NURBS

Justin Unger

May 29, 2022, Mosbach

Contents

1 Einleitung

Produkte haben eine Vielzahl von Phasen, welche sie durchlaufen, bis sie in die Produktion kommen. Besonders bei Autos ist das Design sehr wichtig. Um so komplexe Formen zu generieren, gibt es verschiedene Möglichkeiten. Wenn diese digital modelliert werden, dann müssen diese im Hintergrund mathematisch dargestellt werden.

In dieser Ausarbeitung wird ein Standardverfahren für diese Mathematische Darstellung vorgestellt: *NURBS*. NURBS steht für Non-uniform rational B-Splines. Was diese einzelnen Bestandteile der Abkürzung mit dem Modell zu tun haben, wird im Laufe der Arbeit vorgestellt.

Bevor dies geschehen kann, müssen explizit zwei Vorbetrachtungen zu rate gezogen werden: die *Bézier-Kurven* und die *Tensor Produkt Oberflächen*. Dahingehend wird auch der Rationale Aspekt erläutert.

Anschließend werden im Abschnitt *3 Definition NURBS* die weiteren Bestandteile von NURBS dargestellt. Dabei werden auch B-Splines für sich selbst vorgestellt.

Im Kapitel *4 Modellierung* wird beispielhaft gezeigt, wie in einem Programm mit den NURBS gearbeitet werden kann. Diese Kapitel ist explizit nicht mathematisch. Es soll die Anwendung für Designer und nicht für Mathematiker darstellen.

Diese Arbeit basiert auf den *NURBS Book* von Les Piegl und Wayne Tille. Wenn eine andere Quelle verwendet wird, ist diese Referenziert. Ansonsten ist dieses Buch die Hauptquelle, bei welcher alle (nicht anders markierte Formeln) übernommen wurden.

2 Vorbetrachtung

2.1 Bézier-Kurven

Pierre Etienne Bézier arbeitete bei Renault. Dabei hat er eine mathematische Beschreibung für beliebige Kurven gefunden, welche sich anhand von Kontrollpunkten definieren lassen. Dabei muss die Kurve (Bézier-Kurve) nicht durch jeden Kontrollpunkt verlaufen. Dennoch wird die Form durch die Kontrollpunkte definiert.

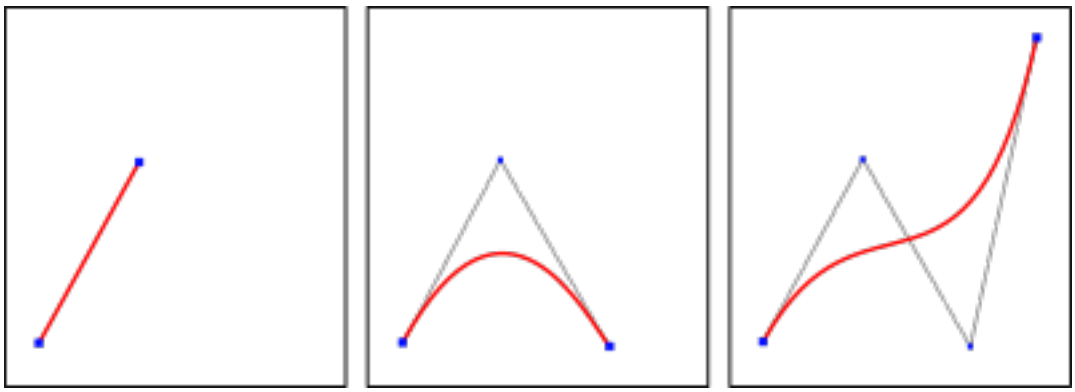

Figure 1: Bezier-Kurven vom 1. bis 3. Grad

Eine Bézier-Kurve hat immer einen bestimmten Grad n. Zudem hat sie $n+1$ Kontrollpunkte P.

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) * P_i \tag{1}$$

$$B_{i,n}(u) = \frac{n!}{i! * (n-i)!} * u^i * (1-u)^{n-i} \tag{2}$$

$$0 \leq u \leq 1 \tag{3}$$

Mit diesen Gleichungen lassen sich viele zwei dimensionale Formen darstellen. Jedoch nicht alle. Bézier-Kurven sind eine andere Darstellung für Polynome. Die "Grunddarstellung" von Polynomen sind:

$$x(u) = a_0 * u^0 + a_1 * u^1 + a_2 * u^2 + ... + a_n * u^n \tag{4}$$

$$x(u) = b_0 * u^0 + b_1 * u^1 + b_2 * u^2 + ... + b_n * u^n \tag{5}$$

Diese können jedoch keine Kreise, Ellipsen oder Hyperbeln exakt darstellen. Dazu sind gebrochen rationale Polynome (Englisch: rational polynomes) notwendig.

