NURBS FAIRING BY KNOT VECTOR OPTIMIZATION [1]

Barnabás Aszódi, Szabolcs Czuczor and László Szirmay-Kalos

Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
Budapest, Magyar Tudósok krt. 2., H-1117, HUNGARY
E-mail: ab011@ural2.hszk.bme.hu, cs007@ural2.hszk.bme.hu, szirmay@iit.bme.hu

ABSTRACT

The shape of a NURBS curve or patch is defined by the location of its control points, the weights associated with these points, and the parameter intervals, also called the knot vector. Most of the curve and patch design methods assume that the knot vector is constant and the user is allowed to modify only the control points and the weights. The possibility of controlling the shape through the change of the knot vector has shown up recently, but it turned out that this approach is less intuitive than either the control vertex or the weight modification. This paper attacks this problem by setting these knot values automatically, taking into account some goodness measures of the shape. In order to find the global optimum, simulated annealing is selected as the basic mechanism of the optimization. The paper reviews the basics of NURBS and simulated annealing, discusses our approach of setting the knot vector and concludes with the experience gained with this algorithm.

Keywords: NURBS, simulated annealing, curve and surface fairing.

1 INTRODUCTION

NURBS curves and surfaces allow to model a dozen of well-known basic objects (e.g. circle, ellipse and conic segments, sphere, etc.) and also to describe free form objects. NURBS stands for *Non-Uniform Rational B-Spline*. NURBS curves are parametric splines, whose main components are the 2D or 3D control points (or control vertices, or CVs for short), the weights of these points, and a knot vector limiting the effect of the control vertices onto a given segment of the curve. Control points can be imagined as small magnets, whose attacking force is proportional to their weight, but is also a function of the actual parameter. The knot vector consists of a non-decreasing sequence of parameter values, which defines for which parameter range a particular control vertex can influence the curve.

If the weights of all control vertices are similar, the NURBS becomes a NUBS (*Non-Uniform B-Spline*). The basis functions of NUBS ($B^{\text{NUBS}}_{CV index,\ level}(t)$) can be defined by the Cox-deBoor recursion formula:

$$B^{\text{NUBS}}_{i,1}(t) = \begin{cases} 1, & \text{if } t_i \le t < t_{i+1}, \\ 0, & \text{otherwise,} \end{cases}$$

$$B^{\text{NUBS}}_{i,k}(t) =$$

$$\frac{(t-t_i)B^{\text{NUBS}}_{i,k-1}(t)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-t)B^{\text{NUBS}}_{i+1,k-1}(t)}{t_{i+k}-t_{i+1}},$$

$$\text{if } k > 1,$$

where t_i is an element of the knot vector and k is the level of NUBS. Since it can happen that

[1] Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972 WSCG'2004, February 2-6, 2004, Plzen, Czech Republic. Copyright UNION Agency - Science Press

subsequent knot values are identical, the recursion formula can result in 0/0, which should be replaced by 1 in the implementation.

Note that if the number of control vertices is n, then the number of weights is also n, but the number of knot values is $n + K - 1$, where K is the level of the curve. In the active range of the parameter domain, the NUBS basis functions sum up to 1. Thus a mechanical analogy can be given to NUBS. Masses of $B_i(t)$ are placed at the control points, and the center of mass is the curve point for this parameter.

When we associate an additional scaling of these masses, called the weights and denoted by w_i, we can define the NURBS curve. Using the same center of mass analogy, the point on the NURBS curve for a given parameter t is obtained as:

$$\vec{r}(t) = \frac{\sum_{i=0}^{n-1} w_i B_i^{\text{NUBS}}(t) \cdot \vec{r}_i}{\sum_{j=0}^{n-1} w_j B_j^{\text{NUBS}}(t)} = \sum_{i=0}^{n-1} B_i^{\text{NURBS}}(t) \cdot \vec{r}_i,$$

where $\vec{r}_i$ is an element of the array of control points (n is the number of elements). Here we talk about NURBS, thus the weights of CVs can be less or greater than 1.

