Table of contents

1 Abstract

In computer graphics, T-spline surface are often used to generate 3D objects. This mathematical model is similar to the better known NURBS: it is defined by control points and knot vectors, but with more freedom in surface manipulation. This extra freedom comes from the freely adjustable knots. How that actually works and what it entails to create a T-spline surface are the scope of this project. I would like to explain the fundamental properties of a T-spline through examples, experiment with knot vectors. The aim of this project is to gain some intuition on how a surface is distorted by knot vector alteration.

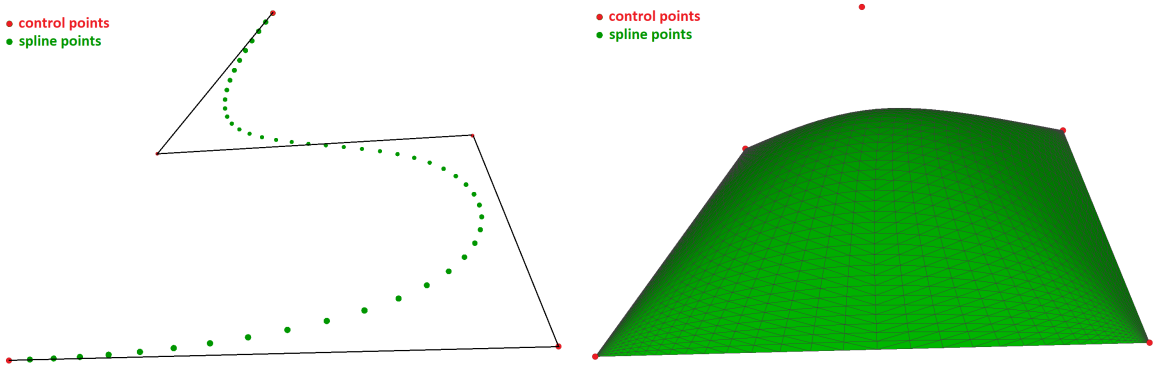
This essay focuses on the mathematics and less on coding. However, it is inevitable to write some code in order to create the necessary examples.

At the end, I will also show that T-spline is just a generalisation of B-spline, and that one can create the same surfaces with both.

2 Introduction

In recent days, a growing number of fields use 3D modelling to visualise materials and even print out various products, such as medical science, manufacturing industry or architecture (*Application of 3D printing*, 2017). From a computer graphics point of view, these models can be represented in several different ways, most of which will not be discussed in this report. However, a widely applied method that uses splines to define two- and three-dimensional objects (i.e. curves and hollow surfaces) will be detailed.

A spline is a mathematical parametric polynomial function that is defined by control points and knots (Kothari et al., 2019).



(a) Spline curve with 1D parameter

(b) Spline surface with 2D parameter

Figure 1: Spline curve and surface consisting the same 5 control points with one- and two-dimensional parametrisation, respectively.

Dragging the control points shifts a restricted part of the surface towards it. How the area of effect (AOE) looks lies in the knots and their distance from each other. Knots are less intuitive to understand,

since control points exist in the 3D real space, meanwhile knots exist in the parametric space that is "invisible". This will be explained in more detail later in Section 3.

The goal of this essay is to give some intuition about how such surfaces change by altering the parametric space, in restriction of T-spline surfaces which have exceptionally high flexibility in terms of knots.

3 Technical background

In order to get a full picture on T-splines, it is best to start with B-splines. Depending on the dimension, one can talk about a spline curve or a spline surface. The fundamental mathematics work similarly, though. Either way, the equation that describes a spline is a parametric equation where the substituted values exist in the parametric space that have nothing to do with the real 3D space. For nice surfaces, however, a loose connection is made between them two, but mathematically speaking, they are separate. That is why it is not intuitive how certain changes in the parametric space alter the real space.

3.1 B-spline

Spline is a generic term. There are several different types, but the base of 3D modelling relies on B-splines. For all those mathematics-oriented readers out there, what it means from a more technical perspective can be found in Gallier (2018). What is important to know is that the B-spline equation is a recursive function defined by Equation (1) (*B-spline*, 2022).

$$
\begin{aligned}
&B_{i,k}(p) = m_{i,k-1}(p)B_{i,k-1}(p) + \big(1 - m_{i+1,k-1}(p)\big)B_{i+1,k-1}(p),\\
&\qquad \text{where}\\
&m_{i,k}(p) = \begin{cases} \frac{p-t_i}{t_{i+k}-t_i}, & t_{i+k} \neq t_i \\ 0, & \text{otherwise} \end{cases}\\
&B_{i,0}(p) = \begin{cases} 1, & t_{i+k} \leq p < t_{i+1} \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{1}
$$

$B_{i,k}$ is called B-spline basis function of degree k, where i goes from 0 to the number of knots. t_i is the (parametric) coordinate of the ith knot, and p denotes the parameter, the variable of a spline equation.

Each recursion of this function is a separated function on its own, but they only have nonzero values between certain knots. Therefore one can look at knots as limits, thresholds between two basis functions.

Let us look at now some basis functions in Figure 2 to understand knots better. A basis function of degree 0 is a simple indicator function that is 1 between two consecutive knots and 0 elsewhere. These two knots are defined by the first index of the basis function. $B_{0,0}$ is nonzero between the first and second knot, $B_{1,0}$ is nonzero between the second and third knot, and so on and so on. $B_{3,0}$ - that is shown in Figure 2 - is nonzero between the fourth and fifth knot. The degree of a basis function signals the spanning of the nonzero interval. Degree 0 spans across two adjacent knots, i.e. one "knot distance".

Degree 1 spans across three adjacent knots, i.e. two "knot distance", meanwhile degree 3 does over five knots - as also shown in Figure 2. The higher the degree, the smoother the basis function is.

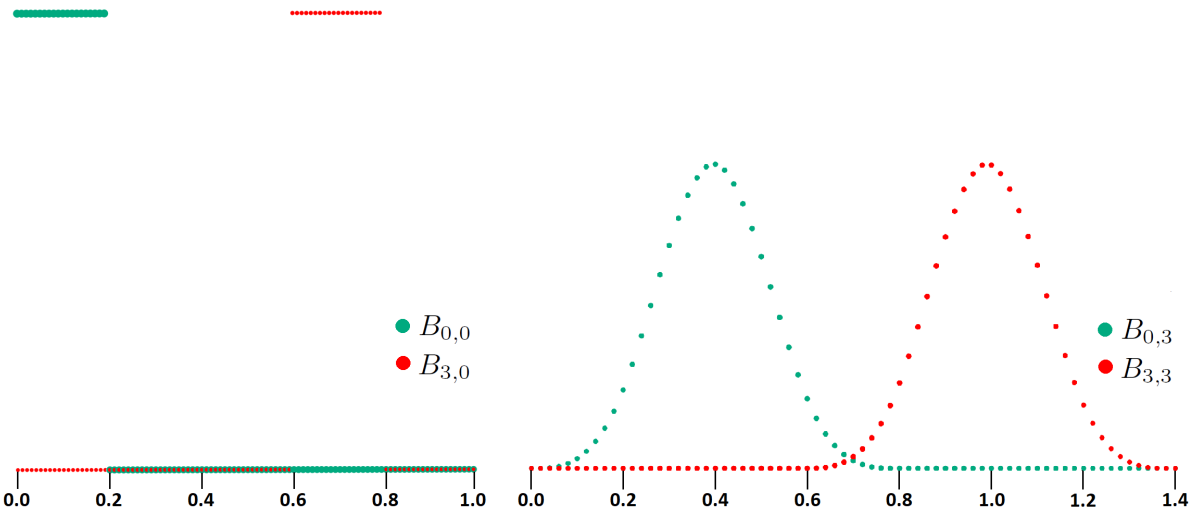


Figure 2: Two-two basis functions of degree 0 and 3, respectively. Indicated points on horizontal line represent the defined knots for these basis functions.

$B_{i,3}$ is more complicated to understand why it looks the way it does. Just by looking at Equation (1) it is not clear, since the recursion is already on a higher level. $B_{i,3}$ consist of several previous $B_{i,k}$. An illustrative explanation can be found on *B-spline basis functions* (2001)[1].

Although the code will not crash, mathematically speaking knots are supposed to follow each other on a nondecreasing order in a *knot vector* (*B-splines*, n.d.). A knot vector contains knots that can be arbitrary, but meaningful knots do exist, and they are advised to be used. What is considered meaningful is up to the individual, the situation, and circumstances. For easier programming[2], knot vector is represented between 0 and 1(*B-spline curve*, 2009) for all upcoming examples[3].

Depending on the differences of adjacent knots, *uniform, open uniform, and non-uniform* knot vectors are distinguished. The difference in a uniform knot vector is constant, in a non-uniform knot vector it is arbitrary. An open uniform knot vector is similar to a uniform one, but the first and last k knots (where k = degree) are equal (*B-splines*, n.d.) (*B-spline basis functions*, 2001).

3.2 B-spline curve

A B-spline is curve is technically a combination of the aforementioned B-spline basis functions that is defined by control points, knots, and degree. Its function is shown in Equation (2)[4].

