UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Raytracing von NURBS

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Richard Markgraf

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	M. Sc. Bastian Krayer (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im Mai 2019

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

1 Kurzfassung

NURBS sind eine Art von Splines, die besondere Eigenschaften besitzen. Das *ray tracen* von NURBS ist eine der Darstellungsmöglichkeiten von NURBS. Dies ist durch das konkrete berechnen von Schnittpunkten mit Strahlen möglich. Durch die vielseitige Möglichkeiten der Modellierung mittels NURBS sind diese beliebt in Anwendungen die im Maschinenbau verwendet werden und auch anderen *CAD*-Programmen. Diese Arbeit befasst sich mit der Berechnung von NURBS-Kurven und -Oberflächen, dem direkten rendern von diesen und wägt ab ob sich der Aufwand dafür im Vergleich zu Tesselierung lohnt.

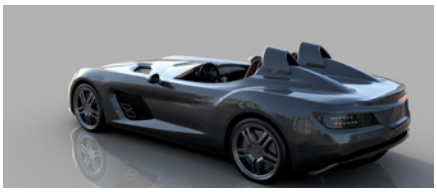
Inhaltsverzeichnis

2 Einleitung

In der Computergraphik werden häufig Dreiecke und Rasterisierung eingesetzt, um Umgebungen und Objekte darzustellen. Dies liegt daran dass das Dreieck ein einfach zu berechnendes, primitives Geometrieobjekt ist, welches relativ geringe Rechenleistung benötigt. Aufgrund der stark gewachsenen Rechenleistung moderner Computer ist es möglich komplexere geometrische Objekte in Echtzeit oder in kurzer Berechnungszeit darzustellen. Eines dieser Objekte wird **Nicht-uniforme rationale Basis-Splines (NURBS)** genannt. Diese ermöglichen es Geometrie mit glatten Oberflächen zu entwerfen, welche durch Dreiecke nur unter Nutzung von hoher Polygonzahl angenähert werden können. NURBS sind ein beliebtes und mächtiges Werkzeug zum Definieren von Kurven und Freiformflächen. NURBS eignen sich als Werkzeug zum Entwurf und der Darstellung von Zeichensätzen und komplexen Fahrzeugkarosserien.

2.1 Anwendungsbereiche

Die Anwendungsbereiche für NURBS-Oberflächen sind vielfältig. Sie sind unerlässlich in der modernen computerunterstützten Modellierung. Im Maschinenbau werden NURBS-Oberflächen ebenfalls wegen ihrer mathematischen Eigenschaften verwendet. Die Anforderungen in diesen Bereichen an die Qualität der Geometrie sind sehr hoch, weshalb für einige Objekte NURBS und andere Splines benötigt werden. Ebenfalls verwendet die Automobilindustrie zum entwurf von Fahrzeugen häufig NURBS. Abbildung 1a zeigt ein Beispiel bei dem das gesamte Fahrzeug durch NURBS modelliert wurde.



(a) Mercedes SLR. [1]

(b) Autodesk Beispielszene. [2]

Abbildung 1: Anwendungsfall Automobilindustrie

3 Grundlagen

3.1 NURBS

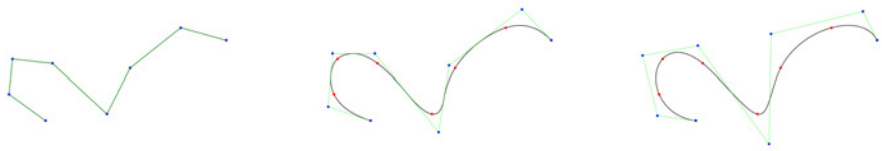
Nicht-uniforme rationale Basis-Splines (NURBS) sind Funktionen, welche stückweise aus Polynomen zusammengesetzt werden. Eine NURBS-Kurve C vom Grad p wird durch

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \leq u \leq b \tag{1}$$

definiert [11] [4]. Hierbei sind $\{P_i\}$ die Kontrollpunkte, $\{w_i\}$ die Gewichte und $\{N_{i,p}(u)\}$ die B-Spline Basisfunktionen vom Grad p, welche auf einem nicht-periodischen (und nicht-uniformen) Knotenvektor

$$U = \{\underbrace{a, \ldots, a}_{\text{p+1}} u_{p+1}, \ldots, u_{m-p-1}, \underbrace{b, \ldots, b}_{\text{p+1}}\}$$

definiert sind. Wenn es nicht anders angegeben wird, dann wird angenommen dass $a = 0, b = 1$ und $w_i > 0$ für alle i ist.



(a) p = 1 **(b)** p = 2 **(c)** p = 3

Abbildung 2: NURBS-Kurven mit Kontrollpunkten

3.2 Basisfunktionen

Die Definition von

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{2}$$

erlaubt es Formel (1) in

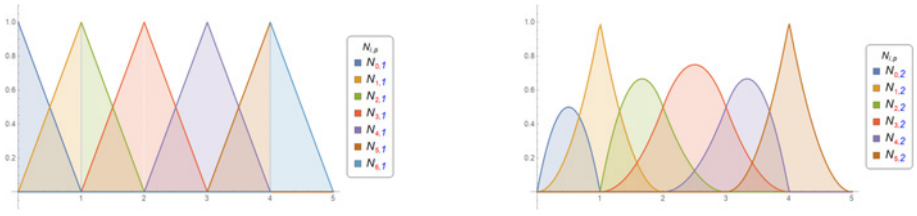
$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) P_i \tag{3}$$

umzuschreiben [10]. $\{R_{i,p}\}$ werden als rationalen Basisfunktionen bezeichnet und besitzen die folgenden Eigenschaften [10] [3]:

1. Nicht negativ: $R_{i,p}(u) \geq 0$ für alle i, p und für $u \in [0, 1]$.
2. $\sum_{i=0}^{n} R_{i,p}(u) = 1$ für alle $u \in [0, 1]$
3. $R_{0,p}(0) = R_{n,p}(1) = 1$.
4. Ist $p > 0$, dann besitzen alle $R_{i,p}(u)$ exakt ein Maximum auf dem Intervall $u \in [0, 1]$.
5. Lokaler Träger/*support*: $R_{i,p}(u) = 0$ für $u \notin [u_i, u_{i+p+1})$.
6. Alle Ableitungen von $R_{i,p}(u)$ existieren innerhalb eines Knotenintervalls bei dem die rationale Funktion keine Null im Nenner besitzt.
7. Ist $w_i = a$ für alle i und $a \neq 0$, dann folgt $R_{i,p}(u) = N_{i,p}(u)$ für alle i.

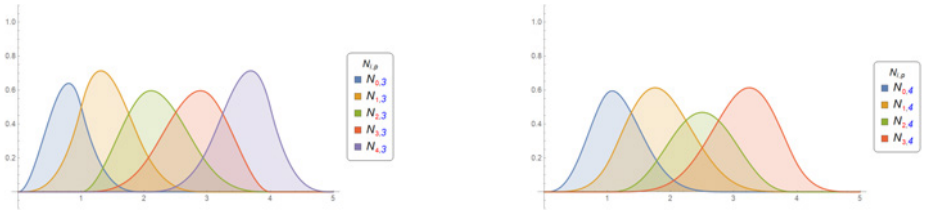
Daraus folgen ebenfalls einige Eigenschaften für die Kurve C [11]:

1. C ist invariant zu affinen Transformationen. Eine Transformation wird auf die Kurve angewendet, indem sie auf die Kontrollpunkte angewendet wird.
2. C ist invariant zu perspektivischer Projektion.
3. Ist $u \in [u_i, u_{i+1})$, dann liegt $C(u)$ innerhalb der konvexen Hülle aus den Kontrollpunkten $P_{i-p}, \ldots, P_i$.
4. Start- und Endpunkt liegen auf dem ersten bzw. dem letzten Kontrollpunkt. $C(0) = P_0$ und $C(1) = P_n$.