$$x(u) = \frac{X(u)}{W(u)} \tag{6}$$

$$y(u) = \frac{Y(u)}{W(u)} \tag{7}$$

Wenn Bézier-Kurven gebrochen rationale Polynome darstellen sollen, kann folgende die Gleichung verwendet werden. Die Funktion $B(u)$ ist die gleiche wie bei rationalen Bézier-Kurven.

$$C(u) = \sum_{i=0}^{n} R_{i,n}(u) * u_i * P_i \tag{8}$$

$$R_{i,n}(u) = \frac{B_{i,n}(u) * w_i}{\sum_{j=0}^{n} B_{j,n}(u) * w_j} \tag{9}$$

$$B_{i,n}(u) = \frac{n!}{i! * (n-i)!} * u^i * (1-u)^{n-i} \tag{10}$$

Bei $C(u)$ ist der Aufbau auch gleich, außer das $B_{i,n}(u)$ durch $R_{i,n}(u)$ ersetzt wurde. Die Funktion $R_{i,n}(u)$ ist der Bruch der Funktion. Dieser stellt das Verhältnis zwischen dem aktuellen Punkt und der Summe aller

Punkte dar. Der Parameter w_i stellt die Gewichtung dar. Wenn die Abbildung 1 erneut betrachtet, kann die Gewichtung als Einflussstärke eines Kontrollpunktes verstanden werden.

Da die Gleichung für Bézier-Kurven immer identisch ist, kann jede mögliche Kurve durch zwei Listen darstellt werden. Eine Liste mit den Kontrollpunkten und eine Liste mit den Gewichtungen für die Punkte. Da jedoch dann immer beachtet werden muss, das beide Listen gleich groß sind, wird eine weitere Darstellungsmöglichkeit für die Punkte gewählt. Dabei werden die Punkte um eine Dimension erweitert. Zwei dimensionale Punkte werden zu drei dimensionalen Punkten.

$$P = (x, y) \tag{11}$$

$$P^w = (w * x, w * y, w) \tag{12}$$

$$H\{P^w\} = P \tag{13}$$

Dadurch ist die Gewichtung direkt mit im Punkt mit angegeben. Diese Umwandlung nennt sich Homogene Koordinaten. Die Geometrische Bedeutung ist das Zuweisen (Mappen) einer $n + 1$-dimensionalen Kurve auf eine n-dimensionale Kurve. Grafisch dargestellt ist dieses Mappen in einem $n = 2$ dimensionalen Beispiel in Figur 2. Die blaue Kurve ist die Kurve in einem n+1-Raum welche durch die Punkte P^w definiert werden. Die rote Kurve hingegen ist die Kurve, welche durch das Mappen generiert wird. Diese wird durch die Punkte P und die Gewichtungen w definiert.

Bézier-Kurven können an diese Punkte entsprechend angepasst werden, sodass sie nicht erst zurückgemapped werden müssen, sondern direkt die gewichteten Punkte P^w verwenden können.

$$C^w(u) = \sum_{i=0}^{n} B_{i,n}(u) * P_i^w \tag{14}$$

Diese Formel hat die exakt gleiche Bedeutung wie die gebrochen rationalen Bézier-Kurven. Der Beweis ist im Anhang vermerkt.

2.2 Tensor Produkt Oberflächen

Wie der Name bereits andeutet, sind Bézier-Kurven nur Kurven. Damit lassen sich noch keine Oberflächen oder Volumen-Körper darstellen. Dies ist mit *Tensor Produkt Oberflächen* (im Weiteren TPO) möglich Diese können