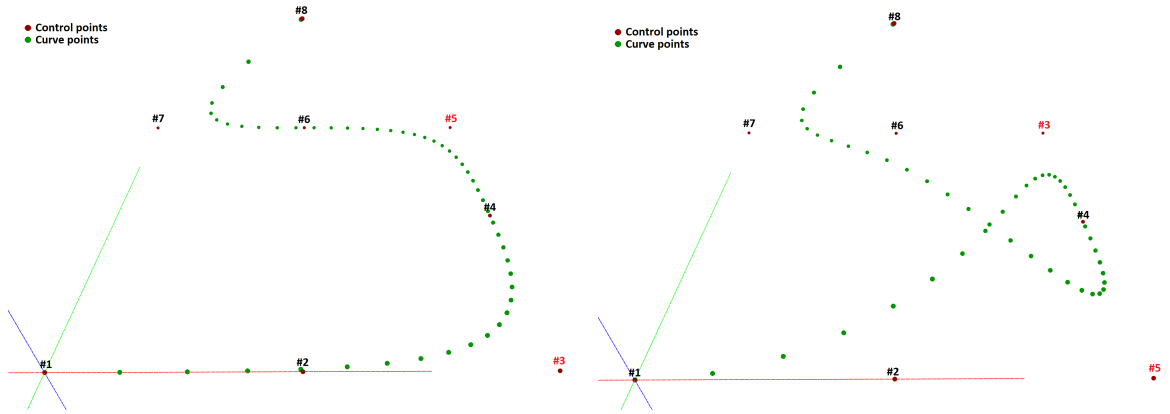
[1]Further illustration and explanation can be found on *B-spline curve* (2009)
[2]and for numerical accuracy
[3]except Figure 2
[4]Assuming each control point has weight = 1

$$C_{k,\mathbf{t}}(p) = \sum_{i}^{n} R_i B_{i,k}(p), \tag{2}$$

where k represents the degree, $\boldsymbol{t}$ is the knot vector, R_i is the ith control point's 3D coordinate, and n is total number of control points.

If the reader is familiar with Fourier or Taylor series, fundamentally they are similar to a B-spline curve equation. In comparison, the basis functions have the role of cosines and sines from Fourier series. They serve the *basis* which have to be multiplied by some "weight"[5]. In a way, a B-spline curve equation is a series of B-splines; in other words, an expansion, but unlike Fourier or Taylor series, the sum is never infinite.

Equation (2) suggests that the order of the control points on a curve affects the end result. Figure 3 supports that the shape of a B-spline curve is indeed different with swapped control points.



(a) (b)

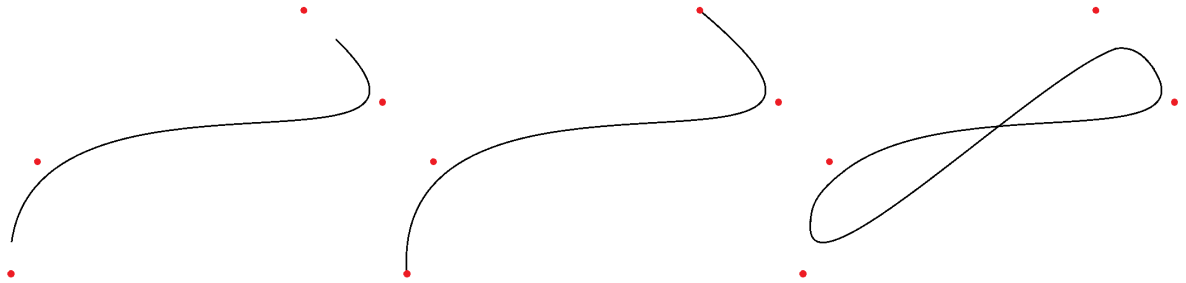
Figure 3: B-spline curves with the same eight control points with different order in function of degree 3.

If a control point is in third place, $B_{2,k}$ is used, and in fifth place $B_{4,k}$ is associated with it. That means that different knots have impact on the control points, so the third and fifth control points' spanning areas are switched.

3.2.1 Knot vector

When a B-spline curve is defined, a knot vector is also initialised with it. Depending on the type of knot vector, there are three categories of a B-spline curve: *open, clamped*, and *closed* (Ching-Kuang, n.d.). Open and closed B-splines will be omitted in this essay, only clamped curves will be discussed. However, even clamped curves are open in the general sense - such as shown in Figure 4.

[5]The "weight" here refers to the control point.



(a) Open curve. (b) Clamped curve. (c) Closed curve.

Figure 4: Different B-spline curve types.

Clamped curve is probably the most intuitive among the three. It starts and ends in a control point, and unless those two controls points are the same, the curve does not circle back to the initial point. Those two control points serve as a sort of guide, so it is easier to imagine the result of a curve even without seeing it.

Now let us assume that the associated knot vector is open uniform, and has a range of [0,1]. If degree = 3 then this looks like (3).

$$\mathbf{t} = (0, 0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1) \tag{3}$$

The more control points are introduced, the more knots are required. In this specific example in (3), there are eight control points - similarly to Figure 3. In case of $k = 3$ degree, there are $k+1$ 0s, $k+1$ 1s, and $n - (k + 1)$ intermediate equidistant knots (*B-splines*, n.d.). That means that $n \geq$ k+1 always.

Let us look at this from the control points' point of view. With $k = 3$, basis functions span across five knots. The first control point therefore also spans across five knots, namely, the first five.

$$\mathbf{t}_1 = (0, 0, 0, 0, 0.2) \longleftrightarrow R_1 \tag{4}$$

That means that the first control point has an impact on the shape until the first nonnull knot. When a parameter ≥ 0.2 is substituted, $B_{1,3}$ will be 0, so R_1 will not affect the curve. Simply, when the order of control points is set in case of a B-spline curve, the appropriate knots are also designated.

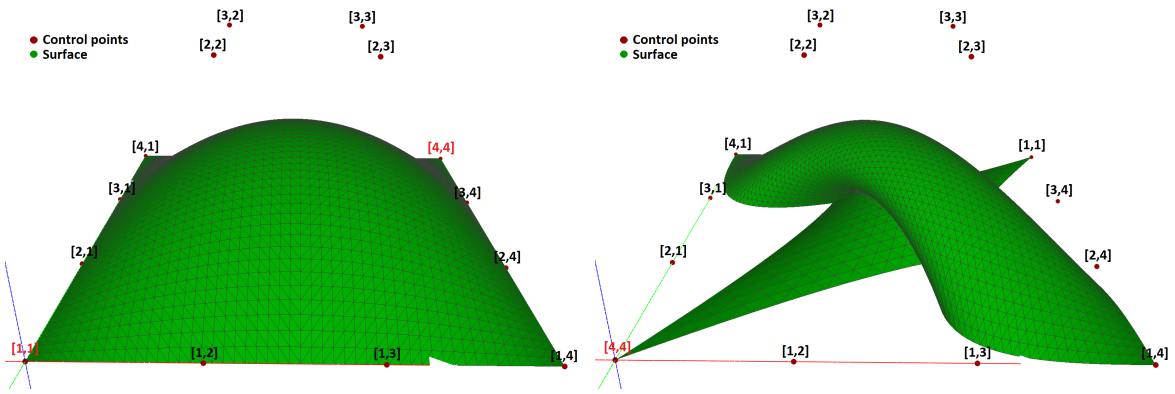
3.3 B-spline surface

B-spline surfaces are very similar to B-spline curves, but their parameter is two dimensional.

$$S_{k,\mathbf{t}}(p_1, p_2) = \sum_{i}^{n} \sum_{j}^{m} B_{i,k}(p_1) B_{j,l}(p_2) R_{i,j} \tag{5}$$

In Equation (5) (*B-spline*, 2022) weight is omitted again, so all control points have weight = 1. In this essay, $l = k$, but mathematically speaking it is not necessarily. This decision was purely arbitrary to simplify the mathematics without losing much generalisation. It also makes sense to have basis functions of same degree in both parametric dimensions.

Let us look at an example that is shown in Figure 5.



(a) (b)

Figure 5: B-spline surfaces with the same 16 control points with different order in function of degree 3. $R_{4,2}$ and $R_{4,3}$ are not visible because they are in the back covered by the surface.

Both surfaces are defined by the same 16 control points, but $R_{1,1}$ and $R_{4,4}$ are swapped, so the order of control points has changed, and with that, the surface has changed, as well. Similarly to a B-spline curve, there is a part on the surface that stayed intact, because not the entire surface is affected by these two control points.

The order in case of a surface is also two dimensional, so instead of a control point vector, a control point matrix is defined - as it is shown in Equation (5) with a double indexed R_{ij}. There are 16 control points and they have been put into a 4×4 control point matrix.

3.3.1 Knot vectors

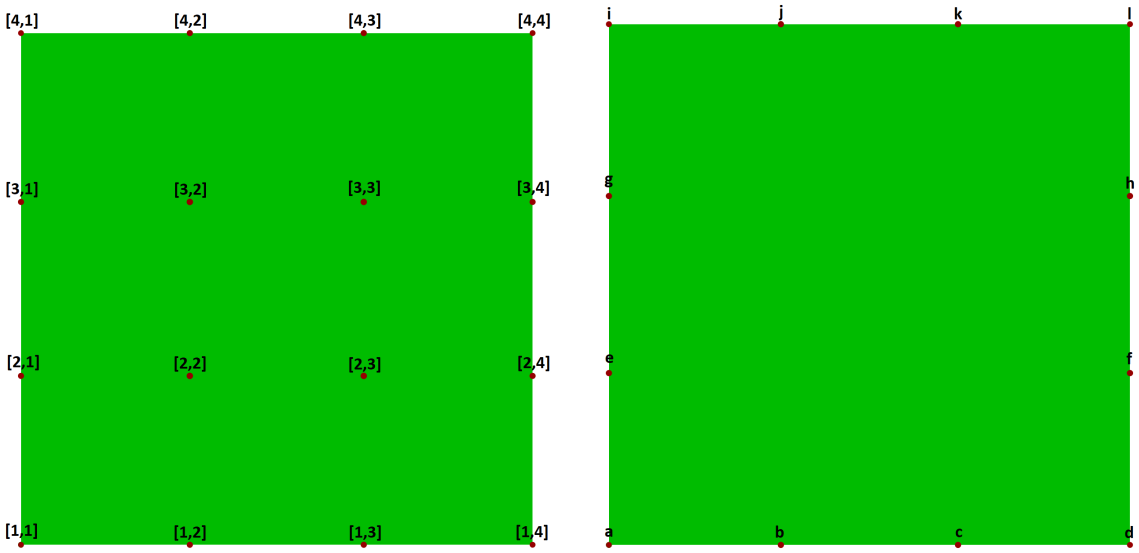
Since a surface is two dimensional[6], there are two knot vectors associated with a B-spline surface. The number of knots are defined by the degree and number of control points in each direction. This surface contains 4 control points in both direction - hence the $4 \times 4 = 16$ points -, and its degree is 3. Therefore the corresponding knot vectors are both as in Equation (6).

$$\mathbf{t} = (0, 0, 0, 0, 1, 1, 1, 1) \tag{6}$$

3.4 T-spline

The fundamental difference between T-spline and B-spline lies in *freedom*. While in a B-spline surface a rigid grid of control points in the right order with their respective knot vectors defines the area, in T-spline it is much more flexible than that. It can, however, describe the same surface with the right settings as shown in Figure 6.