(a) p = 1 **(b)** p = 2

Abbildung 3: Graphen von Basisfunktionen



(a) p = 3 **(b)** p = 4

Abbildung 4: Graphen von Basisfunktionen

3.3 Cox-de Boor

Um die rationalen Basisfunktionen $\{R_{i,p}\}$ berechnen zu können, müssen die Basisfunktionen $\{N_{i,p}\}$ berechnet werden. Dies ist durch Anwenden der Cox-de Boor Formel möglich [11] [4]. Diese ist definiert als

$$N_{i,0}(u) = \begin{cases} 1 & \text{falls } u_i \leq u \leq u_{i+1} \\ 0 & \text{andernfalls} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{4}$$

Anschaulich baut sich zur Berechnung einer Basisfunktion ein Baum aus Basisfunktionen von niedrigerem Grad auf. Möchte man nun einen Punkt auf der Kurve berechnen, dann wird dieser über eine Kombination aus Basisfunktionen und Kontrollpunkten berechnet. Abbildung 6 zeigt visuell wie sich ein Punkt $C(\frac{1}{2})$ auf der Kurve berechnen lässt.

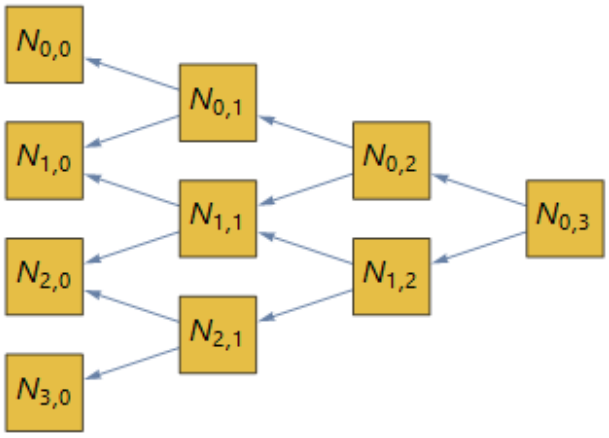


Abbildung 5: Baum aus Basisfunktionen

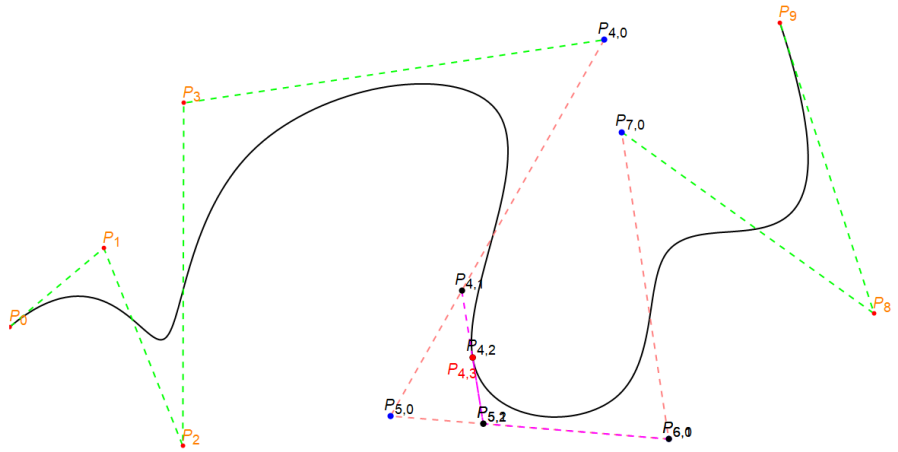


Abbildung 6: NURBS-Kurve mit Kontrollpunkten

3.4 Berechnen von Ableitungen

In vielen Fällen ist es nützlich die Ableitung der Kurve an einer Stelle zu berechnen. Um dies zu berechnen, betrachten wir Formel 1 als eine Kurve im vierdimensionalen Raum [10]. Hierbei bilden die Gewichte $\{w_i\}$ die vierte Koordinate. Da durch diese später wieder geteilt wird, wird diese auch homogene Koordinate genannt. Somit wird $C(u)$ zu $H\{C^w(u)\}$ und über diese lassen sich die Ableitungen berechnen.

$$C(u) = H\{C^w(u)\} = H\left\{\sum_{i=0}^{n} N_{i,p}(u) P_i^w\right\} \tag{5}$$

Aus den Kontrollpunkten $\{P_i\}$ und den Gewichten $\{w_i\}$ werden die gewichteten Kontrollpunkte $P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$ gebildet. H steht für die perspektivische Abbildung

$$\begin{aligned} P &= H\{P^w\} \\ &= H\{(X,Y,Z,W)\} = \begin{cases} (\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}) & \text{falls } W \neq 0 \\ \text{Richtungsvektor}(X,Y,Z) & \text{falls } W = 0 \end{cases} \end{aligned} \tag{6}$$

Sei nun

$$C(u) = \frac{w(u)C(u)}{w(u)} = \frac{A(u)}{w(u)} \tag{7}$$

Hierbei ist $A(u)$ die Funktion, welche den Vektor mit den ersten drei Koordinaten von $C^w(u)$ enthält. $A(u)$ ist hierbei nichts anderes als der Zähler in Formel 1. Dann ist die erste Ableitung von $C(u)$

$$\begin{aligned} C'(u) &= \frac{w(u)A'(u) - w'(u)A(u)}{w(u)^2} \\ &= \frac{w(u)A'(u) - w'(u)w(u)C(u)}{w(u)^2} = \frac{A'(u) - w'(u)C(u)}{w(u)} \end{aligned} \tag{8}$$

Die Formel für $C'(u)$ wird auch hodograph Formel genannt [5]. Um die Ableitungen $A'(u)$ und $w'(u)$ zu erhalten wird die Formel zur Ableitung von B-Splines auf C^w angewendet.

$$C^{w(k)}(u) = \sum_{i=0}^{n} N_{i,p}^{(k)}(u) P_i^w \tag{9}$$

Die Basisfunktionen lassen sich nach [10] durch

$$N_{i,p}^{(k)}(u) = p \left(\frac{N_{i,p-1}^{(k-1)}}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}}{u_{i+p+1} - u_{i+1}} \right) \tag{10}$$

ableiten. Sei $k = 1$, dann enthält $C^{w(k)}(u)$ die Werte von $A'(u)$ und $w'(u)$. Für andere k erhält man dementsprechend $A^{(k)}(u)$ und $w^{(k)}(u)$.

$$C^{w\prime}(u) = (\underbrace{X', Y', Z'}_{A'(u)}, \underbrace{W'}_{w'(u)}) \tag{11}$$

Weitere Ableitungen erhalten wir durch das Anwenden der allgemeinen Produktregel für höhere Ableitungen (auch als Leibniz Regel für Differentiale bekannt) [8]. Seien u und v zwei Funktionen, dann berechnet sich die k-te Ableitung des Produktes durch

$$uv^{(k)} = \sum_{i=0}^{k} \binom{k}{i} u^{(i)} v^{(k-i)} \tag{12}$$

Wenden wir Formel 12 zur Berechnung von $A^k(u)$ an, dann folgt

$$\begin{aligned} A^{(k)}(u) &= (w(u)C(u))^{(k)} = \sum_{i=0}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u) \\ &= w(u)C^{(k)}(u) + \sum_{i=1}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u) \end{aligned} \tag{13}$$

Durch umformen erhält man

$$C^{(k)}(u) = \frac{A^{(k)}(u) - \sum_{i=1}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u)}{w(u)} \tag{14}$$

Welches einem ermöglicht die k-te Ableitung an einem Punkt für einen Wert u von einer NURBS-Kurve zu berechnen.

