DTE-3604 - Applied Geometry and Special Effects

Håkon Berg Borhaug

March 4, 2023

Abstract

This is a technical report which covers the project work done in DTE-3604 at UiT. This includes creating curves and surfaces using different types splines, and creating a special effect on one of the curves using an affine transformation. Implementation details for each step are described and the resulting rendering of each curve and surface is shown in form of screenshots of a demo application.

1 Introduction

This report covers the project work done in the course *Applied Geometry and Special Effects* at UiT. The project contains different programming exercises regarding geometric structures and dynamic objects that continuously change shape. In total there are six parts of this project: a B-spline curve, a closed subdivision curve, a model curve, a blending spline curve, a dynamic visual special effect in the form of affine transformations of local curves, and a blending spline surface. We will cover the implementation details and results from solving these six tasks. The source code can be found at [Borhaug, 2023].

2 Implementation

To implement the mentioned tasks we used the *GMLib* library [Lakså et al., 2006] and the theory found in the course book [Lakså, 2022], and in addition we display our results by extending the provided demo application *qmldemo*. For the first four parts concerning curves we extend the *PCurve* class found in *GMlib*, and for the last point, which concerns surfaces, we extend the *PSurf* class. In all the six cases we generate a knot vector (two in the case of the surface), which is a vector containing *knot values*, which in turn describes the domain for each of the basis functions. This domain is determined by the state of B-spline: open/clamped or closed. For the open/clamped state the domain of the curve is restricted to $[t_d, t_n]$, while for the closed state it's extended to $[t_d, t_{n+d})$.

2.1 B-spline curve

A B-spline curve is a set of local basis functions that sums up to 1 over the whole curve domain, where each basis function is connected to its respective control point. The formula for the curve is

$$c(t) = \sum_{i=0}^{n-1} c_i b_{k,i}(t), \quad \text{where} \quad \sum_{i=0}^{n-1} b_{k,i}(t) = 1, \quad \text{for} \quad t_d \leq t \leq t_n, \tag{1}$$

where c_i are the control points that together form the control polygon, $b_{d,i}$ are the basis functions, d is the polynomial degree, $k = d + 1$ is the order and $\tau = \{t_i\}_{i=0}^{n+d}$ is the knot vector. The order k defines the number of basis functions $b_{k,i}(t) > 0$ for a given knot interval t.

Our implementation consists of constructing a clamped 2nd degree B-spline curve with two constructors: one which takes a vector of control points as input, and one which takes a vector of points and an integer which determines how many control points to generate using the least squares method, where both of these constructors create the knot vector using these inputs.

To draw these B-spline curves we implemented the evaluation function. This function takes a value t as an argument, which is generated by the sample function and is within the curve domain $[t_d, t_n]$. This sample function can be called with different sample sizes, for example when the sample function is called with 20 as sample size argument, the evaluation function is ran with 20 different and uniformly spaced t values, which in turn uses the k basis functions for the given knot interval and their respective control points to find the position of the corresponding point in the curve $c(t)$. In our implementation we achieve this by solving the B-spline factor matrix for $d = 2$:

$$c(t) = \begin{pmatrix} 1 - w_{1,i}(t) & w_{1,i}(t) \end{pmatrix} \begin{pmatrix} 1 - w_{2,i-1}(t) & w_{2,i-1}(t) & 0 \\ 0 & 1 - w_{2,i}(t) & w_{2,i}(t) \end{pmatrix} \begin{pmatrix} c_{i-2} \\ c_{i-1} \\ c_i \end{pmatrix} \tag{2}$$

where $w_{d,i}(t) = \frac{t - t_i}{t_{i+d} - t_i}$ is the linear translation and scaling function, and i is determined by the knot interval $t_i \leq t < t_{i+1}$.

2.2 Subdivision curve

A subdivision curve is a curve where you have used a subdivision scheme to refine the original control polygon (polygon created by the control points) of some curve into a new refined control polygon. One of these schemes is the Lane-Riesenfeld algorithm [Lane and Riesenfeld, 1980]. The algorithm is divided into two parts: doubling the set of points and smoothing/degree raising. The first part in is done by either making two of each point or by inserting new points in the midpoint between every point. The second part is done by replacing the points with new points in the midpoint between the old points,

which is repeated to the desired degree. After each iteration of this procedure we have double the amount of points, and the whole procedure of doubling and smoothing is repeated until we have the desired degree or amount of points. To implement this we have used Algorithm 7 in [Lakså, 2022], specifically the $LaneRiesenfeldClosed$, $doublePart$ and $smoothPartClosed$ functions.