[6] embedded in 3D



(a) B-spline surface with 16 control points.

(b) T-spline surface with 12 control points.

Figure 6: The same surface described by B-spline and T-spline equations.

T-spline surfaces use B-spline basis functions as well, but they are all $B_{0,3}$ with different knot vectors.

$$T(p_1, p_2) = \sum_{i}^{n} B_{0,3,\mathbf{t_{i1}}}(p_1) B_{0,3,\mathbf{t_{i2}}}(p_2) R_i, \tag{7}$$

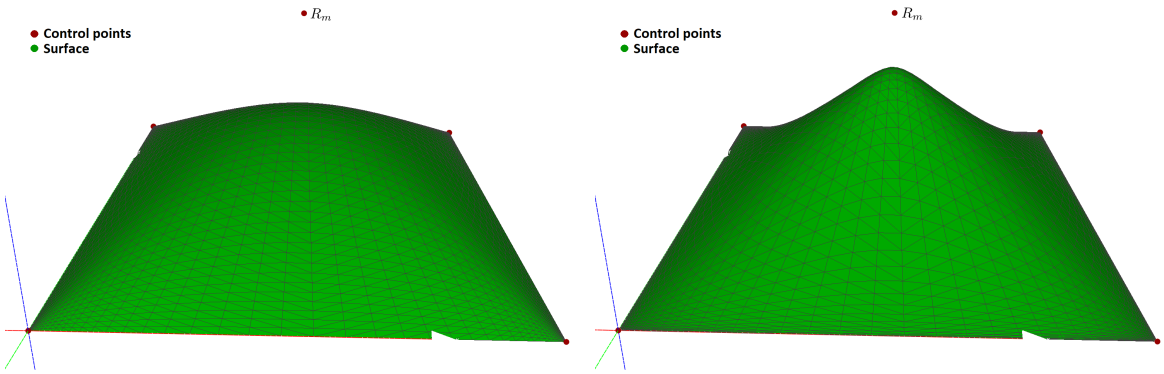
where $\boldsymbol{t}_{i1}$ and $\boldsymbol{t}_{i2}$ indicate the corresponding knots in both directions (Sederberg et al., 2004). These knots are associated to one-one R_i control point. Since $B_{0,3}$ is a basis function of degree 3, $\boldsymbol{t}_{i1}$ and $\boldsymbol{t}_{i2}$ are 5-long knot vectors. It is also worth mentioning, that since the basis functions do not depend on i directly (only through the knot vectors of control points), the order of the control points do not matter. The basis functions are connected to the control point itself and not its place in a sequence of points.

In case of B-spline, the order of control points dictates which knots are associated with certain control points. Here, each control point is separately associated with its knots. A B-spline knot vector is a set of knots designated to define a sequence of subsets of knots, and the order of control points decides which control point receives which subset. Since a T-spline control point has its own "subset"[7], there is no need for a specific order. That is why in Figure 6b each control point was denoted with a letter and not a number, to indicate its orderless nature.

3.4.1 Knot vector

The power of T-spline comes from the individual adjustability of knot vectors. That is clearly illustrated in Figure 7.

[7] It has its own knot vector.



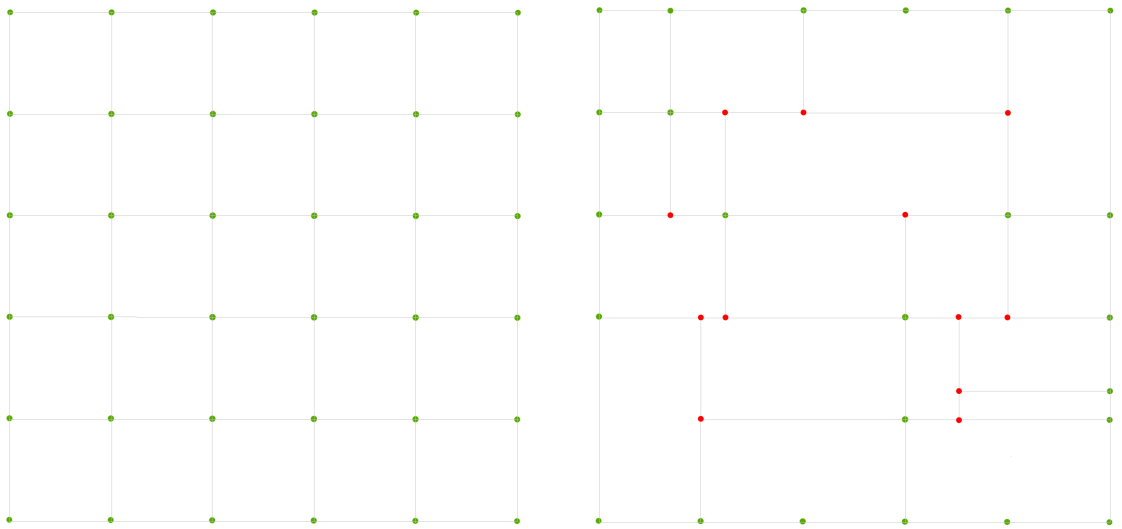
(a) $\boldsymbol{t}_1 = \boldsymbol{t}_2 = (0, 0, 0.5, 1, 1)$ (b) $\boldsymbol{t}_1 = \boldsymbol{t}_2 = (0, 0.4, 0.5, 0.6, 1)$

Figure 7: Surfaces with the same T-spline control points, but with different knot vectors of R_m.

For this kind of change in a B-spline surface, the introduction of more control points are inevitable, but in case of T-spline surface, altering the knot vectors is sufficient. This allows the so-called *local refinement* as opposed to *global refinement* which is the only possible solution with the B-spline approach (Sederberg et al., 2004). This will be illustrated through examples in Section 5 in more detail.

3.4.2 T-mesh

T-mesh is the rectangular grid of parametric space where the knots exist for a specific T-spline surface (Li & Scott, 2014). It is characterised by the so-called *T-junction* which is a T-shaped edge-edge intersection (Sederberg et al., 2003). This is why a T-spline surface is locally refineable.



(a) B-spline knot grid is always a regular mesh without T-junctions or holes.

(b) T-spline knot grid (*T-mesh*) is a generally nonregular mesh with T-junctions in it.

Figure 8: B-spline and T-spline knot meshes of parametric space. The dots represent the knots. Red dots are in a T-junction.

Each control point in a T-spline surface has a central knot in the T-mesh. Their other four knots are not necessarily visible, though. From first knot to last, the full *domain* of a control point can be drawn with a rectangle. This is what is shown in Figure 9.

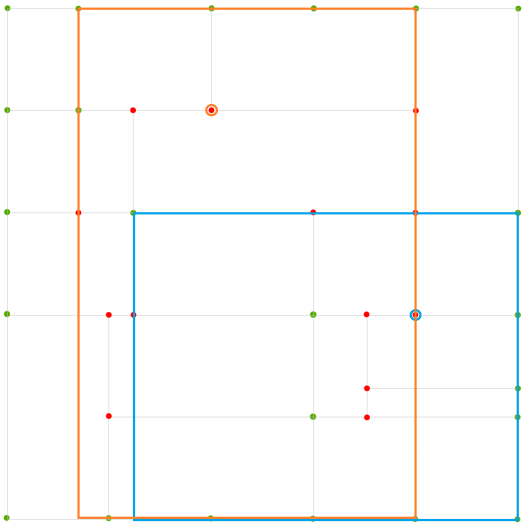


Figure 9: Domains of two (circled) control points. Overlaps indicate common surface area defined by both control points.

There are certain rules, how a T-mesh is formed, but this will be omitted here. This essay focuses on knots solely, and how they shape the surface. T-mesh technicalities are explained in great detail in Sederberg et al. (2003) (2004).

4 Code implementation

The emphasis of this research is not on writing a perfect, sustainable code, however, implementation is inevitable to experiment. The code is written in Go, and is available on `https://github.com/ElleScotZ/project-cs`. This choice was mainly based on my prior knowledge and experience in Go. Also, it is an easily understandable, flexible language. Its syntax is hopefully easy enough for everybody who knows some kind of object oriented programming language.