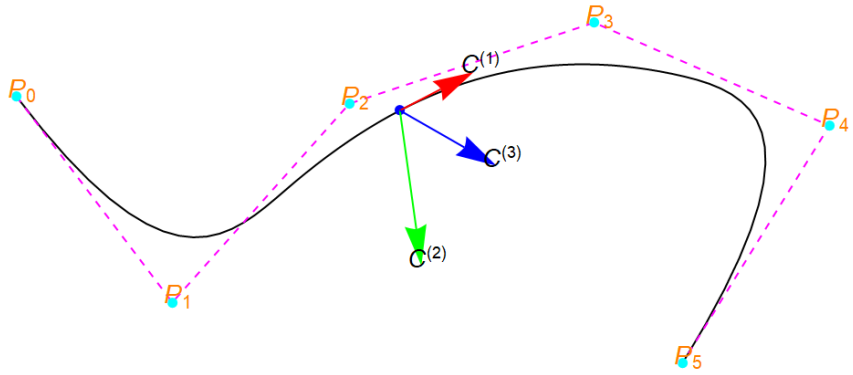


Abbildung 7: Ableitungen einer NURBS-Kurve

3.5 Freiformflächen

Durch NURBS-Flächen lässt sich komplexe Geometrie beschreiben und entwerfen. Eine solche Fläche entsteht durch das Kombinieren mehrerer NURBS-Kurven zu einer NURBS-Fläche. Eine NURBS-Fläche vom Grad p in Richtung u und vom Grad q in v Richtung ist eine bivariate stückweise rationale Funktion der Form [3]

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad 0 \leq u,v \leq 1 \qquad (15)$$

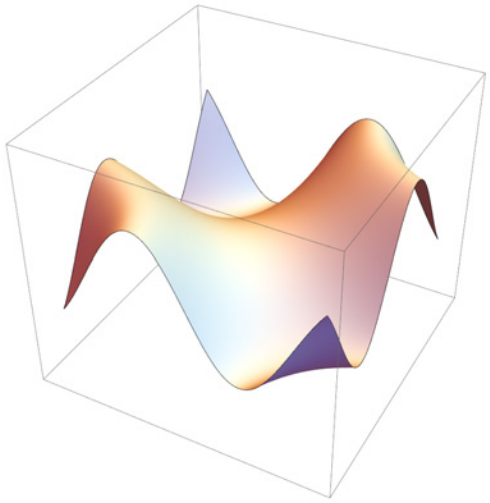


Abbildung 8: NURBS-Freiformfläche

Die Kontrollpunkte $\{P_{i,j}\}$ bilden ein Kontrollnetz in zwei Richtungen, die Gewichte $\{w_{i,j}\}$ bilden ebenfalls ein Netz und $\{N_{i,p}(u)\}$ und $\{N_{j,q}(v)\}$ sind die nicht-rationalen B-Spline Basisfunktionen, welche auf den Knotenvektoren

$$U = \{\underbrace{0,\ldots,0}_{\text{p+1}} u_{p+1},\ldots,u_{r-p-1},\underbrace{1,\ldots,1}_{\text{p+1}}\}$$
$$V = \{\underbrace{0,\ldots,0}_{\text{q+1}} v_{q+1},\ldots,v_{s-q-1},\underbrace{1,\ldots,1}_{\text{q+1}}\}$$

definiert sind. Hierbei ist $r = n+p+1$ und $s = m+q+1$. Ähnlich zu Formel 2 lässt sich auch eine stückweise rationale Basisfunktion formulieren

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,p}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{t=0}^{m} N_{k,p}(u)N_{t,q}(v)w_{k,t}} \tag{16}$$

Somit erhält man zusammen mit Formel 15

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

NURBS-Flächen lassen sich so wie in Formel 5, über homogene Koordinaten darstellen [10]. Sei $P^w_{i,j} = (w_{i,j}x_{i,j}, w_{i,j}y_{i,j}, w_{i,j}z_{i,j}, w_{i,j})$

$$\begin{aligned} S(u,v) &= H\{S^w(u,v)\} \\ &= H\left\{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P^w_{i,j}\right\} \end{aligned} \tag{18}$$

Analog zu Formel 8 und Formel 14 lassen sich auch die Ableitungen von NURBS-Oberflächen berechnen [5]. Allerdings können hier mehrere partielle Ableitungen berechnet werden. Diese sind vom Parameter der zum Ableiten gewählt wird abhängig.

$$S(u,v) = \frac{w(u,v)S(u,v)}{w(u,v)} = \frac{A(u,v)}{w(u,v)} \tag{19}$$

$A(u,v)$ ist der Zähler von $S(u,v)$ in Formel 15.

$$S_\alpha(u,v) = \frac{A_\alpha(u,v) - w_\alpha(u,v)S(u,v)}{w(u,v)} \tag{20}$$

α kann entweder für u oder für v stehen. Nun gilt allgemein

$$\begin{aligned} A^{(k,l)} = [(wS)^k]^l &= \left(\sum_{i=0}^{k}\binom{k}{i}w^{(i,0)}S^{(k-i,0)}\right)^l \\ &= \sum_{i=0}^{k}\binom{k}{i}\sum_{j=0}^{l}\binom{l}{j}w^{(i,j)}S^{(k-i,l-j)} \\ &= w^{(0,0)}S^{(k,l)} + \sum_{i=1}^{k}\binom{k}{i}w^{(i,0)}S^{(k-i,l)} + \sum_{j=1}^{l}\binom{l}{j}w^{(0,j)}S^{(k,l-j)} \\ &+ \sum_{i=1}^{k}\binom{k}{i}\sum_{j=1}^{l}\binom{l}{j}w^{(i,j)}S^{(k-i,l-j)} \end{aligned} \tag{21}$$

Daraus lässt sich durch Umformen folgendes formulieren

$$\begin{aligned} S^{(k,l)} = \frac{1}{w}\Bigg(& A^{(k,l)} - \sum_{i=1}^{k}\binom{k}{i}w^{(i,0)}S^{(k-i,l)} \\ & - \sum_{j=1}^{l}\binom{l}{j}w^{(0,j)}S^{(k,l-j)} - \sum_{i=1}^{k}\binom{k}{i}\sum_{j=1}^{l}\binom{l}{j}w^{(i,j)}S^{(k-i,l-j)}\Bigg) \end{aligned} \tag{22}$$