In case of NURBS surfaces we have two-dimensional arrays of CVs ($\vec{r}_{ij}$) and weights (w_{ij}), whose size is $m \cdot n$. The basis functions of the surface is the product of two curve basis functions $B_{i,k_u}^{\text{NUBS}}(u)$ and $B_{j,k_v}^{\text{NUBS}}(v)$, where k_u and k_v are the levels, u and v are the parameters of the patch in the two dimensions. The surface points can be calculated by the following formula:

$$\vec{r}(u,v) = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{ij} B_{ij}^{\text{NUBS}}(u,v) \cdot \vec{r}_{ij}}{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{ij} B_{ij}^{\text{NUBS}}(u,v)} =$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} B_{ij}^{\text{NURBS}}(u,v) \cdot \vec{r}_{ij}.$$

In order to draw a NURBS curve (or patch), it is tessellated then the resulting line strip (or mesh) is rasterized. The tessellation process requires the calculation of curve (or surface) points for a series of parameter values.

2 SHAPE CONTROL

The most obvious way of setting the shape of a NURBS curve is to place the control points, since the curve will approximate these control points. Changing the weights, we can make certain control points more important than the others. The curve will get closer to these more important points, while approximating others less accurately. If there are more control points than the degree of the curve, then only a subset of the control points affect the curve for a given parameter value. The set of these active control points change at certain parameter values, called knots. If the spline contains quadratic, or higher degree polynomials, then the curve only approximates the control points, and the knot vector also affects the shape.

The control points are responsible for the global shape control, since they have large scale effects. The modification of the weights has less influence, because the curve will always be in the convex hull of the active control points. Changing the knot value also has a similar limited effect, thus it can only be used to fine tune the curve. However, the dependence of the shape on the knot vector is much less intuitive than on the control vertices or on the weights. This is why most of the design methods do not allow the user to alter the knot vector directly, and set it to a default value.

For example a fourth level NURBS curve (whose basis function's degree is three) with six CVs has usually the following knot vector:

$$\{0, 0, 0, 0, 1, 2, 3, 3, 3, 3\}.$$

In this case there are multiple knot values at the beginning and at the end of the curve, making the curve go through the first and the last control vertices.

However, setting the knot vector in such way reduces the freedom of the designer, who would like to take advantage of the additional shape control of the knot vector modification. In [JH01], for example, it was proven that modifying a single, or a few knot values can be given an elegant geometric interpretation. However, when many, or all values are controlled simultaneously, this easy to understand behavior disappears. Thus we believe that the additional control mechanism given by the knot values can be exploited only by numeric techniques, which require just indirect input from the user. For example, the user can set general criteria what he expects from the curve, and a numeric algorithm finds the appropriate knot values automatically. Since the dependence of the shape properties on the large dimensional knot vector is non linear and is usually not even available in a simple algebraic form, we have to use an optimization procedure that can cope with this problem. Particularly, this optimization can have many local minima, from which the optimization is expected to select the global minimum.

The effect and the appropriate definition of the knot vector have already been considered by several researchers. The parametrization problem has been studied extensively in [Hos98, SKH98]. Authors usually treated the placement of the knot vector not as an optimization problem, but as a geometric one [JH01, Pie89, FS90]. The optimization aspect has show up in [LGM93], and [MRV95, RMV97, RA03, GB03] proposed genetic algorithms for the solution. Our approach also attacks the parametrization by optimization but we apply simulated annealing.

In the next section, we review the possible goodness measures that can be selected as an optimization goal function, then in section 4, the simulated annealing algorithm is presented, finally we detail our algorithm to find the knot values with simulated annealing.

3 NURBS QUALITY METRICS

In order to set the knot vector in an “optimal way”, we need a quality or goodness measure that can rank different curves or patches. In this section we define two possible criteria for curves and patches respectively, the curvature integral and the distance from a specified point. On the other hand, we should emphasize that the optimization procedure works with other quality measures as well (e.g. length $\int |d\vec{r}(t)/dt|\ dt$, approximated length $\int |d\vec{r}(t)/dt|^2\ dt$, approximated total curvature $\int |d^2\vec{r}(t)/dt^2|^2\ dt$, the elastic energy, etc. [GB03]).

