Institut für Mathematik
der Universität zu Lübeck

Interaktive und automatisierte Hypergraphenvisualisierung mittels NURBS-Kurven

Diplomarbeit

vorgelegt von
Ronny Bergmann

Betreuer
PD Dr. Hanns-Martin Teichert
Institut für Mathematik

Lübeck, September 2009

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Lübeck, 23. September 2009

Kurzfassung

Für wissenschaftliche Arbeiten in der Graphentheorie, vor allem bei den allgemeineren Hypergraphen, sind Darstellungen zur Erklärung und Erläuterung von großer Bedeutung. Diese Arbeit schafft daher die Grundlagen einer Darstellung von Hypergraphen, die in dem Programm „Gravel“, einem Editor für ebendiese, umgesetzt wurden.

Für Bilder von Hypergraphen werden in dieser Arbeit theoretische Grundlagen entwickelt, die auf Basis der NURBS-Kurven eine Darstellung der Hyperkanten ermöglicht. Dazu werden die Eigenschaften von periodischen NURBS-Kurven betrachtet und Algorithmen zur interaktiven Modifikation vorgestellt. Der Begriff des Hyperkantenumrisses und seiner Gültigkeit wird eingeführt, um formelle Anforderungen an die Darstellung zu schaffen.

Auf Grundlage der genannten Begriffe wird das Programm „Gravel“ zum Zeichnen von Graphen und Hypergraphen entwickelt und implementiert.

Inhaltsverzeichnis

Der Arbeit liegt das Programm „Gravel“ auf einer CD bei.

Kapitel 1.

Einleitung

Die übersichtliche Darstellung von Informationen ist für das Verständnis des präsentierten Sachverhaltes stets von Bedeutung. Besonders in wissenschaftlichen Veröffentlichungen und bei Lernmaterialien ist dies von großem Interesse, da komplizierte Sachverhalte erläutert und verständlich dargebracht werden sollen. In vielen Bereichen finden sich dabei Graphen zur Modellierung von Szenarien und Zusammenhängen.

Graphen stellen in einer Menge von ähnlichen Objekten Beziehungen her, die jeweils zwei der Objekte in einen Kontext setzen. Dies kann in der Betrachtung von Straßen- oder Schienennetzen die Verbindung von Kreuzungen oder Stationen sein oder in einem Ökosystem die Betrachtung von Räuber-Beute-Beziehungen [11]. Das Ziel der Betrachtung als Graph ist dann einerseits, anhand der diskreten Struktur Eigenschaften zu erkennen oder Algorithmen zu implementieren, andererseits aber auch eine vereinfachte Darstellung zu finden. Sie dient der Konzentration auf die betrachteten Merkmale und in einer Ausarbeitung der klaren und übersichtlichen Präsentation der Ergebnisse.

Entstehen in einem Szenario Beziehungen zwischen mehr als zwei Objekten, wie etwa bei der Betrachtung von Konkurrenzbeziehungen der Jäger in einem Ökosystem, so erhält man den allgemeineren Hypergraphen. Fasst man alle Jäger derselben Beute in einer Menge zusammen, erhält man den Konkurrenzhypergraphen (siehe Sonntag u. Teichert [27]). In der Betrachtung von Hypergraphen sind einige Sachverhalte genauer spezifizierbar als in Graphen, etwa das Bindungsverhalten von Molekülen (siehe Danos u. Laneve [12]). Auch hier bleiben die Ziele erhalten, einerseits über die Modellierung Eigenschaften zu betrachten und andererseits eine abstrakte, vereinfachte Darstellung zu erzeugen.

Für die Darstellung von Graphen existieren die ersten Editoren seit Mitte der 1980er Jahre (siehe Rowe u. a. [25]); diese wurden ständig den aktuellen Systemgegebenheiten angepasst. So ist von Himsolt [18] der Editor „Graphlet“ sehr verbreitet im Einsatz und ebenso yEd (der yWorks GmbH [28]), der aufgrund seiner Implementierung in Java plattformunabhängig ist. All diese Editoren bieten vielfältige Möglichkeiten zur Bearbeitung,

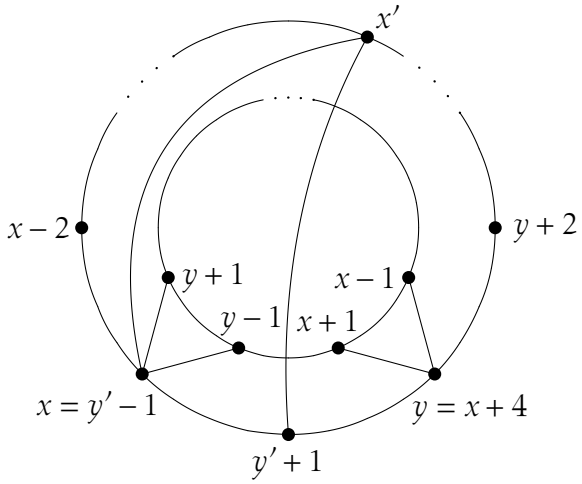


Abbildung 1.1: Eine mit „Gravel“ erstellte Darstellung eines Graphen, aus Schiermeyer u. a. [26]

sind jedoch in ihrem Funktionsumfang derart mächtig, dass eine längere Einarbeitungszeit notwendig ist.

Neben diesen Editoren hat sich in der gleichen Zeit, also seit Mitte der 1980er Jahre, das Gebiet des Graphenzeichnens etabliert, welches sich mit der automatisierten Erzeugung von Darstellungen eines Graphen beschäftigt. Eine umfangreiche Abhandlung dazu findet sich bei di Battista u. a. [3]. Die Ideen und Algorithmen sind in „GraphViz“ (siehe Gansner u. North [16]) implementiert, einer Bibliothek, die sich per Kommandozeile aufrufen und so einbinden lässt. Für Graphen bzw. deren Darstellungen gibt es also bereits umfangreiche Möglichkeiten zur Erzeugung, Be- und Verarbeitung. Die Algorithmen zur automatisierten Erzeugung sind dabei meist entweder für spezielle Szenarien oder für eine Klasse von Graphen konstruiert worden.

Mit der Zielsetzung, einen einerseits plattformunabhängigen Editor für Graphen zu entwickeln, der andererseits eine einfache Bedienbarkeit und Weiterverwendung der Grafiken in wissenschaftlichen Arbeiten bietet, entstand in der vorhergehenden Studienarbeit der Editor „Gravel“ [5]. Ein Schwerpunkt des Programms liegt auf dem Export in ein Vektorgrafikformat, das in LaTeX eingebunden werden kann. „Gravel“ wurde bereits verwendet, so ist die Grafik aus Abbildung 1.1 zusammen mit den restlichen Grafiken von Schiermeyer u. a. [26] mit „Gravel“ erzeugt worden. Der dabei zur Erzeugung notwendige Zeitaufwand war bei weitem geringer, als bei den vorherigen Erarbeitungen der Autoren.

Zielsetzung

Zielsetzung dieser Diplomarbeit ist die Erweiterung der Software „Gravel“, so dass die Erzeugung und Bearbeitung von Hypergraphendarstellungen möglich wird. Dabei soll die Einfachheit des bisherigen Editors beibehalten werden. Für die Darstellung einer Hyperkante bieten sich dabei periodische NURBS-Kurven an, da die NURBS-Kurven eine im Bereich des computergestützten Entwurfs (Computer Aided Design, CAD) weit verbreitete Modellierungsform von Kurven sind. Mit ihnen lassen sich geometrische Grundformen erzeugen, die anschließend bearbeitet werden können. Allerdings benötigen Hyperkanten periodische Kurven, die durch die genannten NURBS-Kurven nicht direkt realisierbar sind. Mit einer Anpassung der ursprünglichen Definition ist aber auch die Periodizität dieser Kurven möglich. Zur Modifikation der NURBS-Kurven und Überprüfung einer Hyperkantendarstellung sollen Algorithmen entworfen werden, die eine interaktive Bearbeitung ermöglichen. Ein weiterer Schwerpunkt liegt im Export der entstandenen Darstellung als Pixel- oder Vektorgrafik, vor allem auch zur Weiterverwendung in LaTeX.

Gliederung dieser Arbeit

Zunächst werden in Kapitel 2 die Grundlagen der Graphen, Hypergraphen und NURBS-Kurven vorgestellt, die eine Verallgemeinerung der B-Splines sind. Aufbauend darauf werden in Kapitel 3 periodische NURBS-Kurven betrachtet. Diese periodischen Kurven bieten die Vorteile der NURBS-Kurven, etwa deren numerische Stabilität in der Auswertung, und liefern außerdem eine wichtige Grundlage für die Darstellung von Hyperkanten. Nach den Anforderungen und Veränderungen, die periodische NURBS-Kurven an allgemeine Algorithmen stellen, werden in Kapitel 4 Algorithmen vorgestellt, die eine teilweise automatisierte, hauptsächlich jedoch interaktive Modifikation ermöglichen. Zentral ist dabei der Algorithmus der Projektion eines Punktes auf die (periodische) NURBS-Kurve, also die Berechnung des Punktes auf der Kurve mit kürzestem Abstand zum erstgenannten Punkt.

Die periodischen NURBS-Kurven ermöglichen dann die in Kapitel 5 ausgeführten Erläuterungen zu der Darstellung von Hyperkanten. Es wird der Begriff des Hyperkantenumrisses eingeführt sowie die Definition seiner Gültigkeit. Zur Überprüfung derselben wird dann ein Algorithmus vorgestellt, welcher auf der Projektion basiert. Abschließend behandelt das Kapitel 6 die Implementierung der vorgestellten Algorithmen und deren Anwendungen im Programm.

Kapitel 2.

Mathematische Grundlagen

In diesem Kapitel werden die grundlegenden Begriffe, Eigenschaften und Algorithmen präsentiert, die in dieser Arbeit verwendet werden. Neben den Graphen und Hypergraphen sind dies vor allem die B-Splines und NURBS. Für die verwendeten numerischen Algorithmen, wie der Newtoniteration, finden sich detaillierte Ausführungen in Opfer [22].

2.1. Graphen

Definition 2.1.
Ein *Graph* $G = (V, E)$ besteht aus einer Menge V von Knoten und einer Menge E von Kanten mit $E \subseteq \{e \mid e \subseteq V \wedge |e| = 2\}$. Für $e = \{p, q\} \in E$ sind p und q *Endknoten* der Kante e. Zwischen zwei Knoten $p, q \in V$ existiert maximal eine Kante.

Je nach Literatur heißen die Knoten auch Ecken des Graphen. Diese werden durch die Kanten zueinander in Beziehung gesetzt, was einer symmetrischen Relation auf den Knoten entspricht.

Zusätzlich gibt es viele Variationen in der Definition, von denen die folgenden Varianten häufig Verwendung finden:

gerichteter Graph. Bei *gerichteten Graphen* oder *Digraphen* $G = (V, A)$ besteht ein Bogen aus dem Tupel $a = (p, q) \in A$. Der Knoten $p \in V$ ist der *Startknoten*, $q \in V$ der *Endknoten* des Bogens. Dieser verläuft also *von p nach q* und ist somit eine *gerichtete Kante*. Bei Digraphen wird die Symmetrie der durch die Kanten gegebenen Relation aufgehoben.

Schlingen. Kanten bzw. Bögen, die einen Knoten $p \in V$ mit sich selbst verbinden, heißen *Schlingen*. Dadurch können Knoten mit sich selbst in Beziehung gesetzt werden. Formell wird dazu die Definition der Kantenmenge verändert zu $E \subseteq \{e \mid e \subseteq V \wedge 1 \leq |e| \leq 2\}$ und eine Schlinge ist eine Kante $e = \{p\} \in E$ mit $|e| = 1$.

Mehrfachkanten. Wird das Einfügen mehrerer Kanten $e_1 = e_2 = \{p, q\}$ zwischen zwei Knoten $p, q \in V$ in die Kanten(multi)menge E zugelassen, spricht man von *Mehrfachkanten* im Graphen G. Analog gibt es bei Digraphen die Erweiterung zu Mehrfachbögen $e'_1 = e'_2 = (p, q) \in A$ zwischen zwei Knoten $p, q \in V$.

Weiter heißen zwei verschiedene Knoten *adjazent*, wenn sie durch eine Kante verbunden sind. Zwei verschiedene Kanten heißen *adjazent*, wenn sie einen Knoten gemeinsam haben. Weitere Begriffe und Variationen eines Graphen G finden sich in Diestel [13], Kapitel 0.

Die mathematische Definition eines Graphen ist unabhängig von ihrer Darstellung. In jener lassen sich hingegen viele Eigenschaften untersuchen und beweisen, etwa der Zusammenhang von Graphen (siehe Diestel [13], Kapitel 2) oder Hamiltonkreise ([13], Kapitel 8). Eine jeweils illustrierende Grafik soll dann den Sachverhalt möglichst klar und verständlich darstellen. Üblicherweise werden Knoten als Punkte dargestellt, Kanten als Verbindungen zwischen den Punkten.

Beispiel 2.1.
Für die Knoten $V = \{v_1, \ldots, v_{10}\}$ entsteht der *Petersen-Graph* $G = (V, E)$ durch die Kantenmenge $E = E_1 \cup E_2$ mit

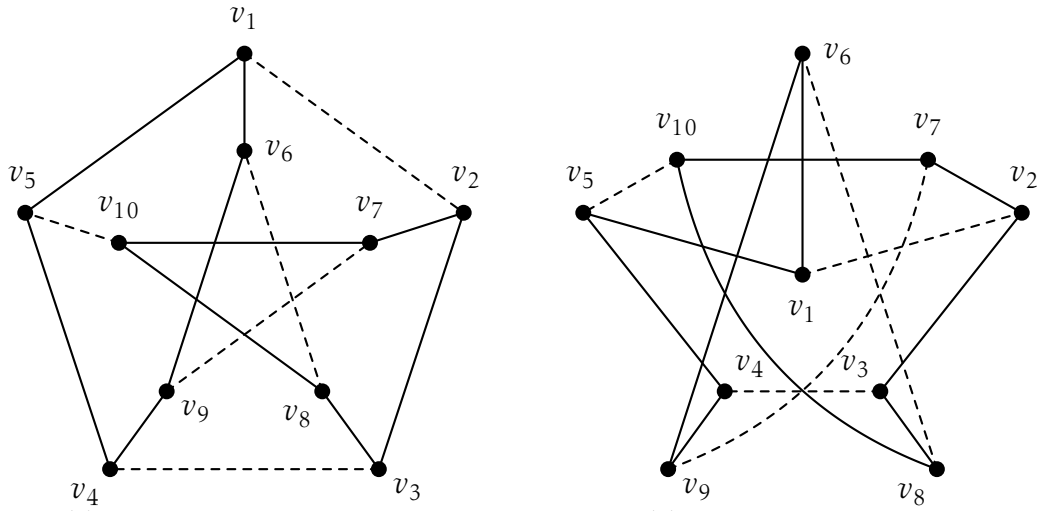
$$E_1 = \{\{v_i, v_{i+1 \mod 5}\}, \{v_i, v_{i+5}\} \mid i = 1, \ldots, 5\} \text{ und}$$
$$E_2 = \{\{v_6, v_8\}, \{v_8, v_{10}\}, \{v_{10}, v_7\}, \{v_7, v_9\}, \{v_9, v_6\}\}.$$

Weiter kommt in der Kantenteilmenge

$$E' = \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_{10}\}, \{v_6, v_8\}, \{v_7, v_9\}\}$$

jeder Knoten genau einmal vor.

In der Abbildung 2.1 sind zwei Bilder dieses Graphen dargestellt. Der äußere Kreis und die Verbindungen nach innen sind in 2.1(a) durch E_1 gegeben, der innere Stern durch E_2. Die Kantenmenge E' ist hervorgehoben. In Abbildung 2.1(b) ist eine Darstellung des gleichen Graphen zu sehen, die diese Erläuterung nicht so klar präsentiert und – aufgrund einer höheren Anzahl Kantenkreuzungen – unübersichtlicher wirkt.



(a) symmetrische Darstellung (b) chaotische Darstellung

Abbildung 2.1: Darstellung des Petersen-Graphen und seiner Kantenteilmenge E' aus Beispiel 2.1 (hervorgehoben): (a) in der üblichen Darstellung und (b) in einer unübersichtlichen Darstellung in der einige Knotenpaare getauscht wurden.

2.2. Hypergraphen

Definition 2.2 (Hypergraph).
Ein *Hypergraph* $\mathcal{H} = (V, \mathcal{E})$ besteht aus einer Menge V von *Knoten* und einer Menge $\mathcal{E} \subseteq \mathcal{P}(V) \backslash \emptyset$ von *Hyperkanten*. Eine Hyperkante $E \in \mathcal{E}$ besteht somit aus einer nichtleeren Menge von Knoten.

Besteht eine Hyperkante $E \in \mathcal{E}$ aus nur einem Knoten ($|E| = 1$), so heißt sie auch *Schlinge*, im Fall $|E| = 2$ entspricht die Hyperkante der normalen Kante eines Graphen.

Beispiel 2.2 (**Ein Hypergraph, aus [4]**).
Der Hypergraph $\mathcal{H} = (V, \mathcal{E})$ mit $V = \{v_i \mid i = 1, \ldots, 8\}$ und den Hyperkanten $\mathcal{E} = \{E_i \mid i = 1, \ldots, 6\}$ mit

$$E_1 = \{v_3, v_4, v_5\}, \qquad E_2 = \{v_5, v_8\}$$
$$E_3 = \{v_6, v_7, v_8\}, \qquad E_4 = \{v_2, v_3, v_7\}$$
$$E_5 = \{v_1, v_2\}, \qquad E_6 = \{v_7\}$$

enthält die Schlinge E_6. Die beiden Hyperkanten E_2 und E_5 entsprechen zwei Kanten eines Graphen G. Eine Darstellung findet sich im Beispiel 5.1, S. 54f.

Den Hypergraph erhält man auf zwei verschiedene Arten aus der Betrachtung eines Graphen heraus:

Verallgemeinerung. Bei Graphen $G = (V, E)$ bzw. Digraphen $G' = (V, A)$ werden jeweils zwei Knoten durch eine Kante bzw. einen Bogen in eine binäre Beziehung gesetzt, die im ersten Fall symmetrisch ist. Für die Betrachten von tertiären oder noch größeren Beziehungen, wie sie etwa in relationalen Datenbanken oder der Molekularbiologie (s. Danos u. Laneve [12]) auftreten, bieten Graphen keine Modellierungsmöglichkeit.

Hypergraphen hingegen bieten diese Möglichkeit der Modellierung. Dabei wird der Graph zu einem Spezialfall des Hypergraphen, nämlich genau dann, wenn gilt

$$\forall E \in \mathcal{E} : |E| = 2.$$

Die gerichteten Beziehungen eines Digraphen lassen sich ebenso in einem *gerichteten Hypergraphen* $\mathcal{H} = (V, \mathcal{A})$ verallgemeinern [15]. Ein Hyperbogen $A = A_S \cup A_E \in \mathcal{A}$ besteht dann aus den disjunkten Mengen A_S und A_E der Start- und Endknoten des Hyperbogens. Auch hier verbleibt der Digraph als Spezialfall des gerichteten Hypergraphen, genau dann, wenn alle Hyperbögen $|A_S| = |A_E| = 1$ erfüllen.

Zusatzinformationen. Für einen Graphen bzw. Digraphen G mit den Knoten V schafft man eine zusätzliche Informationsebene, die über diejenige des Graphen hinausgeht und auf der gleichen Knotenmenge V zusätzliche Beziehungen oder Zusammenhänge mathematisch modelliert.

Beispielsweise wird mit dem *Konkurrenzhypergraph* für einen Digraphen $G = (V, A)$ ein Hypergraph $\mathcal{CH} = (V, \mathcal{E})$ definiert. Es wird je eine Hyperkante $E_v \in \mathcal{E}$ für einen Knoten $v \in V$ eingeführt wird, der mindestens zwei eingehende Kanten besitzt. Diese sind definiert durch

$$E_v = \{u | u \in V \wedge \exists a = (u, v) \in A\}.$$

Somit ist $|E_v| \geq 2$. Nach dieser Konstruktion ist über den Digraphen G ein Hypergraph entstanden, der in seinen Hyperkanten weitere Möglichkeiten zur Analyse des Digraphen bietet. Analog ist für Graphen ein Hypergraph definiert, der dann eine Hyperkante für die Nachbarschaft eines jeden Knotens enthält.

Beide Herleitungen schaffen eine Betrachtung die über den Graphen selbst hinausgeht. Dies hebt auch die Namensgebung durch die Präposition „hyper“ (griech. über, darüber hinaus) hervor. Weitere Eigenschaften von Hypergraphen finden sich in Berge [4]. Da für Graphen viele Eigenschaften

bekannt und bewiesen sind, ist ein spannender Aspekt bei der Betrachtung von Hypergraphen, wie sich diese Eigenschaften übertragen lassen, angepasst werden müssen oder nicht anwendbar sind.

2.3. Graphzeichnen

Als Teilbereich der Computergrafik beschäftigt sich das *Graphzeichnen* (engl. graph drawing) mit der Darstellung von Informationen, die durch einen Graphen gegeben sind. Dabei werden für unterschiedliche Klassen von Graphen Algorithmen zur automatisierten Erzeugung der Darstellungen entwickelt, untersucht und bewertet. Diese Klassen ergeben sich beispielsweise aus der Art und den Eigenschaften des modellierten Sachverhaltes. Neben diesen vollautomatischen Algorithmen existieren auch welche, bei denen der Benutzer im Ablauf des Algorithmus in die Erzeugung der Darstellung eingreifen kann.

Die Darstellung von Graphen begrenzt sich üblicherweise auf ein Bild, also eine Positionierung der Knoten $v \in V$ eines Graphen $G = (V, E)$ bzw. Digraphen $G = (V, A)$ im $\mathbb{R}^2$ und davon ausgehend eine Darstellungsfindung für die Kanten. Es ist, wie in den folgenden Anforderungen dargestellt, nicht immer sinnvoll, für eine Kante $e = \{v_1, v_2\}$ die Strecke $\overline{p_1 p_2}$ der Positionen $p_1, p_2 \in \mathbb{R}^2$ zu verwenden, die den Knoten zugeordnet wurden.

Ein zentraler Punkt des Graphzeichnens sind die Anforderungen an eine Darstellung, die von Anwendungsfall zu Anwendungsfall variieren können. Diese werden von di Battista u. a. [3] in Kapitel 2 dargestellt und umfassen beispielsweise

- Positionierung von Knoten, die, so sie eine Kante gemeinsam haben, nicht zu weit auseinander positioniert werden sollten.
- Vereinbarungen zur Darstellung von Kanten, so wird in technischen Zeichnungen eine Kante stets als Folge von orthogonal zueinander verlaufenden Linien dargestellt.
- Ästhetische Eigenschaften, wie etwa die eingenommene Fläche, die weder zu klein noch zu groß sein sollte, die möglichst minimale Anzahl an Kantenüberkreuzungen oder Darstellung von Symmetrien des Graphen.
- Beschränkungen an die Kantenlänge und den Winkel, den zu einem Knoten inzidente Kanten in der Darstellung an diesem Knoten besitzen.

- Zeitliche Einschränkung an den Algorithmus, der *effizient* sein soll. Andernfalls ist die bereits angesprochene interaktive Erzeugung eines Graphen durch mehrfaches Anwenden verschiedener Algorithmen nicht oder nur noch bedingt möglich.