Aus Formel 22 erhält man

$$S_{uv} = \frac{A_{uv} - w_{uv}S - w_uS_v - w_vS_u}{w} \tag{23}$$

$$S_{uu} = \frac{A_{uu} - 2w_uS_u - w_{uu}S}{w} \tag{24}$$

$$S_{vv} = \frac{A_{vv} - 2w_vS_v - w_{vv}S}{w} \tag{25}$$

Ähnlich zu Formel 9 leiten sich $A^{(k,l)}$ und $w^{(k,l)}$ aus der Formel zum ableiten von B-Spline-Oberflächen ab [11]. Es berechnen sich somit alle Ableitungen bis zum Grad d

$$\frac{\partial^{k+l}}{\partial^k u \partial^l v} S^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}^{(k)} N_{j,q}^{(l)} P_{i,j}^w \qquad 0 \leq k+l \leq d \tag{26}$$

Über die Ableitungen der Oberflächen lassen sich Oberflächennormalen berechnen, welche für Beleuchtung, Reflektion etc. beim *rendern* der Fläche benötigt werden.

$$n(u,v) = \frac{1}{\| S_{vv}^{(1)} \times S_{uu}^{(1)} \|} (S_{vv}^{(1)} \times S_{uu}^{(1)}) \tag{27}$$

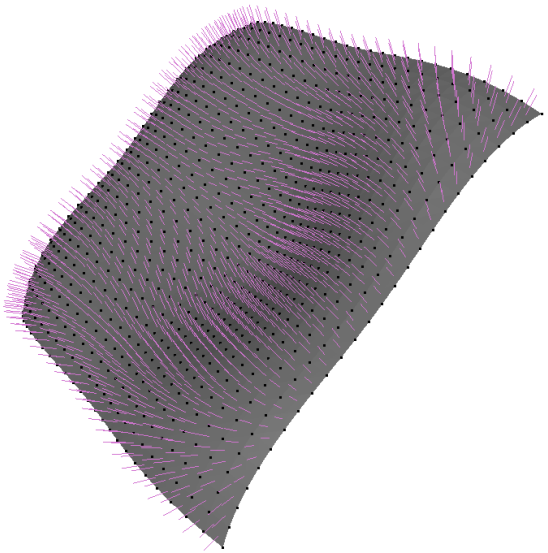


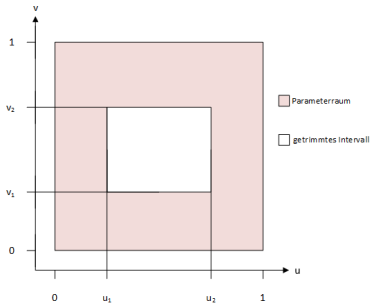
Abbildung 9: NURBS-Freiformfläche mit Oberflächennormalen

4 Trimmen von NURBS-Oberflächen

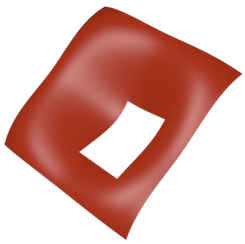
Das Konzept des Trimmens beschreibt das Herausschneiden von Formen aus Spline-Oberflächen. Dies geschieht allerdings nicht direkt auf der Oberfläche, sondern im Parameterraum der Oberfläche, welcher durch u und v aufgespannt wird. Hierbei kann mit einfachen Primitiven, wie z.B. Dreiecken,Rechtecken etc., getrimmt werden oder es können auch zweidimensionale Kurven, welche innerhalb des Parameterraumes definiert sind, zum trimmen verwendet werden.

4.1 Einfaches Trimmen

Für einfache geometrische Objekte, wie z.B. Kreise, Dreiecke, Rechtecke, Ellipsen etc. gibt es die Möglichkeit den Trimmtest direkt über die Formel für das Primitiv auszuführen. Somit ist keine Schnittpunktberechnung nötig. Um eine rechteckige Fläche zu trimmen, werden lediglich zwei u-Werte



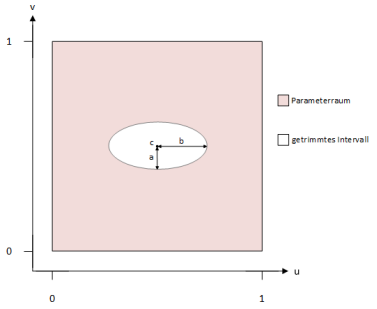
(a) Parameterraum



(b) NURBS-Fläche

Abbildung 10: NURBS-Fläche mit getrimmtem Rechteck

und zwei v-Werte benötigt. Soll nun getestet werden, ob ein Punkt $S(u, v)$ innerhalb des getrimmten Intervalls liegt, dann muss nur geprüft werden ob $u_1 \leq u \leq u_2$ und ob $v_1 \leq v \leq v_2$. Trifft beides zu, dann liegt der Punkt $S(u, v)$ innerhalb der getrimmten Fläche und der Punkt wird getrimmt.



(a) Parameterraum

(b) NURBS-Fläche

Abbildung 11: NURBS-Fläche mit getrimmter Ellipse

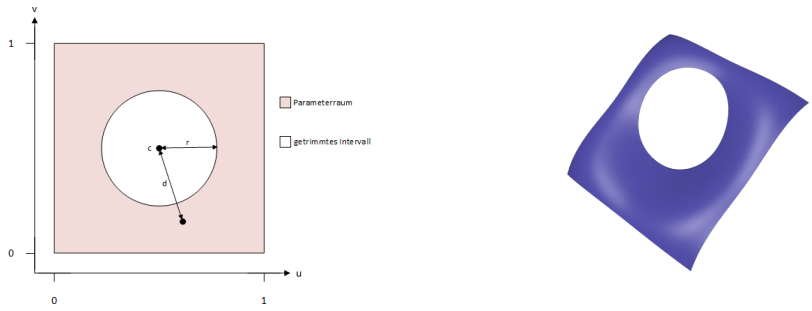
Das Trimmen einer Ellipse lässt sich über die Mittelpunktform einer Ellipse durchführen. Seien zwei Radien a, b und der Mittelpunkt c der Ellipse gegeben, dann gilt

$$\frac{(x - c_x)^2}{a^2} + \frac{(y - c_y)^2}{b^2} = 1 \tag{28}$$

Formuliert man diese Gleichung in die Ungleichung

$$\frac{(u - c_u)^2}{a^2} + \frac{(v - c_v)^2}{b^2} \leq 1 \tag{29}$$

um, dann lässt sich anhand dieser der Trimmtest durchführen. Es wird für alle Punkte $S(u, v)$ geprüft, ob die Ungleichung erfüllt wird. Tut sie dies, dann ist der Punkt nicht mehr Teil der Oberfläche.



(a) Parameterraum **(b)** NURBS-Fläche

Abbildung 12: NURBS-Fläche mit getrimmtem Kreis

Zum Trimmen einer Kreisfläche werden der Kreismittelpunkt c und ein Radius r benötigt. Hierzu wird der Abstand d zwischen der Eingabe (u, v) und dem Mittelpunkt berechnet. Ist nun $d < r$, dann wird der Punkt $S(u, v)$ getrimmt.

4.2 Newton-Verfahren

Das Newton-Verfahren (auch als Newton-Raphson-Verfahren bekannt) ist ein Verfahren zur numerischen Nullstellenfindung von nichtlinearen Gleichungen. Man benötigt zur Anwendung des Verfahrens eine nichtlineare Funktion $f(x)$ und einen initialen Startwert x_0, welcher in der Nähe der Nullstelle liegt. Nun berechnet man die Tangente des Startwertes. Diese Tangente schneidet die x-Achse und liegt näher an der Nullstelle als x_0. Dieser Wert wird als x_1 bezeichnet und man berechnet wieder die Tangente des neuen Wertes, welcher wiederum näher an der Nullstelle liegt. So setzt sich das Verfahren rekursiv fort, bis die Differenz von zwei aufeinanderfolgenden Approximationen kleiner ist als ein bestimmter Schwellwert ϵ. Der neue Approximationswert wird über

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{30}$$

berechnet. Diese Rekursionsvorschrift ist das was als Newton-Iteration bezeichnet wird. Die Konvergenz der Annäherung an die Nullstelle ist qua-

dratisch. Sollte $f'(x) = 0$ sein, dann ist das Verfahren nicht anwendbar.