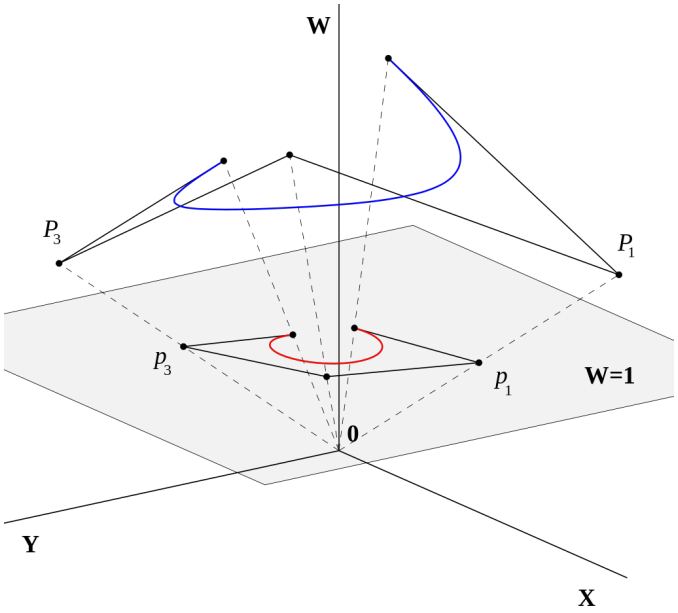

Figure 2: Mapping bei homogenen Koordinaten

sich wie Bézier-Kurven in zwei Richtungen vorgestellt werden. Diese TPOs werden durch das Symbol $S(u, v)$ dargestellt. Dabei sind u und v Elemente einer Fläche im Koordinaten-System. Über dieser Fläche wird die TBO abgebildet.

$$S(u,v) = (x(u,v), y(u,v), z(u,v)) = \sum_{i=0}^{n} \sum_{j=0}^{m} f_i(u) * g_j(v) * b_{i,j} \qquad (15)$$

Diese Gleichung kann auch als Matrix-Multiplikation verstanden werden. Statt mit i und j durch zu iterieren, kann eine nXm ermittelt werden, welche in das Koordinaten-System eingetragen werden kann. Dabei ist $[f_i(u)]^T$ ein Reihenvektor der Größe $(1)X(n+1)$. Dagegen ist $[g_j(u)]$ ein Spaltenvektor der Größe $(m+1)X(1)$. Die Punkte $[b_{i,j}]$ sind bereits in einer Matrix der Größe $(n+1)X(m+1)$ angeordnet. Somit verändert sich die Gleichung wie folgt:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} f_i(u) * g_j(v) * b_{i,j} = [f_i(u)]^T * [b_{i,j}] * [g_j(u)] \tag{16}$$

Wenn diese allgemeine Form der TPO mit Bézier-Kurven als $f_i(u)$ und $g_j(v)$ verwendet werden, kann nahezu jede Oberfläche dargestellt werden.

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) * B_{j,m}(v) * P_{i,j} \tag{17}$$

Für gebrochen rationale Bézier-Kurven können erneut die homogenen Koordinaten verwendet werden, sodass folgende Gleichung entsteht.

$$S^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) * B_{j,m}(v) * P^w_{i,j} \tag{18}$$

Mit dieser Gleichung können alle Oberlfächen dargestellt werden. Jedoch bietet diese Gleichung das Problem, dass sie sehr schnell zu sehr hohen Graden (große n und m) von Bézier-Kurven ausarten. Dadurch ist die Berechnung sehr kompliziert und rechenintensiv. Eine Lösung dafür wird im folgenden Kapitel dargestellt.

3 Definition NURBS

3.1 B-Splines

Eine Möglichkeit Oberflächen mathematisch darzustellen ist mit den Bézier-Kurven und den Tensor Produkt Oberflächen gegeben. Um diese zu optimieren, wird NURBS verwendet. NURBS steht für *non-uniform rational B-splines*. Der Begriff *rational*, bzw. auf Deutsch: *gebrochen rational* wurde im Abschnitt 2.1 dargestellt. Diese Funktionen können eine größere Vielfalt an Kurven darstellen.

Wenn die deutsche Grammatik herangezogen wird, steht das Hauptwort eines zusammengesetzten Wortes am Schluss, weshalb der nächste zu betrachtende Bereich die *B-Splines* sind. B-Splines sind Kurven oder Flächen, welche aus vielen einzelnen Kurven/Flächen bestehen. Dadurch gibt es keinen globalen Einfluss von Kontrollpunkten mehr, sondern nur noch einen lokalen. Wenn bei einem TPO ein Kontrollpunkt verändert wird, dann verändert sich an jedem Punkt der Oberfläche der Wert. Wenn ein B-Spline aus verschiedenen unabhängigen Kurven aufgebaut ist, dann verändert sich

bei einer Kontrollpunktänderung nur lokal die Kurve. Eine kontinuierliche Kurve wird garantiert, indem der letzte Punkt P_n der vorhergehenden Kurve gleich dem ersten Punkt P_0.

Eine Möglichkeit diese B-Splines darzustellen ist sind *B-Spline Basis Funktionen* (Basisfunktionen). Diese werden im folgenden als rekursive Folge dargestellt. Bei einer rekursiven Folge definiert sich ein Element n aus vorangegangenen Elementen $n-k$. Dabei ist k eine positive natürliche Zahl mit $k \leq n$.