2.3 Model curve

The model curve we chose for this assignment is a random 3-dimensional curve with the Cartesian parametrization:

$$\begin{aligned} x &= cos(3t) \\ y &= sin(t) \\ z &= cos(2t) \end{aligned} \tag{3}$$

There is no particular reason for choosing this curve, but due to our professor suggesting this specific one and with the time constraint working on this project, we ended up keeping this curve instead of a more complex one.

2.4 Blending spline curve

A blending spline curve is a curve created by blending several other curves using a B-function (blending function). We have implemented a blending spline curve which is a 2nd order B-spline, which was done by setting the sub-curves to be parts of the model curve mentioned above, and solving:

$$c(t) = \begin{pmatrix} 1 - B \circ w_{1,i}(t) & B \circ w_{1,i}(t) \end{pmatrix} \begin{pmatrix} c_{i-1} \\ c_i \end{pmatrix} \tag{4}$$

where $B(x)$ is the blending function. We chose the symmetric trigonometric B-function of order 2 as the blending function. We can see that eq. (4) is a simplification of eq. (2) where the degree is set to 1 rather than 2, and with the addition of the B-function. The main difference between this and our implementation of the B-spline curve is the use of control/local curves instead of control points, and since the purpose of this task was to copy the model curve, we set the local curves of the blending spline curve to be parts of the model curve.

2.5 Special effect

The special effect made is an affine transformation of of the local curves of the blending spline curve which includes both translation and rotation. This is implemented such that the local curves with even indexes are translated in a way so that they are oscillating up and down and rotated along their Y-axis. The whole curve is also rotated to show the effect on the local curves more clearly. All of this is done by overriding the $localSimulate$ function from the $PCurve$ class we extended.

2.6 Blending spline surface

Blending spline surfaces are an extension of blending spline curves where control points are replaced by local surfaces. The general formula for evaluation is:

$$S(u,v) = \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} s_{ij}(u,v) B \circ w_{1,i}(u) B \circ w_{1,j}(v), \tag{5}$$

where s_{ij} are $n_u \times n_v$ local surfaces, and $B \circ w_{1,i}(u), B \circ w_{1,j}(v)$ are the B-spline basis functions with blending functions. As we can see we are no longer evaluating for t but rather u and v, since a surface is 2-dimensional compared to a 1-dimensional line. This equation can then be reformulated to a similar matrix structure as we have used previously:

$$S(u,v) = \begin{pmatrix} 1 - B \circ w_{1,i}(u) & B \circ w_{1,i}(u) \end{pmatrix} \begin{pmatrix} s_{i-1,j-1}(u,v) & s_{i-1,j}(u,v) \\ s_{i,j-1}(u,v) & s_{i,j}(u,v) \end{pmatrix} \begin{pmatrix} 1 - B \circ w_{1,j}(u) \\ B \circ w_{1,j}(u) \end{pmatrix} \tag{6}$$

To be able to render the surface with light properties we will need to find the surface normals. This is done by calculating the partial derivatives in addition to the position for each evaluation of the u and v values, which in turn is used in the equation:

$$N = \frac{n}{|n|}, \quad \text{where} \quad n = S_u \wedge S_v, \tag{7}$$

where $\wedge$ denotes the 3D vector product, and S_u and S_v are the partial derivatives of S in terms of u and v respectively.

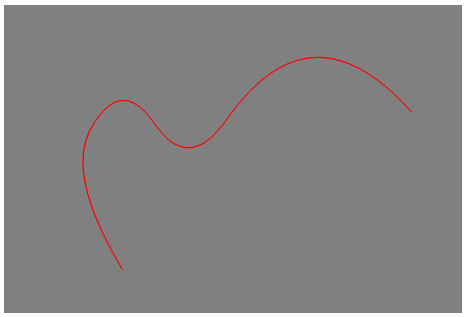
3 Results

In this section we will go over the results of our implementation. The figures presented are all screenshots of the application running our implementation. For all implementations which require a set of points as input we use the set of points

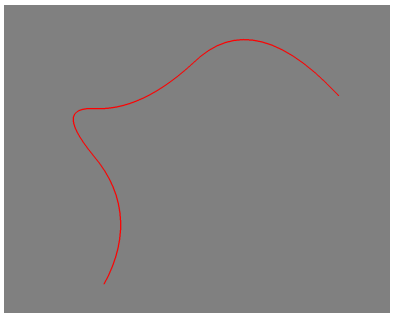
$c = \{(0,0,0), (5,0,0), (10,5,0), (10,10,0), (5,15,0), (10,20,0), (5,25,0), (0,20,0)\}$.

3.1 B-spline curve

In fig. 1 we can see the two types of 2nd order B-spline curves we get using our implementation using c as our input points to the two constructors. We can see that the B-spline using the 8 points as control points creates a curve that looks as expected, but the curve using least squares to create 7 control points deviates quite a bit from the structure of the other curve.