3.1 Curvature integral

The *total curvature* is the integral of the *curvature*[Wei03] along the NURBS curve, which can be computed as

$$\Phi = \int_{t_{start}}^{t_{end}} |d\alpha|\ dt\ ,$$

where $d\alpha$ is the angular deviation of two infinitely small neighboring pieces of the curve around t (see Figure 1), and the integration domain is the union of the intervals where the basis functions sum up to 1. Using multiple knots at the beginning and at the end, and placing other knots uniformly at unit distances, we set $t_{start} = 0$ and $t_{end} = n - K + 1$. The curvature value Φ describes how wavy the curve is. In curve design we usually intend to minimize the oscillation of the curve, thus the curvature integral should be minimized.

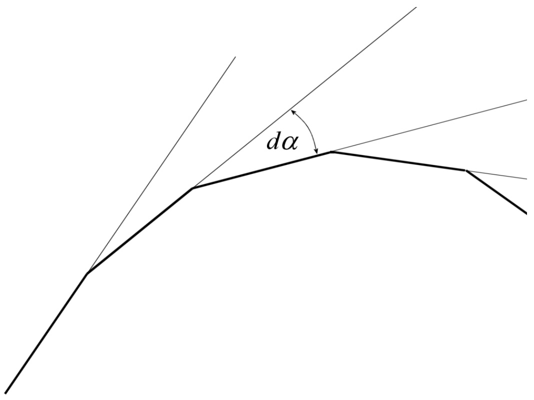


Figure 1: Angular deviation ($d\alpha$) of two infinitely small consecutive pieces of a tessellated curve

The curvature integral can be approximated by a discrete sum if the NURBS curve is tessellated:

$$\Phi = \sum_{\substack{t_{start}<t<t_{end} \\ \Delta t=0.1}} |\Delta\alpha_t| \cdot \Delta t\ ,$$

where

$$\tan\Delta\alpha_t = \frac{\sin\ \Delta\alpha_t}{\cos\ \Delta\alpha_t} = \frac{|\vec{g}_{t,last} \times \vec{g}_{t,next}|}{\vec{g}_{t,last} \cdot \vec{g}_{t,next}},$$

and $\vec{g}_{t,last}$ and $\vec{g}_{t,next}$ are the tangent unit vectors of the curve before and after the tessellated point (at the actual t):

$$\vec{g}_{t,last} = \frac{\vec{r}_t - \vec{r}_{t-\Delta t}}{|\vec{r}_t - \vec{r}_{t-\Delta t}|} \quad \text{and} \quad \vec{g}_{t,next} = \frac{\vec{r}_{t+\Delta t} - \vec{r}_t}{|\vec{r}_{t+\Delta t} - \vec{r}_t|}.$$

Working with the normal vectors, we can calculate a curvature integral for a NURBS patch. Let us examine the change of the normal vector as we move infinitesimally along the two isoparametric curves of the patch. Suppose that the normal vector is rotated by $d\alpha$ if parameter u is modified by du, and by $d\beta$ when parameter v is increased by dv. Notice that we cannot multiply these angles, because the flatness of either direction would set the whole integral to zero (see Figure 2). To avoid this, we average them to characterize the curvature of a parametric surface:

$$\Psi = \int_{v_{start}}^{v_{end}} \int_{u_{start}}^{u_{end}} \frac{|d\alpha| + |d\beta|}{2} \, du \, dv \ . \quad (1)$$

Note that this formula is just an approximation of the surface curvatures usually used in mathematics. The most popular surface curvature in differential geometry is the *mean curvature*[Wei03], which is the average of the principal curvatures, i.e. the minimum and the maximum of normal curvatures. This means that our approximation gives back the mean curvature if the tangent directions of isoparametric lines are the principal directions (i.e. the directions where the normal curvatures are minimum and maximum). The reason of not using the mean curvature is that our Ψ measure (equation 1) is much simpler to compute, but also characterizes the bumpiness of the surface, thus will result in similar optimal configuration as if we used the integrand of the mean curvature.

