process using these modelling operations. To this end, we use seven types of operations widely used in CAD modelling. Five of these operations are shown in Figure 2:

Op: V An arbitrary vertex is translated. Parameters are the vertex and the vector of translation. The vector of translation is defined by three random samples chosen in the interval $[-bbox_d, bbox_d]$, where $bbox_d$ is the length of the bounding box diagonal of the mesh.

Op: E A face is extruded along its normal. Parameters are the face and the distance of extrusion. The maximum distance of extrusion is limited by the length of the bounding box diagonal.

Op: L An edge loop is inserted. Here, the parameter is a single edge. The edge loop operation is applied to the middle of an arbitrary edge but results in a insertion of a number of edges, each splitting along the middle edge of all edge in a mesh which are opposite to each other (Figure 2(a)). It is the only operation which affects the whole mesh. When the edge loop meets a non-quadrilateral face it terminates, creating a T vertex, that is added to the non-quadrilateral face.

Op: B Bevelling creates an angled face between two adjacent faces. Parameters are the edge between two faces and the length to be cut off. Applying the bevel operation on an edge with an extraordinary vertex, an additional extraordinary vertex with the same valence is created.

Op: I Face inset creates another face inside an existing face and connects the new vertices to the vertices of the outer face. Parameters are the face and length of inset.

Op: F Face translate displaces a face along its normal. No new vertices are added. Parameters are the face and the offset of the movement.

Op: S Scale operation scales the whole mesh by a factor. Parameter is the scaling factor.

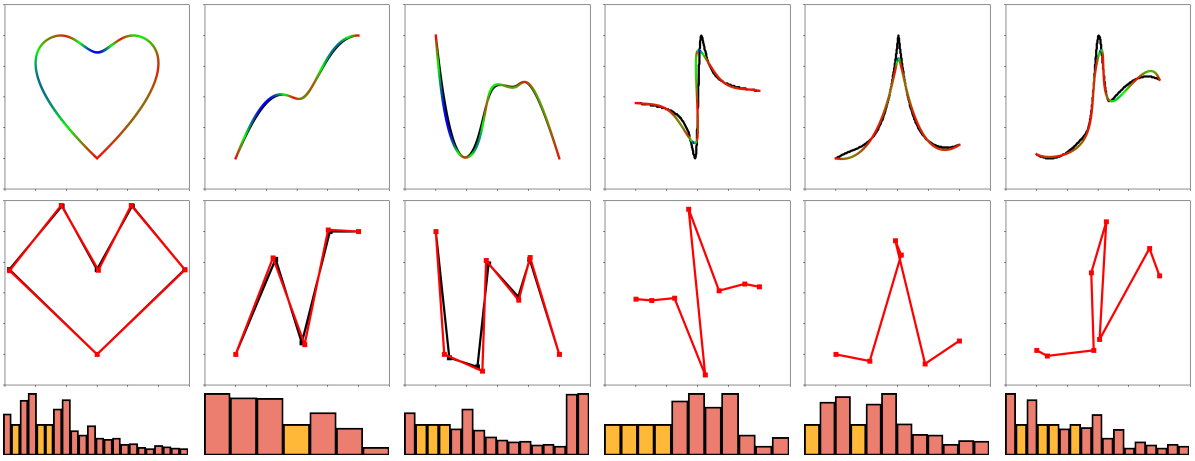
The genome is initialized with one or two random modelling operations which are sequentially applied to arbitrary parts of a mesh. Each operation builds on its predecessors and can change geometry newly introduced by previous operations.

5 RESULTS

Figures 3 and 4 show results derived by our genetic approach. To demonstrate our approach we reconstruct cubic B-spline curves from ordered points sampled from cubic B-spline curves, Figures 3(a) - (c), non-parametric curves, Figures 3(d) - (f), noisy data, Figure 4a, and data sampled at non-equidistant points along a B-spline curve, Figure 4b.