(a) curve using c as control points



(b) curve using least squares method to create 7 control points from c

Figure 1: 2nd degree clamped B-spline curves

3.2 Subdivision curve

In fig. 2 we can see the result of the implementation both a 2nd degree closed subdivision curve and a 3rd degree closed subdivision curve using c as control points. As we can see in the figure, the curve gets more refined the higher the degree.

3.3 Model curve and blending spline curve

In fig. 3a we can see the model curve implementation rendered, and in fig. 3b we can see the blending spline copy of the model curve. The number of local surfaces in the blending spline curve is set to $n = 4$.

3.4 Special effect

The special effect in action is shown in fig. 4 as two example frames of the animation. It is hard to convey in pictures, but we can see during the second frame that compared to the model curve its based on that it is about to return to its original shape after moving down while rotating.

3.5 Blending spline surface

To show our implementation of blending spline surfaces we have constructed three different blending spline surface copies: a plane (open in both parameter directions), a cylinder (open in one parameter direction and closed in the other), and a torus (closed in both parameter directions). In fig. 5 you can see the plane copy with local surfaces highlighted, and you can see a static transformation performed on a few of its local curves. Fig. 6 shows the cylinder surface along with the static transformation done on some of its local curves. Lastly we can see the torus surface and an example of a static transformation done on some local surfaces in fig. 7. In all of these examples we set $n_u = n_v = 4$, so we get in total 16 local surfaces.

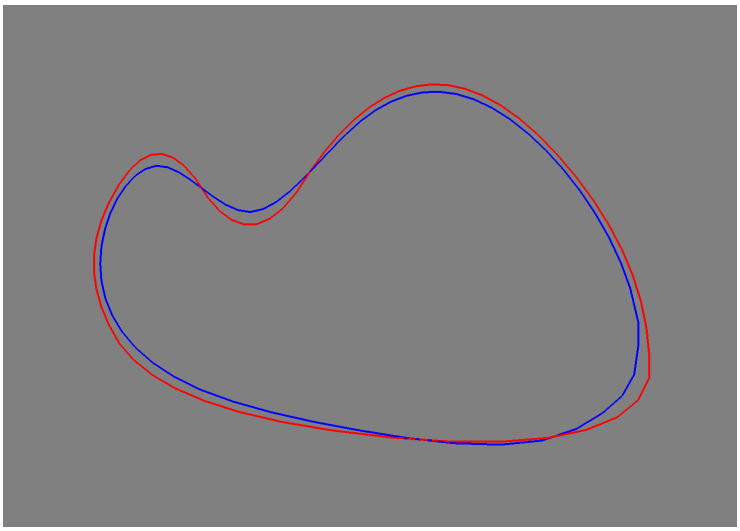
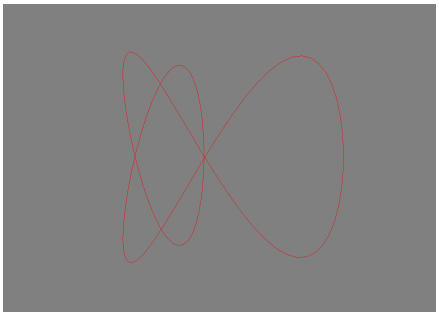
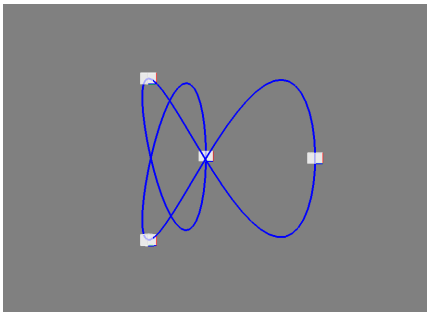


Figure 2: 2nd degree closed subdivision curve in red and 3rd degree closed subdivision curve in blue



(a) Model curve with parametrization eq. (3)

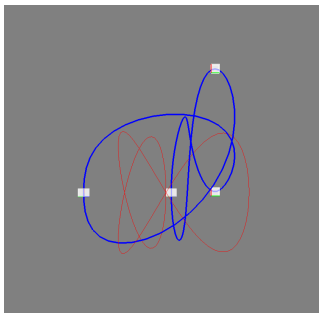


(b) Blending spline curve using four segments of the model curve as local curves, with their local origin highlighted as gray boxes

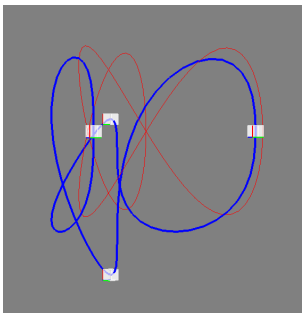
Figure 3: Model curve and its blending spline copy