Einige Anforderungen stehen dabei zueinander in Konflikt. Positioniert man Knoten derart, dass sie der Übersichtlichkeit halber je paarweise einen gewissen Abstand einhalten, kann die Anforderung an die Fläche eventuell nicht mehr eingehalten werden. Hier sind also Kompromisse notwendig.

Für Graphen sind verschiedene Ansätze und Algorithmen zur Erzeugung von Darstellungen bekannt, die wesentlichen davon behandeln di Battista u. a. [3] ausführlich. Auch hier stellt sich für Hypergraphen die Frage, in wiefern die genannten Anforderungen übernommen oder angepasst werden können und wo neue Anforderungen notwendig sind.

2.4. Kurven

Für die Darstellung von Kanten ist oft eine Gerade üblich. Es werden jedoch auch Kurven verwendet, wenn die direkte Strecke zwischen den Knoten bereits von vielen Elementen belegt ist. Das ist zum Beispiel der Fall, wenn die beiden Endknoten einer Kante an gegenüberliegenden Rändern einer Komponente des Bildes platziert worden sind. Dann wird eine Kante verallgemeinert dargestellt durch eine Kurve.

Definition 2.3 (Kurve im $\mathbb{R}^2$).
Seien $a, b \in \mathbb{R}$, $I = [a, b]$ sowie $x : I \to \mathbb{R}$ und $y : I \to \mathbb{R}$ stetige Funktionen. Eine *Kurve* ist die Abbildung

$$C : I \to \mathbb{R}^2, \quad t \mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \quad t \in I. \tag{2.1}$$

Die Kurve heißt *differenzierbar*, wenn ihre Komponentenfunktionen differenzierbar sind.

Weiter heißt eine Kurve *periodisch*, genau dann, wenn ein $h \in \mathbb{R}$ existiert, so dass für ein Intervall $[a, b]$ mit $b - a = h$ gilt:

$$\forall u \in [a, b] \;\; \forall i \in \mathbb{N} : C(u) = C(u + ih).$$

Zur Darstellung von Kurven am Computer sind einige Anforderungen an diese notwendig, da die allgemeine Definition keine einfache und numerisch stabile Speicherung und Verarbeitung bietet.

2.5. B-Splines

Eine Möglichkeit ist die Darstellung von Kurven durch Polynome. Wird eine Kurve durch eine Anzahl von n Punkten spezifiziert, so ist ein Polynom vom Grad $n-1$ notwendig, um eine Kurve zu erhalten, die alle Punkte interpoliert. Dies führt jedoch für größere Grade n zu einem hohen Polynomgrad. Polynome hohen Grades sind jedoch weder effizient zu verarbeiten noch führt eine Interpolation zu schönen Ergebnissen.

Daher werden im folgenden Kurven als Summe von sich überlagernden Polynomen dargestellt, die einen gemeinsamen Grad $d \in \mathbb{N}$ besitzen, der relativ klein ist. Die Polynome werden so konstruiert, dass die entstehende Kurve sich numerisch stabil verarbeiten lässt und dass sie hinreichend oft stetig differenzierbar ist. Die im Folgenden verwendete Notation sowohl der B-Splines als auch der NURBS ist analog zu den Ausführungen in „The $\mathcal{NURBS}$-Book“ verwendet worden [23].

Definition 2.4 (Knotenvektor t).
Seien $a, b \in \mathbb{R}$; $m, d \in \mathbb{N}$ mit $2d \leq m$.
Aus einer Folge reeller Zahlen $t_0 \leq t_1 \leq \ldots \leq t_m$ mit

$$a = t_0 = t_1 = \cdots = t_d, \quad t_{m-d} = t_{m-d-1} = \cdots = t_m = b$$

bildet man den *Knotenvektor* $t = (t_i)_{i=0}^{m}$. Die Elemente $t_i, i = 0, \ldots, m$ des Vektors heißen *Knoten*. Von diesen werden $t_0, t_1, \ldots, t_d, t_{m-d}, t_{m-d+1}, \ldots, t_m$ *Endknoten* und $t_i, i = d+1, \ldots, m-d-1$ *innere Knoten* genannt.

Definition 2.5 (B-Spline-Basisfunktionen).
Bezüglich des Knotenvektors $(t_i)_{i=0}^{m}$ wie in Definition 2.4 sind die *B-Spline-Basisfunktionen* $N_{i,d,t}(u)$ vom Grad d rekursiv definiert durch

$$N_{i,0,t}(u) = \begin{cases} 1 \text{ für } u \in [t_i, t_{i+1}) \text{ falls } t_{i+1} < t_{m-d} \\ 1 \text{ für } u \in [t_i, t_{i+1}] \text{ falls } t_{i+1} = t_{m-d} \\ 0 \text{ sonst} \end{cases} \tag{2.2}$$

und

$$N_{i,d,t}(u) = \frac{u - t_i}{t_{i+d} - t_i} N_{i,d-1,t}(u) + \frac{t_{i+d+1} - u}{t_{i+d+1} - t_{i+1}} N_{i+1,d-1,t}(u) \tag{2.3}$$
$$\text{für } i = 0, 1, \ldots, n \leq m - d - 1 \text{ und } d \geq 1.$$

Für den Fall, dass $u - t_i = t_{i+d} - t_i = 0$ oder $t_{i+d+1} - u = t_{i+d+1} - t_{i+1} = 0$ wird der jeweilige Faktor in (2.3) als 0 definiert.

Einem Element t_i des Knotenvektors wird mit p_i seine *Vielfachheit* zugeordnet. Diese ist die Häufigkeit des Wertes t_i im Knotenvektor $(t_i)_{i=0}^{m}$.

Im Folgenden wird der Fall $i+1=m$ nicht explizit gesondert betrachtet, für diesen Fall ist das Intervall stets geschlossen.

In den meisten Fällen ist der Knotenvektor aus dem Kontext heraus eindeutig. Dann wird t nicht explizit im Index angegeben und die Funktion $N_{i,d,t}(u)$ wird verkürzt beschrieben durch $N_{i,d}(u)$.

Eigenschaften der B-Spline-Basisfunktionen

Die B-Spline-Basisfunktionen $N_{i,d}(u)$ besitzen Eigenschaften, die im folgenden aufgezählt sind. Detaillierte Ausführungen und die Beweise der Eigenschaften finden sich in Piegl u. Tiller [23], S. 55ff.

B1 **Positivität.** Eine Basisfunktion $N_{i,d}(u),\ \ i=0,\dots,n$ ist für $u\in[t_i,t_{i+d+1})$ echt positiv und $N_{i,d}(u)=0$ sonst.

B2 **lokaler Träger.** Für ein $u\in[t_i,t_{i+1}),\ \ d\le i\le m-1$ sind lediglich die Basisfunktionen $N_{j,d}(u),\ \ j=i-d,\dots,i$ echt positiv.

B3 **Polynom.** Die B-Spline-Basisfunktionen $N_{i,d}(u),\ \ i=0,\dots,n$ sind Polynome vom Grad d.

B4 **Zerlegung der Eins.** Für ein festes $i\in\{d,\dots,m\}$ gilt

$$\forall u\in[t_i,t_{i+1}):\ \sum_{j=i-d}^{i} N_{j,d}(u)=1.$$

B5 **Stetigkeit.** $N_{i,d}(u),\ \ i=0,\dots,n$ ist an der Stelle $u=t_j$ $(d-p_j)$-mal stetig differenzierbar $(j=i-d,\dots,i)$ und beliebig oft sonst.

Beispiel 2.3 (B-Spline-Basisfunktionen).
Sei der Knotenvektor gegeben durch

$$t=(t_i)_{i=0}^{9}=\left(0\quad 0\quad 0\quad 0\quad \tfrac{1}{3}\quad \tfrac{2}{3}\quad 1\quad 1\quad 1\quad 1\right).$$

So ergibt sich $m=9, a=0$ und $b=1$. Für jeden Grad $d=0,\dots,3$ sind Basisfunktionen definiert. Die drei B-Spline-Basisfunktionen vom Grad 0 $N_{i,0}(u), i=3,4,5$ sind die einzigen von der Nullfunktion verschiedenen, da nur für diese i ein echtes Intervall existiert. Auf diesen Basisfunktionen ergeben sich über die rekursive Definition (s. Gleichung (2.2)) die Basisfunktionen vom Grad 1, wobei zusätzlich $N_{2,1}(u)\not\equiv 0$ ist.

Zusammen mit den Basisfunktionen vom Grad 2, $N_{i,2}(u),\ i = 1,\ldots,5$ finden sich diese Basen in Abbildung 2.2. Da die echten Intervalle $[0,\frac{1}{3}],[\frac{1}{3},\frac{2}{3}]$, $[\frac{2}{3},1]$ alle die gleiche Größe besitzen, heißen die Basisfunktionen in diesem Fall auch *uniform*, sonst spricht man von *nicht uniformen* B-Spline-Basisfunktionen.

Beispiel 2.4 (nicht uniforme B-Spline-Basisfunktionen).
Ein Beispiel für die *nicht uniformen* Basisfunktionen ergibt sich mit dem Knotenvektor

$$t = (t_i)_{i=0}^{9} = \Big(0 \quad 0 \quad 0 \quad 0 \quad \tfrac{1}{5} \quad \tfrac{3}{5} \quad 1 \quad 1 \quad 1 \quad 1\Big).$$

Berechnet man auf diesem Knotenvektor die Basisfunktionen $N_{i,2}(u)$, so sind sie für $i = 1,\ldots,5$ nicht identisch zur Nullfunktion. $N_{0,2}(u)$ und $N_{6,2}(u)$ entsprechen der Nullfunktion. Da $n = m-d-1 = 9-2-1 = 6$, sind dies per Definition alle Basisfunktionen. Eine Darstellung dieser Funktionen findet sich in Abbildung 2.3.

Ist der Knotenvektor $(t_i)_{i=0}^{m}$ von der Form

$$t = (\underbrace{a \quad a \quad \ldots \quad a}_{d+1-\text{Elemente}} \quad \underbrace{b \quad b \quad \ldots \quad b}_{d+1-\text{Elemente}}), \quad m = 2d+1, \tag{2.4}$$

so vereinfachen sich die B-Spline-Basisfunktionen. Der Knotenvektor wird nicht mehr benötigt, denn es existiert eine geschlossene Form der Polynome, die in der folgenden Definition genannt ist.

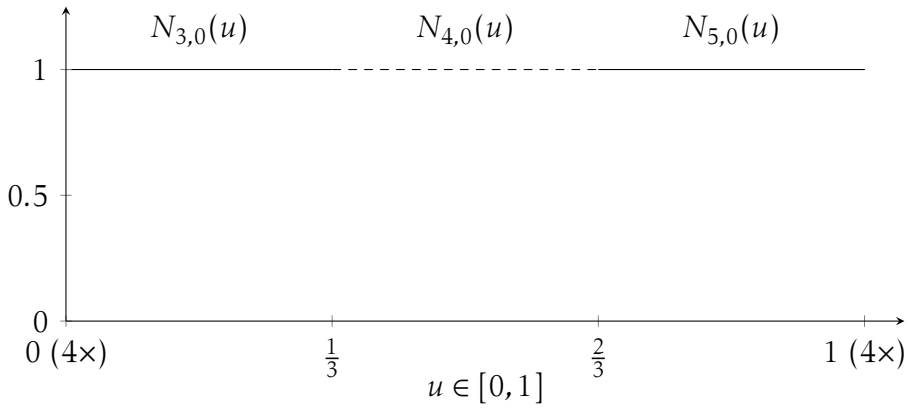
Definition 2.6 (Bernstein-Polynome).
Die Bernsteinpolynome vom Grad d sind definiert durch

$$B_{i,d}(u) = \begin{cases} a + (b-a)\binom{d}{i}(1-u)^{d-i}u^i \text{ für } 0 \le i \le n \\ 0 \text{ sonst} \end{cases}$$

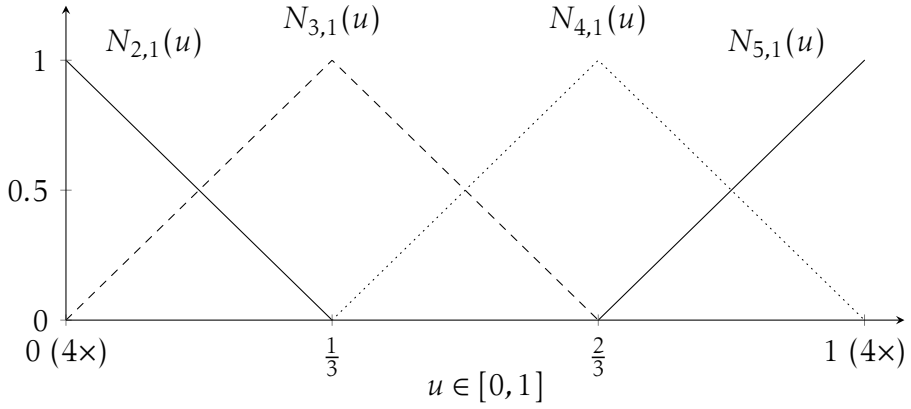
Auf den Basispolynomen $N_{i,d}(u)$ bzw. im Spezialfall den $B_{i,d}(u)$ wird nun eine Kurve definiert.

Definition 2.7 (B-Spline-Kurve).
Die *B-Spline-Kurve* vom Grad d auf dem Knotenvektor $(t_i)_{i=0}^{m}$ mit den Kontrollpunkten $P_0,\ldots,P_n, P_i \in \mathbb{R}^2,\ n = m-d-1$ ist definiert durch

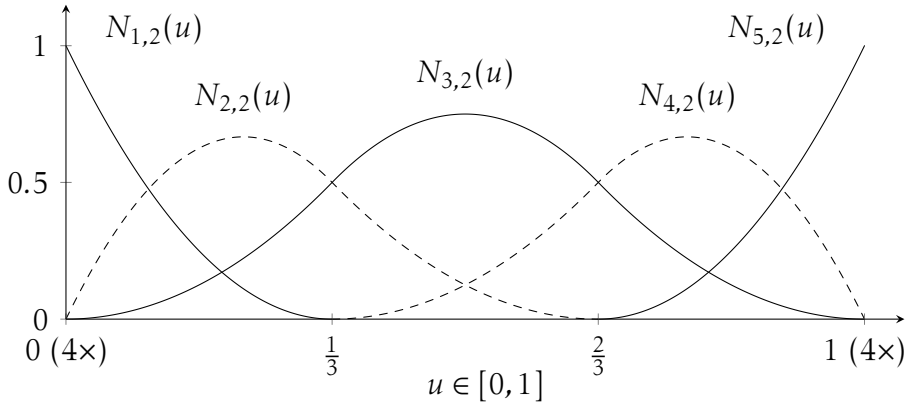
$$B(u) = \sum_{i=0}^{n} N_{i,d}(u)P_i,\ u \in [a,b]. \tag{2.5}$$



(a) Grad 0



(b) Grad 1



(c) Grad 2

Abbildung 2.2: Die B-Spline-Basisfunktionen aus dem Beispiel 2.3 mit (a) Grad 0, (b) Grad 1 und (c) Grad 2.

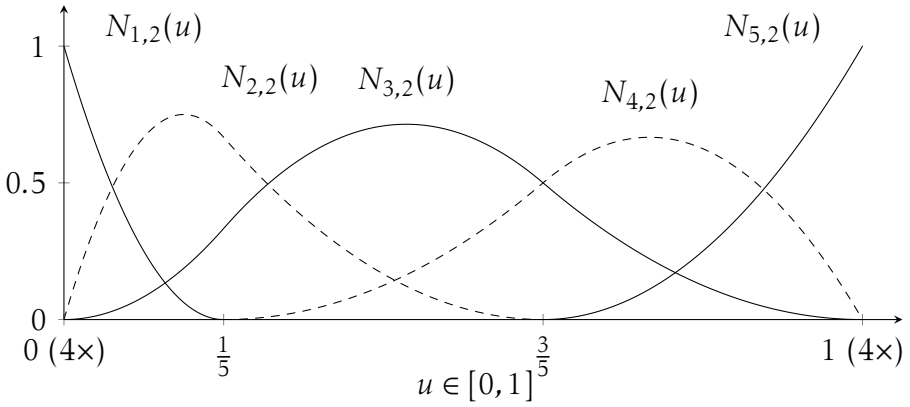


Abbildung 2.3: nichtuniforme B-Spline-Basisfunktionen vom Grad 2 auf dem Knotenvektor aus Beispiel 2.4.

Aufgrund der Eigenschaft B4, dass die B-Spline-Basisfunktionen an einer Stelle $u \in [a,b]$ eine Zerlegung der 1 bilden, werden in der B-Spline-Kurve an einer Stelle u stets $d+1$ Kontrollpunkte (durch die Basisfunktionen) gewichtet aufsummiert. Ein Punkt P_i hat genau dann den größten Einfluss auf die Kurve, wenn $N_{i,d}(u)$ maximal wird. Dadurch definieren die Kontrollpunkte zusammen mit den Basisfunktionen die Kurve. Der Knotenvektor bestimmt dabei die Verteilung auf dem Intervall $[a,b]$.

Eigenschaften der B-Spline-Kurve

Zusätzlich zu den Eigenschaften der Basisfunktionen gelten für die Kurve noch die folgenden Eigenschaften:

B6 **lokale Kontrolle.** Auf jedem Segment $[t_i, t_{i+1})$, $d \le i \le m-d-1$ wird die Kurve $B(u)$ durch die Kontrollpunkte $P_{i-d},\ldots,P_i$ bestimmt. Dies ergibt sich aus B2.

B7 **konvexe Hülle.** Für $u \in [t_i, t_{i+1})$, $d \le i < m-d-1$ liegen alle Punkte der Kurve $B(u)$ in der konvexen Hülle der Kontrollpunkte $P_{i-d},\ldots,P_i$.

B8 **Stetigkeit.** Nach B5 und B2 sind die Basisfunktionen $(d-p_i)$-mal stetig differenzierbar, daher gilt dies auch für die durch die Kontrollpunkte gewichtete Summe. Die B-Spline-Kurve ist also insgesamt $(d-p)$–mal stetig differenzierbar auf $[a,b]$, wobei $p = \max\limits_{i\in\{0,\ldots,m\}} \{p_i\}$.

B9 **Invarianz unter affin-linearen Transformationen.** Für eine affin-lineare Transformation $T(x) = Ax + b$ gilt

$$T(B(u)) = T\left(\sum_{i=0}^{n} N_{i,d}(u)P_i\right) = \sum_{i=0}^{n} N_{i,d}(u)T(P_i).$$

B10 **Endpunktinterpolation.** Es gilt stets $B(a) = P_0$, $B(b) = P_n$.

B11 **variationsvermindernde Eigenschaft.** Eine Gerade $g : \mathbb{R} \to \mathbb{R}^2$ schneidet die Kurve nicht öfter, als sie das Polygon der Kontrollpunkte schneidet.

Außer den Endpunkten werden die Kontrollpunkte im Allgemeinen nicht interpoliert, außer, wenn ein innerer Knoten eine Vielfachheit von mindestens d besitzt.

Ist der Knotenvektor wie in Gleichung (2.4), so heißt die auf den Bernstein-Polynomen basierende Kurve *Beziér-Kurve* vom Grad d mit den Kontrollpunkten P_i. Die *Beziér-Kurve* lässt sich auch als einzelnes Polynom vom Grad d angeben.

Beispiel 2.5 (Eine kubische B-Spline-Kurve).
Seien der Knotenvektor und die Kontrollpunkte gegeben durch

$$t = (t_i)_{i=0}^{9} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 & 1 & 1 & 1 \end{pmatrix},$$
$$P_0 = \begin{pmatrix} 20 \\ 70 \end{pmatrix}, \quad P_1 = \begin{pmatrix} 23 \\ 119 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 100 \\ 119 \end{pmatrix}, \quad P_3 = \begin{pmatrix} 100 \\ 21 \end{pmatrix},$$
$$P_4 = \begin{pmatrix} 177 \\ 21 \end{pmatrix}, \quad P_5 = \begin{pmatrix} 180 \\ 21 \end{pmatrix}.$$

Dann ist $m = 9$ und $n = 5$, also ergibt sich der Grad d zu $d = m - n - 1 = 3$. Zusätzlich zur Kurve sind in Abbildung 2.4 das Polygon, das von den Kontrollpunkten gebildet wird, sowie die Bilder der inneren Knoten $t_4 = \frac{1}{3}$ und $t_5 = \frac{2}{3}$ dargestellt.

Algorithmus zur Berechnung der B-Spline-Kurve

Der Algorithmus von de Boor (siehe de Boor [8], S. 131ff.) ermöglicht nun die Auswertung der entstandenen Kurve. Er ist im Algorithmus 2.1 in der obigen Notation aufgeführt und ist eine iterative Implementierung der Gleichungen (2.2), (2.3) und (2.5).

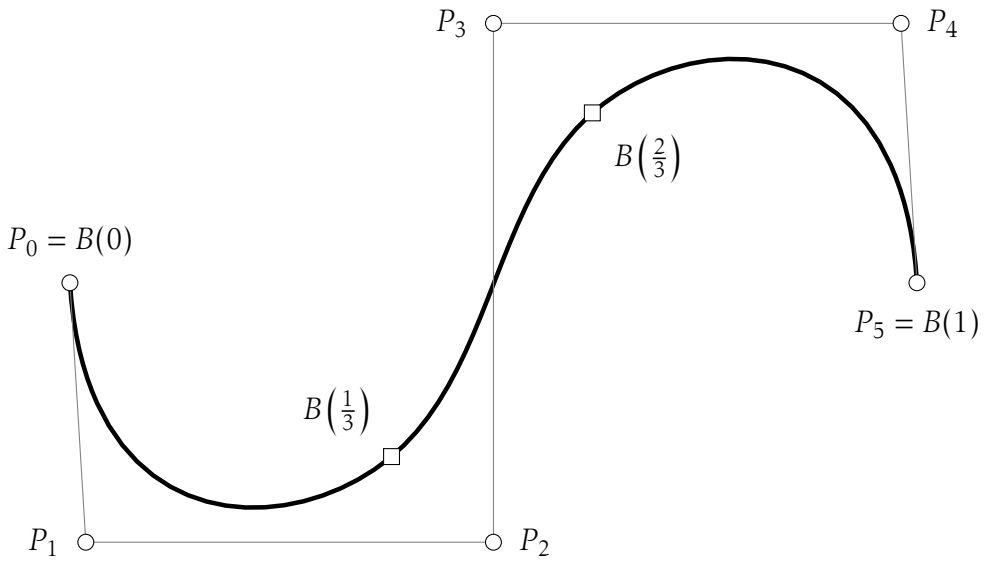


Abbildung 2.4: B-Spline-Kurve aus Beispiel 2.5: Die Kontrollpunkte sind durch ∘, die Bilder der inneren Knoten durch □ gekennzeichnet.

Der Algorithmus berechnet stets auf Grundlage der (bereits mit den Punkten gewichteten) Basispolynome vom Grad $i-1$ diejenigen vom Grad i. Dabei werden von Beginn an nur diejenigen Basispolynome betrachtet, die für das entsprechende $u \in [t_r, t_{r+1}] \subset [a,b]$ von Null verschieden sind. Im letzten Schritt entsteht in P_r^d der Punkt auf der Kurve. Dieser ist aufgrund der Lokalitätseigenschaft B2 eine gewichtete Summe der Punkte $P_r, P_{r-1}, \ldots, P_{r-d}$.