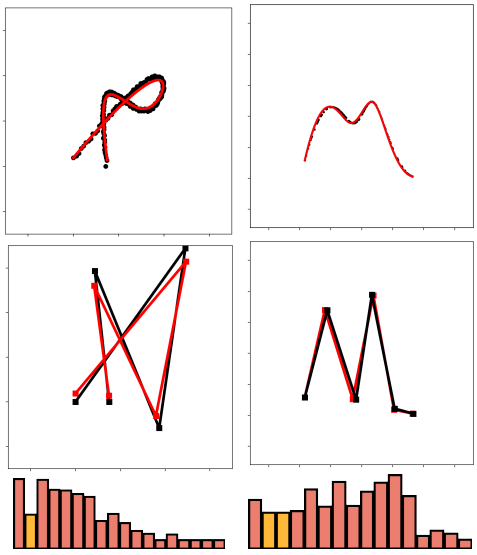
The top row of Figures 3 and 4 show the reconstructed curve (red or colored) plotted on top of the target B-spline curve (black). The colors of the reconstructed curve indicate how close it is to the target curve, scaled by the maximum distance measured. The color red corresponds to a smaller distance, whereas blue describes a larger distance.

The center row in Figures 3 and 4 shows a comparison between the control polygon used to create the target curve (black) and the control points generated by the



(a) Heart (open curve, $n = 7$) (b) Random curve ($n = 5$) (c) Random curve ($n = 7$) (d) Non-parametric curve f1 from [15] (e) Non-parametric curve f3 from [15] (f) Non-parametric curve f4 from [15]

Figure 3: Curves reconstructed using our algorithm. Top: target data points (black) and derived B-spline curve colored by the Euclidean distance to the target curve (red = low, blue = high). Center: result control polygon (red) overlaid on the control polygon of the reference B-spline curve sampled by the target points (black). Bottom: Visualization of the resulting chromosome as a sequence of genes. Yellow bar: vertex insertion operation. Red bar: vertex movement (bar height indicating movement distance).



(a) Noisy input (b) Sparse input

Figure 4: (a) Result after adding Gaussian noise ($\sigma = 0.7\%$ of the bounding box diagonal) to the target samples (black). (b) Reconstruction from sparse data, after randomly removing 75% of the point samples. Top: Result curve (red) overlaid on target points (black). Middle: Comparison of control polygons. Bottom: Chromosome visualization as in Figure 3.

algorithm (red). Because curves 3(d) - (f) are not derived from B-splines, a comparison is not possible.

As evident from these examples, the algorithm derives the same number of control points as was used to create the target curve and matches their positions closely, without any prior knowledge of the control polygon structure. Since the algorithm is not restricted on how many points are to be used in the solution, we expect curves to also be derived by more or less control points than the original target curve.

The bottom row in Figures 3 and 4 depicts the chromosome of each solution, consisting of operations that are applied to generate the control polygons shown below of each of them. The operations are applied to the template mesh in sequence starting from left. The last operation is on the very right. A vertex insertion is indicated by a yellow bar, while a vertex translation is shown in red. The size of the bar indicates the distance of the translation. We observe that translation operations applied to vertices at a later stage the modelling process tends to decrease. This is not explicitly encoded in the algorithm and shows that the algorithm tries to first move the control points reasonably close and then looks more closely to refine the position.

In Figure 4a the data sampled for curves has been altered by introducing noise to each data point. While the algorithm found control points close to those from which the target curve has been defined, they are not as precise as without noise. The average distance between our reconstructions from the noisy data points and the original curves, have an average Euclidean distance of 0.075.

To test the performance of our algorithm with sparse data points sampled from the target curve, we reduced the sampled points by 75% by randomly deleting data points. Results are displayed in Figure 4b. In the reconstruction the algorithm created control points similar to

the ones used in the original input. The example results in an average Euclidean distance of around 0.02 measured from the generated to the original curve.

Although noisy data effects our algorithm more than sparse data, it is quite robust to both.

The parameters used in the calculation of the results of our 3D reconstruction shown on top in Figure 5 for the genetic algorithm were 100 individuals, 200 generations and 20 parents which were combined to generate up to 10 children. The operations of the 10 best individuals were mutated specifically. 10 newly created individuals were added per generation and a limit of 3 individuals with the same genome was enforced. For the bottom row of Figure 5 the parameters were the same with the exception of the number of iterations which was 1000 and the limit of individuals with the same genome which was 10.