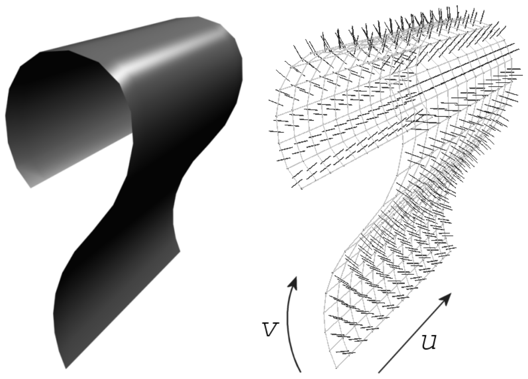


Figure 2: A NURBS patch with one dimensional flatness. (The small sticks on the surface are the normal vectors of the facets.) Along parameter u we have zero curvature.

3.2 Distance from a specified point

The distance from a user specified point $\vec{p}$, defined as

$$\min_t |\vec{p} - \vec{r}(t)|$$

is a quality measure that can be intuitively interpreted and applied by the designer. The user specifies the point and wants the curve to pass this point as closely as possible. In this case the knot vector should be found to minimize this distance. When we calculate this distance for a given knot vector, an approximation of the distance function is evaluated. The curve is tessellated, and the distance is estimated by the minimum distance from the vertices of the generated line strip or mesh. If the tessellation is not fine, the calculation can be less accurate (see Figure 3).

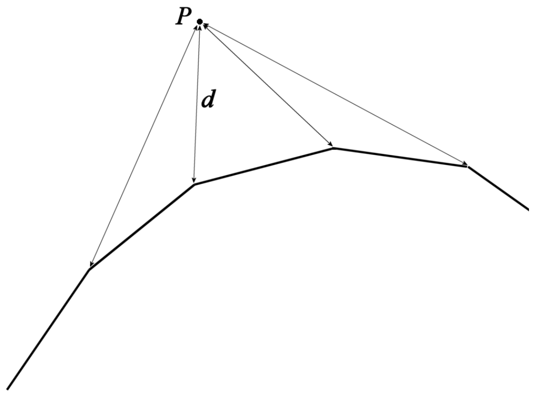


Figure 3: Distance (d) from a specified point (P)

4 OPTIMIZATION PROCESS: SIMULATED ANNEALING

The optimization process tries to find a knot vector for which the quality of the curve or patch is optimum. We have to take into account that this is a large dimensional, non-linear optimization, with many local minima and maxima. If we used some greedy optimization method, then we would end up in a local optimum. On the other hand, the function to be optimized is not available in a simple algebraic form, but we can only evaluate it at certain points using numeric approximation.

Such hard optimization problems can be successfully attacked by the simulated annealing algorithm, which is based on the metaphor of the cooling process of a material composed by particles. If the absolute temperature is T, then the probability that a particle is at energy level E is proportional to $\exp(-E/kT)$, where k is the Boltzmann constant. At the beginning of the cooling process the temperature is high, and the particles can reach high energy states easily. When cooling starts, the particles should find lower energy states. Since even at lower temperatures, a particle can have higher energy with a small probability, we give the particles the chance to jump over local minima. The particles finally find the optimal, i.e. the lowest energy state.

Note that due to accepting higher energy states, such an algorithm would never converge for a given temperature. To force convergence, the temperature should be decreased in subsequent iteration steps.

When it comes to optimizing the knot values, the particle states, the temperature and the Boltzmann constant should be translated to the actual terms of our geometric problem. The actual state of the particles will be the actual knot vector, and the energy is an appropriate quality measure of the resulting curve, e.g. the curvature integral. The initial temperature and the Boltzmann constant should be selected to set the acceptance probability to about 80% at the beginning of the algorithm.

In order to vary the state randomly proportional to $\exp(-E/kT)$, Metropolis sampling is used [MRR+53, SK99]. Based on the actual state, it selects a new random, tentative state in the neighborhood of the actual state. The tentative state can be imagined as a small perturbation of the actual state. Then the algorithm compares the energy of the tentative and the actual states. If the tentative state has lower energy, than it is accepted as the new next state. On the other hand, if the tentative state has higher energy, it might still be accepted randomly, with probability $\exp(-\Delta E/kT)$, where ΔE is the energy increase. Such sampling would asymptotically sample a state with a probability proportional to $\exp(-E/kT)$.

5 SHAPE CONTROL BY OPTIMIZING THE KNOT VECTOR

In this section we put the steps of the previous section together and discuss how the knot vector can be optimized. We start with a NURBS curve whose CVs and weights are known, and which has the default knot vector (the knots are placed at unit intervals).