Vor- und Nachteil der B-Spline-Kurve

Ein großer Vorteil der B-Splines ist, dass diese numerisch stabil berechenbar sind, indem man zur Bestimmung eines Punktes $B(u) \in \mathbb{R}^2, \quad u \in [a,b]$ den Algorithmus von de Boor [8] verwendet. Damit kann ebenso eine Darstellung der gesamten Kurve erzeugt werden, indem man eine Menge von Punkten $(u_1, u_2, \ldots, u_k) \subset [a,b]$ derart wählt, dass

$$\exists \epsilon \in \mathbb{R}\ \forall i \in [1, \ldots, k-1] : \|B(u_i) - B(u_{i+1})\|_2 \leq \epsilon.$$

Dies ist durch stückweise Verfeinerung der Punkte möglich, indem man Mittelpunkte zwischen u_i und u_{i+1} einfügt, bis das Kriterium für ein vorgegebenes ϵ erfüllt ist. Das ϵ gibt dabei die Genauigkeit vor, die ein durch die berechneten Punkte gebildetes Polynom besitzt.

Algorithmus 2.1: De Boor-Algorithmus, nach [8, S. 131ff.]

Eingabe: Eine Position $u \in [a, b]$,
der Knotenvektor $t = (t_i)_0^m$, $t_i \in \mathbb{R}$,
die Kontrollpunkte $P_0, \ldots, P_n, \quad n < m - 1 \in \mathbb{R}, \quad P_i \in \mathbb{R}^2$

Ausgabe: Der Punkt $B(u)$ auf der B–Spline–Kurve

Daten: Der Grad d ergibt sich aus m und n mittels $d = m - n - 1$

Algorithmus
Bestimme r, so dass $u \in [t_r, t_r + 1)$
Initialisiere $P_i^0 = P_i$ und $P_{-1} = P_{n+1} = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$
Für jedes $j \leftarrow 1$ **bis** d
Für jedes $i \leftarrow r - d + j$ **bis** r

$$\alpha_i^j \leftarrow \frac{u - t_i}{t_{i+d-j+1} - t_i}$$

$$P_i^j \leftarrow \frac{u - t_i}{t_{i+d-j}}(1 - \alpha_i^j)P_{i-1}^{j-1} + \alpha_i^j P_i^{j-1}$$

Rückgabewert: P_r^d

Alle Operationen, die durch linear-affine Transformationen dargestellt werden können, lassen sich zur Manipulation von B-Spline-Kurven sehr gut verwenden, da nach Eigenschaft B9 dazu lediglich die Kontrollpunkte verändert werden müssen.

Nachteil der B-Splines ist, dass sich Kreise nur näherungsweise darstellen lassen, so dass diese Näherung zwar in der Erzeugung optisch annehmbar aussehen mag, aber durch Skalierung dann eventuell unschöne Resultate entstehen.

Dieser Nachteil führt zu einer Erweiterung der B-Splines, die im folgenden Abschnitt behandelt wird.

2.6. Nicht-uniforme rationale B-Splines (NURBS)

Für die Darstellung von Kreisen und Ellipsen ist eine Verallgemeinerung der B-Splines notwendig.

Definition 2.8 (NURBS-Kurve).
Seien $m, n \in \mathbb{N}, m > n-1$ gegeben. Die *nichtuniforme rationale B-Spline-Kurve* (NURBS-Kurve) vom Grad $d = m-n-1$ mit den Kontrollpunkten $P_0,\ldots,P_n$, den ihnen zugeordneten Gewichten $w_0,\ldots,w_n, w_i > 0$ auf dem Knotenvektor $t = (t_i)_{i=0}^{m}$ ist definiert als

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,d}(u) w_i P_i}{\sum_{j=0}^{n} N_{j,d}(u) w_j}, \quad u \in [a,b]. \tag{2.6}$$

Der Name nicht uniforme rationale B-Splines (NURBS) entsteht aus den Eigenschaften, dass der Knotenvektor nicht uniform sein muss, also die Abstände $t_{i+1} - t_i$ nicht unbedingt für alle i identisch sein müssen, und aus dem rationalen Polynom, das die Kurve definiert.

Setzt man alle Gewichte $w_i = 1,\ \ i = 0,\ldots,n$, so erhält man eine B-Spline-Kurve, denn der Divisor in Gleichung (2.6) ergibt sich aufgrund der Eigenschaft B4 stets zu 1. Für die NURBS-Kurve und ihre rationalen B-Spline-Basisfunktionen gelten ebenso die Eigenschaften B1 - B11. Details und weitere Ausführungen finden sich in Piegl u. Tiller [23], S. 117ff.

Die geometrische Interpretation der Gewichte w_i ist, dass jedem Kontrollpunkt P_i zusätzlich ein Einfluss auf die Kurve gegeben wird. Ist w_i relativ groß, hat der Kontrollpunkt P_i viel Einfluss, ist w_i relativ klein, so hat er nur geringen Einfluss auf den Verlauf der Kurve. Da je durch die Summe der Gewichte beteiligter Kontrollpunkte normiert wird, ist die Größe eines Gewichtes w_i je im Verhältnis zu den restlichen Gewichten zu sehen. Es wäre zwar möglich, einem Kontrollpunkt auch keinen ($w_i = 0$) oder negativen ($w_i < 0$) Einfluss zu geben, jedoch sind dann einige Eigenschaften nicht mehr von den B-Splines auf die NURBS übertragbar.

Da diese Darstellung manchmal etwas umständlich ist, lässt sich eine NURBS-Kurve auch auf der folgenden Menge von Basisfunktionen definieren, die aus Gleichung (2.6) entstehen.

Definition 2.9 (rationale B-Spline-Basisfunktionen).
Sei $d \in \mathbb{N}$ ein Grad. Bezüglich des Knotenvektors $(t_i)_{i=0}^{m}$ und der Gewichte $w_0,\dots,w_n$, $n = m-d-1$ sind die *rationalen B-Spline-Basisfunktionen* definiert durch

$$R_{i,d}(u) = \frac{N_{i,d}(u)w_i}{\sum_{j=0}^{n} N_{j,d}(u)w_j}, \quad u \in [a,b]. \tag{2.7}$$

Dann lässt sich die Kurve aus Gleichung (2.6) auch über diese Basisfunktionen angeben und man erhält die Form

$$C(u) = \sum_{i=0}^{n} R_{i,d}(u)P_i, \quad u \in [a,b]. \tag{2.8}$$

Analog zur Beziér-Kurve heißt die Kurve $C(u)$ mit einem Knotenvektor wie in Gleichung (2.4) *rationale Beziér-Kurve* und auch in diesem Fall vereinfachen sich die B-Spline-Basispolynome in Gleichung (2.6) zu den Bernsteinpolynomen; der Knotenvektor wird nicht mehr explizit benötigt. Die Kurve lässt sich in diesem Fall als ein rationales Polynom angeben.

Beispiel 2.6 (Variation eines Gewichtes).
Seien der Knotenvektor $t = (t_i)_{i=0}^{9}$ und die Kontrollpunkte $P_o,\dots,P_5$ gegeben wie in Beispiel 2.5. Mit den Gewichten $w_i = 1, i = 0,\dots,5$ erhält man dann dieselbe Kurve wie in dem eben genannten Beispiel.

Verändert man ausgehend von dieser Kurve das Gewicht w_3, so ändert sich der Einfluss des Kontrollpunkte $P_3 = \begin{pmatrix} 100 \\ 21 \end{pmatrix}$ auf die Kurve:

(a) Setzt man das Gewicht auf $w_3 = \frac{1}{2}$, so verringert sich der Einfluss und die Kurve verläuft näher an den restlichen Kontrollpunkten.

(b) Bei einem größeren Gewicht von $w_3 = 2$ hat der Kontrollpunkt mehr Einfluss und die Kurve verläuft näher an dem Kontrollpunkt.

Diese beiden Fälle sind in Abbildung 2.5 dargestellt, wobei der erste Fall durch die gestrichelte, der zweite durch die gepunktete Kurve verdeutlicht wird. Zusätzlich ist zu sehen, wie das Bild des Knoten $t_5 = \frac{2}{3}$ für ein höheres Gewicht Richtung P_3, für ein geringeres Gewicht entgegengesetzt bewegt wird.

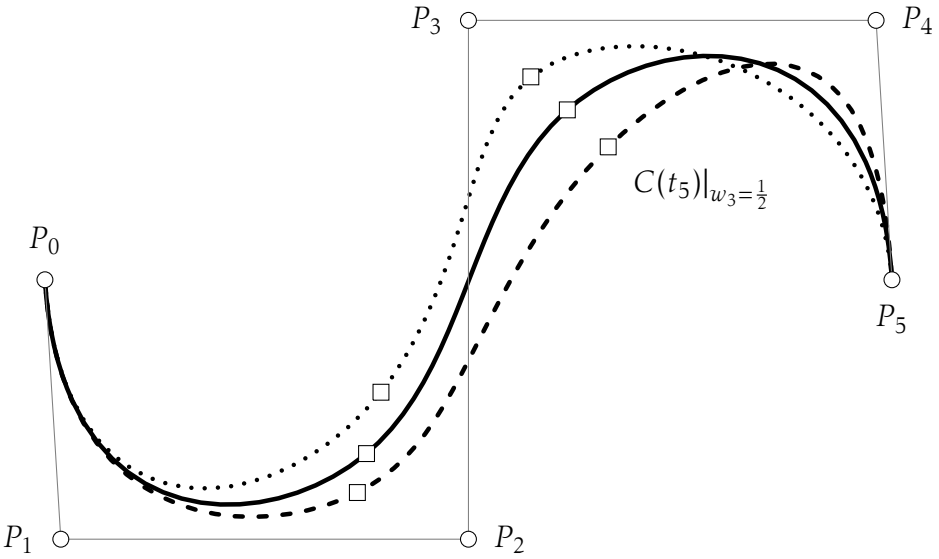


Abbildung 2.5: NURBS-Kurve auf Basis der B-Spline-Kurve aus Beispiel 2.5. Dabei ist das Gewicht des Punktes P_3 einmal erhöht (gepunktete Kurve, $w_3 = 2$) und einmal verringert (gestrichelte Kurve, $w_3 = \frac{1}{2}$). Das Bild $C\left(\frac{2}{3}\right)$ des Knoten t_5 bewegt sich dabei auf P_3 zu bzw. entfernt sich.

Algorithmus zur Berechnung einer NURBS-Kurve

Der Berechnung eines Punktes $C(u), u \in [a,b]$ liegt eine Beobachtung zu Grunde: Stellt man sich die $w_i > 0, i = 0,\ldots,n$ anstatt als Gewichte als weitere Dimension der Kontrollpunkte einer (dann dreidimensionalen) Kurve vor, so lässt sich über den Algorithmus von de Boor (s. Algorithmus 2.1) ein Punkt auf dieser Kurve berechnen. Dabei entspricht die neue Dimension dem Divisors in Gleichung (2.6). Analog lässt sich ein Punkt auf der Kurve mit den gewichteten Kontrollpunkten $w_i P_i$ berechnen, indem die ersten beiden Komponenten dieser dreidimensionalen Kontrollpunkte mit w_i gewichtet werden. Dies entspricht dem Dividend in Gleichung (2.6).

Somit ergibt sich der Punkt auf einer NURBS-Kurve durch Modifikation des für B-Splines bekannten Algorithmus. Der Ablauf ist in Algorithmus 2.2 dargestellt und berechnet zunächst den Punkt auf einer Kurve in *homogenen Koordinaten*. Dazu werden die Kontrollpunkte

$$P_i^w = \begin{pmatrix} P_{i,1} w_i, & P_{i,2} w_i & w_i \end{pmatrix}^T$$

eingeführt und mit ihnen der Punkt auf der Kurve $C^w(u)$ ebenso in homogenen Koordinaten berechnet. C^w hat die Form $C^w(u) = \begin{pmatrix} x & y & w \end{pmatrix}^T$, wobei

der Dividend $\begin{pmatrix} x & y \end{pmatrix}^T \in \mathbb{R}^2$ und Divisor $w \in \mathbb{R}$ aus der Gleichung (2.6) berechnet wurden. Die abschließende Division ist stets möglich, denn nach Eigenschaft B1 sind die Basisfunktionen nicht negativ, $d+1$ Funktionen sogar echt positiv und die Gewichte sind per Definition ebenso echt positiv. Somit wird durch eine Summe geteilt, die nur positive Summanden enthält und echt positiv ist.

Algorithmus 2.2: De Boor-Algorithmus für NURBS-Kurven, nach [23, S. 124f.]

Eingabe: Eine Position $u \in [a, b]$,
der Knotenvektor $t = (t_i)_0^m,\ t_i \in \mathbb{R}$,
die Kontrollpunkte $P_0, \ldots, P_n, \quad n < m-1 \in \mathbb{R}, \quad P_i \in \mathbb{R}^2$
die Gewichte $w_0, \ldots, w_n,\ w_i > 0$ der Kontrollpunkte

Ausgabe: Der Punkt $C(u)$ auf der NURBS–Kurve

Daten: Der Grad d ergibt sich aus m und n mittels $d = m - n - 1$
Der Punkt $C^w(u)$ auf der NURBS–Kurve in homogenen Koordinaten

Algorithmus

Setze $P_i^w = \begin{pmatrix} P_{i,1} w_i, & P_{i,2} w_i & w_i \end{pmatrix}^T, i = 0, \ldots, n$

Berechne $C^w(u) = \begin{pmatrix} x & y & w \end{pmatrix}^T$ nach Algorithmus 2.1 auf Basis der Punkte P_i^w

Rückgabewert: $Q = \begin{pmatrix} \frac{x}{w} & \frac{y}{w} \end{pmatrix}^T$

Die Idee der Rechnung in homogenen Koordinaten ermöglicht eine Anpassung fast aller Algorithmen der B-Spline-Kurven auf NURBS-Kurven. So bringt die Erweiterung der B-Splines auf NURBS-Kurven den Gewinn der Kreisdarstellung. Trotzdem können die meisten Algorithmen weiterhin – leicht modifiziert – angewandt werden und zusätzlich gelten die Eigenschaften der B-Splines ebenso für NURBS.

Ableitungen

Die Ableitung einer B-Spline-Kurve $B(u)$ vom Grad d auf einem Knotenvektor $t = (\underbrace{a \ \dots \ a}_{d+1} \ t_{d+1} \ \dots \ t_{m-d-1} \ \underbrace{b \ \dots \ b}_{d+1})$ und den Kontrollpunkten $P_0,\dots,P_n$ ist nach Piegl u. Tiller [23], S. 93

$$
\begin{aligned}
B'(u) &= \sum_{i=0}^{n-1} N_{i,d-1,t'}(u) Q_i \\
&\text{mit} \\
t' &= (\underbrace{a \ \dots \ a}_{d} \ t_{d+1} \ \dots \ t_{m-d-1} \ \underbrace{b \ \dots \ b}_{d}) \\
Q_i &= d \frac{P_{i+1} - P_i}{t_{i+d+1} - t_{i+1}}, \ i = 0,\dots,n-1.
\end{aligned}
\tag{2.9}
$$

Für eine NURBS-Kurve $C(u)$ ist die Ableitung analog zur Adaption des Algorithmus von de Boor in homogenen Koordinaten berechenbar, jedoch ist zusätzlich aufgrund des Quotienten in Gleichung (2.6) die Leibnizsche Regel zur Bestimmung der Ableitung notwendig.

Sei $C^w(u) = \begin{pmatrix} B(u) \\ w(u) \end{pmatrix}$, wobei $B(u)$ dem Divisor und $w(u)$ dem Dividend aus Gleichung (2.6) entspricht. Damit erhält man die Ableitung für C^w nach Gleichung (2.9) beziehungsweise komponentenweise die k–ten Ableitungen $B^{(k)}(u)$ und $w^{(k)}(u)$ durch mehrfaches Anwenden. Über den Zusammenhang

$$B^{(k)}(u) = (w(u)C(u))^{(k)}$$

ergibt sich mit der Leibnizschen Regel und Umstellen nach $C^{(k)}$ als Ableitung für die NURBS-Kurve

$$C^{(k)}(u) = \frac{B^{(k)}(u) - \sum_{i=1}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u)}{w(u)}.$$

Kapitel 3.

Periodische NURBS-Kurven

Für die Darstellung geschlossener Formen oder Umrisse sind die NURBS-Kurven aus Abschnitt 2.6 ein guter Ausgangspunkt. Setzt man jedoch lediglich $C(a) = P_0 = P_n = C(b)$, erhält man zwar eine geschlossene Form, da jedoch die Ableitungen $C'(a)$ und $C'(b)$ im Allgemeinen nicht identisch sind, ist diese Stelle der geschlossenen Kurve nicht stetig differenzierbar. Dies kann man erreichen, indem man $C'(a) = \alpha(P_1 - P_0) = -\beta(P_n - P_{n-1}) = C'(b)$ setzt. $\alpha, \beta \in \mathbb{R}$ hängen dabei vom Grad der Kurve und dem Knotenvektor ab. Für eine allgemeine Definition der m-maligen stetigen Differenzierbarkeit im Start- und Endpunkt wird dieser Ansatz jedoch sehr kompliziert, da bei Betrachtung der Ableitung zunächst die Leibnizsche Regel angewandt werden muss und diese allgemeine Form dann durch Gleichsetzen von $C^{(k)}(a) = C^{(k)}(b)$ auf ein Gleichungssystem in den Kontrollpunkten, Knoten und Gewichten führt.

Dieses Kapitel behandelt zunächst die Idee zu periodischen NURBS-Kurven, den veränderten Anforderungen an geschlossene NURBS-Kurven und stellt abschließend deren Auswirkungen auf Algorithmen dar.

3.1. Offene und geschlossene NURBS-Kurven

Die im letzten Kapitel eingeführten NURBS-Kurven basieren auf einem Knotenvektor nach Definition 2.4. Eine solche NURBS-Kurve und ihr Knotenvektor heißen auch *offen*, im Englischen hat sich der Begriff *clamped* (dt. eingespannt) etabliert. Im Allgemeinen werden in der Literatur offene Knotenvektoren verwendet, so dass auf diese Unterscheidung nicht immer eingegangen wird. Im Gegensatz dazu steht die folgende Definition, die für periodische NURBS-Kurven eine wichtige Grundlage bildet.

Definition 3.1 (geschlossener Knotenvektor).
Seien $a, b \in \mathbb{R}$, $m, d \in \mathbb{N}$ mit $2d \leq m$.
Eine Folge reeller Zahlen $t_0 \leq t_1 \leq \ldots \leq t_m$ mit

$$t_0 \leq t_1 \leq \ldots \leq t_d = a \leq t_{d+1} \leq \ldots \leq t_{m-d} = b \leq t_{m-d+1} \leq t_m$$

bildet den *geschlossenen Knotenvektor* $t = (t_i)_{i=0}^{m}$.

Die Begriffe sind in der Literatur nicht immer einheitlich verwendet worden. Diese Arbeit hält sich an die in Piegl u. Tiller [23], S. 575 gegebenen Ausführungen und die deutschen Übersetzungen nach Farin [14], S. 106. In der englischen Literatur hat sich hierfür der Begriff *unclamped* (dt. ausgespannt, losgebunden) etabliert. Die englischen Begriffe beziehen sich eher auf die visuelle Darstellung der Basisfunktionen, wohingegen die deutschen Begriffe Bezug auf die Kurve nehmen.

Die *geschlossene NURBS-Kurve* ist jedoch nicht dahingehend geschlossen, dass sie im Sinne einer geschlossenen Formgebung gesehen werden kann oder periodisch ist, sondern eher dadurch, dass Start- und Endpunkt echt innerhalb der konvexen Hülle des Kontrollpolygons liegen.

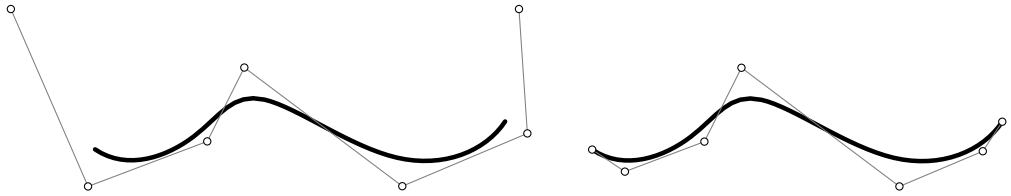
Der geschlossene Knotenvektor besitzt keine Endknoten mehr, sondern besteht nur noch aus inneren Knoten. Die Basisfunktion $N_{0,d}(u)$, die auf dem Intervall $[a, t_{2d}] = [t_d, t_{2d}]$ definiert ist, existiert jedoch weiterhin. Analog existiert ebenso die letzte Basisfunktion $N_{n,d}(u)$ auf $[b, t_m] = [t_{m-d}, t_m]$, an der Menge der Basisfunktionen ändert sich also wenig. Daher bleiben Kurven mit diesem Knotenvektor weiterhin definiert auf dem Intervall $[a, b]$.

Für eine auf einem geschlossenen Knotenvektor definierte B-Spline- oder NURBS-Kurve gelten ebenso die Eigenschaften B1 bis B11 mit Ausnahme der Endpunktinterpolation aus Eigenschaft B10.

Beispiel 3.1 (offene und geschlossene Kurve).
Eine beliebige Kurve lässt sich sowohl mit geschlossenem als auch mit offenem Knotenvektor darstellen. Dies ist in Abbildung 3.1 dargestellt.

Für die offene Kurve in Abbildung 3.1(b), sind aufgrund der Endpunktinterpolation die Kontrollpunkte P_0 und P_n fest gegeben als Endpunkte der Kurve. Die restlichen Kontrollpunkte sind nicht fest gegeben, denn über den Knotenvektor und die Gewichte der Kontrollpunkte lassen sich diese verschieben. Damit die gleiche Kurve entsteht, ist dann ein lineares Gleichungssystem zu lösen. Erhöht man etwa das Gewicht eines Kontrollpunktes, so erhält man die Kurve dadurch, dass man den Knoten weiter von der Kurve entfernt.

Bei der geschlossenen Kurve gilt diese Veränderung ebenso, jedoch auch für die Endknoten, da diese nicht mehr interpoliert werden. Es entstehen also mehr Freiheitsgrade bei der geschlossenen Kurve, man verliert aber auch eine Eigenschaft.



(a) geschlossen (b) offen

Abbildung 3.1: Eine Kurve einmal als (a) geschlossene Kurve, einmal als (b) offene Kurve.

3.2. Mathematische Idee periodischer NURBS-Kurven

Damit ein stetiger Übergang vom Start- zum Endpunkt möglich ist, müssen alle in den Knotenpunkten $t_d = a$ und $t_{m-d} = b$ existierenden Ableitungen identisch sein. Sei also $C(u)$ eine NURBS-Kurve auf $[a, b]$ vom Grad d mit n gewichteten Kontrollpunkten und für $m = n - d - 1$ sei $(t_i)_{i=0}^{m}$ der geschlossene Knotenvektor mit $t_d = a$ und $t_{m-d} = b$. Ist nun neben

$$C(a) \;=\; \sum_{i=0}^{d-1} R_{i,d}(a) P_i \;=\; \sum_{i=0}^{d-1} R_{n-d+i+1,d}(b) P_{n-d+i+1} \;=\; C(b) \tag{3.1}$$

das an diesen Stellen entstandene rationale Polynom identisch, beispielsweise durch Identität der Basispolynome, so ist die maximale Stetigkeit im Start- und Endpunkt erreicht. Diese hängt von den Vielfachheiten der Werte t_d und t_{m-d} im Knotenvektor ab.