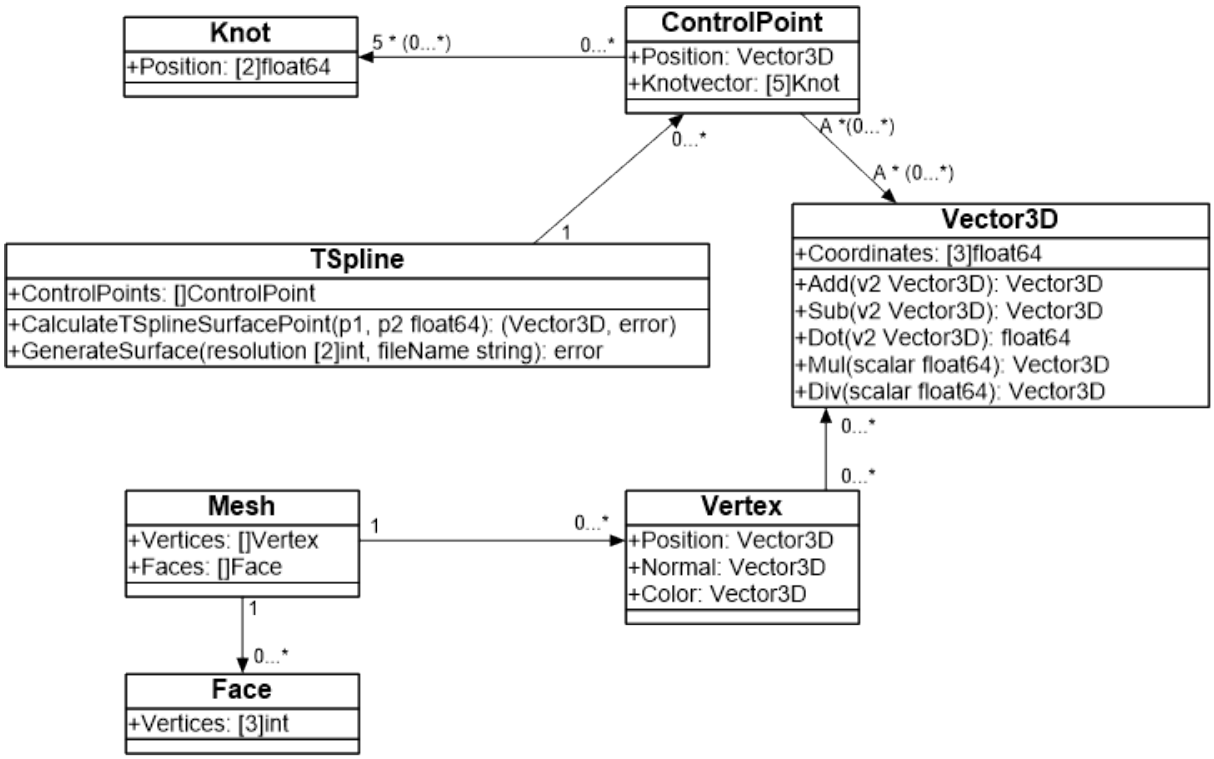


Figure 10: Class diagram of basic code structure. A denotes any non-negative integer.

Class name	Class description	Member variable name	Member variable description
`Vector3D`	3D spatial or other type of vector	`Coordinates`	x, y, z or other meaningful coordinates
`ControlPoint`	A control point in 3D	`Position`	Coordinates of control point
		`Knotvector`	Associated 5-knot long 2D knot vector
`Knot`	A knot in the surface	`Position`	2D parametric position
`TSpline`	T-spline surface	`ControlPoints`	Control points that define the T-spline surface
`Mesh`	Triangle mesh	`Vertices`	Vertices that define surface of mesh
		`Faces`	Triangle (3-vertex sized) faces of mesh
`Vertex`	Vertex of a mesh	`Position`	Coordinates of a vertex
		`Normal`	Normal vector of vertex
		`Color`	RGB colour of vertex
`Face`	3-vertex sized triangle face	`Vertices`	Order number of vertices in `Mesh`

Table 1: Short description of each class and their member variables.

4.1 Code execution

Assuming you have already downloaded Visual Studio Code and Go, and you know how to use basic Git, follow the instructions below on how to run the code.

1. Clone the GitHub repository `https://github.com/ElleScotZ/project-cs`.
2. Open a terminal and in the `root` repository, type in `go run main.go`.
3. Open the created `out` folder with the generated PLY files in it.
4. Open PLY with Print 3D (on Windows), if you do not have anything else[8].

5 T-spline surface variations

It is time to experiment with different knot vectors on the same set of 9 control points to understand

[8] I personally use Meshlab.

their effects on the T-spline surface. The coordinate of these control points are available in `main.go` in `line 1474-1482` and also in Table 2. The order of the control points indicates only their order in the array in the code.

R_1	(-1.00, 0.00, 5.00)
R_2	(1.00, 0.00, 5.00)
R_3	(2.50, 3.50, 0.00)
R_4	(1.00, 5.00, 5.00)
R_5	(-2.50, 3.50, 0.00)
R_6	(-2.50, 1.50, 0.00)
R_7	(1.75, 0.75, 2.50)
R_8	(1.75, 4.25, 2.50)
R_9	(0.00, 2.50, 4.00)

Table 2: The 9 control points that are used to create T-spline surfaces during this essay.

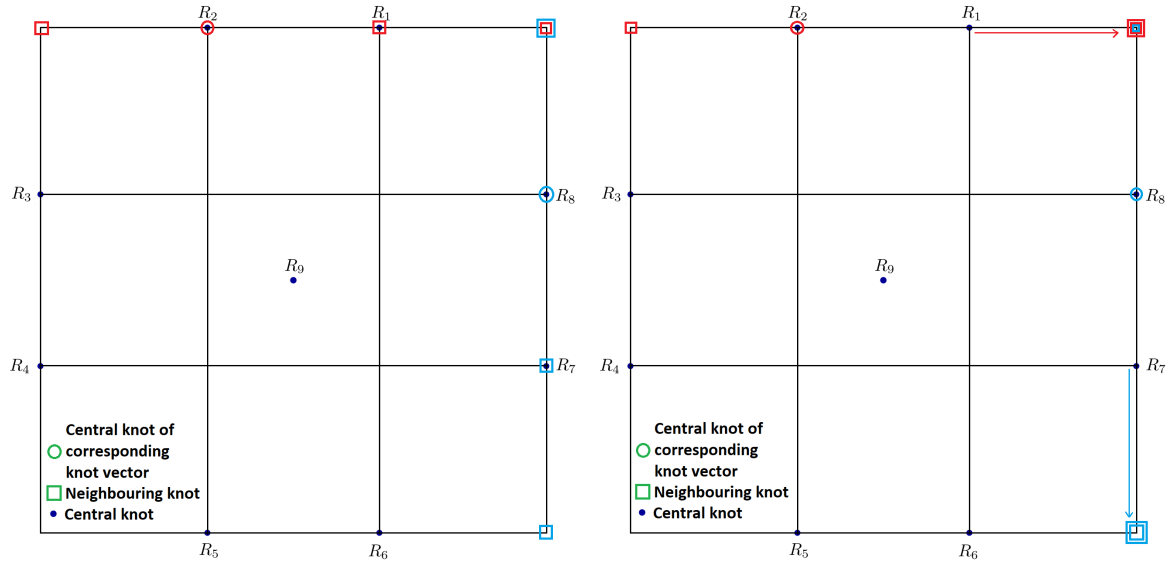
Throughout the examples, $\mathbf{T}$ will denote a generic two-dimensional knot vector that does not belong to any specific control point. The $\mathbf{t}_i$, however, will mean a specific knot vector that belongs to R_i. For further reference, see Table 3.

Symbol	**Description**	**Example**	
$\mathbf{T}$	not control point specific 2D knot vector	([0,0], [0,1], [1,1], [2,1], [2,2])	
$\mathbf{T}_k$	kth knot in $\mathbf{T}$	$k = 2$	[0,1]
$\mathbf{T}_{k,1}$	first dimensional coordinate of $\mathbf{T}_k$	$k = 2$	0
$\mathbf{T}_{k,2}$	second dimensional coordinate of $\mathbf{T}_k$	$k = 2$	1
$\mathbf{t}_i$	2D knot vector of R_i	([0,0], [0,1], [1,1], [2,1], [2,2])	
$\mathbf{t}_{i,k}$	kth knot in $\mathbf{t}_i$	$k = 4$	[2,1]
$\mathbf{t}_{i,k,1}$	first dimensional coordinate of $\mathbf{t}_{i,k}$ $\mathbf{T}_k$	$k = 4$	2
$\mathbf{t}_{i,k,2}$	second dimensional coordinate of $\mathbf{t}_{i,k}$	$k = 4$	1

Table 3: Meaning of different notions.

5.1 $\mathbf{T}_1$ and $\mathbf{T}_3$ shift to $\mathbf{T}_0$ and $\mathbf{T}_4$ along border

Let us first demonstrate the shape shifting of a surface of the aforementioned control points from Table 2 in the following scenario shown in Figure 11, and set in details in `saddle1a()` and `saddle1b()` in `main.go`.

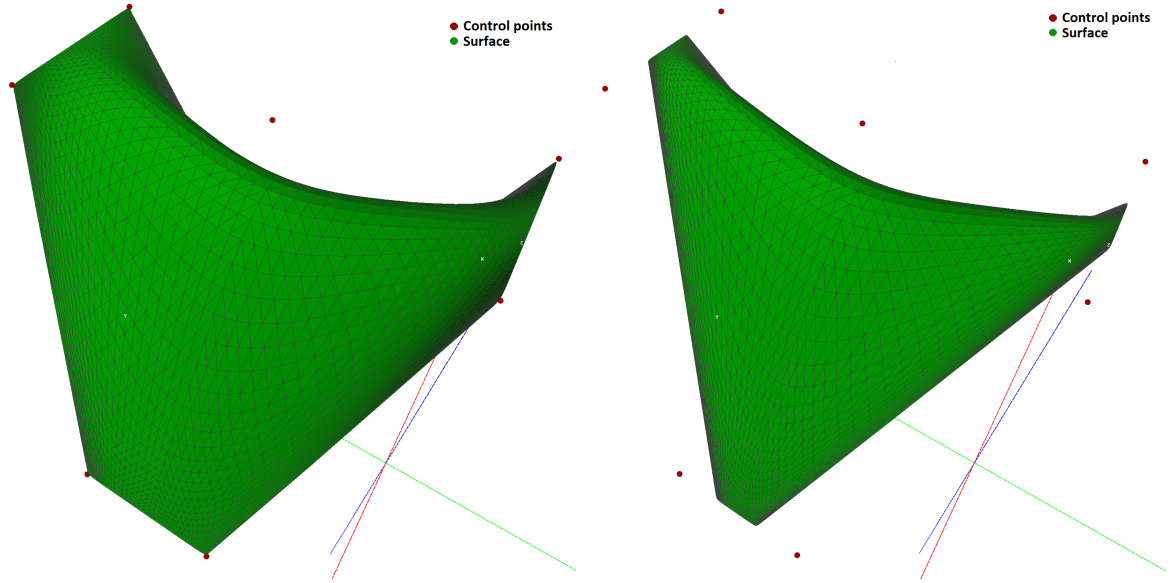


(a) Parametric space of `saddle1a` with two examples (red and blue) of a knot vector distributed across the grid.