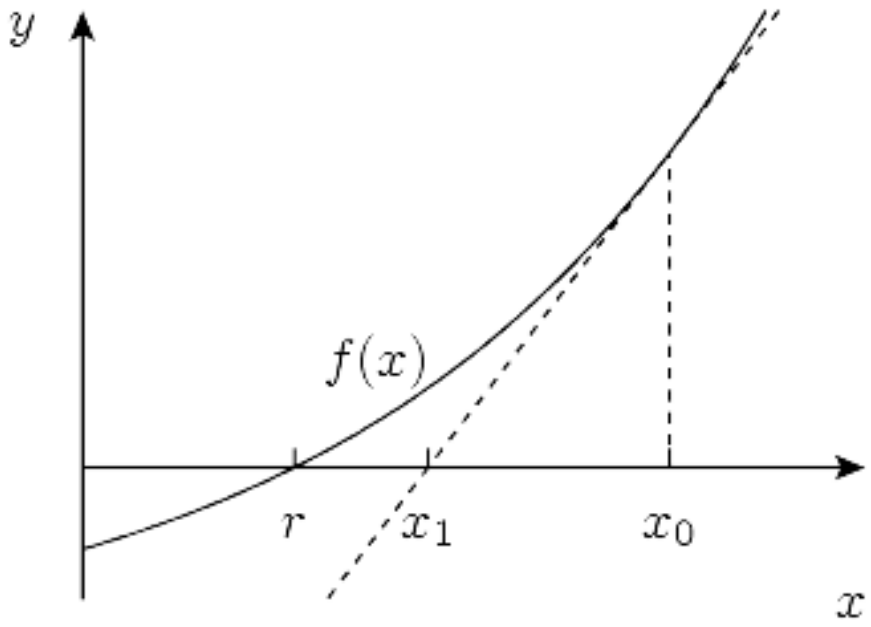


Abbildung 13: Newton-Iteration mit Nullstelle r

Das Newton-Verfahren lässt sich auch auf mehrdimensionale Probleme erweitern. Hierzu wird anstelle der Ableitung die Jacobi-Matrix, also die Matrix der partiellen Ableitungen, berechnet. Anstatt nach Nullstellen der nichtlinearen Funktion f zu suchen, wird nach Nullstellen der linearen Anpassung von f im Punkt x gesucht.

$$x_{n+1} = x_n - (J(x_n))^{-1} f(x_n) \tag{31}$$

Alternativ wird auch häufig

$$J(x_n)\Delta x_n = -f(x_n)$$

gelöst, um die Berechnung der Inversen der Jacobi-Matrix zu vermeiden. Man erhält dann x_{n+1} durch

$$x_{n+1} = x_n + \Delta x_n$$

4.3 Trimmkurven

Das Herausschneiden von Flächen, welche durch Kurven eingeschlossen sind, aus NURBS-Oberflächen benötigt eine sogenannte Trimmkurve. Die Trimmkurve ist in diesem Fall eine NURBS-Kurve, welche in dem Parameterraum der Oberfläche definiert ist. Hinzu kommt dass diese Kurve geschlossen sein muss und eine definierte Richtung besitzen muss. Die Richtung wird benötigt um festzustellen welcher Teil der Oberfläche behalten

Abbildung 14: NURBS-Fläche mit getrimmter Kurve

werden soll. Entweder der Teil innerhalb der Kurve oder der Teil außerhalb wird dann erhalten. Es wird dann festgelegt dass der Teil links der Kurve behalten wird. Die Kontrollpunkte der Kurve $C(u)$

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{32}$$

liegen auf dem Parameterraum der Oberfläche $S(a, b)$. Sie besitzen die Form $P_i = (a, b)$. Um nun zu Prüfen ob ein Punkt $x = S(a, b)$ getrimmt werden muss, werden alle Schnittpunkte zwischen der Kurve $C(u)$ und einem Strahl $r(t) = x + d * t$ berechnet. Der Ursprung des Strahls ist der Punkt x und die Richtung d wird generiert indem ein zweiter zufälliger Punkt $z \neq x$ generiert wird. Aus den Punkten wird die Richtung $d = z - x$ berechnet. Zum verifizieren aller Schnittpunkte wird das Newton-Verfahren verwendet. Da dieses Startwerte benötigt, generiert man einen Umriss der Kurve. Dieser Umriss wird erstellt indem einige Punkte auf der Kurve berechnet werden und diese über Liniensegmente verbunden werden. Zusätzlich zu den Punkten wird auch der Parameter u, durch welchen die Punkte berechnet werden, gespeichert. Nun berechnet man alle Schnittpunkte zwischen dem Umriss und dem Strahl r. Zu jedem Schnittpunkt wird ebenfalls geprüft ob der Strahl in den Umriss eindringt oder diesen verlässt. Dies wird mit -1 beim Eindringen und mit $+1$ beim Verlassen notiert. Anhand der definierten Richtung der Kurve lassen sich Normalen auf den Liniensegmente berechnen, die nach außen zeigen. Durch diese lässt sich bestimmen,

ob der Strahl aus dem vorderen oder hinteren Halbraum kommt. Für den

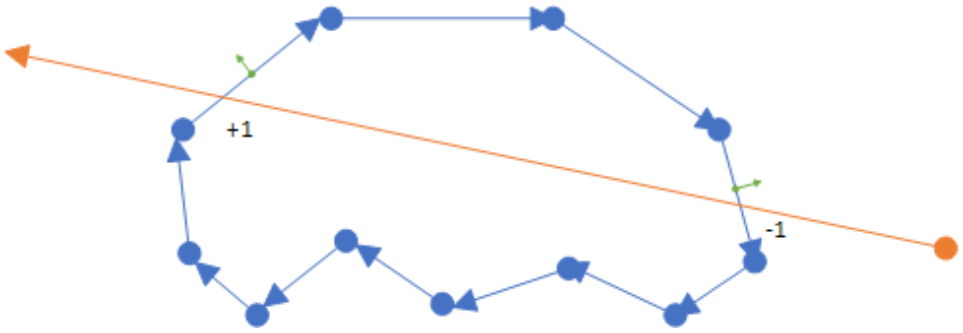


Abbildung 15: Liniensegment für den Kurventrimmtest

Fall dass der Strahl genau entlang der Richtung eines Segmentes verläuft und es ein komplettes Segment als Intersektion gibt, wird einfach eine neue zufällige Richtung für den Strahl gewählt und der Test wird von vorne gestartet. Sobald alle Schnittpunkte bekannt sind, werden die aus diesen berechneten u-Werte als Startwerte für die Newton-Iteration verwendet. Über die Newton-Iteration wird nun für jeden Schnittpunkt überprüft ob dieser ein gültiger Schnittpunkt zwischen Kurve und Strahl ist. Die Newton-Iteration nimmt die Form

$$u_{n+1} = u - \frac{f(u_n)}{f'(u_n)} \tag{33}$$

mit

$$f(u) = (C(u) \cdot n + d)$$

Der Strahl wurde hierbei über die hessesche Normalform dargestellt. n ist die Normale zum Strahl. Dann wird aus den Werten der gültigen Schnittpunkte für das Verlassen und Eindringen der Kurve die Summe gebildet. Ist die Summe 1, dann wird der Punkt getrimmt. Ist die Summe 0, dann wird der Punkt nicht getrimmt.