Es gibt viele unterschiedliche Möglichkeiten, diese Gleichheit zu erbringen. Dazu gibt es eine kurze Ausführung von Piegl u. Tiller [23], S. 576, die an dieser Stelle ausführlich dargestellt werden soll, um ein besseres Verständnis im Umgang mit periodischen NURBS-Kurven zu vermitteln. Die dort vorgestellte Variante ist, zunächst die gewichteten Kontrollpunkte gleichzusetzen, was bildlich dem Namen der periodischen NURBS-Kurven entspricht, da – etwas informell gesprochen – *gegen Ende der Kurve die gleiche Situation erzeugt wird wie zu Beginn*. Setzt man also

$$P_i = P_{n-d+1+i},\ w_i = w_{n-d+1+i}, \quad i = 0, \ldots, d-1 \tag{3.2}$$

so werden die Kontrollpunkte periodisch fortgesetzt. Wählt man dann die Knoten

$$t_{d-i-1} = t_{d-i} - (t_{n-i+1} - t_{n-i}),\ \text{und}\ t_{n+i+2} = t_{n+i+1} - (t_{d+i+1} - t_{d+i}) \qquad i = 0, \ldots, d-1 \tag{3.3}$$

so werden $t_d = a$ und $t_{m-d} = t_{n+1} = b$ (wegen $n + d + 1 = m$) nicht verändert, es gilt jedoch

- Die Abstände zwischen den Knoten $(t_i)_{i=0}^{d}$ entsprechen denen von $(t_{m-2d+i})_{i=0}^{d} = (t_{n+d+1+i})_{i=0}^{d}$, bzw. in anderen Worten:
  Die Abstände *vor* dem Knoten $t_d = a$ sind identisch mit denen *vor* $t_{m-d} = b$.

- Die Abstände zwischen den Knoten $(t_{d+i})_{i=0}^{d}$ sind identisch mit denen zwischen $(t_{n+i+1})_{i=0}^{d}$ oder in anderen Worten:
  Die Abstände *nach* dem Knoten $t_{m-d} = b$ sind identisch mit denen *nach* $t_d = a$.

Da die B-Spline-Basisfunktionen genau über diese Abstände definiert sind (siehe Gleichung (2.3), S. 10), gilt somit:

$$\forall u \in [a, b] : R_{i,d}(u) = R_{n-d+i+1,d}(u), \quad i = 0, \ldots, d-1 \tag{3.4}$$

da die Knotenfolge $t_0, \ldots, t_{2d}$ der Knotenfolge $t_{m-2d}, \ldots, t_m$ bis auf eine Translation um $t_{m-d} - t_d$ entspricht. Da jede Basisfunktion $R_{i,d}(u)$ bzw. $R_{n-d+1+i,d}(u)$ je auf einer Teilknotenfolge definiert ist, ergibt sich die Gleichheit. Mit den Gleichungen (3.2) und (3.4) folgt die Gleichheit der Ableitung im Start- und Endpunkt.

Sind für eine offene NURBS-Kurve die Ableitungen in den Punkten a und b einer Kurve $C(u)$ komplett identisch, so lässt sich mit dem Algorithmus 12.1 von Piegl u. Tiller [23], S. 577 genau diese Form erzeugen.

Ebenso lassen sich zwei NURBS-Kurven $C_1 : [a, b] \to \mathbb{R}^2$ und $C_2 : [a', b'] \to \mathbb{R}^2$ verbinden, wenn in den Punkten $C_1(a) = C_2(b')$ oder $C_1(b) = C_2(a')$ die Ableitungen identisch sind. Dazu müssen vor Anwendung des Algorithmus 12.1 die beiden Intervalle zu einem verbunden werden. Dies erreicht man, indem man die Intervalle derart verschiebt, dass $b' = a$ bzw. $a' = b$ gilt.

Die Konstruktion ist nicht eindeutig. Wählt man die Basisfunktionen nicht identisch, lassen sich trotzdem periodische Funktionen konstruieren, indem die Kontrollpunkte entsprechend verändert werden. Mit der Identität der Kontrollpunkte ist jedoch die bildliche Vorstellung einer periodischen Kurve (mit der Periode $b - a$) gegeben.

Beispiel 3.2 (periodische NURBS-Kurve).
Sei eine NURBS-Kurve $C(u)$ gegeben mit Grad $d = 3$ und $n = 9$ Kontrollpunkten, so ist $m = n + d + 1 = 13$. Die Kurve ist definiert auf dem Intervall $[a, b] = [t_3, t_{10}]$. Sind die Kontrollpunkte so gewählt, dass $P_0 = P_7$, $P_1 = P_8$, $P_2 = P_9$ und ist ein periodischer Knotenvektor $(t_i)_{i=0}^{m}$ nach Gleichung (3.3)

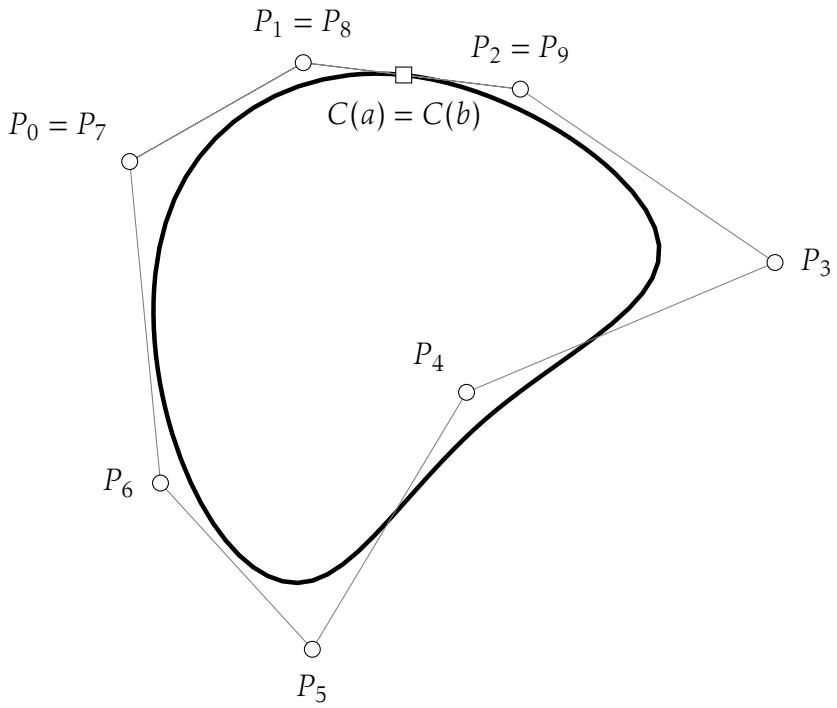


Abbildung 3.2: Eine periodische NURBS-Kurve vom Grad $d = 3$ mit $n = 9$ Kontrollpunkten

gegeben, ergibt sich eine periodische Kurve, wie exemplarisch im Abbildung 3.2 dargestellt.

3.3. Der Kreis als periodische NURBS-Kurve

In der Literatur sind verschiedene Möglichkeiten und Algorithmen benannt, mit der man Kreise erzeugen kann, wobei jedoch immer im Knotenvektor Werte mehrfach vorkommen, siehe Piegl u. Tiller [24]. Basierend darauf beschreiben Bangert u. Prautzsch [2] eine Möglichkeit zur Konstruktion von Kreisen als periodische NURBS-Kurve. Sie zeigen für ihre Konstruktion, dass ein Kreis, der k-mal stetig differenzierbar sein soll, mindestens eine NURBS-Kurve vom Grad $2k + 2$ benötigt.

Beispiel 3.3 (Ein Kreis als periodische NURBS-Kurve).
Da NURBS-Kurven nach B9 invariant gegenüber affin-linearen Transformationen sind, ist eine einfache Konstruktion mit folgendem Ablauf möglich:

1. Erzeugen des Einheitskreises im Ursprung

2. Skalieren mit dem Faktor $r \in \mathbb{R}$, dem gewünschten Radius des Kreises

3. Verschieben des Kreismittelpunktes an einen Punkt $p \in \mathbb{R}^2$

Das Ergebnis ist – mit seinen Kontrollpunkten und -knoten – dargestellt in Abbildung 3.3.

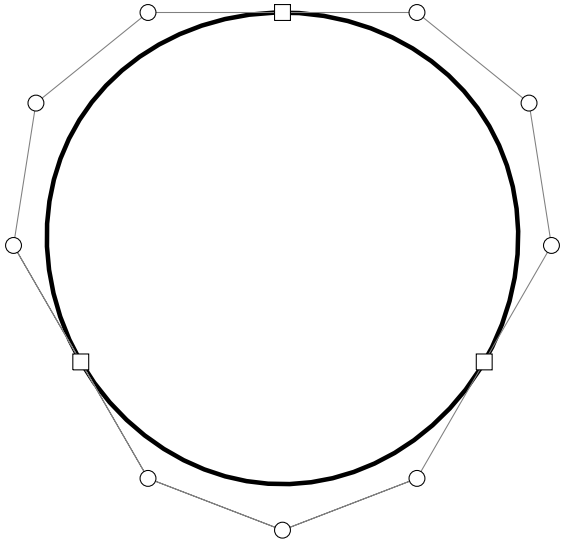


Abbildung 3.3: Ein Kreis in Form einer NURBS-Kurve vom Grad $d = 4$, der einmal stetig differenzierbar ist.

In dieser Arbeit werden Kreise mit NURBS-Kurven vom Grad 4 eingesetzt. Dadurch ist nach Bangert u. Prautzsch [2] die erste Ableitung stetig. Dies genügt allen Algorithmen, da vor Verwendung der zweiten Ableitung die NURBS-Kurve in eine Folge von rationalen Beziér-Kurven zerlegt wird. Jede dieser Kurven ist bei dem Kreis dann dreimal stetig differenzierbar. Dies ist für die verwendeten und präsentierten Algorithmen in Kapitel 4 ausreichend.

Ausgehend von dem Kreis lassen sich durch Streckung in eine Richtung beliebige Ellipsen erzeugen. Da dies ebenso durch eine affin-lineare Transformation realisiert werden kann, sind neben Kreisen auch sehr einfach Ellipsen möglich.

3.4. Anforderungen an Algorithmen

Da bis auf die Endpunktinterpolation die Eigenschaften der B-Splines sich ebenso auf die geschlossenen und somit auch die periodischen NURBS-Kurven übertragen lassen, kann man Algorithmen für B-Splines für die periodischen Kurven verwenden. Dabei muss man jedoch in einigen Fällen Anpassungen vornehmen. Bei den Algorithmen, die lediglich verarbeiten, wie etwa dem Algorithmus von de Boor, sind keine Anpassungen notwendig.

Bei manipulierenden Algorithmen gilt es jedoch zwei Aspekte zu berücksichtigen:

Veränderung der Kontrollpunkte. Für jeden Kontrollpunkt P_i mit $i < d$ muss analog der Kontrollpunkt $P_{n-d+1+i}$ verändert werden, damit diese gleich bleiben. Analog der umgekehrte Fall, wird ein Kontrollpunkt $P_{n-i}, i < d$ manipuliert, muss diese Manipulation auch für den Knoten P_{d-1-i} durchgeführt werden.

Veränderung von Knoten. Wird ein Knoten $t_i, i \leq 2d$ um den Wert $\Delta t \in \mathbb{R}$ verschoben, so dass also $t_i' = t_i + \Delta t$ ist, so muss Δt so gewählt sein, dass weiterhin die Monotonie des Knotenvektors erfüllt ist (vgl. Def. 2.4, S. 10). Ist $i < 2d$ oder $i > m-2d$, so muss zusätzlich der Knoten t_{m-2d+i} bzw. t_{i-m+2d} ebenfalls um Δt verändert werden, um die Kurve periodisch zu erhalten.

Auswirkungen auf Algorithmen

Bei Algorithmen, die nur einzelne Kontrollpunkte oder Knoten verändern, genügt eine einfache Behandlung der eben genannten Fälle. Arbeitet ein Algorithmus auf mehreren Knoten bzw. Kontrollpunkten, so kann eine atomare Betrachtung der genannten Fälle zu nicht erwünschten Nebeneffekten führen, falls gleichzeitig auf beiden identisch gesetzten Kurventeilen Veränderungen vorgenommen werden. Dies tritt bei den Algorithmen im nächsten Kapitel lediglich beim Einfügen von Knoten in Abschnitt 4.1 auf. Eine mögliche Lösung ist, vor Beginn des Algorithmus sicherzustellen, dass entweder nur auf dem vorderen Teil ($P_i, i < d$ sowie $t_j, j \leq 2d$) oder nur auf dem hinteren Teil Manipulationen stattfinden. Im Anschluss an eine solche Manipulation ist dann eine Aktualisierung der jeweils anderen Menge notwendig, damit die Kurve erneut periodisch ist.

Beim Einfügen von Knoten lässt sich dies dahingehend realisieren, dass die Menge der einzufügenden Knoten in zwei Mengen aufgeteilt wird, so dass die erste Menge nur den Anfang, die zweite nur das Ende beeinflusst.

Nach Einfügen der Ersten der beiden Mengen wird die Periodizität wiederhergestellt, bevor mit dem Einfügen der Knoten aus der zweiten Menge begonnen wird.

Bei sondierenden Algorithmen, die also lediglich Eigenschaften prüfen oder auf Basis einer periodischen NURBS-Kurve eine Berechnung vollziehen, die Kurve selbst jedoch unverändert belassen, ist eine Alternative das Erstellen einer offenen identischen Kurve. Dabei wird aus der periodischen NURBS-Kurve $C : [a, b] \to \mathbb{R}^2$ eine offene Kurve $C_o : [a, b] \to \mathbb{R}^2$ erzeugt, welche in allen Punkten $u \in [a, b]$ mit C übereinstimmt ($C(u) = C_o(u)$), jedoch auf einem offenen Knotenvektor definiert ist. Dies ist etwa bei der Projektion in Abschnitt 4.3 der Fall. Die Erzeugung von C_o wird im Beispiel 4.2 erläutert.

Entfernen der Redundanz

Anstelle der dargestellten Aktualisierungen ist eine alternative Realisierung die Definition der periodischen NURBS-Kurven über Restklassen bezüglich der Knoten- und Kontrollpunkt-Indizes, so dass formell keine doppelten Knoten und Kontrollpunkte mehr vorliegen.

Dann müsste jedoch jeder Algorithmus darauf angepasst werden und in der Datenmodellierung sind dann für offene und periodische NURBS-Kurven unterschiedliche Formate notwendig. Dies liegt darin begründet, dass die Knotenanzahl sich um $2d$ reduziert, die Anzahl Kontrollpunkte jedoch nur um d. Das Entfernen von Redundanz würde also zu mehr Implementierungsaufwand und zwei verschiedenen Datenmodellen führen.

Kapitel 4.

Algorithmen auf NURBS-Kurven

Die folgenden Algorithmen bilden eine Grundlage der Erzeugung periodischer NURBS-Kurven und deren anschließender Manipulation. Die meisten Algorithmen werden hier der Einfachheit wegen über B-Splines und Kontrollpunkte $P_i \in \mathbb{R}^2$ erläutert. Analog zum Algorithmus 2.1 zur Auswertung der Kurve lassen sich die Algorithmen auf NURBS-Kurven durch Transformation der Kontrollpunkte und Gewichte in homogene Koordinaten anwenden. Dabei muss lediglich darauf geachtet werden, dass bei den Rechnungen die Gewichte positiv bleiben. Die Darstellung der Algorithmen auf B-Splines ist also lediglich der formellen Einfachheit halber gewählt. Treten dabei spezielle Anforderungen für die Gewichte auf, wird dies dargestellt. Ebenso gilt dies für die Adaption der Algorithmen auf periodische NURBS-Kurven.

4.1. Hinzufügen und Entfernen von Knoten

Einfügen und Entfernen von Knoten ist für die Bearbeitung von Kurven von großer Bedeutung. Da die Veränderung eines Kontrollpunktes P_i einer B-Spline-Kurve $B(u)$ bzw. NURBS-Kurve $C(u)$ vom Grad d Einfluss auf ein Intervall von $d+1$ Knoten hat, kann durch Einfügen von Knoten dieses Intervall (bezüglich der Länge des Kurvensegmentes) verringert werden. Ist eine Kurve zu „umständlich“ geworden, da sie aus zu vielen Kontrollpunkten gebildet wird, bietet eine Reduktion der Knoten ein gutes Werkzeug zur Vereinfachung.

Einfügen von Knoten

Zum Einfügen eines Knotens $\hat{t} \in [a,b]$ in den Knotenvektor $t = (t_i)_{i=0}^{m}$ einer B-Spline-Kurve vom Grad d, welche also die Kontrollpunkte $P_0,\dots,P_n$, $n = m-d-1$ besitzt, ist nach Piegl u. Tiller [23], S. 142 ein lineares Gleichungssystem notwendig. Sei $k \in d,\dots,m-d$ derjenige Index des Knotenvektors,

so dass $\hat{t} \in [t_k, t_{k+1})$ liegt. Da durch Einfügen des Knotens die Kurve unverändert bleiben soll, muss gelten:

$$\forall u \in [u_k, u_{k-1}) \quad \sum_{i=k-p}^{k} N_{i,d,t}(u) = \sum_{i=k-p}^{k+1} N_{i,d,t'}(u) Q_i$$
$$\text{mit } t' = \begin{pmatrix} t_0 & t_1 & \dots & t_k & \hat{t} & t_{k+1} & \dots & t_m \end{pmatrix} \tag{4.1}$$

Es werden also $d+1$ Kontrollpunkte verändert und ein neuer Kontrollpunkt benötigt. Durch einen Koeffizientenvergleich bezüglich der Basisfunktionen ergibt sich

$$Q_i = \alpha_i P_i + (1-\alpha_i) P_{i-1}, i = 0, \dots, n+1$$
$$\text{wobei } \alpha_i = \begin{cases} 1 \text{ falls } i \le k-d \\ \frac{\hat{t}-t_i}{t_{i+d}-t_i} \text{ falls } k-d+1 \le i \le k \\ 0 \text{ sonst} \end{cases} \tag{4.2}$$

Beispiel 4.1 (Einfügen eines Knotens).
Für die Kurve aus Beispiel 2.5 sei das Einfügen des Knotens $\hat{t} = \frac{4}{25}$. Da $\hat{t} \in [0, \frac{1}{3}) = [t_3, t_4)$, ist $k = 3$. Daher werden die Kontrollpunkte Q_1, Q_2 und Q_3 nach dem zweiten Fall der Formel 4.1 berechnet, für alle nachfolgenden Punkte ergibt sich mit $\alpha_i = 0$ lediglich eine Änderung im Index ($Q_i = P_{i-1}$). Die neuen Kontrollpunkte Q_i liegen jeweils auf den Verbindungsgeraden der beiden vorherigen Kontrollpunkte P_i und P_{i-1}.

Bemerkung 4.1.
Führt man das Einfügen an einer Stelle $\hat{t}$ d-mal durch, so entsteht ein Kontrollpunkt, der auf der Kurve liegt. Diese Idee liegt auch dem Algorithmus 2.1 zu Grunde.

Besitzt eine B-Spline-Kurve einen inneren Knoten der Vielfachheit d, lässt sich die Kurve an dieser Stelle aufspalten und in zwei Kurven auftrennen. Dazu wird der innere Knoten der Vielfachheit d Endknoten der beiden entstehenden Kurven. Dies ist im nachfolgenden Beispiel 4.2 dargestellt.

Beispiel 4.2 (Zerlegung in eine Folge von Beziér-Kurven).
Die Zerlegung in Beziér-Kurven ist in Abbildung 4.2 dargestellt, in dem die Kurve aus Beispiel 2.5 durch Einfügen von Knoten verändert wird. Eingefügt wurden die Knoten $\{\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}\}$, so dass die Vielfachheit beider innerer Knoten 3 beträgt.

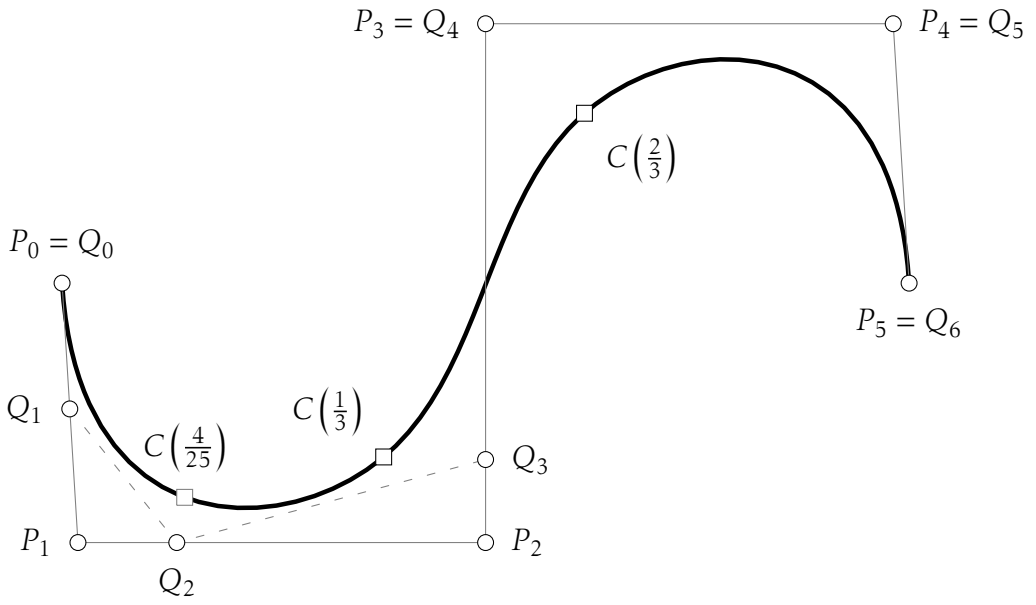


Abbildung 4.1: In die B-Spline-Kurve aus Beispiel 2.5 wird der Knoten $\hat{t} = \frac{4}{25}$ eingefügt. Wegen $d = 3$ und $k = 4$ werden Q_1, Q_2, Q_3 verändert und alle weiteren um einen verschoben.

Da keines der drei Teilstücke innere Knoten besitzt, besteht die Kurve aus einer Folge von Beziér-Kurven. Dabei haben zwei aufeinanderfolgende Beziér-Kurven je genau einen Kontrollpunkt gemeinsam. Auf diese Art lässt sich jede NURBS-Kurve durch Einfügen von Knoten in eine Folge von Beziér-Kurven zerlegen.

Entfernen von Knoten

Das Entfernen von Knoten ist – analog zum Einfügen – das Gleichsetzen zweier Kurven. Im Gegensatz zum Einfügen ist das Entfernen jedoch nicht immer möglich, möchte man die Gleichheit der Kurve beibehalten (siehe Piegl u. Tiller [23], S. 179ff.). Lässt man eine Veränderung der Kurve zu, lassen sich Knoten entfernen, so die anfängliche Kurve nicht schon eine Beziér-Kurve ist. Es können nur innere Knoten entfernt werden. Die Veränderung der Kurve ist hier tolerierbar, da es später in der Implementierung die Möglichkeit gibt, das Entfernen rückgängig zu machen. Bei periodischen Kurven können so lange Knoten entfernt werden, bis lediglich $2d$ Knoten verblieben sind.