(b) Parametric space of `saddle1b` with the same two examples with an emphasis on change of T_1 and T_3 knots with arrows.

Figure 11: Parametric space of each surface. The central knots of control points are represented in a blue point. Circle denotes the central, rectangle does the neighbouring (all, but not central) knots of a knot vector of the corresponding control point.

The knot vector of border control points R_1, ..., R_8 change from `saddle1a` to `saddle1b`. T_1 and T_3 along the border shift, and merge to T_0 and T_4, respectively. This will extend the primary AOE of each border control point along the border. Since $T_0 = T_1$ and $T_3 = T_4$, more $m_i, k(p) = 0$. Parts that were primarily affected by 2 control points now are affected by 3, so the curve of the border shrinks. Why? The more points attract a specific domain (curve), the shorter the curve tries to be. With that, the curve becomes flatter, so the surface shrinks along the border. This is illustrated in Figure 12.

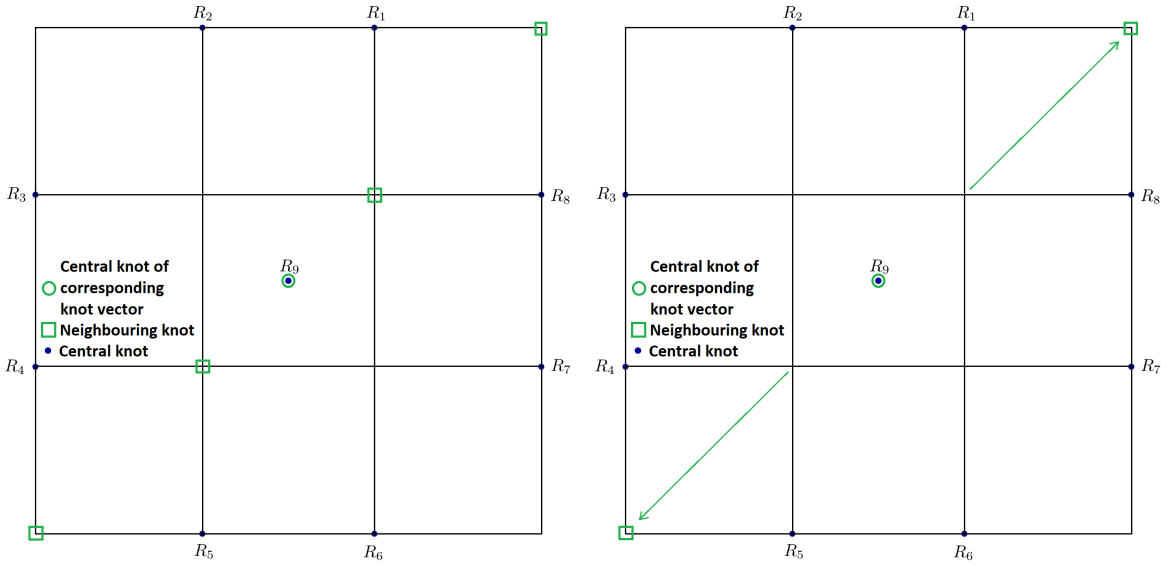


(a) Surface of `saddle1a`. (b) Surface of `saddle1b`.

Figure 12: T-spline surfaces of `saddle1`. Surface shrinkage is visible along the border.

5.2 T_1 and T_3 shift to T_0 and T_4 inside surface

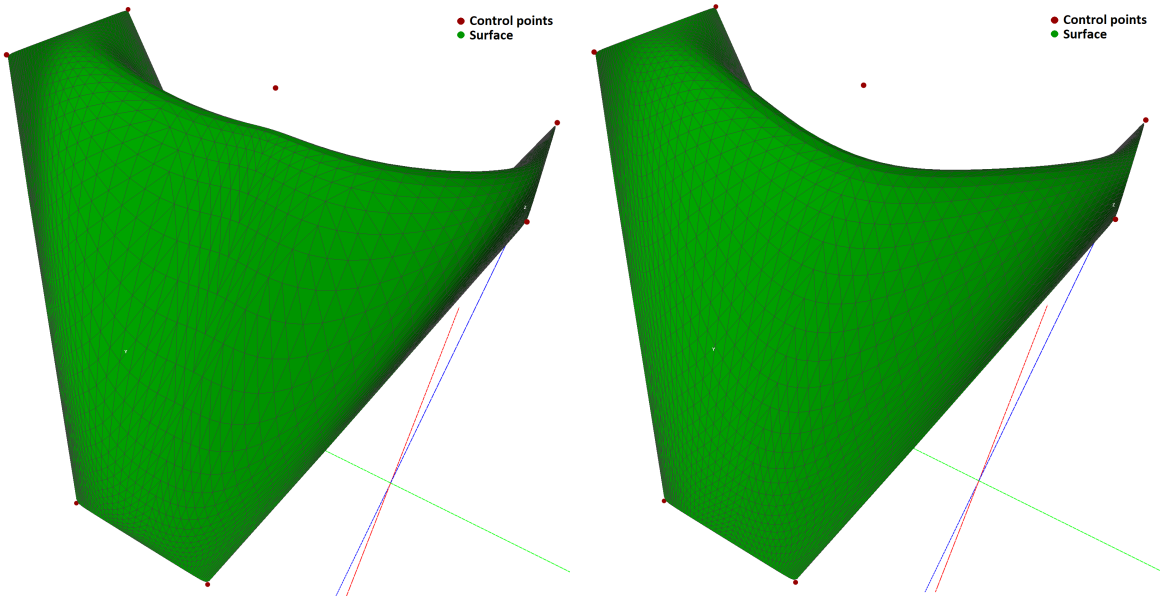
In the second example, the same control points are used, but the knot vector of the middle one R_9 is shifted: $t_{9,1}$ to $t_{9,0}$, and $t_{9,3}$ to $t_{9,4}$.



(a) `saddle2a` (b) `saddle2b`

Figure 13: Parametric space of each surface.

The result is expected to be similar to `saddle1`: the surface will get flatter. The central part of the surface will be less affected by R_9, because the primary AOE will have been extended, so more control points will have a say in the shape of the surface.



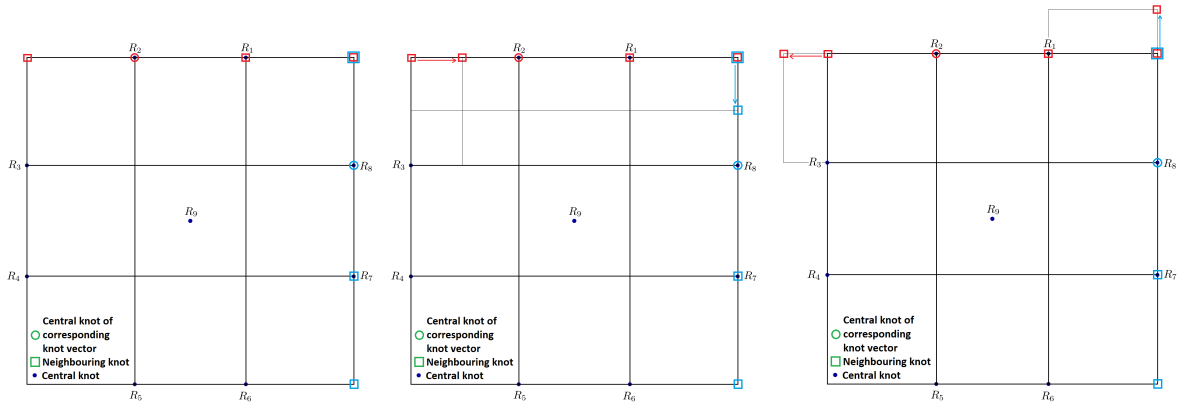
(a) Bump in the middle in `saddle2a`.

(b) Smoother central part in `saddle2b`.

Figure 14: T-spline surfaces of `saddle2`. Flatter saddle shape in the middle after knot shift.

5.3 Knot change along border off the grid

In this section, let us start with `saddle3a` at the following setup: $T_{0,d} = T_{1,d}$ in case of control points closer to the lower end of a border, and $T_{3,d} = T_{4,d}$ in case of control points closer to the higher end of a border, and d represents the appropriate dimension along which the corresponding border lies. So for example: $t_{2,0,1} = t_{2,1,1}$ (red in Figure 15), and $t_{8,3,2} = t_{8,4,2}$ (blue in Figure 15).



(a) Parametric space of `saddle3a` with knot vectors distributed across the grid.

(b) Parametric space of `saddle3b` with change in duplicated knots inwards the border dimension.

(c) Parametric space of `saddle3c` change in duplicated knots outwards the border dimension.