5 Schnittpunktberechnung

Um eine NURBS-Oberfläche zu *ray tracen*, muss der Schnittpunkt zwischen dem Strahl r und der Oberfläche $S(u, v)$ berechnet werden. Der Strahl $r(t) = o + d * t$ besitzt einen Ursprung o und eine Richtung d. Nach [6] stellt man den Strahl als einen Schnitt von zwei Ebenen $P_1 = (N_1, d_1)$ und $P_2 = (N_2, d_2)$ dar. Die Normale zur Ebene P_1 ist definiert als [7]

$$N_1 = \begin{cases} (d_y, -d_x, 0) & \text{falls } |d_x| > |d_y| \text{ und } |d_x| > |d_z| \\ (0, d_z, -d_y) & \text{andernfalls} \end{cases} \tag{34}$$

Somit ist N_1 immer orthogonal zur Richtung d des Strahls r. Dann ist N_2

$$N_2 = N_1 \times d$$

Beide Ebenen gehen durch den Ursprung o somit ergibt sich

$$\begin{aligned} d_1 &= -N_1 \cdot o \\ d_2 &= -N_2 \cdot o \end{aligned}$$

Ein Schnittpunkt mit der Oberfläche S muss nun

$$\begin{aligned} P_1 \cdot S(u,v) &= 1 \\ P_2 \cdot S(u,v) &= 1 \end{aligned} \tag{35}$$

erfüllen. Daraus lässt sich die Funktion $F(u,v)$ für die Newton-Iteration formulieren.

$$F(u,v) = \begin{pmatrix} N_1 \cdot S(u,v) + d_1 \\ N_2 \cdot S(u,v) + d_2 \end{pmatrix} \tag{36}$$

Das Problem liegt nun dabei das Gleichungssystem nach den Nullstellen (u_*, v_*) zu lösen. Hierzu wird die mehrdimensionale Newton-Iteration verwendet. Somit nimmt die Iteration nach Formel 31 die Form

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} u_n \\ v_n \end{pmatrix} - J^{-1}(u_n, v_n) * F(u_n, v_n) \tag{37}$$

J ist die Jacobi-Matrix von F

$$J = (F_u, F_v) \tag{38}$$

F_u und F_v sind die Vektoren

$$\begin{aligned} F_u &= \begin{pmatrix} N_1 \cdot S_u(u,v) \\ N_2 \cdot S_u(u,v) \end{pmatrix} \\ F_v &= \begin{pmatrix} N_1 \cdot S_v(u,v) \\ N_2 \cdot S_v(u,v) \end{pmatrix} \end{aligned} \tag{39}$$

Die Inverse der Jacobi-Matrix ist

$$J^{-1} = \frac{adj(J)}{det(J)} \tag{40}$$

Die Adjunkte $adj(J)$ ist gleich der transponierten Kofaktormatrix

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

mit $C_{ij} = (-1)^{i+j} det(J_{ij})$. J_{ij} ist die verbleibende Matrix, wenn man die i-te Reihe und die j-te Spalte entfernt. Somit ist die $adj(J)$ nun [7]

$$adj(J) = \begin{pmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{pmatrix}$$

Für die Iteration wurden mehrere Abbruchkriterien nach [12] und [7] gewählt. Falls man so nahe an der Nullstelle ist, dass der Wert kleiner ist als ein vorbestimmtes ϵ,

$$\|F(u_n, v_n)\| < \epsilon \tag{41}$$

dann wird ein Treffer registriert. Die Anzahl der Iterationen wird limitiert für den Fall dass nur eine schwache Konvergenz vorliegt. Meistens sind nur wenige Iterationen nötig, um den Schwellwert zu erreichen. Dies liegt an der allgemein quadratischen Konvergenz des Verfahrens. Ebenfalls sollte die Iteration konvergieren und nicht divergieren. Also darf der nachfolgende Iterationsschritt nicht weiter weg von der Nullstelle (u_*, v_*) führen.

$$\|F(u_{n+1}, v_{n+1})\| > \|F(u_n, v_n)\| \tag{42}$$

Die Iteration darf nicht außerhalb des Parameterraums der Oberfläche laufen:

$$u \neq [u_p, u_n), v \neq [v_q, v_m) \tag{43}$$

Schließlich noch wird geprüft ob die Jacobi-Matrix Singulär ist, denn dann lässt sich die Inverse nicht berechnen. An solchen stellen ist entweder die Oberfläche nicht regulär ($S_u \times S_v = 0$) oder der Strahl verläuft parallel in der Nähe zum Umriss der Oberfläche. Es wird geprüft ob $|det(j)| < \epsilon$. Sollte dies der Fall sein, dann wird der momentane Parameter um einen kleinen Zufallswert verschoben, um Problemregionen zu umgehen.

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix} + 0,1 * \begin{pmatrix} drand48() * (u_0 - u_k) \\ drand48() * (v_0 - v_k) \end{pmatrix} \tag{44}$$

$drand48()$ ist eine Funktion, welche nicht-negative Zufallswerte vom Typ *double* liefert und diese sind gleichverteilt auf dem Intervall $[0.0, 1.0]$. Mit dem neu berechneten Vektor wird dann die Iteration weitergeführt. Da die Newton-Iteration nur einen angenäherten Wert des Schnittpunktes (u_*, v_*) liefert, liegt dieser mit sehr hoher Wahrscheinlichkeit nicht exakt auf dem Pfad des Strahls $r = o + t * d$. Um zu gewährleisten dass der Punkt auf dem Strahl liegt kann man den Punkt P zurück auf den Strahl projezieren.

$$t = (P - o) \cdot d$$

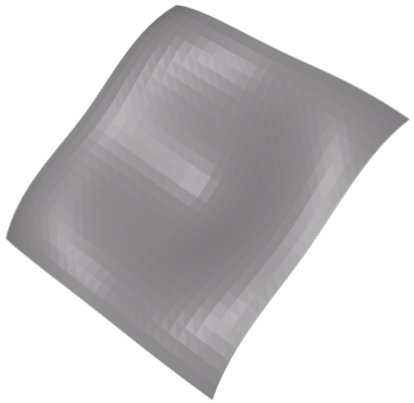


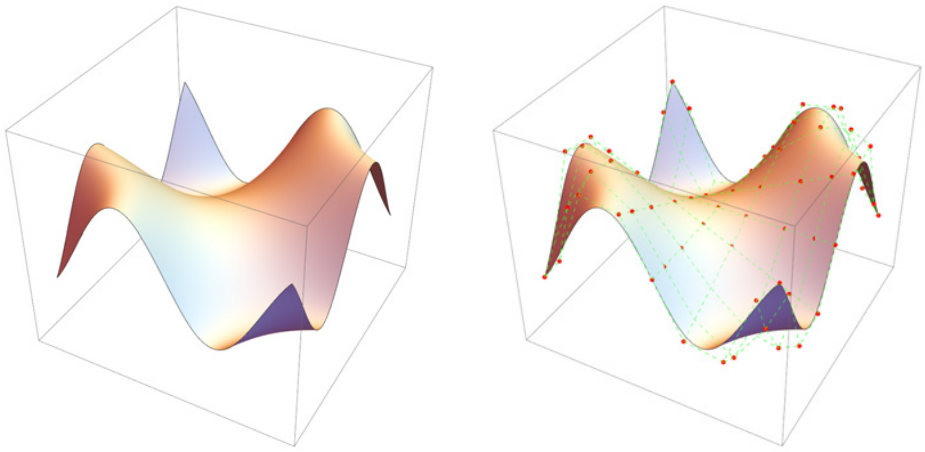
Abbildung 16: Tesselierte NURBS-Oberfläche