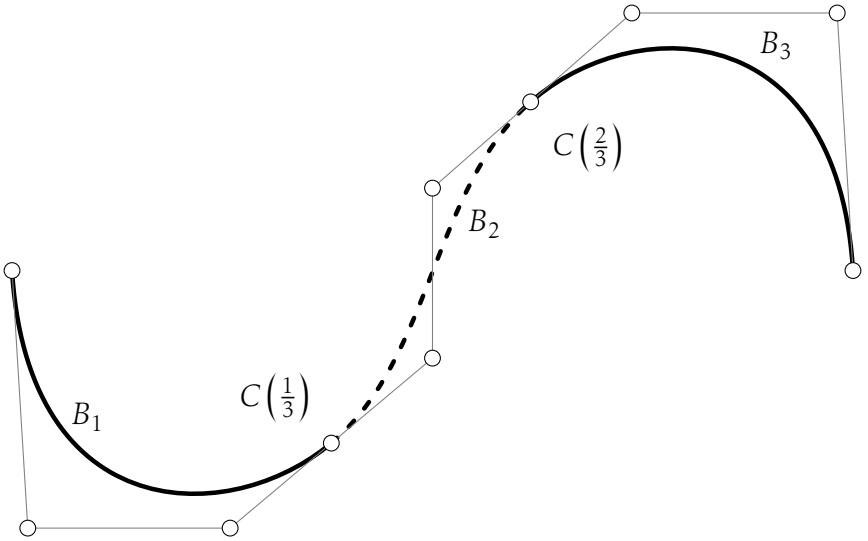


Abbildung 4.2: Die Zerlegung des Beispiels 2.5 in drei Beziér-Kurven B_1, B_2 und B_3 durch doppeltes Einfügen von $\frac{1}{3}$ und $\frac{2}{3}$. Dadurch werden $C\left(\frac{1}{3}\right)$ und $C\left(\frac{2}{3}\right)$ interpoliert.

Bei NURBS-Kurven kann zusätzlich der Fall auftreten, dass in der Berechnung der neuen Kontrollpunkte negative Gewichte auftreten. Dann kann der Knoten selbst dann nicht entfernt werden, wenn eine Veränderung der Kurve toleriert würde.

Erzeugen von Teilkurven

Das Einfügen von Knoten stellt auch die Möglichkeit zur Verfügung, eine beliebige Teilkurve einer NURBS-Kurve zu erhalten. Die ursprüngliche Kurve kann dabei offen, geschlossen oder sogar periodisch sein. Die Teilkurve entsteht durch d–faches Einfügen eines Knotens am Startpunkt $u_1 \in [a, b]$ und analog am Endpunkt $u_1 < u_2 \leq b$ des Teilstücks.

Danach erhält man die Teilkurve als offene NURBS-Kurve, indem man als Knotenvektor denjenigen Teilvektor wählt, der mit u_1 beginnt und mit u_2 endet. Diese werden, im Unterschied zur Kurve nach dem Einfügen, $d+1$-fach im neuen Knotenvektor verwendet und bilden die neuen Endknoten. Sie kommen nach dem Einfügen nur d-mal in der ursprünglichen Kurve vor. Bei den Kontrollpunkten existieren zwei Kontrollpunkte auf der Kurve:

Sei $P_j = C(u_1)$ und $P_k = C(u_2)$. Da $u_1 < u_2$ ist $j < k$. All jene Punkte $P_i, j \leq i \leq k$ bilden die Kontrollpunkte der Teilkurve.

Ist u_1 oder u_2 bereits im Knotenvektor vorhanden, so muss anstelle des d–maligen Einfügens nur so oft eingefügt werden, dass die Vielfachheit im Knotenvektor sich nach dem Einfügen zu d ergibt.

Einzig im Fall der periodischen NURBS-Kurve ist es möglich, dass $u_2 < u_1$ gewählt werden kann. Dies ist zu verstehen als diejenige Teilkurve, die verbleibt, wenn man die Teilkurve von $[u_2, u_1]$ entfernt. Das entspricht der Übernahme genau der anderen Knoten- und Kontrollpunkte im Vergleich zum eben beschriebenen Fall. Der Knotenvektor muss dabei ein wenig modifiziert werden, denn in dem Bereich von u_2 bis u_1 liegt der Start- und Endpunkt. An diesem muss nach dem Endpunkt der Knotenvektor entsprechend der Differenzen vom Anfang periodisch fortgesetzt werden.

4.2. Die Objective Squared Distance Function

Der Abstand eines Punktes p zu einer Kurve $C(u)$ ist vor allem für die Projektion im Abschnitt 4.3 von Bedeutung. Für eine analytische Betrachtung des Abstandes wird dieser wieder als Kurve, genauer als B-Spline bzw. NURBS-Kurve, erzeugt. Zusätzlich wird eine Vereinfachung der vorgestellten Formeln im Falle von (rationalen) Beziér-Kurven präsentiert.

Der Abstand in Abhängigkeit von $u \in [a, b]$ ist nach Chen u. a. [10] wie folgt definiert:

Definition 4.1 (Objective Squared Distance Function (OSD), aus [10]).
Für eine B-Spline-Kurve $C(u)$, mit Knotenvektor – also auch jede (rationale) Beziér- und B-Spline-Kurve – und einen Punkt $p \in \mathbb{R}^2$ heißt die Funktion

$$f(u) = (C(u) - p)(C(u) - p)^T, u \in [a, b]$$

Objective Squared Distance Function (OSD-Funktion). Diese gibt das Quadrat des Abstandes in Abhängigkeit von u an.

Für eine rationale Beziér-Kurve $C(u)$ vom Grad d mit Kontrollpunkten P_i und Gewichten $w_i, w_i > 0, i = 0, \ldots . d$ lässt sich nach [21] die OSD-Funktion wieder als rationale Beziér-Kurve schreiben und hat dann die Form

$$f(u) = (C(u) - p)(C(u) - p)^T = \frac{\sum_{i=0}^{2d} B_{i,2d}(u)\hat{w}_i\hat{P}_i}{\sum_{j=0}^{n} B_{j,2d}(u)w_j}, \quad u \in [a, b]. \tag{4.3}$$

Die entstandenen Kontrollpunkte $\hat{P}_i$ sind reelle Zahlen, da die Funktion das Quadrat des Abstandes von der Kurve $C(u)$ zum Punkt p angibt.

Außerdem ist die Verschiebung um $-p$ der beiden Faktoren eine lineare Transformation und kann somit nach Eigenschaft B9 vor der Berechnung des Produktes durch Verschiebung der Kontrollpunkte durchgeführt werden. Dadurch verbleibt im Folgenden die Betrachtung des Produktes und seiner Berechnung.

Nach Mørken [21] lässt sich das Produkt zweier B-Spline-Kurven über diskrete B-Spline-Koeffizienten berechnen. Im Folgenden seien die Kontrollpunkte $P_i \in \mathbb{R}^2$. Für die Berechnung des Produktes von NURBS-Kurven verwendet man analog zum Algorithmus von de Boor (siehe Algorithmus 2.2.1, S. 17) homogene Koordinaten.

Zunächst benötigt man für die Berechnung des Produktes den neuen Knotenvektor. Dieser entsteht aus dem Zusammenfügen der beiden Knotenvektoren der Faktoren. Im Folgenden sei stets t der Knotenvektor des Produktes und τ derjenige eines der beiden Faktoren.

Satz 4.1 (diskrete B-Spline-Basisfunktionen, nach [21]).
Seien $t = (t_i)_{i=0}^{m_t}$ ein Knotenvektor und $N_{i,d,t}(u)$ die dazugehörigen B-Spline-Basisfunktionen ($i = 0,\dots,n_t$, $n_t = m_t - d - 1$, siehe Gleichung (2.5)). Seien weiter $\tau = (\tau_j)_{j=0}^{m_\tau}$ eine Teilfolge von t. Dann gilt für die B-Spline-Basisfunktionen von τ

$$N_{j,d,\tau}(u) = \sum_{i=0}^{n_t} \alpha_{j,d,t,\tau}(i) N_{i,d,t}(u), \quad j = 0,\dots,n_\tau = m_\tau - d - 1.$$

Die $\alpha_{j,d,t,\tau}(i)$ heißen *diskrete B-Splines vom Grad d auf t mit den Knoten* τ und erfüllen ebenso einen rekursiven Zusammenhang, wie die Basisfunktionen:

$$\alpha_{j,d,t,\tau}(i) = \frac{t_{i+d} - \tau_j}{\tau_{j+d} - \tau_j} \alpha_{j,d-1,t,\tau}(i) + \frac{\tau_{j+d+1} - t_{i+d}}{\tau_{j+d+1} - \tau_{j+1}} \alpha_{j+1,d-1,t,\tau}(i)$$
$$j = 0,\dots,n_\tau, \quad i = 0,\dots,n_t$$

und

$$\alpha_{j,0,t,\tau}(i) = N_{j,0,\tau}(t_i), \quad j = 0,\dots,n_\tau.$$

Damit lässt sich eine B-Spline-Kurve $B_\tau(u)$ auch auf dem Knotenvektor t angeben.

Satz 4.2 (diskrete B-Spline-Kurven, nach [21]).
Seien t und τ wie in Satz 4.1 mit ihren Basisfunktionen gegeben und

$$B_\tau(u) = \sum_{j=0}^{n_\tau} N_{j,d,\tau}(u) P_{j,\tau}$$

eine B-Spline-Kurve mit den Kontrollpunkten $P_{i,\tau}$. Dann kann die Kurve auch über dem Knotenvektor t geschrieben werden:

$$B_\tau(u) = B_t(u) = \sum_{i=0}^{n_t} N_{i,d,t}(u) P_{i,t}$$

wobei sich die Kontrollpunkte wie folgt berechnen lassen:

$$P_{i,t} = \sum_{j=0}^{n_\tau} P_{j,\tau} \alpha_{j,d,t,\tau}(i),\ i = 0,\ldots,n_t.$$

Mit Hilfe dieser beiden Sätze ist das Produkt zweier B-Spline-Kurven berechenbar. Für Beziér-Kurven vereinfacht sich die Formel zusätzlich, vor allem hinsichtlich der Laufzeit. Die folgende Betrachtung ist also speziell auf die Beziér-Kurven bezogen: Sei

$$B_\tau(u) = \sum_{j=0}^{d} B_{j,d}(u) P_j = \sum_{j=0}^{d} N_{j,d,\tau}(u) P_j$$

mit

$$\tau = (\tau_i)_{i=0}^{2d+1} = (\underbrace{a \quad a \quad \ldots \quad a}_{d+1 \text{ Elemente}} \quad \underbrace{b \quad b \quad \ldots \quad b}_{d+1 \text{ Elemente}})$$

eine Beziér-Kurve vom Grad d.

Das B-Spline-Produkt $B_t(u) = B_\tau(u) B_\tau(u)^T$ ist nach Mørken [21] wieder eine Bezièr-Kurve, allerdings vom Grad $2d$ und besitzt somit den Knotenvektor

$$t = (t_i)_{i=0}^{4d+1}, \quad t_i = (\underbrace{a \quad a \quad \ldots \quad a}_{2d+1 \text{ Elemente}} \quad \underbrace{b \quad b \quad \ldots \quad b}_{2d+1 \text{ Elemente}}). \tag{4.4}$$

Dieser besitzt $m = 4d + 2$ Elemente. Die Anzahl Kontrollpunkte ergibt sich also zu $n = 4d + 2 - 2d - 1 = 2d + 1$. Die Kurve des Produktes berechnet

sich wie im nachfolgenden Satz genannt durch Summation aller möglichen Diskretisierungen nach Satz 4.2. Dazu sei $i \in \{0,\dots,2d\}$ fest gewählt. Dies ist die Position im Knotenvektor t, hinter der eine Diskretisierung stattfindet. Dabei bleiben der erste und letzte Wert t_0 und t_{4d+1} stets erhalten. Sie gehören also beide stets sowohl zu t^P als auch zu t^Q.

Sei $P = \{p_1,\dots,p_d\}$ eine Menge von d aufsteigend sortierten natürlichen Zahlen aus der Menge $I_{2d} = \{1,\dots,2d\}$. Seien die restlichen d Werte in der Menge $Q = \{q_1,\dots,q_d\} = I_{2d}\backslash P$ ebenfalls aufsteigend sortiert. Damit definieren sich die Knotenvektoren

$$\begin{aligned} t^P &= (\dots, t_{i-1}, t_i, t_{i+p_1}, t_{i+p_2}, \dots, t_{i+p_d}, t_{i+2d+1}, \dots), \\ t^Q &= (\dots, t_{i-1}, t_i, t_{i+q_1}, t_{i+q_2}, \dots, t_{i+q_d}, t_{i+2d+1}, \dots). \end{aligned} \tag{4.5}$$

Sei Π die Menge aller möglichen P, also aller d-elementigen Teilmengen von I_{2d}. Dann gilt folgender Satz:

Satz 4.3 (Das Produkt zweier B-Splines, nach [21]).
Die Kontrollpunkte $\hat{P}_i$ der Beziér-Kurve

$$B_t(u) = B_\tau(u) B_\tau(u)^T = \sum_{i=0}^{2d} N_{i,2d,t}(u) \hat{P}_i$$

sind gegeben durch die Formel

$$\hat{P}_i = \frac{1}{\binom{2d}{d}} \sum_{P\in\Pi} \sum_{k=0}^{d} \sum_{l=0}^{d} \alpha_{k,d,t,t^P}(i) \alpha_{l,d,t,t^Q}(i) P_k P_l^T, \quad i = 0,\dots,2d. \tag{4.6}$$

Die in t^P und t^Q vorkommenden Werte sind für die Beziér-Kurven stets nur a, b, also die Intervallgrenzen der Kurve. Da t^P und t^Q außerdem sortiert sind, ergibt sich aus der Anzahl gewählter Werte für a eindeutig t^P. Analog ergibt sich t^P ebenso über die Anzahl gewählter Werte b. Ist t^P definiert, ergibt sich über die restlichen Werte t^Q.

Daher lässt sich die erste Summe in der allgemeinen Formel (4.6) vereinfachen:

Satz 4.4 (Das Produkt zweier Beziér-Kurven).
Sei t der Knotenvektor des Produktes zweier Beziér-Kurven (s. Gleichung 4.4). Für ein festes $i \in [0,\dots,2d]$ seien $r_i, b_{i,j} \in \mathbb{N}$ definiert als

$$r_i = \min\{2d - i, i\} = d - |i - d|$$

$$b_{i,j} = \begin{cases} \binom{2d-i}{d-j}\binom{i}{j} & \text{falls} \quad 0 \le i \le d \\ \binom{2d-i}{j}\binom{i}{d-j} & \text{falls} \quad d < i \le 2d \end{cases} \qquad \text{wobei} \;\; 0 \le j \le r_i$$

und weiter für festes j mit $0 \leq j \leq r_i$ seien t_j^P, t_j^Q diejenigen Vektoren t^P, t^Q, bei denen

(i) für $i \leq d$ in $t_{p_1}, \ldots, t_{p_d}$ exakt j-mal der Wert b vorkommt, und

(ii) für $d < i \leq 2d$ in $t_{p_1}, \ldots, t_{p_d}$ exakt j-mal der Wert a vorkommt und

(iii) t_j^Q der nach Konstruktion aus Gleichung (4.5) von t_j^P analoge Vektor ist.

Mit diesen Definitionen vereinfacht sich die Formel für die Kontrollpunkte zu

$$\hat{P}_i = \frac{1}{\binom{2d}{d}} \sum_{j=0}^{r_i} b_{i,j} \sum_{k=0}^{d} \sum_{l=0}^{d} \alpha_{k,d,t,t_j^P}(i) \alpha_{l,d,t,t_j^Q}(i) P_k P_l^T, \quad i = 0, \ldots, 2d. \tag{4.7}$$

Beweis.

Sei $i \in \{0, \ldots, 2d\}$ beliebig aber fest gewählt. Dann ist $2d - i$ die Anzahl des Vorkommens von a und i die Anzahl des Vorkommens von b in dem Ausschnitt der Länge $2d$, der auf den Knoten t_i folgt. Aus diesem Ausschnitt sind nun die Knoten t_{i+p_k} zu wählen. Anders formuliert werden diejenigen Indizes gewählt, durch welche die Menge P und damit der Vektor t^P erzeugt werden.

Da $a < b$ ist, ergibt sich über eine der beiden genannten Anzahlen eindeutig der Vektor t^P (und damit analog der Vektor t^Q). In der Wahl der Werte lässt sich das Minimum $r_i = \min\{2d - i, i\}$ stets komplett aus den t_{i+p_k} entfernen, denn es müssen d Indizes für P gewählt werden. Es stehen dafür $2d$ Indizes zur Verfügung und $r_i \leq d$. Somit lassen sich für dieses i alle Knotenvektoren t^P bestimmen, indem man ein j fest aber beliebig, $0 \leq j \leq r_i$ wie folgt wählt:

Sei im Folgenden zunächst a dasjenige Element mit dem geringeren Vorkommen im Ausschnitt $t_{i+1}, \ldots, t_{i+2d}$ des Vektors t. Dann sind $t_{i+1} = \ldots = t_{i+r_i} = a$ und alle nachfolgenden identisch zu b. Für jedes $j \in \{0, \ldots, r_i\}$ ergibt sich dann ein Vektor t_j^P definiert nach dem Fall (ii) aus der Behauptung.

Um in der Summation der Diskretisierungen in Gleichung (4.6) alle Wahlmöglichkeiten zu erhalten, bleibt nun noch zu betrachten, wie häufig ein einzelner Vektor t_j^P mit unterschiedlichen Indizes p_k gewählt werden kann. Dies entspricht allen $P \in \Pi$, die dieses t_j^P erzeugen. Dabei gilt:

- Es sollen j Knoten mit dem Wert a gewählt werden und in dem Ausschnitt kommt a exakt r_i mal vor.

- Da a dasjenige Element geringeren Vorkommens ist, gilt: $r_i = 2d - i$.
- b kommt in dem Ausschnitt i-mal vor und von diesen müssen, um die d Elemente für t_j^P zu erhalten, noch $d - j$ weitere ausgewählt werden.

Damit ergibt sich für die kombinatorische Häufigkeit, mit der t_j^P gewählt wird der Wert $\binom{2d-i}{j}\binom{i}{d-j}$. Dies entspricht dem zweiten Fall des $b_{i,j}$ in der Aussage. Außerdem ist $i > 2d - i$ und somit $i > d$. Weiter gilt per definitionem $i \leq 2d$.

Analog ergibt sich der erste Fall von $b_{i,j}$, wenn b dasjenige Element ist, das in dem Ausschnitt $t_{i+1}, \ldots, t_{i+2d}$ seltener vorkommt.

Damit ergibt sich für Beziér-Kurven die Gleichung (4.7) als Spezialfall von Gleichung (4.6) und der Satz ist bewiesen.

□

4.3. Projektion auf eine NURBS-Kurve

Die Projektion liefert für einen Punkt $p \in \mathbb{R}^2$ und eine NURBS-Kurve $C(u)$ denjenigen Parameter $u^\star \in [a, b]$, so dass gilt

$$\forall u \in [a, b]: \quad \|C(u^\star) - p\|_2 \leq \|C(u) - p\|_2$$

Man erhält also den minimalen Abstand des Punktes zur Kurve, den Punkt $C(u^\star)$ auf der Kurve und dessen Urbild $u^\star \in n[a, b]$. Die *Projektion eines Punktes p auf die Kurve C* sei im folgenden definiert durch die Funktionen:

$$\text{proj}_C : \mathbb{R}^2 \to C([a, b]), \quad \text{proj}_C(p) = C(u^\star)$$

Zur interaktiven Bearbeitung von NURBS-Kurven ist ein schneller Algorithmus für die Projektion aus zwei Gründen sehr wichtig:

Auf Basis des Abstandes lassen sich weitere Algorithmen implementieren, zum Beispiel die Bestimmung des minimalen Abstandes einer Menge von Punkten zur Kurve. Außerdem lässt sich über die Projektion eine Approximation der Umkehrfunktion $C^{-1}(p) : \mathbb{R}^2 \to [a, b]$ angeben, die eine Toleranz zulässt. Ist $C(u)$ nicht injektiv, so existiert eine solche Umkehrfunktion nicht. Dann liefert die Projektion für einen Punkt auf der Kurve eines seiner Urbilder aus dem Intervall $[a, b]$, auch dies wieder mit der Möglichkeit einer Toleranz. Dadurch kann in der Interaktion beispielsweise ein Mausklick auf eine Kurve erkannt werden, je nach Detailstufe der Ansicht mit entsprechender Toleranz.

Das wesentliche Problem bei der Projektion eines Punktes p auf die NURBS-Kurve C ist, dass diese nicht explizit vorliegt, sondern als implizites, durch Knoten und Kontrollpunkte definiertes stückweises Polynom.

Idee

Die Projektion nach Chen u. a. [10] basiert auf der Idee des Clipping aus der Computergrafik, bei dem nicht benötigte Teile der Kurve verworfen werden. Dazu wird ein Kreis mit dem Punkt p als Mittelpunkt und einem möglichst kleinen Radius $\sqrt{\alpha} \in \mathbb{R}$ definiert. Alle Teile der NURBS-Kurve, die außerhalb dieses Kreises liegen, werden verworfen. Dieser Kreis heißt deswegen auch *Eliminationskreis*. Anschließend werden bei verbleibenden Kurvenstücken, von denen kein eindeutiges Minimum bestimmbar ist, zwei Hälften in die Betrachtung einbezogen. Ist auf einem der verbleibenden Segmente das Minimum eindeutig, so bildet dies ein lokales Minimum. Im Laufe des Algorithmus wird der Radius des Kreises um p ständig verkleinert, so dass möglichst viele Teile der Kurve ausgeschlossen werden können. Abschließend wird aus den lokalen Minima das globale bestimmt.

Da eine B-Spline bzw. NURBS-Kurve durch Einfügen von Knoten in mehrere (rationale) Beziér-Kurven unterteilt werden kann, genügt eine Betrachtung des Clipping auf einer Menge von (rationalen) Beziér-Kurven.

Ausschlusskriterien

Satz 4.5 (Ausschluss eines Kurventeils, nach [10]).
Für $\alpha > 0 \in \mathbb{R}$, einen Punkt $p \in \mathbb{R}^2$ und eine Beziér-Kurve $B(u), u \in [a,b]$ vom Grad d ist die OSD-Funktion $f(u)$ aus Abschnitt 4.4 ebenso eine Kurve in Beziérform, siehe Definition 4.1, Gleichung (4.3). Sie besitzt die Kontrollpunkte $\hat{P}_i,\ \ i = 0,\ldots,2d$. Falls gilt:

$$\forall i \in \{0,\ldots,2d\} : \hat{P}_i \geq \alpha$$

so ist $\forall u \in [a,b] : f(u) \geq \alpha$ und somit die Kurve $B(u)$ außerhalb des Kreises um p mit Radius $\sqrt{\alpha}$.

Satz 4.6 (Eindeutigkeit des Minimums in einem Kurventeil, nach [10]).
Sei $f(u)$ wie im vorherigen Satz gegeben. Existiert nun ein $k \in \{1,\ldots,2d-1\}$, für das gilt

$$\forall i < k :\ \hat{P}_i > \hat{P}_{i+1} \text{ sowie}$$
$$\forall i \geq k :\ \hat{P}_i < \hat{P}_{i+1}$$

so besitzt $f(u)$ ein eindeutiges Minimum.

Ist dieser Satz erfüllt, so konvergiert eine Newton-Iteration begonnen auf diesem Kurventeil gegen denjenigen Punkt mit minimalem Abstand.