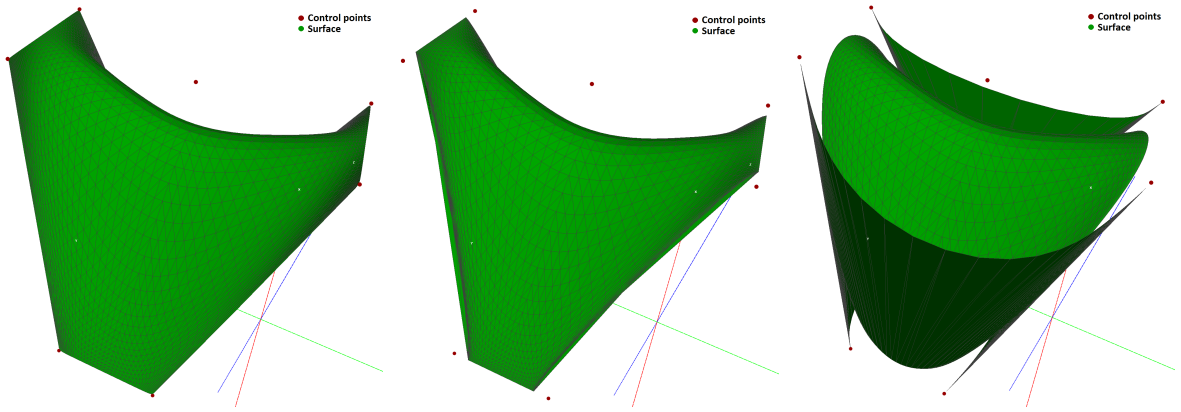
Figure 15: Parametric space of each surface with two examples (red and blue).

In `saddle3a` the setup is the same as in `saddle1a` or `saddle2a`. The are border knot coordinates - as it was mentioned above - that are equal. This will be changed in `saddle3b`. The red and blue examples are as follows:

$$
\begin{aligned}
t_{2,0,1} &= 0 & t_{8,3,2} &= 0.85 \\
t_{2,1,1} &= 0.15 & t_{8,4,2} &= 1
\end{aligned}
\tag{8}
$$

The "problem" with that is there is no central knot associated with 0.15 and 0.85. $t_{2,0} = t_{1,0}$, so the secondary AOE between T_0 and T_1 is the same, having the same effect on the surface without extra control point. That will result in a "cut". This is essentially different from the `saddle1` example due to the lack of junction in the T-mesh. The reader is advised to take a closer look at `saddle1b` and `saddle3b`.

In `saddle3c`, knot vectors leave the [0,1] parametric space, but the surface is still generated on that area only. Therefore, the result is quite strange. I thought it might be helpful to illustrate the importance of right range. In order to get the full picture of how a T-spline surface looks, the variables along the whole parametric surface should be substituted.



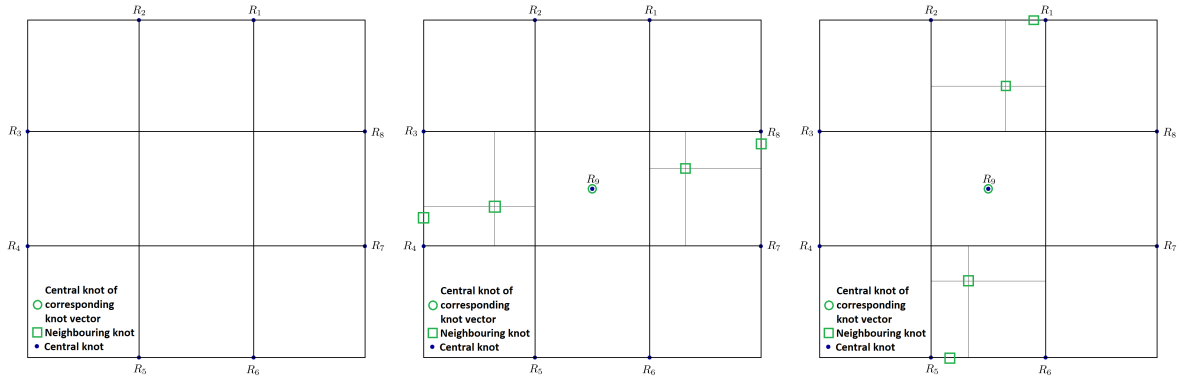
(a) Surface with duplicated knots. (b) Off grid surface. (c) Out of range surface.

Figure 16: T-spline surfaces of `saddle3`.

5.4 Local refinement inside surface

Let us talk about local refinement now. Local refinement means that the surface is *refined*, changed in a local space on the surface. In `saddle4a` only R_1, ..., R_8 control points are used, and R_9 will be added to the surface later. The knot vector of R_9 in `saddle4b` and `saddle4c` is shown in Equation 9.

$$
\begin{aligned}
\texttt{saddle4b}: &\quad t_9 = ([0, 0.4], [0.2, 0.45], [0.5, 0.5], [0.8, 0.55], [1, 0.6]) \\
\texttt{saddle4c}: &\quad t_9 = ([0.4, 0], [0.45, 0.2], [0.5, 0.5], [0.55, 0.8], [0.6, 1])
\end{aligned}
\tag{9}
$$



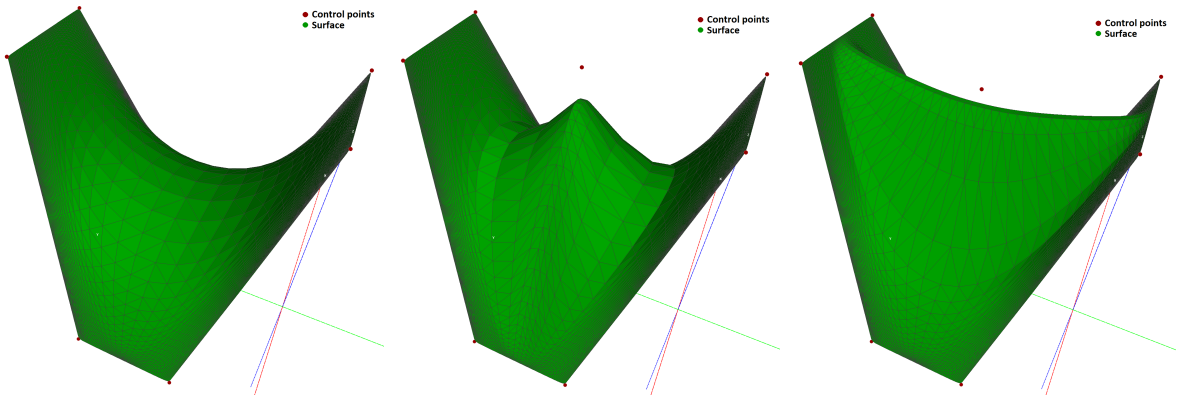
(a) Without local refinement, `saddle4a`.

(b) Horizontally elongated local refinement, `saddle4b`.

(c) Vertically elongated local refinement, `saddle4c`.

Figure 17: Parametric space of `saddle4` surfaces.

In `saddle4b` the bump on the surface is elongated in the horizontal parametric dimension (x-axis in real space), in `saddle4c` it is elongated in the vertical parametric dimension (y-axis in real space). The explanation is simple: when a control point is more restricted in one dimension and less in the other, its AOE will be elongated. It is simply because over the restriction, the respective control point has no affect on the surface, so over that restriction, the surface remains as it was without the additional control point. This is shown in Figure 18.



(a) Surface without R_9.

(b) R_9 with horizontal elongation.

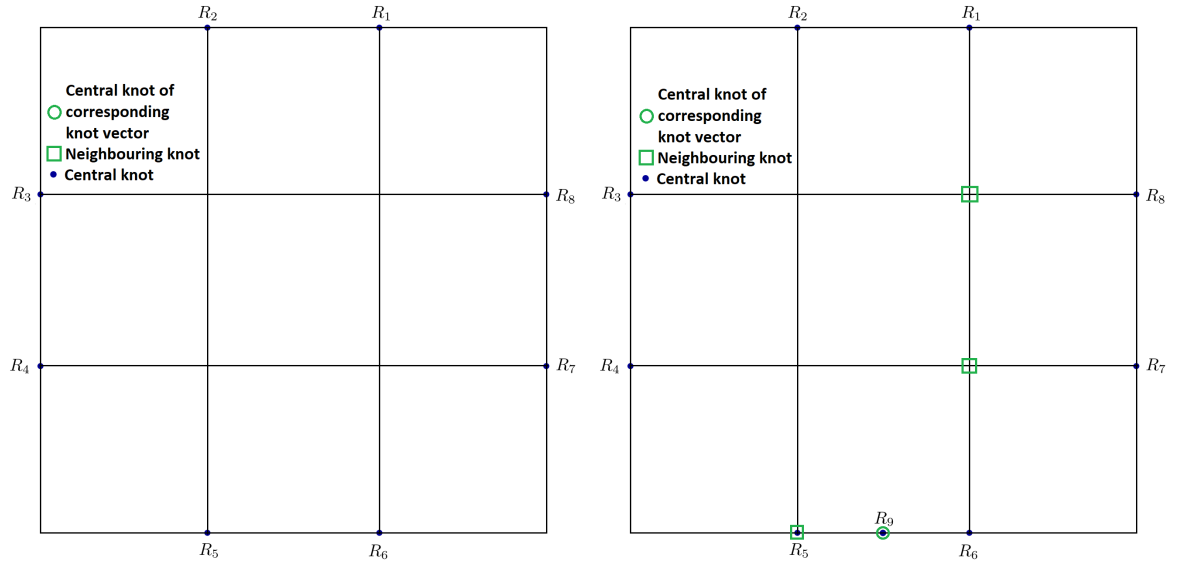
(c) R_9 with vertical elongation.

Figure 18: T-spline surfaces of `saddle4` with and without local refinement.