Figure 5 show results of the algorithm performing the fitment of a straightforward control mesh to a limit surface. It has no information about the control mesh which was used to generate the target and compare their resulting limit surfaces to the given target. The coloring indicates the relative distance from the point on the generated mesh to the target. The red color indicates a small distance to the target. The closer the color is to blue the further the target mesh is away. In Figure 5 top we observed distances from 0 to 0.001 and a mean of 0.0001 for the best individual of the genetic algorithm. Execution time was 240s. In Figure 5 bottom we observed distances from 0 to 0.005 and a mean of 0.002 for the best individual of the genetic algorithm. Execution time was about 1400 seconds. The distances were normalized by the bounding box diagonal. The control meshes show that our algorithm is able to identify the specific operations needed to generate to target control polygon.

In Figure 6 we test our genetic approach with slightly more challenging reconstruction problems. The shape on top was specifically modelled to enforce the algorithm to reconstruct exact sets of operations. While the best individual in the first generation has an extrusion without a prior face inset, by the 50th generation the correct set was identified and only positions were refined afterwards. The shape on the bottom was produced using a random set of operations. After 1000 generations our method was able to find the combination of operations very similar to the ones used to create the mesh. However, an extra edge loop was inserted in the top part, which, after 1000 generations remained in the shape. This demonstrates a limitation of the genetic reconstruction approach.

6 CONCLUSION

In this paper we introduce a genetic approach to the reconstruction of B-spline or subdivision curves and subdivision surfaces. The genetic algorithm takes a list of data points as input and evolves a control polygon or control mesh by applying a set of operations. The subdivision curves or surfaces corresponding to the derived control polygon or mesh approximates the input data closely. The algorithm derives the number of control points required as well as their position to optimise the approximation to the input data.

We proposed a genetic algorithm based on a modelling process, which is novel in the context of reconstruction surfaces or curves.

For the case of planar curve reconstruction our proposed method using two operations showed some promising results in reconstruction. The algorithm handles noise and sparsity well, although it struggles with more complex shapes like in Figure 3 (f), where it does tend to get stuck in local optima.

For the reconstruction of highly complex curves we suggest a pre-processing step in form of a curve segmentation algorithm at high curvature region to split up the reconstruction problem. Additionally, to speed up reconstruction of highly complex curves, future work will include extending this algorithm by a pre-processing step which derives an approximate minimum to the number of control points required to reconstruct the curve.

The 3D case is not a straightforward extension of the 2D case and requires a larger set of operations. Although results were impressive for the examples given in this paper, the algorithm struggles with convergence for more complex shapes. Future work will focus on solving the convergence problem by employing a more sophisticated approach, e.g. using NSGA-II [3], and multiple fitness functions, constraining the algorithm to converge faster.

REFERENCES

[1] Catmull, E., Clark, J., 1978. Recursively generated b-spline surfaces on arbitrary topological meshes. Computer-aided design 10, 350–355.

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

[3] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6, 182–197.

[4] Diaz-Gomez, P.A., Hougen, D.F., 2007. Initial population for genetic algorithms: A metric approach., in: Gem, Citeseer. pp. 43–49.

[5] Doo, D., 1978. A subdivision algorithm for smoothing down irregularly shaped polyhederons. Computer Aided Design , 157–165.