In seltenen Fällen, wenn $f(a) = \alpha$ oder $f(b) = \alpha$ das einzige Minimum im Satz 4.5 darstellt, kann der Satz 4.6 keine klare Aussage treffen, da das eindeutige Minimum im Satz 4.6 auf dem Rand des Intervalls liegt und dort keine Konvergenz gesichert ist. Da jedoch $B(a)$ bzw. $B(b)$ interpoliert werden (Bézier-Kurven sind offene Kurven) ist das entsprechende Minimum bereits ohne Newton-Iteration ein Kandidat.

Der Algorithmus

In einem Intervall mit eindeutigem Minimum konvergiert die Newton-Iteration gegen das Minimum. Als Funktion verwenden Piegl u. Tiller [23], S. 230 $g(u) = C'(u)(C(u) - p)^T$, wobei in einem Schritt der nächste Wert u_{i+1} dann berechnet wird durch

$$u_{i+1} = u_i - \frac{g(u_i)}{g'(u_i)} = \frac{C'(u_i)(C(u) - p)}{C''(u_i)(C(u_i) - p) + |C'(u_i)|^2)}.$$

Die Bestimmung des Startwertes u_0 beruht auf den bisherigen Sätzen dieses Abschnitts. Abbruchkriterien der Newton-Iteration sind eine geringe Veränderung im Parameter oder eine geringe Bewegung auf der Kurve. Formell

$$|u_{i+1} - u_i| < \epsilon_1 \quad \|C(u_{i+1}) - C(u_i)\|_2 < \epsilon_2, \quad \epsilon_1, \epsilon_2 \in \mathbb{R}$$

Dabei kann es mehrere Kurvenstücke geben, auf denen das Minimum bestimmt wird, falls verschiedene Kurventeile durch den Eliminationskreis verlaufen. Dann wird aus diesen lokalen Minima das globale Minimum bestimmt.

Eine Darstellung im Pseudoquelltext ist in Algorithmus 4.1 zu finden.

Beispiel 4.3 (Projektion eines Punktes).
Ausgehend von der Kurve aus Beispiel 4.2 und deren Zerlegung in die Beziér-Kurven B_1, B_2 und B_3 wird im folgenden die Projektion des Punktes $p = \begin{pmatrix} 120 \\ 100 \end{pmatrix}$ auf die Kurve dargestellt. Die dabei betrachteten Eliminationskreise mit p als Mittelpunkt und das Ergebnis der Projektion sind in Abbildung 4.3 dargestellt.

Zunächst wird $\alpha = \min\{\|C(0) - p\|_2^2, \|C(1) - p\|_2^2\} = 4500$ gesetzt, denn $\|C(1) - p\|_2 = \sqrt{4500} \approx 67,08$. Dies entspricht dem ersten Eliminationskreis K_1.

Dann werden nacheinander die Beziér-Kurven betrachtet und dabei gilt:

Algorithmus 4.1: Projektion auf eine NURBS-Kurve, nach [10]

Eingabe: Eine NURBS–Kurve $C(u), u \in [a,b]$ vom Grad d mit ihren Knoten, Kontrollpunkten und Gewichten,
Ein Punkt $p \in \mathbb{R}^2$

Ausgabe: Die Projektion $u^\star \in [a,b]$ des Punktes p auf die Kurve

Daten: Eine Menge $\mathcal{M}$ von Beziér–Kurven

Algorithmus
Zerlege $C(u)$ in Beziér–Kurven (s. Bem. 4.1)
Füge diese Kurventeile in $\mathcal{M}$ ein.
Initialisiere $\alpha = \min\{\|C(a)-p\|_2^2, \|C(b)-p\|_2^2\}$ und $u^\star = a$
Solange $\mathcal{M} \neq \emptyset$
Nehme ein Kurventeil $B(u), u \in [a',b']$ aus $\mathcal{M}$ heraus.
Berechne die OSD–Funktion $f(u)$ für p und $B(u)$
Setze $\alpha = \min\{\alpha, \|B(a')-p\|_2^2, \|B(b')-P\|_2^2\}$
Falls $B(u)$ außerhalb des Eliminationskreises liegt (Satz 4.5 erfüllt)
Falls $B(a') = \alpha$ oder $B(b') = \alpha$ das eindeutige Minimum ist
Setze $u^\star = a'$ bzw. $u^\star = b'$ und $\alpha = \min\{\alpha, \|C(u^\star)-p\|_2^2\}$
sonst
Verwerfe $B(u)$
sonst
Falls Das Minimum auf $B(u)$ eindeutig ist (Satz 4.6 erfüllt)
Setze u' = Ergebnis Newton–Iteration mit Startwert $u_0 \in [a',b']$
Falls $\|C(u')-p\|_2 \leq \|C(u^\star)-p\|_2$
Setze $u^\star = u'$
Setze $\alpha = \min\{\alpha, \|C(u^\star)-p\|_2^2\}$
sonst
Füge $\frac{b'-a'}{2}$ d-mal als neuen Knoten in B ein.
Teile B in die Beziér-Kurven $B_1(u), u \in [a', \frac{b'-a'}{2}]$ u. $B_2(u), u \in [\frac{b'-a'}{2}, b']$ auf.
Füge die Beziér–Kurven $B_1(u)$ und $B_2(u)$ zu $\mathcal{M}$ hinzu.
Ende Falls
Ende Falls
Ende Solange
Rückgabewert: $u^\star$

- Für B_1 ist wegen $b = \frac{1}{3}$ der Wert $\|B_1\left(\frac{1}{3}\right) - p\| \approx \sqrt{1548} \approx 39,35$ kleiner als der bisherige Radius und somit wird $\alpha \approx 1548$ gesetzt. Der daraus resultierende Kreis um p mit dem Radius $\sqrt{\alpha}$ ist als K_2 bezeichnet.

$\Rightarrow$ Die OSD-Funktion von B_1 und p erfüllt den Satz 4.5 bezüglich des aktuellen α, B_1 liegt also vollständig außerhalb von K_2 und wird verworfen.

- Für B_2 ist Satz 4.5 nicht erfüllt, jedoch ist nach Satz 4.6 das Minimum eindeutig und eine Newton-Iteration auf dem Intervall $[\frac{1}{3}, \frac{2}{3}]$ konvergiert.

$\Rightarrow$ Das Minimum auf dem Kurvenstück B_2 lautet $p_q = \begin{pmatrix} 93,11 \\ 86,36 \end{pmatrix}$ mit minimalem Abstand zu p. Dieser Punkt p_q hat als Parameter $u \approx 0,4213$ und einem Abstand $\|p - p_C\|_2 \approx 30,14$ zum Punkt p. Damit kann $\alpha = \|p - p_C\|_2^2$ gesetzt werden, was zu einem kleineren Eliminationskreis K_3 führt.

- B_3 wird nach Satz 4.5 ausgeschlossen, da keine Überschneidung mit K_3 vorliegt.

Insgesamt ergibt sich also als Projektion $p_C \approx \begin{pmatrix} 93,11 \\ 86,36 \end{pmatrix} \approx C(0,4213)$.

Fazit

Die Laufzeit der Berechnung einer OSD-Funktion nach der Formel 4.7 ist $O(d^4)$. Das führt bei Kurven mit niedrigem Grad (beispielsweise 3 oder 4) zu Laufzeiten, die im Rahmen eines Mausklicks durchführbar sind. Bei höheren Graden nimmt die Laufzeit jedoch sehr schnell zu.

Da der Algorithmus auf den gleichen Prinzipien basiert wie der Algorithmus von de Boor (siehe S. 17), ist er numerisch stabil und liefert somit zuverlässige Ergebnisse. Er ist sowohl für NURBS als auch periodische NURBS anwendbar, da die Zerlegung in rationale Beziér-Kurven immer möglich ist. Im periodischen Fall werden lediglich die Knoten aus dem Intervall $[a, b] = [t_d, t_{m-d}]$ bei der Zerlegung verwendet.

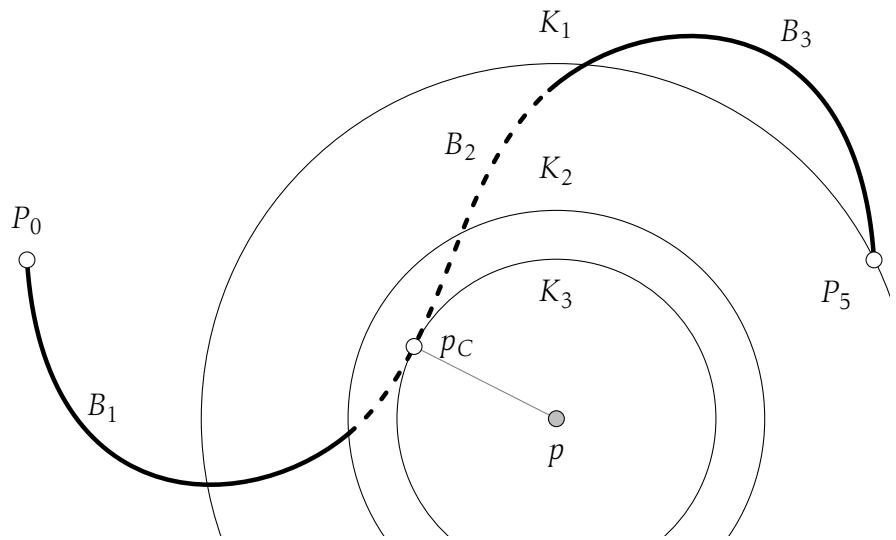


Abbildung 4.3: Projektion des Punktes p auf die Kurve C, die in die Beziér-Kurven B_1, B_2, B_3 zerlegt wurde. Die Kreise K_1 (Initialisierung) und K_2 (nach Betrachten von B_1) schließen alles außerhalb der Kreisscheibe von der weiteren Betrachtung aus. Durch Newton-Iteration auf B_2 (hervorgehoben) erhält man die Projektion p_C.

4.4. Offene und periodische Interpolation

Interpolation offener NURBS-Kurven

Die Interpolation ist ein zentraler Algorithmus zur Erzeugung von NURBS-Kurven. Dabei ist eine Folge von Punkten $Q_0, \ldots, Q_n$ gegeben mit $Q_i \in \mathbb{R}^2$, durch welche eine B-Spline- bzw. NURBS-Kurve $C(u)$ verlaufen soll. Für diese Kurve wird zusätzlich der Grad d gegeben. Dann lässt sich nach Piegl u. Tiller [23], S. 364ff. für einen noch zu bestimmenden Knotenvektor $(t_i)_{i=0}^m$, $m = n + d + 1$ folgendes lineares Gleichungssystem aufstellen

$$Q_k = C(\hat{t}_k) = \sum_{i=0}^{n} N_{i,d}(\hat{t}_k) P_i, \quad k = 0, \ldots, n. \tag{4.8}$$

Damit ist für $\hat{t}_k \in [t_d, t_{m-d}]$, $k = 1, \ldots, n$ die Interpolation gegeben.

Diese Stützstellen werden nach der präsentierten zentripetalen Methode gewählt, um einerseits den Abständen zwischen den Interpolationspunkten Q_k, andererseits auch engen Kurven gerecht zu werden. Seien

$$\hat{t}_0 = 0, \quad \hat{t}_n = 1$$

und

$$d = \sum_{k=1}^{n} \sqrt{\|Q_k - Q_{k-1}\|},$$

dann ergeben sich die mittleren Stützstellen zu

$$\hat{t}_k = \hat{t}_{k-1} + \frac{\sqrt{\|Q_k - Q_{k-1}\|}}{d}, \quad k = 1, \ldots, n-1. \tag{4.9}$$

Daraus lässt sich der Knotenvektor bestimmen, der ebenso die Abstände der Interpolationspunkte einbezieht, damit eine gleichmäßige Geschwindigkeit der Kurve auf dem Intervall $[t_d, t_{m-d}]$ vorherrscht. Dies ist deswegen sinnvoll, da dann relative Angaben bezüglich des Urbildintervalles auch näherungsweise in einer relativen Angabe bezüglich der Gesamtstrecke der Kurve resultieren.

Daher empfehlen Piegl u. Tiller [23] folgende Berechnung des Knotenvektors durch Bildung des Mittelwertes

$$\begin{aligned} t_0 = \ldots = t_d &= 0 \\ t_{m-d} = \ldots = t_m &= 1 \\ t_{j+d} = \frac{1}{d} \sum_{i=j}^{j+d+1} \hat{t}_i, \quad j &= 1, \ldots, n-d \end{aligned} \tag{4.10}$$

Damit bleiben in der Gleichung (4.8) lediglich die Kontrollpunkte $P_i, i = 0, \ldots, n$ zu bestimmen. Es entsteht ein lineares Gleichungssystem mit folgenden Eigenschaften:

- Aufgrund der Lokalität der Basisfunktionen (Eigenschaft B2) sind in jeder Zeile lediglich $d + 1$ Werte von Null verschieden.
- Diese Werte sind jeweils um die Hauptdiagonale gruppiert, es entsteht eine Bandmatrix.
- Durch die Positivität (B1) sind alle Einträge der Matrix positiv.
- Für NURBS-Kurven lässt sich die Gleichung (4.8) ebenso anwenden, indem man die Gewichte $w_i = 1, \ i = 0, \ldots, n$ setzt.

Eine positive Bandmatrix lässt sich mit dem Gaußsches Eliminationsverfahren ohne Pivotisierung lösen, beispielsweise durch eine vereinfachte LR-Zerlegung.

Beispiel 4.4 (Interpolation).
Für die Punkte

$$Q_0 = \begin{pmatrix}125\\63\end{pmatrix},\ Q_1 = \begin{pmatrix}100\\88\end{pmatrix},\ Q_2 = \begin{pmatrix}63\\88\end{pmatrix},\ Q_3 = \begin{pmatrix}25\\100\end{pmatrix},$$
$$Q_4 = \begin{pmatrix}25\\63\end{pmatrix},\ Q_5 = \begin{pmatrix}75\\25\end{pmatrix},\ Q_6 = \begin{pmatrix}113\\50\end{pmatrix} \tag{4.11}$$

ist $n = 6$. Setzt man weiter $d = 3$, so lassen sich $\hat{t}_i,\ i = 0,\dots,6$ und der Knotenvektor $t = (t_i)_{i=0}^{10}$ bestimmen.

Daraus ergibt ist ein Gleichungssystem mit 14 Gleichungen und ebenso vielen Unbekannten, je Interpolationspunkt und Koordinate eine. Löst man dieses, ergibt sich die offene Kurve aus Abbildung 4.4(a).

Für eine B-Spline-Kurve vom Grad d sind mindestens $d + 1$ Interpolationspunkte notwendig. Dann entsteht exakt eine Beziér-Kurve.

Interpolation periodischer NURBS-Kurven

Die im letzten Abschnitt vorgestellte Berechnung ist für periodische NURBS-Kurven ebenso anwendbar, jedoch gibt es einige zusätzliche Anforderungen an die zu berechnenden Knoten und Kontrollpunkte. Die wesentliche Änderung im Resultat ist, dass zwischen Q_n und Q_0 ein zusätzlicher Spline benötigt wird und insgesamt alle Ableitungen im Start- und Endpunkt Q_0 der Kurve identisch sein müssen.

Setzt man die Interpolationspunkte periodisch fort, so ergibt sich eine Kurve, die bis auf die ersten und letzten d Punkte periodisch ist. Das ergibt sich aus dem in Abschnitt 3.2 vorgestelltem Ansatz der periodischen NURBS und daraus, dass die ersten bzw. letzten d Kontrollpunkte Einfluss auf die Endpunktinterpolation offener NURBS-Kurven haben.

Damit also für den ersten Interpolationspunkt Q_0 ein identischer zweiter Punkt Q_{n+1} vorliegt, in dem später die Kurve in allen Ableitungen identisch ist, benötigt man folgende Interpolationspunkte

$$\begin{aligned}\hat{Q}_i &= Q_{n-d+i}, \quad i = 0,\dots,d\\ \hat{Q}_{d+1+i} &= Q_i, \quad i = 0,\dots,n\\ \hat{Q}_{n+d+2+i} &= Q_i, \quad i = 0,\dots,d\end{aligned}$$

Damit ist $Q_0 = \hat{Q}_{d+1} = \hat{Q}_{n+d+2}$. Die Kontrollpunkte wiederholen sich also mit der Periode $n + 1$. Für die Interpolation ergibt sich dann eine neue Anzahl $n' = n + 2d + 3$ anstelle der vorherigen $n + 1$ Interpolationspunkte, da vor und hinter den bisherigen jeweils $d + 1$ Punkte hinzugefügt wurden.

Mit diesen neuen Interpolationspunkten $\hat{Q}_i,\ i = 0,\dots n+2d+2$ gilt dann folgender Satz.

Satz 4.7 (Ableitungen einer periodisch interpolierten Kurve).
Für eine interpolierte NURBS-Kurve basierend auf den Interpolationspunkten $\hat{Q}$ nach (4.8) mit den Stützstellen (4.9) und Knoten (4.10) gilt:

$$C^{(k)}(\hat{t}_d) = C^{(k)}(\hat{t}_{n+d+1}), \quad k = 0,\dots,d-1$$

Beweis.

Aus der periodischen Fortsetzung der Interpolationspunkte $\hat{Q}_i$ ergibt sich der Zusammenhang

$$\begin{aligned} \hat{Q}_i &= \hat{Q}_{n+1+i}, \\ \Rightarrow C(\hat{t}_i) &= C(\hat{t}_{n+1+i}), \quad i = 0,\dots,2d \end{aligned}$$

Für $i = d$ ergibt sich damit der Fall $k = 0$, denn $\hat{Q}_{d+1} = Q_0 = \hat{Q}_{n+d+2}$ (★).
Weiter gilt für die Differenzen der Stützpunkte

$$\hat{t}_k - \hat{t}_{k-1} = \frac{\sqrt{\|\hat{Q}_k - \hat{Q}_{k-1}\|}}{d} = \frac{\sqrt{\|\hat{Q}_{n+1+k} - \hat{Q}_{n+k}\|}}{d} = t_{n+k+1} - t_{n+k} \qquad k = 1,\dots,2d$$

Setzt man dies in die Gleichung (4.10) ein, so bleiben auch dort die Differenzen der ersten d Kontrollpunkte mit den Differenzen der letzten d identisch. Diese Verringerung auf d statt $2d$ ergibt sich aus dem Mittelwert.

$$t_{i+d} - t_{i+d-1} = t_{n+d+1+i} - t_{n+d+i}, \quad i = 1,\dots,d \tag{4.12}$$

Der Summand d im Index ergibt sich dabei aus der Verschiebung des Start- und Endpunktes $Q_0 = \hat{Q}_d$. Aus der Gleichheit (4.12) ergibt sich für die B-Spline bzw. NURBS-Basisfunktionen für den Fall (★) die gleichen Werte, womit sich ebenso die Kontrollpunkte als identisch ergeben.

Daraus und aus der Tatsache, dass sich die Ableitungen jeweils auf einem Teil der Differenzen und aus Differenzen der Kontrollpunkte errechnen lassen (siehe Gleichung (2.9), S. 22), erhält man die Behauptung.

□

Nun sind am Anfang und Ende noch die Anteile der Kurve, die bei der Interpolation der offenen Kurve entstanden sind. Diese lassen sich jedoch

durch Konstruktion der Teilkurve $[\hat{t}_d, \hat{t}_{n+d+1}]$ eliminieren. Wendet man dann auf diese Teilkurve den Algorithmus 12.1 von Piegl u. Tiller [23] an, so periodisiert man die Kurve und erhält insgesamt eine periodische NURBS-Kurve, welche die Punkte $Q_0, \ldots, Q_n$ interpoliert.

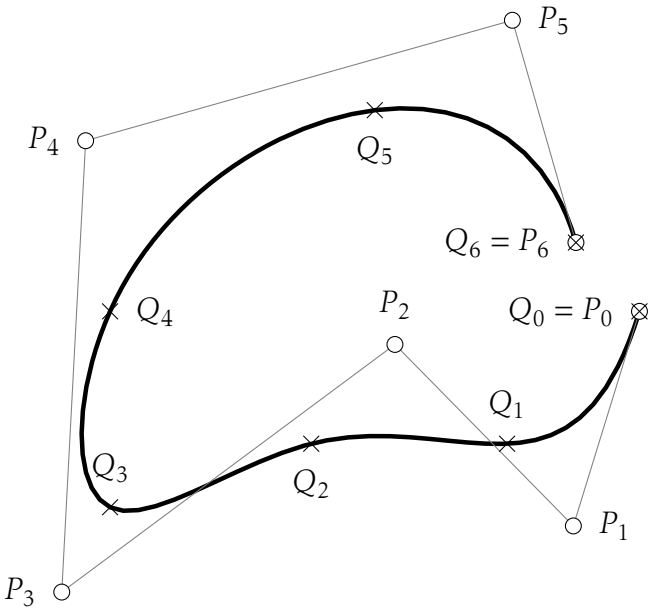
Beispiel 4.5 (periodische Interpolation).
In Fortführung des Beispiels 4.4 konstruiert man aus den dort gegebenen Interpolationspunkten diejenigen für die periodische Interpolation. Es ist $n = 6$, $d = 3$ und so entstehen die Interpolationspunkte $\hat{Q}_i, i = 0, \ldots 14$ wie folgt:

$$\begin{aligned}
&\hat{Q}_0 = \hat{Q}_7 = \hat{Q}_{14} = Q_3, && \hat{Q}_1 = \hat{Q}_8 = Q_4,\\
&\hat{Q}_2 = \hat{Q}_9 = Q_5, && \hat{Q}_3 = \hat{Q}_{10} = Q_6,\\
&\hat{Q}_4 = \hat{Q}_{11} = Q_0, && \hat{Q}_5 = \hat{Q}_{12} = Q_1,\\
&\hat{Q}_6 = \hat{Q}_{13} = Q_2
\end{aligned}$$

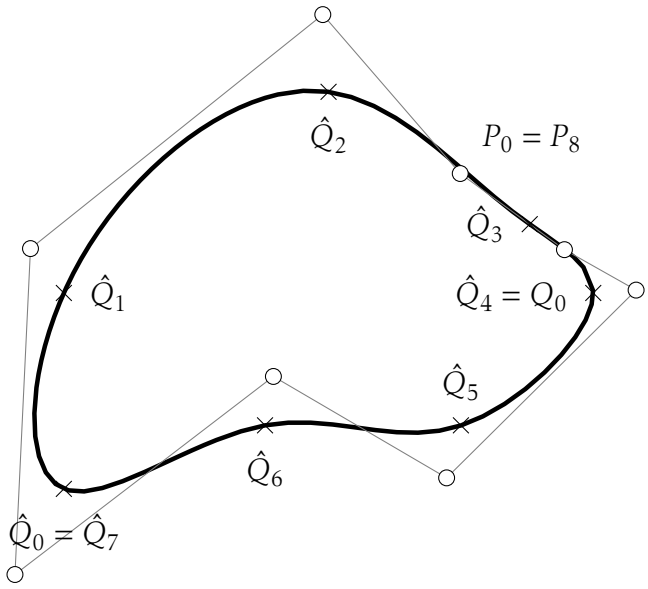
$\hat{Q}_0$, bis $\hat{Q}_3$ sind dabei der neue Anfang, $\hat{Q}_{11}$ bis Q_{14} das neue Ende, welche vor der Periodisierung abgeschnitten werden, die allerdings auch die Gleichheit der Ableitungen sicherstellen. Löst man damit das lineare Gleichungssystem und nimmt danach die Teilkurve zwischen den Punkten $\hat{Q}_3$ und $\hat{Q}_{10}$, so ist diese im Start- und Endpunkt nach dem Satz 4.7 in allen existierenden Ableitungen identisch. Weiter ist per definitionem $\hat{Q}_3 = \hat{Q}_{10}$. Schließt man dann die Kurve, entsteht eine periodische NURBS-Kurve, wie sie exemplarisch in Abbildung 4.4(b) dargestellt ist.