In each step of the Metropolis iteration the knot values are perturbed a little. In our implementation we allow a knot value to go at most distance 0.5 from its default setting, thus the order of knot values can never change.

Before we get the elements in random order and start perturbing them, we measure the specified NURBS property defined in section 3, and consider it as the energy of the actual curve. After perturbing the elements we compare the actual value of the energy with the original one. If this energy is lower, we keep this new knot vector. If the new energy is higher, i.e. new property is worse than the original one, then we keep the new configuration only with a calculated probability. We can formalize this decision in a very simple way: let us generate a pseudo-random number between 0 and 1. If this value is less then the calculated probability, we are lucky and keep the new settings. Otherwise we throw them away.

We gradually decrease the temperature giving smaller chance to increase the energy. If the cooling speed is low, then the algorithm will need a long time to converge. On the other hand, if the cooling is fast, the algorithm can miss the global minimum and end up in a local minimum. However, the quality of the NURBS could still be much better in this local minimum than at the beginning of the process. The optimization process is summarized in the following code:

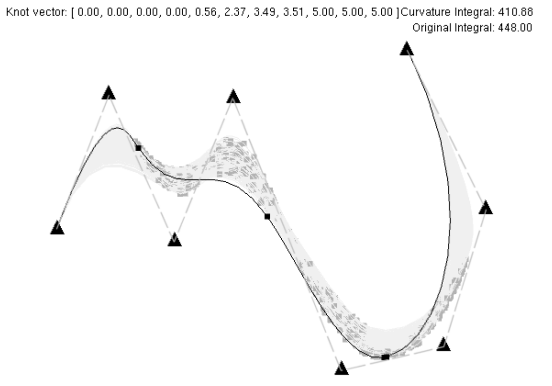
```
Set the knot vector $t[]$ to its default;
Set temperature $T$ and Boltzmann constant $k$;
repeat
    repeat                    // *Metropolis*
        perturb $t[]$ randomly to obtain $tnew[]$;
        $\Delta E = \Phi(tnew[]) - \Phi(t[])$;
        $P = \min\{1, \exp(-\Delta E/kT)\}$;
        if (random() $<$ $P$) $t[] = tnew[]$;
    until (equilibrium);
    decrease $T$;
until ($T > Tmin$);
return $t[]$;
```

The Metropolis sampling generates samples with the required probability only in an asymptotic state, that is why we have to iterate until "equilibrium". Unfortunately, it is not easy to decide when we reached that state. In practice, the number of Metropolis iterations equals to 5–10 times the number of elements in the knot vector. On the other hand, if the cooling is slow, then the probability distribution changes slowly, thus it is enough to iterate only once in the Metropolis loop.

6 IMPLEMENTATION

We have implemented the presented algorithms in C++ using OpenGL graphics support. Example snapshots of the application are shown in Figures 4 and 5.

Figure 4 shows the CVs of the NURBS curve as big black triangles and the border of curve segments as smaller black rectangles. The progress of the curve can also be seen by the trajectories of the curve. In Figure 5 the black triangles indicate the CVs of the patch.



Knot vector: [0.00, 0.00, 0.00, 0.00, 0.56, 2.37, 3.49, 3.51, 5.00, 5.00, 5.00]Curvature Integral: 410.88
Original Integral: 448.00

Figure 4: A snapshot of the curve optimization program

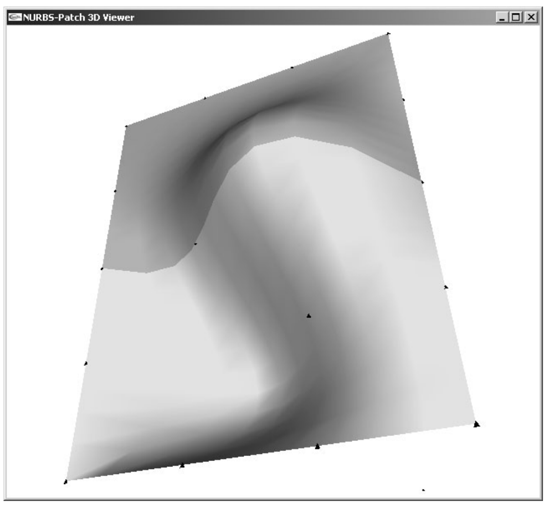


Figure 5: A snapshot of the surface optimization application

7 RESULTS

We have made tests on four different NURBS curves (see Figure 6) and four different NURBS patches (see Figure 7). Table 1 shows the results. The second column contains the original value of curvature integrals. In the third column the improved (iterated) values can be seen, and the last column shows the number of iteration cycles.