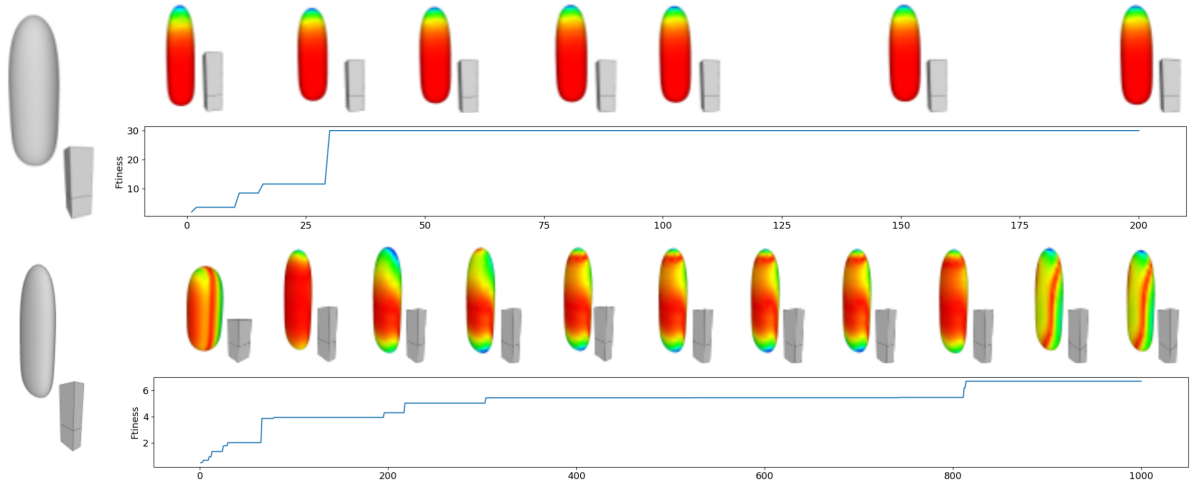


Figure 5: Comparison on the convergence behaviour of the algorithm using two examples. Top: cube extended in one direction. Bottom: cube flattened and then extended. For each, the fitness of the best solution is plotted as a function of generations. Results above the fitness graph show the best individual of the generation together with its limit surface. Limit surfaces are colored by the distance of a point on the mesh to the target mesh (red closer, blue further away). Since the the flattened shape needs very specific operations to be determined it takes many more generations as for the top shape.

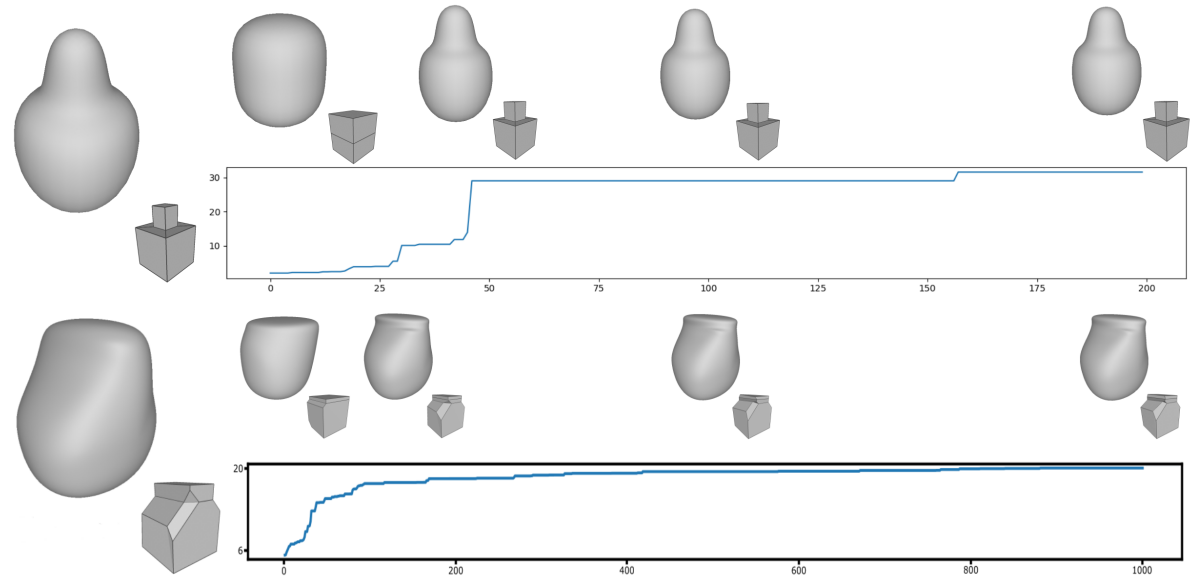


Figure 6: Two examples more complex examples of freeform geometry, generated using a random set of operations. Fitness and solutions are shown as in Figure 5.

[6] Farin, G.E., 2002. Curves and surfaces for CAGD: a practical guide. Morgan Kaufmann.

[7] Foundation, B., 2022. Blender python api (version 3.0.1). URL: `https://docs.blender.org/api/current/index.html`.