5.5 Local refinement along border

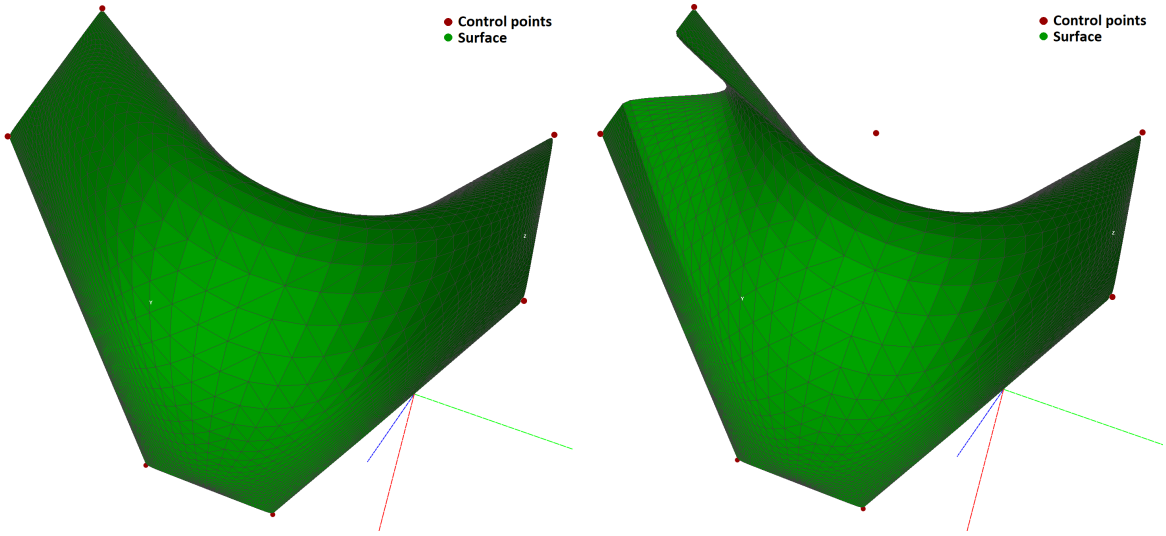
The next example will be similar to the previous one, but in this case, the border will be locally refined. The same R_9 is used with the following knot vector:

$$t_9 = ([0.33, 0], [0.33, 0], [0.5, 0], [0.67, 0.33], [0.67, 0.67]) \tag{10}$$



(a) (b) T.

Figure 19: Saddle5



(a) Without local refinement, `saddle5a`. (b) With local refinement, `saddle5b`.

Figure 20: T-spline surfaces of `saddle5` with and without local refinement along border.

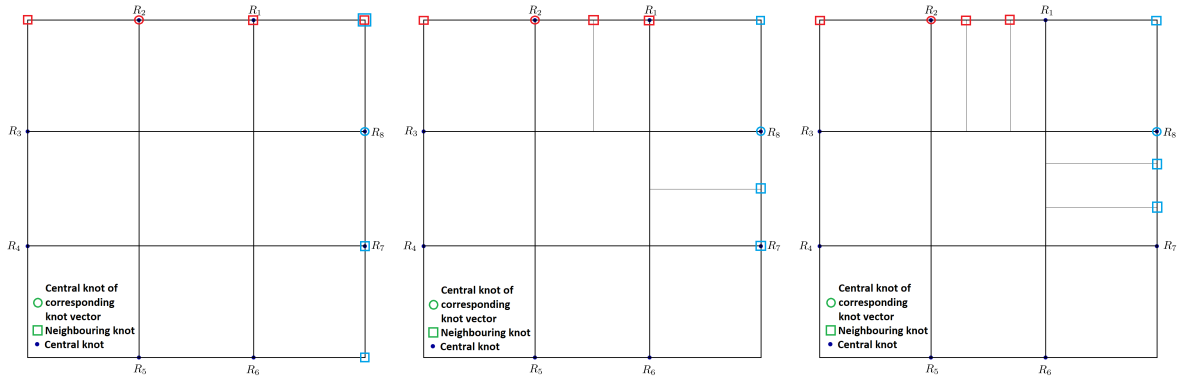
As it was anticipated, the AOE of R_9 is mainly[9] visible along the border.

5.6 $t_{i,k}$ and $t_{i,k+1}$ change

In this next example, let us shift non-duplicated knots of border control points. In Figure 21 one can see this specific change, and in (11) the related knots.

[9]Not entirely, because its knot vector goes until 0.67 in the perpendicular parametric direction.

$$
\begin{aligned}
\texttt{saddle6a}: \quad & t_{2,k.2} = (0, 0, 0.33, 0.67, 1) \\
& t_{8,k.1} = (0, 0.33, 0.67, 1, 1) \\
\texttt{saddle6b}: \quad & t_{2,k.2} = (0, 0, 0.33, 0.50, 0.67) \\
& t_{8,k.1} = (0.33, 0.50, 0.67, 1, 1) \\
\texttt{saddle6c}: \quad & t_{2,k.2} = (0, 0, 0.33, 0.45, 0.55) \\
& t_{8,k.1} = (0.45, 0.55, 0.67, 1, 1)
\end{aligned}
\tag{11}
$$



(a) `saddle6a`: adjacent central knots of adjacent control points along borders.

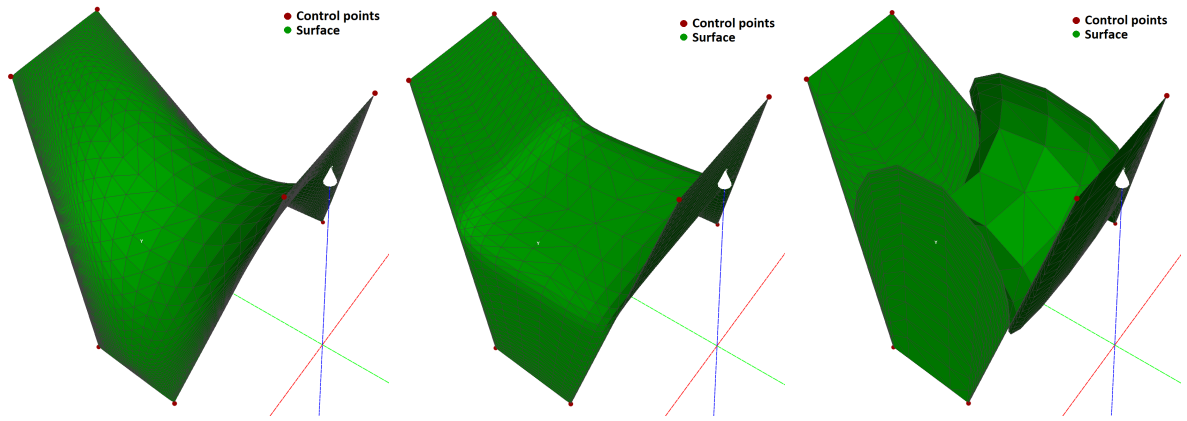
(b) `saddle6b`: overlapping central-adjacent knots of adjacent control points.

(c) `saddle6c`: overlapping secondary domains of adjacent control points.

Figure 21: Parametric space of `saddle6`. Shifts of two knots further from corner of red and blue examples are highlighted.

With each shift, fewer and fewer domains overlap between adjacent control points, until no secondary knot can reach the adjacent control point's central knot. `saddle6a` is the basic setting for R_1, ..., R_8 as before. In `saddle6b` knots are already further from each other. For example $t_{2,4,2} = t_{1,1,2}$, so R_2's secondary domain is R_1's primary domain. However, $t_{2,4,2}$ has no real meaning, no central knot associated with it, so the surface is expected to be less smooth, flatter as it is shown in Figure 22b. The corners are more angular since the AOE of relevant control points do no reach the other end of the border.

Then another new knot is introduced between two adjacent central knots, so there are already two new knots without associated central knots. But these changes occur only along the border, in the border dimension, other remains intact. Therefore, in one dimension, we would still gain a saddle-looking shape, only in the other dimension the shape is really distorted. At this point, it gets harder and harder to predict what the surface will look like. On the other hand, it is also very questionable whether or not it makes sense to set the knots to values that are not associated with any other central knots. The reader is encouraged to try even stranger setups where adjacent control points share even fewer or no knots in the parametric space.

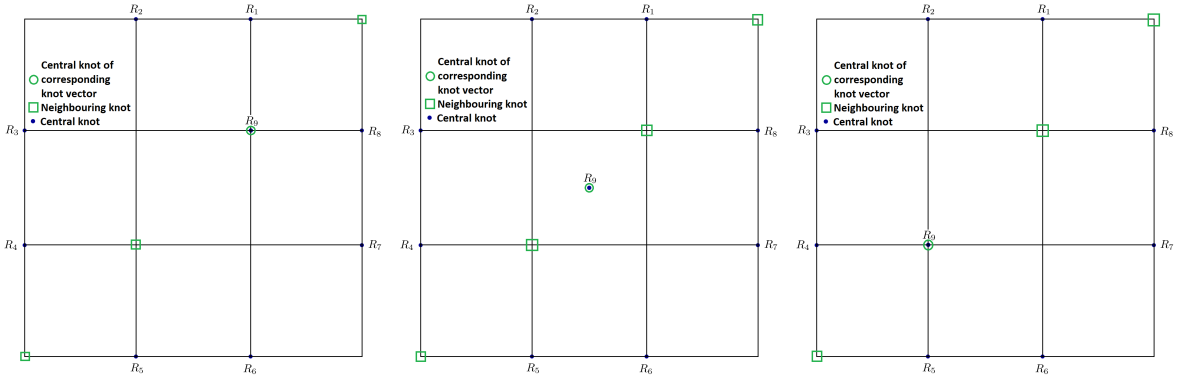


(a) Adjacent central knots. (b) Adjacent primary knots. (c) Adjacent secondary knots.