Die bekannteste rekursive Folge ist vermutlich die Fibonacci folge: $fib(n) = fib(n-1)+fib(n-2)$. Dabei muss ein Anker defineirt werden. Bei Fibonacci ist es: $fib(0) = fib(1) = 1$.

Die Basisfunktionen können wie folgt definiert werden:

$$N_{i,p}(u) = \frac{u - u_i}{u_{u+i} - u_i} * N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} * N_{i+1,p-1}(u) \tag{19}$$

Dabei ist der Anker für $p = 0$ definiert.

$$N_{i,0}(u) = \left\{ \begin{array}{l} 1 \text{ wenn } u_i \leq u_{i+1} \\ 0 \text{ wenn } sonst \end{array} \right\} \tag{20}$$

Dabei wird ein Knotenvektor $U = \{u_0, u_1, ..., u_n\}$ benötigt, welcher die *Step-Funktionen* $N_{i,p}(u)$ ihren nicht-Null-Bereich definiert. Dadurch ist die Lokalität möglich. Die B-Spline Basisfunktionen können nur in den Bereich eine Wirkung auf die endgültige *B-Spline*-Kurve haben, in welcher sie nicht Null sind. Dadurch ist eine Änderung im Bereich von u_i bis u_{i+1} nur durch die Basisfunktionen veränderbar, welche die Funktion $N_{i,0}(u)$ in ihrer Berechnung nutzen.

Wenn eine Basisfunktion berechnet wird, dann bildet sich eine Art Dreieck. Dabei ist die Basis des Dreiecks das i und der Grad p ist die Höhe des Dreiecks.

$$\begin{array}{ccccc} & & N_{0,2}(u) & & \\ & N_{0,1}(u) & & N_{1,1}(u) & \\ N_{0,0}(u) & & N_{1,0}(u) & & N_{2,0}(u) \end{array}$$

Die Anzahl der Basisfunktionen in jedem Grad nimmt ab, desto höher der Grad wird. Somit ist auch nur ein maximaler Grad $p \leq n-1$ möglich. Die Anzahl der Knoten im Knotenvektor Es sind dabei $n-i$ Basisfunktionen mit dem Grad Null möglich.

3.2 Eigenschaften von NURBS

Zusammenfassend können NURBS-Kurven mit folgender Gleichung dargestellt werden. Dabei sind $\{P_i\}$ die Kontrollpunkte, welche durch die $\{w_i\}$ gewichtet werden. Die B-Spline Basis-Funktionen sind durch $\{N_{i,p}(u)\}$ definiert.

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) * w_i * P_i}{\sum_{i=0}^{n} N_{i,p}(u) * w_i} \tag{21}$$

Dabei können die gebrochen rationalen Basisfunktionen $R_{i,p}(u)$ herausgenommen werden, sodass darüber einige Eigenschaften abgeleitet werden können.

$$R_{i,p}(u) = \frac{N_{i,p}(u) * w_i}{\sum_{j=0}^{n} N_{j,p}(u) * w_j} \tag{22}$$

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) * P_i \tag{23}$$

Folgende Eigenschaften lassen sich auf diese gebrochen rationalen Basisfunktionen ableiten:

Nichtnegativ

Es gilt:

$$\forall u \in [0,1], i, p : R_{i,p}(u) \geq 0 \tag{24}$$

Dies leitet sich aus der nichtnegativen Eigenschaft von $\{N_i(u)\}$ ab. Alle $\{w_i\}$ müssen dahingehend immer positiv oder Null sein.

Somit hat jeder Punkt $\{P_i\}$ niemals einen abstoßenden Effekt auf die Kurve, solange die Gewichtung nicht negativ ist.

Teil eines Ganzes

Es gilt:

$$\sum_{i=0}^{n} R_{i,p}(u) = 1; u \in [0,1] \tag{25}$$

Im Zusammenhang mit der Nichtnegativität gilt, dass $0 \leq R_{i,p}(u) \leq 1$ gilt.

Anfang- und Endpunkt-Gleichheit

Es gilt:

$$R_{0,p}(0) = R_{n,p}(1) = 1 \tag{26}$$

Dadurch werden am Ende die Kontrollpunkte P_0 und P_n durch die Kurve geschnitten. Den es gilt für die NURBS-Kurven:

$$C(0) = P_0; C(1) = P_n \tag{27}$$

In der Modellierung ist es dahingehend sinnvoll, wenn die Anfang- und Endpunkte auch den Anfang und das Ende des Modells darstellen.