The cooling speed and minimum temperature $Tmin$ have been set empirically to obtain globally optimal results with very high probability. Thus the cooling speed was low (see section 5), the temperature was multiplied by 0.999 in each iteration step. We have set the minimum temperature to guarantee that the found solution does not change practically (it changes only with a very small probability). These explain why we needed so many iteration steps (Table 1) to reach the supposed global minimum. These iteration numbers correspond to acceptable running times in curve optimization, but the surface optimization is too slow for interactive applications. For example, on an AMD Athlon 1.4 GHz computer the time of 1000 iteration steps to fair a 7×7 patch using tessellation factor 5 took approximately 4 minutes. If we use fast annealing, the possibility of reaching a local minimum increases, but we can still expect significant improvements in the surface quality.

Name	Original	Improved	Iteration steps
Wave	408.83	352.42	12000
Helix	841.33	833.90	10000
Gauss	280.46	267.50	11000
Circle	344.26	333.96	13000

Table 1: Results of measuring the curvature integral of different types of NURBS curves

Name	Original	Improved	Iteration steps
Gauss	7109.1	6867.1	6000
Rugby	9480.9	9205.0	7000
Terrain	13564.3	12799.9	5200
Cloth	5464.0	5433.7	3200

Table 2: Results of measuring the curvature integral of different types of NURBS patches

8 CONCLUSIONS AND FUTURE WORK

The results prove that optimizing the elements of the knot vector of a NURBS can improve the curve, thus this technique is feasible. The simulated annealing process was found to be efficient to solve this optimization problem. We have concluded that even with higher cooling speeds, when there is no guarantee that the global optimum is obtained, significant fairing can be achieved quickly.

When extending the idea to patches, we concluded that this optimization is less powerful than for curves. This can be explained by the fact that the two dimensional parametrization leaves smaller freedom for the knot vector to affect the shape of the surface. On the other hand, the simulated annealing gets less effective when the dimension of the search domain is increased. In the future we intend to attack this problem by allowing the annealing process to proceed only in a lower dimensional sub-space at a time. According to the first experiments, this can improve the robustness and the speed considerably.