5.1 Berechnung von Startwerten

Die Newton-Iteration benötigt einen Startwert, welcher in der Nähe des Schnittpunktes liegt um zu konvergieren. Hierzu wird die Oberfläche tesseliert [9]. Die Fläche muss entsprechend fein tesseliert werden, um die Nähe zum Schnittpunkt zu garantieren. Zusätzlich zu den generierten Dreieckspunkten wird auch der Parameterbereich in dem das Dreieck liegt gespeichert. Nun wird das Dreiecksnetz *geraytraced*. Der daraus resultierende Schnittpunkt wird in baryzentrische Koordinaten, in Relation zum Dreieck ABC in dem der Schnittpunkt liegt, überführt. Nun lässt sich der Startwert (u_0, v_0) über baryzentrische Interpolation berechnen. Da die Eckpunkte alle einen gespeicherten (u, v)-Wert besitzen, lässt sich der (u, v)-Wert für den Schnittpunkt ebenfalls berechnen.

$$f(x) = a(f(x_A)) + b(f(x_B)) + c(f(x_C)) \tag{45}$$

$f(x)$ bildet einen Punkt x auf seinen (u, v)-Wert ab. (a, b, c) sind hierbei die normierten baryzentrischen Koordinaten des Punktes. $f(x_A), f(x_B), f(x_C)$ sind die (u, v)-Werte der Eckpunkte des Dreiecks ABC in dem der Schnittpunkt liegt.



(a) NURBS-Oberfläche **(b)** Oberfläche mit Dreiecksnetz

Abbildung 17: NURBS-Oberflächen

6 Implementation

Die Implementation der Schnittpunktberechnung bezieht sich stark auf die Formeln. Es wurde ebenfalls eine *bounding box* für die Oberfläche generiert um schneller Strahlen auswerfen zu können, die nicht die Oberfläche treffen. In den Zeilen 16-22 wird der Startwert für die Newton-Iteration über das Dreiecksnetz berechnet. In Zeile 23 wird $F(u, v)$ nach Formel 36 berechnet. Ebenso wird bevor die Iteration startet geprüft ob die Jacobi-Matrix singulär ist. Für die Newton Iteration wird immer der alte Wert von $F(u, v)$ und (u_k, v_k) gespeichert, um die Tests für den Iterationsabbruch zu berechnen können. Die Zeilen 32-46 entsprechen den Formeln 42,43 und 44.

```
1  double NURBSSurface::GetIntersection(const Ray &ray){
2  if (bbox.GetIntersection(ray)) {
3    Vector3d d = ray.GetDirection();
4    Vector3d N1, N2;
5    if (abs(d.x) > abs(d.y) && abs(d.x) > abs(d.z)) {
6      N1 = (d.y, -1.0*d.x, 0);
7      N1.Normalize();
8    }
9    else {
10     N1 = (0, d.z, -1.0*d.y);
11     N1.Normalize();
12   }
13   N2 = glm::cross(N1,d);
14   double d1 = glm::dot(-1.0*N1,ray.GetOrigin());
15   double d2 = glm::dot(-1.0*N2, ray.GetOrigin());
16   mesh.intersect(ray);
17   if (mesh.hasintersection(ray)) {
18     Vector2d uv = mesh.GetIntersection(ray);
```

```
19    }
20    else {
21      return -1;
22    }
23    Vector2d uv_start = uv;
24    Vector2d Fuv = evalFuv(N1,N2,d1,d2,uv);
25    mat2d J = jacobian(N1,N2,srf,uv);
26    if (det(jacobian) < EPSILON) {
27      return -1;
28    }
29    int iter = 0;
30    while (glm::length(Fuv)>EPSILON) {
31      Vector2d Fuv_old = Fuv;
32      Vector2d uv_old = uv;
33      if (iter > iterMAX) {
34        return -1;
35      }
36      Fuv = evalFuv(N1, N2, d1, d2, uv);
37      J = jacobian(N1, N2, uv);
38      if (glm::length(Fuv) > glm::length(Fuv_old)) {
39        return -1;
40      }
41      if (det(jacobian) < EPSILON2) {
42        uv = jitter(uv_old,uv_start);
43        continue;
44      }
45      if (paramcheck(uv, srf)) {
46        return -1;
47      }
48
49      mat2d invJ = invJacobian(J);
50      uv = uv_old - (mvmult(invJ, Fuv));
51      iter++;
52    }
53    return glm::dot(srf.evaluate(uv)-ray.GetOrigin(),ray.
        GetDirection());
54  }
55  else
56    return -1;
57  }
```

7 Auswertung und Ergebnisse

Um bewerten zu können ob es sich lohnt die Schnittpunkte beim *ray tracen* von NURBS-Oberflächen zu berechnen, ist ein Vergleich der Renderzeiten zwischen zwei Methoden angebracht. Man könnte NURBS-Oberflächen rendern indem man diese ausreichend fein tesseliert und dann das Dreiecksnetz dazu rendert. Hierzu einige Vergleiche anhand von Bildern.

Alle diese Flächen wurden generiert indem Dreiecke aus abgetasteten Punkten der Oberfläche geformt wurden. Die Punkte wurden in uniformen

(a) 1250 Dreiecke **(b)** 5000 Dreiecke

Abbildung 18: Tesselierte NURBS-Oberflächen

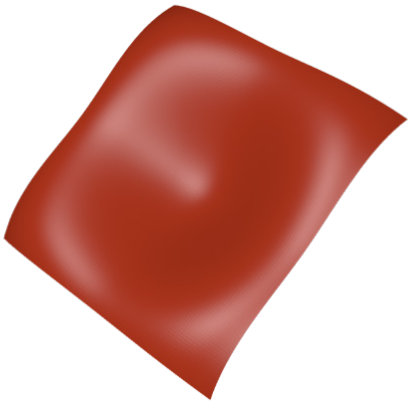


Abbildung 19: Tesselierte NURBS-Oberfläche mit 20000 Dreiecken

Abständen generiert. Beim Rendern der Dreiecksnetze wurde eine *bounding box* verwendet um zu prüfen ob das Netzt überhaupt getroffen wird. Dies gilt ebenfalls für die direkt gerenderte NURBS-Oberfläche. Beim betrachten der Oberflächen war erst ab ca. 31250 Dreiecken (bei einer Auflösung von 1920x1080px) kaum noch zu erkennen dass diese aus Dreiecken besteht. Bei allen anderen mit weniger Dreiecken ist bei genauerem betrachten zu erkennen dass diese aus Dreiecken besteht. Dies fällt voralllem bei den spekularen Teilen der Beleuchtung auf.