Figure 22: T-spline surfaces of `saddle6`.

5.7 T_i shift inside surface

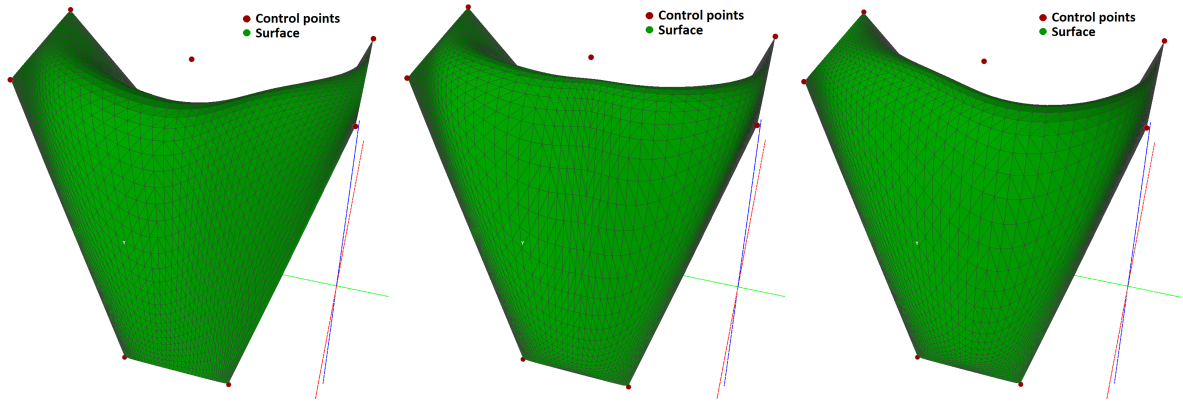
Let us look at the following setup: $t_{9,1}$ and $t_{9,5}$ are in the two corners of the parametric grid. Then $t_{9,2}$, $t_{9,3}$, and $t_{9,4}$ will be shifted diagonally between those two corners as it is shown in Figure 23.



(a) Shift close to upper corner in `saddle7a`. (b) Knots are in the middle in `saddle7b`. (c) Shift close to lower corner in `saddle7c`.

Figure 23: Parametric grids of `saddle7`.

This is in fact a local refinement in three different scenarios. The additional bump on the saddle is defined by the knots of R_9. Without surprise, `saddle7a` would result in a bump closer to R_1 and R_8, meanwhile in `saddle7c` the bump is closer to R_4 and R_5 as shown in Figure 24. `saddle7b` is an example how to place a local bump on the surface in the middle.

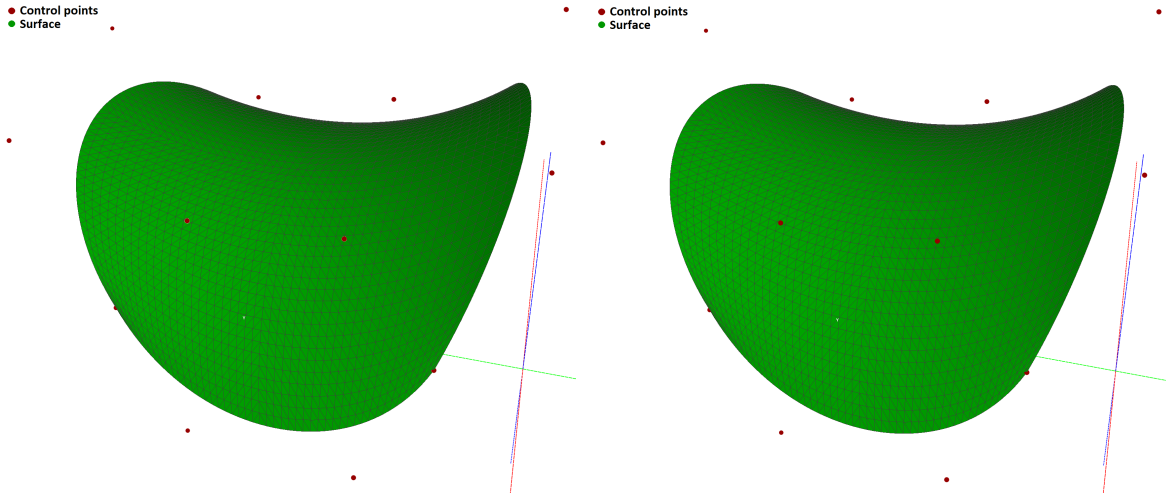


(a) Upper bump in `saddle7a`. (b) Middle bump in `saddle7b`. (c) Lower bump in `saddle7c`.

Figure 24: Saddle7

5.8 T-spline limit goes to B-spline

Last, but not least, let us demonstrate how T-spline goes to B-spline in the limit. In Figure 25, a Pringle is created out of 16 control points that are available in `main.go/main()`.



(a) T-spline Pringle surface. (b) B-spline Pringle surface.

Figure 25: Pringle surfaces made with different splines out of the same control points and knot vectors.

Every B-spline surface can be transformed to a T-spline surface without losing any information about the surface. Using the same knot vectors, same control points, the exact same surface can be created. That is why T-splines are extraordinary, because they are capable of creating the same surface, but with extra freedom of choosing costumised knot vectors, or adding local refinement.

5.8.1 T-spline or B-spline?

So if both can achieve the same, which one should we use? A big advantage of T-spline is that one can achieve the desired surface with fewer control points. That can save some computational power, which

might be a bigger issue with B-splines. On the other hand, its high flexibility is also problematic when it comes to automation. In each `saddle` example, I have set up the knot vectors manually. B-spline does not require manual setup, it pairs knot vectors and control points automatically after the initial setup. That is why B-spline surfaces might be more appealing.

Local refinement cannot be achieved by B-splines. If an extra little bump is needed on a B-spline saddle, more then just one control point has to be introduced, so it would be a way more tedious work to gain the right surface. That is probably the biggest pro for T-spline surfaces.

There is a technique that is used to avoid the high level of complexity of T-splines, namely, the Catmull-Clark subdivision (*Catmull-Clark subdivision surface*, 2022). It is used in 3D modeling softwares, such as *Blender* (*Catmull-Clark Subdivision Surfaces*, 2003), and it uses T-spline control points without the hassle of knot vector adjustments.

6 Final thoughts

Another interesting thing that this essay did not cover is the vertex distribution on a T-spline surface. A vigilant eye might have noticed that even though equidistant values were substituted in the T-spline equations, vertices were usually not equally distributed on the surface. This is also related to the knot vectors, but due to lack of space, this essay has omitted how they could affect surface resolution.

With prior knowledge and some practice behind, it is possible to predict how the surface would react to different knot vectors and local refinements. The extent of a local refinement is absolutely up to the person, mathematically there are very few restrictions that have to be followed, even those can be broken technically in a code without crashing it, so it is important to pay attention to them.

This high degree of freedom of T-spline is also dangerous. If one follows the rules of B-spline, it is quite hard to go wrong, but once knot vectors are adjusted freely, it can be challenging to keep up with the transformed surface. If one wants to create a T-spline surface from scratch, my suggestion would be to start from a B-spline structure, and reduce the unnecessary control points and change the knot vectors later, once a basis form is already acceptable.

7 References

Applications of 3D printing. (2017, March). Wikipedia. `https://en.wikipedia.org/wiki/Applications_of_3D_printing`

Kothari, D. P., Awari, G., Shrimankar, D., & Bhende, A. (2019). *Mathematics for Computer Graphics and Game Programming: A Self-Teaching Introduction.* Mercury Learning and Information.

Gallier, J. (2018). *Curves and Surfaces In Geometric Modeling: Theory And Algorithms.* Morgan

Kaufmann.

B-spline. (2022, January). Wikipedia. https://en.wikipedia.org/wiki/B-spline

B-spline basis functions. (2001, August). Ibiblio. https://www.ibiblio.org/e-notes/Splines/basis.html

B-splines. (n. d.). University of Cambridge, Department of Computer Science and Technology. https://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node4.html

B-spline curve. (2009, December). Massachusetts Institute of Technology. https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node17.html

Ching-Kuang, S. (n.d.) *B-spline Curves: Definition*. Michigan Technological University, Information Technology https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve.html

Sederberg, T. W., Cardon, D. L., Finnigan, G. T., North, N. S., Zheng, J., & Lyche, T. (2004). *T-spline simplification and local refinement*. ACM Transactions on Graphics, 23(3), 276–283. https://doi.org/10.1145/1015706.1015715

Li, X., & Scott, M. A. (2014). *Analysis-suitable T-splines: Characterization, refineability, and approximation*. Mathematical Models and Methods in Applied Sciences, 24(06), 1141–1164. https://doi.org/10.1142/s0218202513500796

Sederberg, T. W., Zheng, J., Bakenov, A., & Nasri, A. (2003). *T-splines and T-NURCCs*. ACM Transactions on Graphics, 22(3), 477–484. https://doi.org/10.1145/882262.882295

Catmull-Clark subdivision surface. (2022, July). Wikipedia. https://en.wikipedia.org/wiki/Catmull\OT1\textendashClark_subdivision_surface

Catmull-Clark Subdivision Surfaces. (2003, September). Blender. https://download.blender.org/documentation/html/x2615.html

8 Index

Area of effect	(AOE) Surface area which a control point affects.
Central knot	T_3
Domain	Parametric space area of a control points.
Primary area of effect	Surface area which a control point affects in the range of T_2-T_3 or T_3-T_4.
Primary domain	Parametric space area of a control points in the range of T_2-T_3 or T_3-T_4.
Primary knot	T_1 or T_4
Secondary area of effect	Surface area which a control point affects in the range of T_1-T_2 or T_4-T_5.
Secondary domain	Parametric space area of a control points in the range of T_1-T_2 or T_4-T_5.
Secondary knot	T_0 or T_5