Einschränkungen an periodische Interpolation

Wie schon in dem Beispiel 4.5 zu erkennen ist, gibt es Fälle, in denen Interpolationspunkte dreifach vorkommen, genauer, wenn $n \leq 2d$ Interpolationspunkte für einen Grad d existieren. Das lineare Gleichungssystem ist von diesen Mehrfachvorkommen nicht betroffen, es ergeben sich jedoch auch einige Kontrollpunkte dabei dann doppelt bzw. dreifach. Für $n < 2d - 1$ treten dann in der Umrechnung zur periodischen Kurve (Teilkurve berechnen und diese schließen nach Algorithmus 12.1 von Piegl u. Tiller [23]) Artefakte in der Berechnung auf. Diese entstehen, da in dem Algorithmus gewichtete Differenzen der Kontrollpunkte verwendet werden. Daher ist das dargestellte Beispiel fast minimal. Lediglich mit einem Punkt weniger ließe sich noch eine Interpolation berechnen, die nach der Periodisierung weiterhin alle Interpolationspunkte durchläuft.



(a) offene Interpolation



(b) periodische Interpolation

Abbildung 4.4: Für die Interpolationspunkte $Q_0, \ldots, Q_6$ sind die beiden Interpolationen dargestellt: (a) zeigt die offene Interpolation nach Piegl u. Tiller [23], (b) die Erweiterung zur periodischen Interpolation.

4.5. Ersetzen von Teilkurven

Definieren einer Teilkurve

Die Erzeugung von Teilkurven wurde bereits in Abschnitt 4.1 angesprochen. Bei periodischen NURBS-Kurven entstehen dabei zwei offene NURBS-Kurven. Ist C eine periodische NURBS-Kurve auf dem Intervall $[a,b]$ und bildet man die Teilkurve auf $[a_1,b_1], a < a_1 < b_1 \leq b$, so entsteht eine zweite Teilkurve, die jedoch insgesamt auch wieder eine offene NURBS-Kurve ist. Diese ist definiert auf dem Intervall $[b_1, b + (a_1 - a)]$, da die Kurve eine Periode von $b - a$ besitzt.

Durch Angabe zweier Punkte zerfällt eine periodische NURBS-Kurve somit in zwei offene NURBS-Kurven. Seien u_1 und u_2 die beiden Parameter aus dem Intervall $[a,b]$. Für $u_1 < u_2$ ergibt sich exakt der oben angegebene Fall, für $u_1 > u_2$ geben die beiden Parameter denjenigen Kurventeil an, der über die Periodengrenze hinaus verläuft, also auf dem Intervall $[u_1, b + (u_2 - a)]$ definiert ist. Die zweite Teilkurve ergibt sich dann analog zum ersten Absatz.

Ersetzen einer Teilkurve

Ist eine periodische NURBS-Kurve in zwei offene NURBS-Kurven zerlegt, so kann eine der beiden ersetzt werden. Seien C_1, C_2 die beiden Teilkurven, wie exemplarisch in Abbildung 4.5 dargestellt. Nun soll die Teilkurve C_2 durch eine Kurve ersetzt werden, so dass dann wieder eine periodische NURBS-Kurve entsteht. Dazu erzeugt man mit der in Abschnitt 4.4 beschriebenen Interpolation eine offene NURBS-Kurve $\hat{C}_3$, die an beiden Enden mit der offenen Kurve C_1 so weit überlappt, dass sie an den Endpunkten von C_1 die gleichen Ableitungen besitzt. Die Kurve $\hat{C}_3$ beschneidet man dann analog zur periodischen Interpolation und erhält eine Kurve C_3, die auf dem gleichen Intervall definiert ist, wie C_2. Im Gegensatz zur periodischen Interpolation ist diese Überlappung und Beschneidung allerdings in zwei (statt nur in einem) Punkt notwendig, namentlich den Punkten $C_1(u_1)$ und $C_1(u_2)$. Ebenso analog zur Interpolation lassen sich dann C_1 und C_3 zunächst an dem einen gemeinsamen Ende, dann an dem anderen verbinden. Auch dies geschieht wie bei der Interpolation einer gesamten periodischen Kurve, nur dass hier die beiden Knotenvektoren verbunden und die Kontrollpunkte entsprechend zusammengefasst werden.

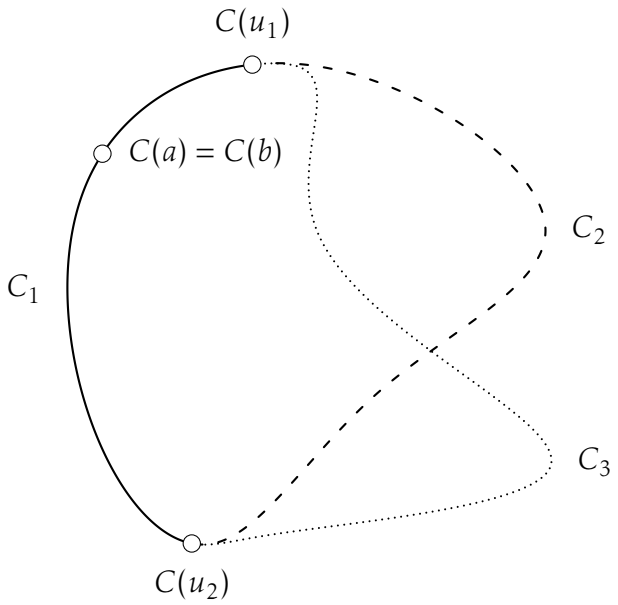


Abbildung 4.5: Die Kurve C (im Uhrzeigersinn definiert), wird mit $u_1 < u_2$ unterteilt in zwei Teilkurven C_1 (auf $[u_2, b + (u_1 - a)]$ und C_2 (auf $[u_1, u_2]$, gestrichelt). Ersetzt man C_2 durch C_3 (gepunktet), so entsteht wieder eine periodische Kurve C' in C_1 und C_3

Kapitel 5.

Darstellung von Hyperkanten

Nach Mäkinen [20] gibt es zwei Darstellungsmöglichkeiten für Hypergraphen: Die Kantenmetapher und die Teilmengenmetapher. In der Kantenmetapher wird eine Hyperkante als Bündel von Kanten dargestellt, das jeweils alle Knoten der Hyperkante paarweise miteinander verbindet. Vorteil dieser Metapher ist, dass auch gerichtete Hypergraphen gezeichnet werden können. Da ein Bündel im Verlauf seiner Kanten auch wiedererkannt werden muss, üblicherweise durch ein allen Kanten gemeinsames Kurvenstück, ist die Kantenmetapher jedoch eher unübersichtlich. Weitere Darstellungsformen finden sich in Kaufmann u. a. [19].

Diese Arbeit konzentriert sich auf die Darstellung einer Hyperkante in der Vorstellung als Teilmenge der Knoten, also der Teilmengenmetapher. Dabei werden Hyperkanten durch eine kontinuierliche, periodische Kurve dargestellt, welche genau die Knoten umfasst, die zu einer Hyperkante gehören [20][4, S. 1].

5.1. Der Hyperkantenumriss

Für einen Hypergraphen $\mathcal{H} = (V, \mathcal{E})$ ist eine einzelne Hyperkante $E' \subset V,\ E' \in \mathcal{E}$ eine nichtleere Teilmenge der Knoten. Angenommen für die Knoten $v_i \in E'$ sind die Positionen in einer Darstellung bekannt. Dann ist die Aufgabe der Darstellung nach der Teilmengenmethapher, einen Zusammenhang zwischen diesen Knoten in der Darstellung herzustellen. Dies wird durch eine Kurve C verwirklicht, die um alle Knoten herum verläuft. Diese Kurve $C = C(u), u \in [a, b] \subset \mathbb{R}$ heißt *Hyperkantenumriss* (engl. hyperedge shape). Für eine ästhetische Darstellung gibt es nun keine eindeutige Lösung, es lassen sich jedoch einige Anforderungen festlegen.

Anforderungen an den Umriss

Der Umriss sollte eine stetige, periodische Kurve sein, so dass die Kurve eine bestimme Fläche umreißt. Weiterhin sollte die Kurve injektiv sind. Dann bildet die Kurve, da sie periodisch ist, die Grenze einer Fläche, die zur Kreisscheibe homeomorph ist. Dies bedeutet, dass kein „Zerfallspunkt“

existiert und somit eine einzelne Fläche von der Kurve umrissen wird. Dies führt insgesamt zu mehr Übersichtlichkeit. Weitere Ausführungen zu Flächen als Spezialfall von Oberflächen finden sich in Bloch [7], Kapitel 2.

Ist nun also die Kurve Grenze einer Fläche, so lassen sich weiter die Begriffe *innerhalb* und *außerhalb* eines Hyperkantenumrisses definieren: Innerhalb ist derjenige endliche Flächeninhalt, der von der Kurve umfasst wird. Außerhalb hingegen ist die restliche Fläche des $\mathbb{R}^2$, die jedoch auch auf eine endliche Fläche reduziert werden kann, indem man die gesamte vom Hypergraphen eingenommene Fläche mit dem Äußeren schneidet. Innerhalb der Kurve liegen dann genau diejenigen Knoten $v \in V$, die zur Hyperkante E' gehören. Alle anderen Knoten $u \in V \backslash E'$ müssen außerhalb liegen.

Eine weniger formelle Forderung ist die Übersichtlichkeit. Ein Hyperkantenumriss sollte übersichtlich sein, also eine möglichst einfache Form haben, damit die zusammenhängenden Knoten schnell erfasst werden können. Diese Übersichtlichkeit bedeutet zusätzlich, dass nicht unverhältnismäßig viel „freie Fläche“ innerhalb des Umrisses liegen sollte. Außerdem sollten sich verschiedene Hyperkantenumrisse nicht zu häufig kreuzen, denn dies verringert die Übersichtlichkeit.

Neben diesen Anforderungen gibt es zwei Spezialfälle, bei denen zusätzlich eine andere Darstellung üblich ist, so dabei nicht die Übersichtlichkeit leidet. Im Allgemeinen sind die folgenden beiden Spezialfälle jedoch übersichtlicher als ein Umriss.

- Gilt $|E'| = 2$, so entspricht die Hyperkante formell einer Kante eines ungerichteten Graphen, da sie zwei Knoten verbindet. In diesem Fall kann die Hyperkante anstelle eines Umrisses auch durch eine offene Kurve dargestellt werden, deren Endpunkte die entsprechenden beiden Knoten sind.

- Ist die Hyperkante eine Schlinge, enthält also nur einen Knoten ($|E'| = 1$), so ist auch die Darstellung als Kurve üblich, die durch diesen einen Knoten verläuft. Dabei wird häufig der Übersichtlichkeit wegen eine kleine Ellipse verwendet.

Beispiel 5.1 (Ein Hypergraph, aus [4]).
Der Hypergraph $\mathcal{H} = (V, \mathcal{E})$ aus Beispiel 2.2 (s. S. 6) mit $V = \{x_i \mid i = 1, \ldots, 8\}$ und den Hyperkanten $\mathcal{E} = \{E_i \mid i = 1, \ldots, 6\}$ mit

$$E_1 = \{v_3, v_4, v_5\}, \qquad E_2 = \{x_5, x_8\}$$
$$E_3 = \{x_6, x_7, x_8\}, \qquad E_4 = \{x_2, x_3, x_7\}$$
$$E_5 = \{x_1, x_2\}, \qquad E_6 = \{x_7\}$$

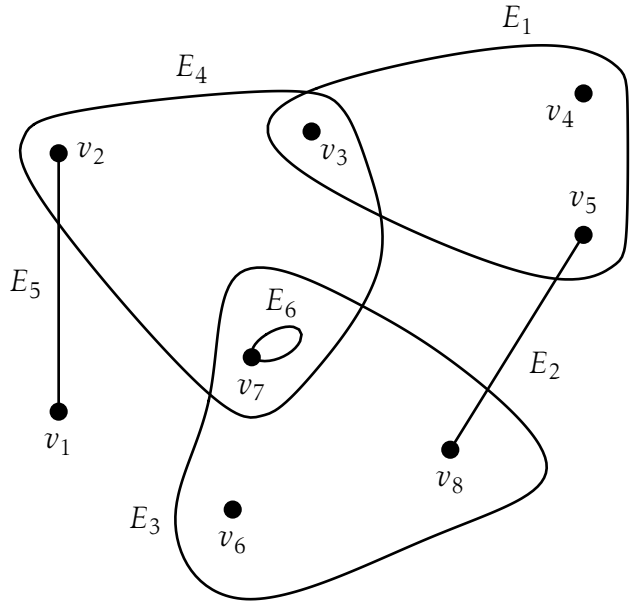


Abbildung 5.1: Darstellung eines Hypergraphen (nach Berge [4], S. 2). Dabei ist E_6 eine Schlinge, E_5 un E_2 werden als Kanten dargestellt.

enthält die Schlinge E_6, die als Ellipse dargestellt wird. Die Hyperkanten E_2 und E_5 werden als direkte Linien in Abbildung 5.1 dargestellt. Die Darstellung orientiert sich an derjenigen aus Berge [4], S. 2.

Eine weitere Eigenschaft eines Hyperkantenumrisses ist der Innenabstand. Dazu sei ein Hypergraph $\mathcal{H} = (V, \mathcal{E})$ mit Hyperkante $E = \{v_1, \ldots, v_k\}$, $v_i \in V$ gegeben. Für jeden Knoten $v_i \in E$ sei $p_i \in \mathbb{R}^2$ dessen Position in einer Darstellung und $s_i \in \mathbb{R}$ dessen Größe (Durchmesser des Kreises um p_i, als formelle Größe des Knotens).

Der Hyperkantenumriss $C(u)$ der Hyperkante E besitzt dann den *Innenabstand* δ, definiert durch

$$\delta = \min \left\{ \|\mathrm{proj}_C(p_i) - p_i\|_2 - \frac{s_i}{2} \,\middle|\, v_i \in E \right\}$$

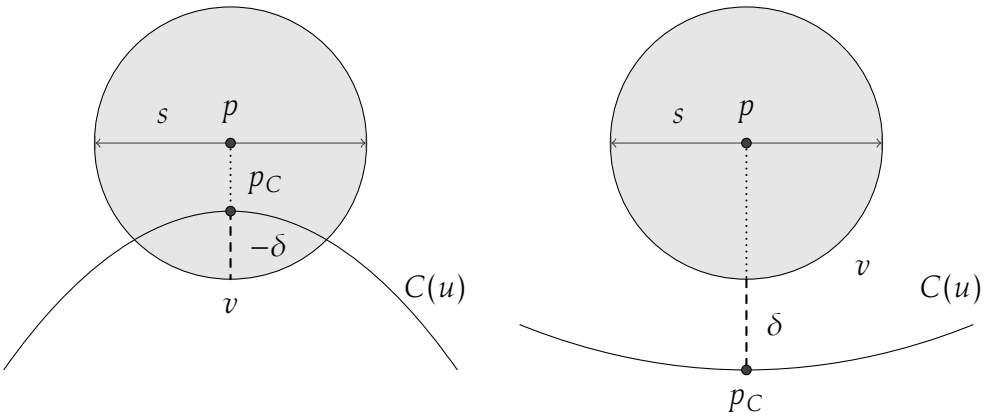
Der Wert δ entspricht der Länge der kürzesten Strecke vom Rand eines Knotens zur Kurve, denn die Projektion liefert denjenigen Punkt auf der Kurve mit dem kürzesten Abstand, von dem dann noch der Radius der Knotendarstellung subtrahiert wird.

Lemma 5.1.
Der Innenabstand δ wird genau dann negativ, wenn der Hyperkantenumriss durch einen Knoten verläuft.

Beweis.

Ist $\delta < 0$, so gibt es einen Knoten v', dessen Position p' einen kleineren Abstand zur Kurve hat als sein Radius $\frac{s'}{2}$. Somit verläuft die Kurve durch den Kreis, der den Knoten darstellt.

□



(a) Innenabstand $\delta < 0$ (b) Innenabstand $\delta > 0$

Abbildung 5.2: Der Innenabstand δ am Beispiel eines Knotens v mit Position p und Durchmesser s. Da in (a) die Strecke $\overline{p_C p}$ kürzer ist als $\frac{s}{2}$, ist $\delta < 0$, genauer der negative Wert der gestrichelten Strecke. In (b) hingegen ist die Strecke $\overline{p_C p}$ länger als $\frac{s}{2}$ wodurch $\delta > 0$ die Länge der gestrichelten Strecke ist.

Der Innenabstand lässt sich ein wenig erweitern, wenn man dem Hyperkantenumriss eine Linienbreite hinzufügt, mit der dieser in der Darstellung gezeichnet wird. Diese würde dann in dem Minimum ebenso von dem Abstand des Knotens zur Kurve subtrahiert werden.

In Abbildung 5.2 ist der Innenabstand visualisiert, einmal für ein positives, einmal für ein negatives δ.

Ein negativer Innenabstand führt zu einer sehr unübersichtlichen Darstellung, da häufig eine Darstellung in Graustufen verwendet wird, in der sowohl Knoten als auch die Umrisse schwarz dargestellt sind. Daher soll dieser durch folgende Definition verhindert werden, die zumindest die formell fassbaren Eigenschaften des Hyperkantenumrisses zusammenfasst:

Definition 5.1 (Validität einer Hyperkante).
Der Hyperkantenumriss $C(u)$ einer Hyperkante E aus einem Hypergraphen $\mathcal{H} = (V, \mathcal{E})$ heißt *valide* oder *gültig* in einer Darstellung, falls eine der folgenden Bedingungen erfüllt ist:

(i) Die Kante wird von genau einem Knoten gebildet ($|E| = 1$) und die Kurve $C(u)$ stellt eine Schlinge dar, das heißt, dass für die Position p des einen Knotens $v \in E$ und dessen Größe s gilt:

$$\|\text{proj}_C(p) - p\|_2 \leq \frac{s}{2}$$

(ii) Die Kante enthält zwei Knoten ($|E| = 2$) und die Kurve $C(u), u \in [a, b]$ ist eine offene Kurve, so dass für die beiden Positionen p_1, p_2 der Knoten von E gilt:

$$(C(a) = p_1 \wedge C(b) = p_2) \vee (C(b) = p_1 \wedge C(a) = p_2)$$

(iii) Die Kante besteht aus $k \in \mathbb{N}\backslash\{0\}$ Knoten ($E = \{v_1, \ldots, v_k\}$, $v_i \in V$, $i = 1, \ldots, k$). Für die Positionen $p_1, \ldots, p_k$ der Knoten gilt, dass diese innerhalb des Hyperkantenumrisses und die Positionen p_j aller Knoten $v_j \notin E$ außerhalb des Umrisses liegen. Zusätzlich ist der Innenabstand $\delta > 0$.

Die Definition der Validität bietet der Darstellungen von Schlingen zwei Möglichkeiten: Neben dem allgemeinen Fall (iii), die Schlinge durch eine Kurve um den einen Knoten herum darzustellen, ermöglich der Fall (i) eine für Schlingen übliche spezielle Darstellung. Ebenso lassen sich Hyperkanten, die zwei Knoten umfassen, entweder als Kante analog zu einem Graphen – Fall (ii) – oder als Umriss um die beiden Knoten herum – Fall (iii) – darstellen.

Für die Definition der Validität gibt es zwei Erweiterungsmöglichkeiten: Analog zum Innenabstand lässt sich ein Außenabstand definieren. Dieser muss im Gegensatz zum Innenabstand für alle Knoten $v \notin E$ erfüllt sein. In der Betrachtung des Innen- und Außenabstandes kann zusätzlich eine Linienbreite b des Hyperkantenumrisses eingebracht werden. Diese erweitert die Forderung derart, dass beispielsweise für den Innenabstand $\delta > \frac{b}{2}$ erfüllt sein muss. Beide Erweiterungen sind mit wenig Aufwand möglich und sollen der Einfachheit halber im Folgenden nicht weiter behandelt werden.

5.2. NURBS-Kurven als Hyperkantenumriss

Um in einem Programm den Umriss einer Hyperkante zu erzeugen und automatisch oder interaktiv zu gestalten, benötigt man eine Repräsentation der im letzten Abschnitt vorgestellten Kurve, die den Umriss angibt.

Dazu sind NURBS-Kurven sehr gut geeignet, wenngleich sie einige Nachteile mit sich bringen. Diese Vor- und Nachteile werden im Folgenden genannt und erläutert.

Detailgenauigkeit. NURBS-Kurven, die nach der periodischen Interpolation aus Abschnitt 4.4 erzeugt worden sind, erfüllen die im letzten Abschnitt genannte Periodizität und bilden somit einen Umriss. Da NURBS-Kurven die Kurve implizit durch ihre Knoten und Kontrollpunkte definieren kann in der Darstellung eine beliebig hohe Auflösung berechnet werden. Dies ist vor allem zum Erzeugen von Vektorgrafiken von Vorteil. Es gibt zwar kein offenes Format für Vektorgrafiken, das NURBS-Kurven explizit verarbeiten kann, jedoch ist die Auflösung und Länge bei der Angabe von Pfaden derart genau möglich, dass NURBS-Kurven durch einen sehr hochauflösenden Pfad gezeichnet werden können.

Lokalität. Die NURBS-Kurven bieten in der Bearbeitung den großen Vorteil, dass eine Veränderung eines Kontrollpunktes nach Eigenschaft B2 (siehe Seite 11) nur lokalen Einfluss auf die Kurve hat. Ist ein Umriss also beinahe zufriedenstellend vorhanden, lassen sich Änderungen auf kleinen Bereichen durchführen, ohne den Rest der Kurve noch zu verändern. Ist der Einfluss einer Änderung zu groß, kann dies durch Einfügen von Knoten behoben werden.

Stetigkeit. Je nach gewünschter Stetigkeit kann eine NURBS-Kurve mit Grad d erzeugt werden. Dabei ist die Existenz einer stetigen zweiten Ableitung für einige Algorithmen notwendig, etwa die Projektion aus Abschnitt 4.3, jedoch ist dies für eine NURBS-Kurve vom Grad 2 größtenteils gegeben. An den Knotenstellen auf dem Intervall $[a, b]$ ist diese zwar nur einmal stetig differenzierbar, an diesen wird die Kurve vor der Projektion jedoch aufgetrennt. Die Trennstellen werden dann zu einem Sonderfall im Algorithmus. Ein zu hoher Grad hingegen führt – vor allem bei der Projektion – zu längerer Laufzeit.

Polynome. Obwohl NURBS in ihrer Definition ein wenig kompliziert sind, bestehen sie lediglich aus sich stückweise überlagernden rationalen Poly-

nomen, die somit in mathematischer Hinsicht einfach sind. Dadurch sind Beweise, die auf NURBS-Kurven basieren, nur in der Formalisierung kompliziert, da man dann – beispielsweise für genau einen Punkt auf der Kurve – in den richtigen Intervallen auf Knotenvektor und beteiligten Kontrollpunkten agieren muss.

Mangelnde Intuitivität. An einer Stelle auf der NURBS-Kurve ist es möglich, die Daten der Kurve zu verändern und trotzdem die gleiche Kurve zu behalten. Zusätzlich sind die Kontrollpunkte mit unterschiedlichen Gewichten (und somit Einflüssen auf die Kurve) nur bedingt intuitiv, wenn man eine Bearbeitung am Computer vornehmen möchte. Hier ist es also von Bedeutung, den Benutzer eines Programms nicht mit der Fülle an Daten zu überfordern.