Lokalität

Eine Veränderung ändert immer nur einen fest abgegrenzten Bereich . Dadurch ist eine komplexe Modellierung erst möglich. Wenn also der Punkt P_i oder die Gewichtung dieses Punktes verändert wird, ändert sich die Kurve nur im Bereich $[u_i, u_{i-1}]$.

Ableitungsregeln

Es ist möglich, $R_{i,p}(u)$ an dem Knoten u_i k-mal abzuleiten, wenn es keine Nullstelle ist. Die Variable k ist hierbei die Multiplizität des Knotens u_i. Dadurch ist ein flüssiger Übergang zwischen den unterschiedlichen Knoten-Abschnitten möglich.

Bei einer NURBS Kurve ist es nicht mehr k-mal Ableitbar, sondern $k-p$ mal ableitbar. Dabei ist k wieder die Multiplizität des Knotens und p der Grad. Desto höher der Grad einer NURBS-Kurve ist, desto geringer ist die Kontinuität.

Vereinfachung

Wenn die Gewichtungen $\{w_i\}$ konstant sind, dann gilt:

$$R_{i,p}(u) = N_{i,p}(u) \tag{28}$$

Dabei sind $\{N_{i,p}(u)\}$ die B-Spline Basis Funktionen. Somit kann der Rechenaufwand deutlich verringert werden, solange die Gewichtungen gleich sind.

4 Modellierung

5 Fazit

6 Anhang

Umformung Bézier-Kurven mit P_i^w auf Bézier-Kurven mit $w_i * P_i$

$$C^w(u) = \sum_{i=0}^{n} B_{i,n}(u) * P_i^w$$

$$X(u) = \sum_{i=0}^{n} B_{i,n}(u) * w_i * x_i$$

$$Y(u) = \sum_{i=0}^{n} B_{i,n}(u) * w_i * y_i$$

$$Z(u) = \sum_{i=0}^{n} B_{i,n}(u) * w_i * z_i$$

$$W(u) = \sum_{i=0}^{n} B_{i,n}(u) * w_i$$

$$x(u) = \frac{X(u)}{W(u)} = \frac{\sum_{i=0}^{n} B_{i,n}(u) * w_i * x_i}{\sum_{i=0}^{n} B_{i,n}(u) * w_i}$$

$$y(u) = \frac{Y(u)}{W(u)} = \frac{\sum_{i=0}^{n} B_{i,n}(u) * w_i * y_i}{\sum_{i=0}^{n} B_{i,n}(u) * w_i}$$

$$z(u) = \frac{Z(u)}{W(u)} = \frac{\sum_{i=0}^{n} B_{i,n}(u) * w_i * z_i}{\sum_{i=0}^{n} B_{i,n}(u) * w_i}$$

$$C(u) = (x(u), y(u), z(u)) = \frac{\sum_{i=0}^{n} B_{i,n}(u) * w_i * (x_i, y_i, z_i)}{\sum_{i=0}^{n} B_{i,n}(u) * w_i} = \frac{\sum_{i=0}^{n} B_{i,n}(u) * w_i * P_i}{\sum_{i=0}^{n} B_{i,n}(u) * w_i}$$

B-Spline Basisfunktions Programm

Mit *jupyter-notebook* wurde das Programm ausgeführt, um die entsprechenden Bilder zu erzeugen. Die Variable p anpassen, um den Grad zu ändern. Die Liste *us* anpassen, um den Knoten-Vektor zu ändern.

```
import matplotlib.pyplot as plt

def bsplinebase(i,p,us,u):
    if p == 0:
        if u < us[i+1] and u >= us[i]:
            return 1
        else:
            return 0
    else:
        try:
            a = (u - us[i])/(us[i+p] - us[i]) * bsplinebase(i,
    p-1, us, u)
        except ZeroDivisionError:
            a = 0
        try:
            b = (us[i+p+1] - u)/(us[i+p+1] - us[i+1]) *
    bsplinebase(i+1, p-1, us, u)
        except:
            b = 0
        return a+b

us = [0,0,0,1,2,3,4,4,5,5,5]
p = 5

fig, ax = plt.subplots(ncols=1, figsize=(40, 5))

u = 0
a = [0]
xs = [0]

for i in range(0,10-p):
    while u < max(us)+ .1:
        xs.append(u)
        a.append(bsplinebase(i,p,us,u))
        u += 0.001
    u = 0
    ax.plot(xs,a)
    a = [0]
    xs = [0]
```

Listing 1: B-Splines Basisfunktionen