[8] Gálvez, A., Iglesias, A., 2012. Particle swarm optimization for non-uniform rational b-spline surface reconstruction from clouds of 3d data points. Information Sciences 192, 174–192.

[9] Gálvez, A., Iglesias, A., Puig-Pey, J., 2012. Iterative two-step genetic-algorithm-based method for efficient polynomial b-spline surface reconstruction. Information Sciences 182, 56–76.

[10] Hassan, M.F., Dodgson, N.A., 2005. Reverse subdivision, in: Advances in multiresolution for geometric modelling, Springer. pp. 271–283.

[11] Holland, J.H., 1976. Adaptation in natural and artificial systems.

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

[13] Katoch, S., Chauhan, S.S., Kumar, V., 2021. A review on genetic algorithm: past, present, and future. Multimedia tools and applications 80, 8091–8126.

[14] Kobbelt, L., 2000. Square root 3 subdivision, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 103–112. doi:10.1145/344779.344835.

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

[16] Kumar, M., Husain, D.M., Upreti, N., Gupta, D., 2010. Genetic algorithm: Review and application. Available at SSRN 3529843 .

[17] Lanquetin, S., Neveu, M., 2006. Reverse catmull-clark subdivision .

[18] Loop, C., 1987. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics .

[19] Ma, W., 2005. Subdivision surfaces for cad - an overview. Computer-Aided Design 37, 693–709. doi:10.1016/j.cad.2004.08.008.

[20] Moulaeifard, M., Bernard, S., Wellmann, F., 2023. Pysubdiv 1.0: open-source geological modeling and reconstruction by non-manifold subdivision surfaces. Geoscientific Model Development 16, 3565–3579.

[21] Rahamathulla, N.H., Misro, M.Y., 2023. Curve fitting using genetic algorithm and its application in craniofacial reconstruction. Pertanika Journal of Science & Technology 31.

[22] Sabsch, T., Braune, C., Dockhorn, A., Kruse, R., 2017. Using a multiobjective genetic algorithm for curve approximation, in: 2017 IEEE symposium series on computational intelligence (SSCI), IEEE. pp. 1–6.

[23] Sadeghi, J., Samavati, F.F., 2011. Smooth reverse loop and catmull-clark subdivision. Graphical Models 73, 202–217. doi:10.1016/j.gmod.2011.03.004.

[24] Samavati, F., Pakdel, H.R., Smith, C., Prusinkiewicz, P., 2003. Reverse loop subdivision .

[25] Sastry, K., Goldberg, D., Kendall, G., 2005. Genetic algorithms. Search methodologies: Introductory tutorials in optimization and decision support techniques , 97–125.

[26] Soon, G.K., Guan, T.T., On, C.K., Alfred, R., Anthony, P., 2013. A comparison on the performance of crossover techniques in video game, in: 2013 IEEE international conference on control system, computing and engineering, IEEE. pp. 493–498.

[27] Srinivas, M., Patnaik, L., 1994. Genetic algorithms: a survey. Computer 27, 17–26. doi:10.1109/2.294849.

[28] Takeuchi, S., Kanai, T., Suzuki, H., Shimada, K., Kimura, F., 2000. Subdivision surface fitting with qem-based mesh simplification and reconstruction of approximated b-spline surfaces, in: Proceedings the Eighth Pacific Conference on Computer Graphics and Applications, IEEE. pp. 202–212.

[29] Thengade, A., Dondal, R., 2012. Genetic algorithm–survey paper, in: MPGI national multi conference, Citeseer. pp. 7–8.

[30] Wang, X., Ang, K.D., Samavati, F.F., 2021. 4d surface mesh reconstruction from segmented cardiac images using subdivision surfaces, in: Proceedings of the 8th International Conference on Bioinformatics Research and Applications, pp. 56–62.

[31] Whitley, D., Rana, S., Dzubera, J., Mathias, K.E., 1996. Evaluating evolutionary algorithms. Artificial intelligence 85, 245–276. doi:https://doi.org/10.1016/0004-3702(95)00124-7.

[32] Xiyu, L., Mingxi, T., Hamilton Frazer, J., 2003. Shape reconstruction by genetic algorithms and artificial neural networks. Engineering Computations 20, 129–151.