4 Discussion and concluding remarks

From our results we can say that we managed to successfully implement every exercise on some level. The B-spline using the least squares method shows a somewhat big deviation from the B-spline created using the defined points as control points. If this is an error in our implementation, just a limitation of the points chosen as input or actually a "correct" result is hard to say. In addition, the special effect implemented is not particularly complex or interesting compared to what is possible to achieve by using affine transformations, but ac-

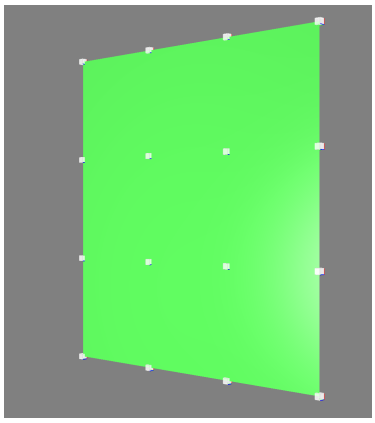


(a)

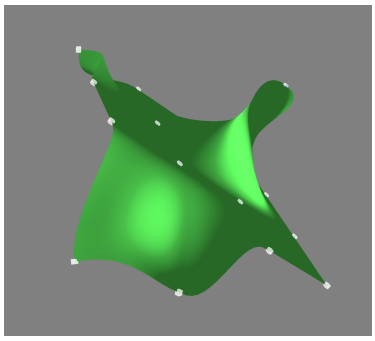


(b)

Figure 4: Special effect on blending spline curve in blue and model curve in red



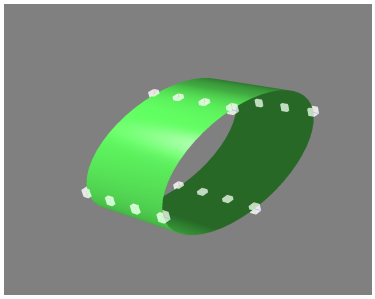
(a) Plane surface with local surfaces highlighted in gray



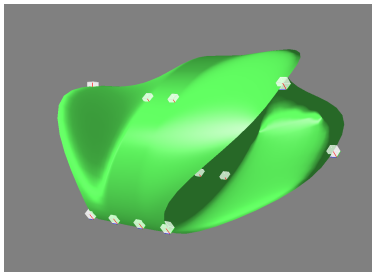
(b) Same plane surface with four of the local surfaces translated and rotated

Figure 5: Blending spline surface copy of a plane

complishes the task of both performing translation and rotation of local curves. The choice of another model curve might have led to more interesting ideas for a visually complex special effect. All in all, we found the project challenging, especially due to the limited time available to work on the project, but at the same time the work was interesting and instructive in our understanding of geometric modelling.

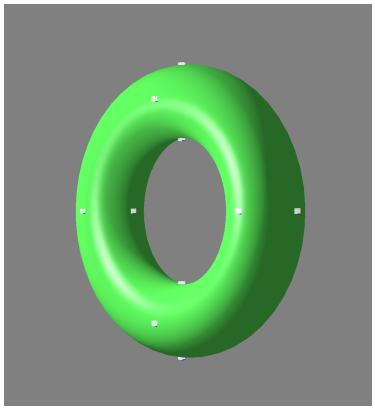


(a) Cylinder surface with local surfaces highlighted in gray

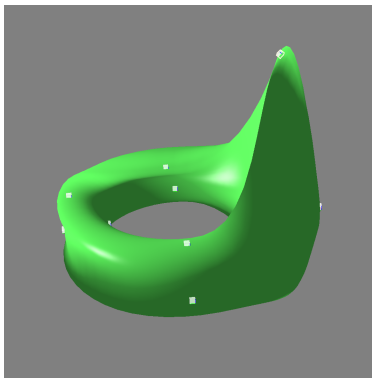


(b) Same cylinder with three of the local surfaces translated and rotated

Figure 6: Blending spline surface copy of a cylinder



(a) Torus surface with local surfaces highlighted in gray



(b) Same torus with three of the local surfaces translated and rotated

Figure 7: Blending spline surface copy of a torus

References

[Borhaug, 2023] Borhaug, H. B. (2023). Source code for the project. https://github.com/haakonbor/applied_geometry.

[Lakså, 2022] Lakså, A. (2022). *Blending techniques in Curve and Surface constructions*. Geofo.

[Lakså et al., 2006] Lakså, A., Bang, B., and Kristoffersen, A. R. (2006). Gm_lib, a c++ library for geometric modelling. Technical report, Narvik University College, Narvik, Norway.

[Lane and Riesenfeld, 1980] Lane, J. M. and Riesenfeld, R. F. (1980). A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):35–46.