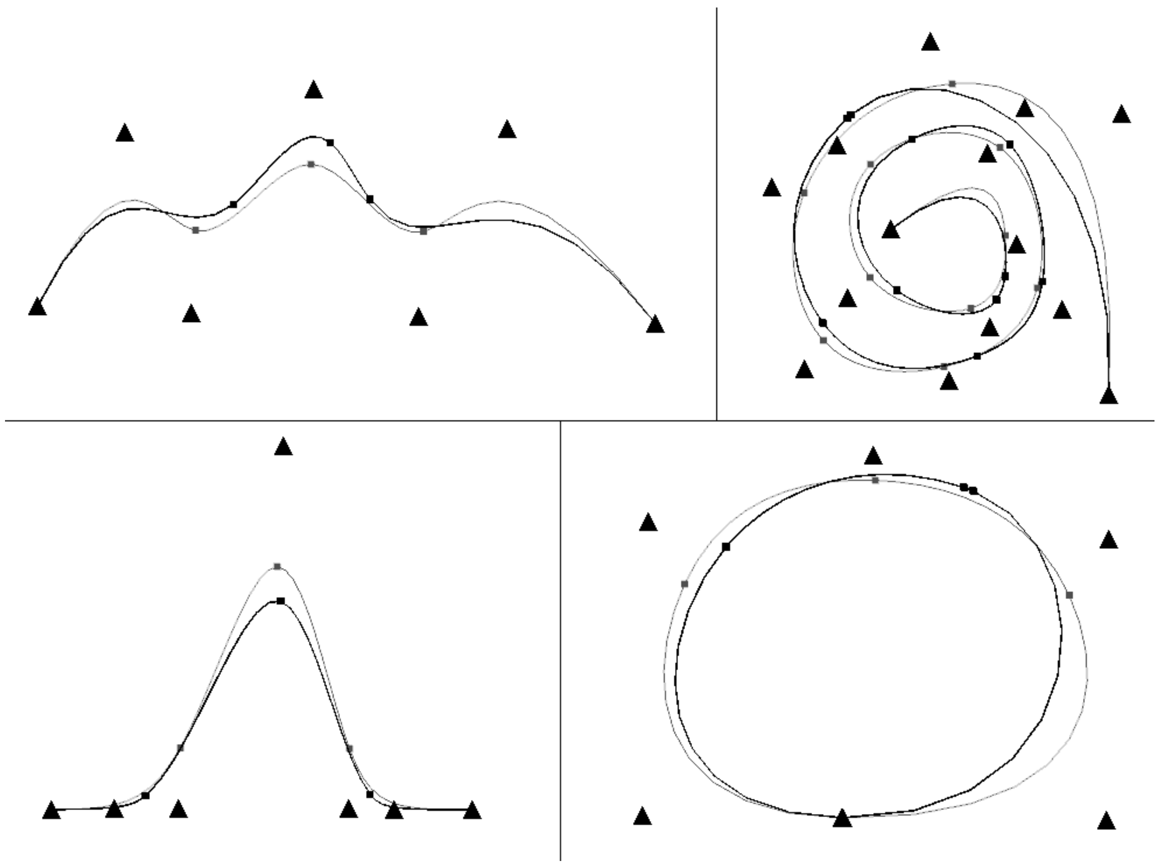


Figure 6: Tested NURBS curves: Wave (top left), Helix (top right), Gauss (bottom left), Circle (bottom right). We show the curves with grey color before the iteration process, and with black color after the iteration process.

9 ACKNOWLEDGEMENT

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T042735), IKTA ref. No.: 00159/2002, the Bolyai Scholarship and the Slovene-Hungarian Fund.

REFERENCES

[FS90] G. Farin and N. Sapidis. Automatic fairing algorithm for B-spline curves. *Computer-Aided Design*, 22(2):121–129, 1990.

[GB03] R. Goldenthal and M. Bercovier. Spline curve approximation and design by optimal control over the knots using genetic algorithms. In *EROGEN*, 2003.

[Hos98] J. Hoschek. Intrinsic parameterization for approximation. *Computer Aided Geometric Design*, 5:27–31, 1998.

[JH01] I. Juhász and M. Hoffmann. The effect of knot modifications on the shape of B-spline curves. *Journal for Geometry and Graphics*, 5:111–119, 2001.

[LGM93] P. Laurent-Gengoux and M. Mekhilef. Optimization of a NURBS representation. *Computer-Aided Design*, 25(11):699–710, 1993.

[MRR$^+$53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[MRV95] A. Márkus, G. Renner, and J. Váncza. Genetic algorithms in free form curve design. In *Mathematical Methods for Curves and Surfaces*, pages 343–354, 1995.

[Pie89] L. Piegl. Modifying the shape of rational B-splines. *Computer-Aided Design*, 21:509–518, 1989.

[RA03] G. Renner and Ekárt A. Genetic algorithms in computer aided design. *Computer Aided Design*, 35:709–726, 2003.