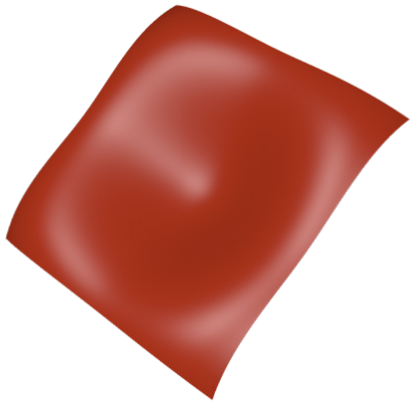


Abbildung 20: Tesselierte NURBS-Oberfläche mit 31250 Dreiecken

Allerdings ist nicht nur die Qualität der Ergebnisse wichtig, sondern auch wie lange die Zeit zum rendern ist.

Berechnungsmethode	Oberflächen	Dreiecke	Renderzeit(in s)
Tesselierung	1	1250	1.87
Tesselierung	1	5000	2.13
Tesselierung	1	11250	2.67
Tesselierung	1	20000	3.04
Tesselierung	1	31250	3.69
Schnittpunktberechnung	1	1250	4.04
Schnittpunktberechnung	3	3 x 1250	15.11

Tabelle 1: Renderzeiten im Vergleich

Alle Tests wurden auf einem Intel Core i3-4130 unter gleichen Bedingungen gerendert. Somit ist das Rendern über die direkte Schnittpunktberechnung in diesem Fall ca. 17,6% langsamer als die tesselierte Fläche mit nahezu ähnlicher Qualität. Der Mehraufwand hierbei entsteht durch die im Vergleich teure Newton-Iteration. Hierbei müssen unter Umständen mehrfach die Ableitungen für einen einzigen Punkt berechnet werden. Daraus lässt sich schließen dass sich die direkte Schnittpunktberechnung nur dann lohnt, wenn man die höhere Qualität benötigt oder die Eigenschaften der Oberfläche für weitere Effekte benötigt.

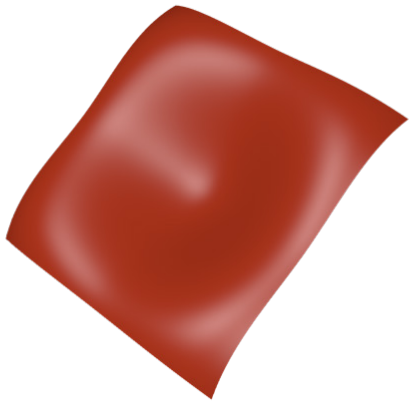


Abbildung 21: NURBS-Oberfläche mittels Newton-Iteration gerendert

8 Fazit

In dieser Arbeit wurde sich damit befasst wie NURBS-Kurven und -Oberflächen aufgebaut sind und wie sich diese berechnen lassen. Zusätzlich wurde gezeigt wie Basisfunktionen berechnet und abgeleitet werden können. Ebenfalls wurde auf das Trimmen von Oberflächen mittels verschiedener Primitive beschrieben und gezeigt. Hinzu kommt noch das Trimmen mittels Spline-Kurven, welches zur Berechnung Verwendung vom Newton-Verfahren macht. Das Newton-Verfahren wurde in normaler Form und auch in mehrdimensionaler Form beschrieben. Über das mehrdimensionale Newton-Verfahren wurde ein Verfahren zur Schnittpunktberechnung von Strahlen mit NURBS-Oberflächen hergeleitet. Abschließend wurde der Aufwand des renderns der Oberflächen analysiert und mit einer alternative abgewogen. Zum Schluss sollte erwähnt werden dass moderne *Ray Tracer* oftmals zum rendern von Dreiecksnetzen Bäume verwenden um die Renderzeit zu reduzieren. Allerdings lassen sich ähnliche Ideen auch auf die direkte Schnittpunktfindung anwenden indem man zum Beispiel *bounding volume* Hierarchien aufbaut. Jedoch hat bereits die vorhandene Implementierung einige Grenzen beider Methoden aufgezeigt.

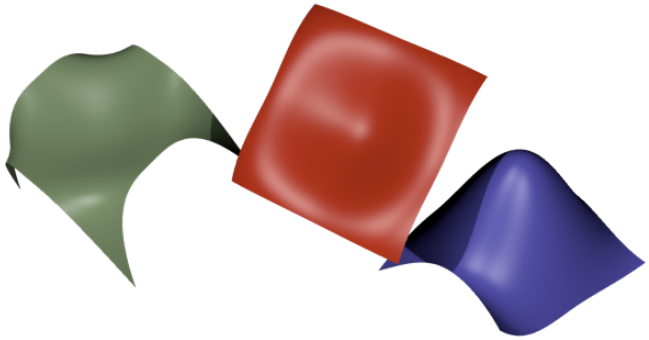


Abbildung 22: Szene mit drei Oberflächen

Literatur

[1] Autodesk Mercedes render, . URL https://gallery.autodesk.com/fusion360/projects/25031/mercedes-slr-stirling-moss. Online;Aufgerufen am: 2019-02-01.

[2] Autodesk show room car render, . URL https://www.autodesk.com/solutions/product-design/automotive. Online;Aufgerufen am: 2019-02-01.

[3] Bartels, R.H., J.C. Beatty und B.A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Series in Comp. Elsevier Science, 1995. ISBN 9781558604001. URL https://books.google.de/books?id=9bQ0f8sYqaAC.

[4] de Boor, C. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York, 2001. ISBN 9780387953663. URL https://books.google.de/books?id=m0QDJvBI_ecC.

[5] Floater, Michael S. Mathematical methods in computer aided geometric design ii. Kapitel Evaluation and Properties of the Derivative of a NURBS Curve, Seite 261–274. Academic Press Professional, Inc., San Diego, CA, USA, 1992. ISBN 0-12-460510-9. URL http://dl.acm.org/citation.cfm?id=133698.133717.

[6] Kajiya, James T. Ray tracing parametric patches. *SIGGRAPH Comput. Graph.*, 16(3):245–254, July 1982. ISSN 0097-8930. URL http://doi.acm.org/10.1145/965145.801287.

[7] Martin, William, Elaine Cohen, Russell Fish und Peter Shirley. Practical ray tracing of trimmed nurbs surfaces. *J. Graph. Tools*, 5(1):27–52, January 2000. ISSN 1086-7651. URL http://dx.doi.org/10.1080/10867651.2000.10487519.

[8] Olver, P.J. *Applications of Lie Groups to Differential Equations*. Applications of Lie Groups to Differential Equations. Springer New York, 2000. ISBN 9780387950006. URL https://books.google.de/books?id=sI2bAxgLMXYC.

[9] Peterson, John W. Graphics gems iv. Kapitel Tessellation of NURB Surfaces, Seite 286–320. Academic Press Professional, Inc., San Diego, CA, USA, 1994. ISBN 0-12-336155-9. URL http://dl.acm.org/citation.cfm?id=180895.180921.

[10] Piegl, Les und Wayne Tiller. *The NURBS Book (2Nd Ed.)*. Springer-Verlag, Berlin, Heidelberg, 1997. ISBN 3-540-61545-8.

[11] Rogers, D.F. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Elsevier Science, 2001. ISBN 9781558606692. URL https://books.google.de/books?id=DiyxPUiKvB8C.

[12] Yang, C. G. On speeding up ray tracing of b-spline surfaces. *Comput. Aided Des.*, 19(3):122–130, April 1987. ISSN 0010-4485. URL http://dx.doi.org/10.1016/0010-4485(87)90196-5.