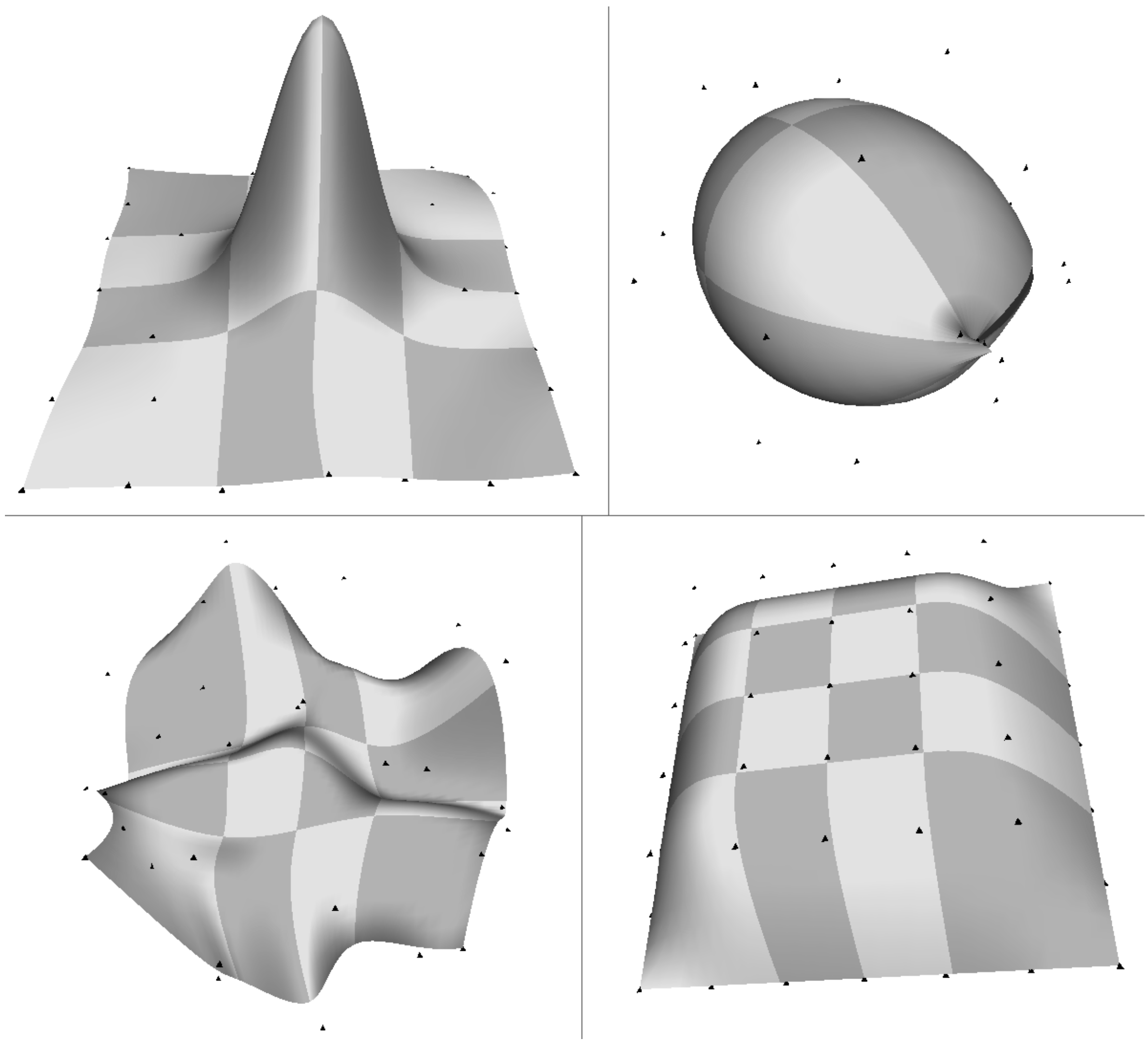


Figure 7: Tested NURBS patches: 7×7 Gauss-bump (top left), 7×5 Rugby-ball (top right), 7×7 Terrain (bottom left), 7×7 Table-cloth (bottom right). In the case of the first three shapes the positions of the CVs were randomized as if they were located by a human modeler. Note that the Question-mark (in Figure 2) is equivalent to a NURBS curve because of its one dimensional flatness.

[RMV97] G. Renner, A. Márkus, and J. Váncza. Spline interpolation with genetic algorithms. In *IEEE, Shape Modelling, 1997*, 1997.

[SK99] L. Szirmay-Kalos. *Monte-Carlo Methods in Global Illumination.* Institute of Computer Graphics, Vienna University of Technology, Vienna, 1999. http://www.iit.bme.hu/~szirmay/script.pdf.

[SKH98] T. Speer, M. Kuppe, and J. Hoschek. Global reparameterization for curve approximation. *Computer Aided Geometric Design*, 15:869–877, 1998.

[Wei03] E. Weisstein. World of mathematics. Technical report, 2003. http://mathworld.wolfram.com.