Implementierungsaufwand. Obwohl die NURBS-Kurven lediglich aus stückweisen Polynomen bestehen, ist für den Umgang mit ihnen doch ein Aufwand in der Implementierung notwendig, der ausgehend von den Basisfunktionen und deren Ableitungen bis hin zu den Algorithmen in Kapitel 4 einige Einarbeitungszeit benötigt.

Trotz einiger Nachteile bilden die NURBS-Kurven sowohl bezüglich ihrer Erzeugung als auch Manipulation und Speicherung einen sehr guten Ansatz zur Darstellung von Hyperkantenumrissen.

5.3. Automatische Erzeugung

Auf Basis der Interpolation periodischer NURBS-Kurven aus Abschnitt 4.4 lässt sich eine automatische Erzeugung des Umrisses angeben. Nach Berechnung der konvexen Hülle, die von den Positionen der Knoten einer Hyperkante definiert wird, erweitert man dieses Polygon, so dass $\delta > 0$ erfüllt ist. Dazu eignet sich folgendes Vorgehen:

In einer Knotenposition der konvexen Hülle verlängert man die beiden Strecken des Polygons der konvexen Hülle um die gewünschte Länge über den Knoten hinaus und erhält so zwei Interpolationspunkte. Liegen die beiden Strecken auf einer Geraden wie in Abbildung 5.3 bei Knoten v_5, so wählt man einen Interpolationspunkt auf der Geraden orthogonal zu den Polygonstrecken außerhalb der konvexen Hülle. Bei dem Knoten v_5 entspricht dies der Position direkt oberhalb des Knotens. Auf Basis dieser Punkte ergibt sich mit der periodischen Interpolation ein Umriss wie bei der Kante E_1 in Abbildung 5.3.

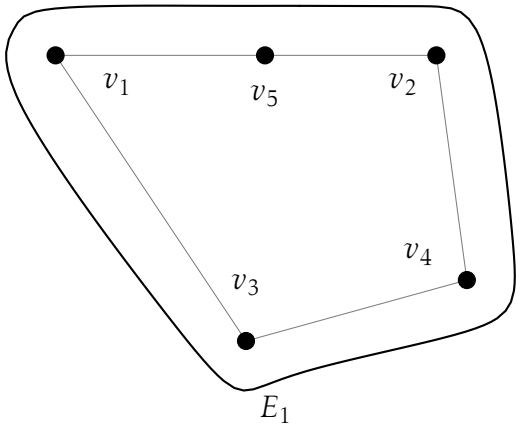


Abbildung 5.3: Ausgehend von der konvexen Hülle der fünf Knoten erzeugter Umriss der Hyperkante E_1.

Eine Einschränkung ist dabei, dass auch hier für die Interpolation genügend Punkte verfügbar sein müssen. Die Konstruktion über die Verlängerung der Strecken liefert pro Strecke der konvexen Hülle zwei Interpolationspunkte.

Liegen in dieser konvexen Hülle auch Knoten, die nicht zur entsprechenden Hyperkante gehören, so entsteht kein gültiger Hyperkantenumriss. Trotzdem kann die konvexe Hülle dann ein guter Ausgangspunkt sein, um einen Umriss zu erstellen. Diese hat den Vorteil, dass aufgrund der Konstruktion der Abstand des Umrisses zu den Knoten relativ einheitlich ist. Daher eignet sich ein solcher Umriss gut als Ausgangspunkt, wenn dann nur geringe Teile verändert werden müssen.

5.4. Validierung eines Hyperkantenumrisses

In größeren Hypergraphen oder in Algorithmen, die eine Erzeugung der Darstellung iterativ durchführen, ist es sinnvoll, die Validität eines Hyperkantenumrisses zu prüfen. Die Abstände der Knoten und somit der minimale Abstand eines Knotens zur Kurve lässt sich über die Projektion bestimmen. Ebenso lassen sich die Fälle (i) und (ii) in Definition 5.1 über eine Projektion beziehungsweise eine Auswertung der Kurve an Anfangs- und Endpunkt bestimmen.

Die Zuordnung, welche Knoten innerhalb und außerhalb des Hyperkantenumrisses sind, ist hingegen nicht sofort ersichtlich. In der Computergrafik sind bei geschlossenen Polygonzügen Scan-Line-Algorithmen üblich. Diese durchlaufen das Bild zeilenweise und entscheiden anhand der Anzahl Überschreitungen der Strecken des Polygons, ob ein Punkt innerhalb

liegt oder nicht. Diese Algorithmen sind für einen Hyperkantenumriss nur dann anwendbar, wenn mit einer passenden Genauigkeit ein Polygonzug daraus erzeugt werden kann. Diese Genauigkeit führt in dem einen Extremum zu Fehlentscheidungen, wenn der Polygonzug nur grob der Kurve folgt, und in dem anderen zu einer sehr langen Laufzeit, da der Polygonzug bei zu hoher Genauigkeit oder einem langen Umriss sehr umfangreich ausfällt.

Um diese beiden Nachteile zu umgehen, basiert der nun vorgestellte Algorithmus auf der Projektion und den Eigenschaften periodischer NURBS-Kurven. Dazu sei im folgenden $C(u), \quad u \in [a,b]$ ein Hyperkantenumriss in Form einer NURBS-Kurve vom Grad d mit n Kontrollpunkten, $p \in \mathbb{R}^2$ ein Punkt in der Ebene und $p_C = \text{proj}_C(p)$ seine Projektion auf die Kurve.

Idee

Die Idee des Algorithmus ist, ausgehend von einer Menge von (noch zu bestimmenden) Punkten die beiden Flächen, die das Innere und Äußere des Umrisses bilden, so weit abzudecken, dass sich entscheiden lässt, welche Knoten des Graphen innerhalb und außerhalb liegen.

Der Algorithmus basiert auf einer grundlegenden Feststellung, die in dem folgenden Lemma präsentiert wird:

Lemma 5.2.
Überschneiden sich für zwei Punkte $p,q \in \mathbb{R}^2$ die Kreise um p und q mit den Radien $\overline{p_C p}$ bzw. $\overline{q_C q}$ um mehr als einen Punkt, so liegen beide Kreise entweder komplett innerhalb oder komplett außerhalb des Hyperkantenumrisses.

Beweis.

Angenommen p und q liegen nicht beide innerhalb oder beide außerhalb des Hyperkantenumrisses $C(u)$. Dann schneidet die Verbindungsstrecke von p und q den Hyperkantenumriss mindestens einmal in einem Punkt $t \in \overline{pq} \cap C(u)$. Überschneiden sich die genannten Kreise in mehr als einem Punkt, so gilt $\overline{pq} = \overline{pt} + \overline{qt} < \overline{p_C p} + \overline{q_C q}$; ein Widerspruch zur Definition der Projektionen p_C und q_C.

Somit liegen p und q entweder beide innerhalb oder aber beide außerhalb des Hyperkantenumrisses.

□

Die Idee des Algorithmus ist nun, mit einer Menge von sich überschneidenden Kreisen um gewisse Punkte $p \in \mathbb{R}^2$ mit dem Radius $\overline{p_C p}$ die Fläche innerhalb (und analog außerhalb) abzudecken und dabei sowohl die Eigenschaften der NURBS-Kurve als auch die Verteilung der Knoten des Graphen zu berücksichtigen.

Dazu werden zunächst alle Kontrollpunkte $P_0,\ldots,P_n \in \mathbb{R}^2$ der periodischen NURBS-Kurve und alle Knotenpositionen $p_1,\ldots,p_{|V|}$ als Anfangspunkte gewählt. Überschneiden sich von diesen einige der Kreise, werden sie wegen Lemma 5.2 zu einer gemeinsamen Menge zusammengefügt. Dies wird realisiert durch eine partielle Funktion f (d.h. für jedes Element der Urbildmenge existiert höchstens ein Bild, der Definitionsbereich verändert sich im Laufe des Algorithmus), welche Mengen von Kreisen anhand ihrer Mittelpunkte bildet und initialisiert wird durch

$$f : \mathbb{R}^2 \to \{1,\ldots,|V|+n+1\}, \quad f(p_i) = i, \quad f(P_i) = |V|+1+i \tag{5.1}$$

Die Zuordnung der Kreismittelpunkte zu der beschriebenen Menge reicht hier aus, da sich der Radius stets als Abstand zum Hyperkantenumriss ergibt.

Überschneiden sich zwei Kreise, wird deren Funktionswert auf das Minimum der beiden bisherigen Funktionswerte gesetzt. Allen Punkten, die auf das Maximum der bisherigen beiden Funktionswerte abgebildet wurden, wird ebenso das Minimum als neuer Funktionswert zugewiesen.

Bildet die Funktion f nur noch auf exakt zwei Werte ab, so sind zwei Punktmengen bzw. Flächen entstanden, von denen die eine innerhalb des Hyperkantenumrisses liegt, die andere außerhalb. Zur Unterscheidung zwischen inner- und außerhalb hilft folgendes Lemma:

Lemma 5.3.
Der Kontrollpunkt $P_j = \begin{pmatrix} x_j & y_j \end{pmatrix}^T$ einer periodischen NURBS-Kurve $C(u)$ mit $y_j \leq y_i, \;\; i = 0,\ldots,n$ liegt außerhalb der von $C(u)$ umschlossenen Fläche.

Beweis.

Aufgrund der Eigenschaft B7, dass die NURBS-Kurve innerhalb der konvexen Hülle ihrer Kontrollpunkte liegt, folgt das Lemma sofort.

□

Wahl eines Nachfolgers

Da ausgehend von der genannten anfänglichen Menge $\{p_1,\ldots,p_{|V|},P_0,\ldots,P_n\}$ an Punkten sehr selten der gewünschte Fall $|\text{Bild}\, f| = 2$ entsteht, ist die Wahl

von Nachfolgern notwendig. Für einen Punkt p lassen sich dabei alle Punkte p' innerhalb und auf dem Kreis mit Radius $\overline{p_C p}$ um p wählen. Um eine möglichst große neue Fläche durch den Nachfolger abzudecken, sollte p' weit von p entfernt sein und liegt daher auf dem Kreis. Für einen großen Radius $r_{p'}$ ist es sinnvoll, lediglich die dem Projektionspunkt gegenüberliegende Kreishälfte für die Wahl von p' zu verwenden (siehe Abbildung 5.4(a)).

Wählt man p' „gegenüber" von p_C, wie in Abbildung 5.4(b) und (c), so ergibt sich der Radius zu $\overline{p'_C p'} \leq 2\overline{p_C p}$. Gleichheit tritt dabei im Fall $p_C = p'_C$ auf. Ein größerer Radius von p' ist nicht möglich. Ein kleinerer Radius für p' entsteht, sobald ein Teil der Kurve innerhalb des in Abbildung 5.4(b) dargestellten Kreises um p' liegt.

Verläuft dieser Teil der Kurve nah an dem eben gewählten p' entlang, so wird der Radius $r_{p'}$ klein und p' ist als Nachfolger ungeeignet. Dies ist in Abbildung 5.4(c) dargestellt und entspricht optisch einer Art Tunnel oder Engpass, den die Fläche des Hyperkantenumrisses an dieser Stelle besitzt. In diesem Fall ist es sinnvoll, einen der beiden seitlichen Punkte q_1, q_2 zu wählen.

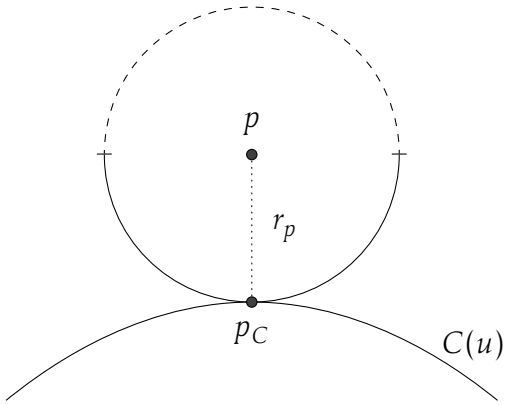
Eine algorithmisch einfache Wahl der Nachfolger, welche die ausgeführten Situationen berücksichtig, verläuft wie folgt:

Sei $r_{p'}$ die Länge der Strecke $\overline{p'_C p'}$ und r_p analog diejenige von $\overline{p_C p}$. Dann ist $r_{p'} \in [0, 2r_p]$. Mit $\alpha = \frac{\pi}{2} + \frac{\pi r_{p'}}{4 r_p}$ sind

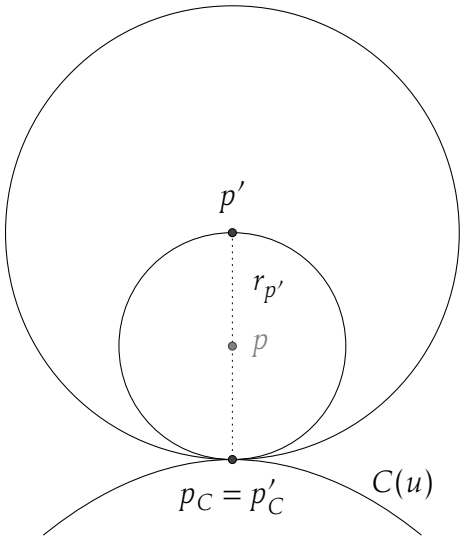
$$q_1 = p + \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}(p_C - p), \quad q_2 = p + \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}(p_C - p) \quad (5.2)$$

zwei mögliche Nachfolger auf dem Kreis. Die beiden Punkte q_1 und q_2 werden dabei unter Berücksichtigung des Radius $r_{p'}$ gewählt. Ist dieser Radius maximal ($r_{p'} = 2r_p$) so ist $\alpha = \pi$, die Strecke $p_C - p$ wird für beide Nachfolger um 180° gedreht und es gilt $p' = q_1 = q_2$. Dieser Fall entspricht der Abbildung 5.4(b). Für einen kleineren Radius wie etwa $r_{p'} = r_p$ ergibt sich $\alpha = \frac{3\pi}{4}$, die beiden Nachfolger sind auf dem Kreis sich „schräg entfernt" von der Strecke $p_C - p$. Liegt an dieser Stelle ein Engpass vor, so wird er durch diese Wahl von dem nachfolgenden Kreis „weiter ausgefüllt". Im Minimum ist $r_{p'} = 0$, $\alpha = \frac{\pi}{2}$ und die beiden Nachfolger sind komplett „seitlich" angeordnet, da „gegenüber" gar kein nachfolgender Kreis gegeben ist. In Abbildung 5.4(c) ist $r_{p'}$ sehr klein, wodurch die beiden Nachfolger nur sehr wenig „oberhalb" des seitlichen Falles liegen.

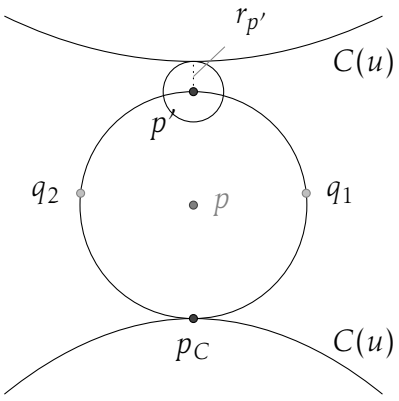
Alle Kreise um q_1 und q_2 schneiden definitiv den Kreis um p und liegen daher in der gleichen Menge bezüglich f wie p. Die Wahl der beiden



(a) Tangentialkreis an C um p mit gestricheltem Bereich für die Nachfolgerwahl



(b) maximaler Fall: $r_{p'} = 2r_p$



(c) sehr kleiner Radius $r_{p'}$

Abbildung 5.4: Kreise und ihre Nachfolger in der Validierung: (a) p und der Tangentialkreis an p_C auf der Kurve. Die Wahl des Nachfolgers erfolgt dabei aus dem gestrichelten Kreisbogen. Exemplarisch sind dazu zwei Fälle dargestellt: (b) maximaler Radius $r_{p'} = 2r_p$ und (c) sehr kleiner Radius $r_{p'}$ und die daraus resultierenden Nachfolger q_1, q_2.

Punkte q_1, q_2 berücksichtigen dabei das Umfeld um den Punkt p, um zu vermeiden, dass der Nachfolgerkreis zu klein wird.

Im Algorithmus 5.1 werden nun maximal zweimal die Nachfolger q_1 und q_2 gewählt: Einmal zu Beginn, also wenn p keinen Vorgänger hat (Zeile 23 und 24, da nur Anfangspunkte keinen Vorgänger haben) und höchstens einmal danach, falls $r_{p'} = 0$ ist, also p' auf der Kurve liegt (Zeile 25 bzw. die Werte w_p). Dies entspricht der Situation, dass der Kreis einen gesamten „Tunnel“ in seiner Höhe einnimmt. Andernfalls wird derjenige Punkt auf dem Kreis gewählt, der weiter vom Vorgänger von p (im Quelltext pre(p)) entfernt ist. Dies entspricht dem ständigen Vorangehen in etwa derselben Richtung. Zusätzlich werden Kreise, die komplett in anderen Kreisen liegen, verworfen. Dies verkleinert sukzessive die Menge $\mathcal{M}$ an Punkten, die noch keinen Nachfolger haben. Damit gilt

Satz 5.1 (Terminierung der Validierung).
Der Algorithmus 5.1 zur Validierung eines Hyperkantenumrisses terminiert.

Beweis.

Zu Beginn besteht die Menge $\mathcal{M}$ aus allen Knotenpositionen p_i und allen Kontrollpunkten P_i. Danach wird der aktuell behandelte Punkt aus $\mathcal{M}$ entfernt (Zeile 17) und dessen Nachfolger werden eingefügt (Zeilen 24, 26 bzw. 28). Dadurch enthält $\mathcal{M}$ immer all diejenigen Punkte, deren Kreise noch nicht betrachtet worden sind. Sei die Mächtigkeit der Anfangsmenge $|\mathcal{M}| = k$. Für jeden dieser Punkte aus der anfänglichen Menge $\mathcal{M}$ gilt nach der Einschränkung über die Variablen w_p, dass höchstens zweimal jeweils zwei Nachfolger zu $\mathcal{M}$ in die Menge gelegt werden (zu Beginn – Zeile 24 – und genau einmal für einen Pfad in Zeile 26), andernfalls höchstens ein Nachfolger (Zeile 28). Damit gilt zu jedem Zeitpunkt des Algorithmus $|\mathcal{M}| \leq 4k$.

Für den Fall, dass $|\text{Bild}\, f| = 2$ ist, terminiert der Algorithmus, da dies die Schleife (Zeile 16–33) beendet. Zusätzlich wird die Menge $\mathcal{M}$ in ihrer Mächtigkeit verringert, wenn für einen Punkt kein Nachfolger gewählt wird. Dies ist der Fall, wenn für einen Punkt $\hat{p} \in \mathcal{M}$ dessen Kreis von irgendeinem vorherigen Kreis komplett überdeckt wird. Dazu dient die Menge $\mathcal{N}$ alle derjenigen Punkte, die einen Nachfolger besitzen sowie die Bedingung in Zeile 18. Dass Kreise ausgeschlossen werden, liegt an der Endlichkeit der inneren Fläche. Die äußere Fläche lässt sich ebenso auf eine endliche begrenzen durch die maximalen und minimalen Eckpunkte des Hypergraphen.

Algorithmus 5.1: Validierung des Hyperkantenumrisses

Eingabe: Ein Hypergraph $\mathcal{H} = (V, \mathcal{E})$, $V = \{v_1, \ldots . v_n\}$,
eine spezielle Hyperkante $E \in \mathcal{E}$ mit ihrem Umriss $C(u)$
Position p_i und Größe s_i eines jeden Knotens $v_i \in V$

Ausgabe: Die Korrektheit des Hyperkantenumrisses.

Daten: Der Vorgänger $\text{pre}(q_i)$ von q_i
w_{q_i}, der wahr ist, gdw. es auf dem Weg zu $q_i \in \mathcal{N} \cup \mathcal{M}$ einmal zwei Nachfolger gab
Die Mengen $\mathcal{M}$ der aktuellen Punkte und $\mathcal{N}$ aller bereits betrachteten Punkte
Eine Funktion $f : \mathcal{N} \cup \mathcal{M} \to \mathbb{N}$ wie in Gleichung (5.1).

Algorithmus
Initialisiere $\mathcal{M}$ mit allen Punkten aus Gleichung (5.1) und initialisiere f analog.
Setze dabei je den Vorgänger $\text{pre}(p)$ auf nicht existent und w_p =falsch.
Setze $\mathcal{N} = \emptyset$ und bestimme den Kontrollpunkt $\hat{P}$ mit minimalem y.
Solange $\mathcal{M} \neq \emptyset$ und $|f(V \cup \{\hat{P}\})| > 2$
Wähle $p \in \mathcal{M}$, entferne diesen aus $\mathcal{M}$ und berechne $\text{proj}_C(p)$ sowie r_p.
Falls der Kreis mit Radius r_p um p nicht komplett in einem aus $\mathcal{N}$ liegt
Füge p zu $\mathcal{N}$ hinzu.
Falls der Kreis um p den Kreis eines $y \in \mathcal{N}$ schneidet
Setze $\forall x \in \mathcal{N} : f(x) \in \{f(y), f(p)\}\ f(x) = \min\{f(y), f(p)\}$
Bestimme q_1 und q_2 nach Gleichung (5.2)
Falls $\text{pre}(p)$ nicht existiert
Füge q_1 und q_2 zu $\mathcal{M}$ hinzu, setze $w_{q_1} = w_{q_2}$ = falsch
Falls der Punkt $p' = p - (p_c - p)$ auf der Kurve liegt und w_p =falsch
Füge q_1 und q_2 zu $\mathcal{M}$ hinzu, setze $w_{q_1} = w_{q_2}$ = wahr
sonst
Füge q_i zu $\mathcal{M}$ hinzu, mit $\|q_i - \text{pre}(p)\|_2 > \|q_j - \text{pre}(p)\|_2, i \neq j$
Setze $w_{q_i} = w_p$
Ende Falls
Für die neuen Werte in $\mathcal{M}$ setze $\text{pre}(q_i) = p$ und $f(q_i) = f(p)$
Ende Falls
Ende Solange
Falls $|f(V \cup \{\hat{P}\})| > 2$
Keine Entscheidung, da zu viele Teilmengen
sonst
Falls $\exists x : \forall v_i \in E : f(p_i) = x \wedge \forall v_j \notin E : f(p_j) = f(\hat{P})$ und $\delta > 0$
Rückgabewert: Der Umriss ist *korrekt*.
sonst
Rückgabewert: Der Umriss ist *nicht korrekt*.
Ende Falls

Somit verringert sich $|\mathcal{M}|$ während der gesamten Abarbeitung des Algorithmus und enthält in diesem Ablauf maximal $4k$ Punkte. Falls der Zustand $|\text{Bild}f| = 2$ nicht erreicht wird, terminiert die Schleife (Zeile 16–33) sobald $|\mathcal{M}| = 0$ erfüllt ist. Im Fall $|\mathcal{M}| = 0$ und $|\text{Bild}f| > 2$ kann die Validität nicht entschieden werden und dies wird ausgegeben. Andernfalls existieren exakt zwei Mengen bezüglich f und die Prüfung der inneren und äußeren Knoten ist möglich.

□

Der Fall, dass mehr als nur die zwei Mengen (eine von inneren, eine von äußeren Punkten) entstehen, kommt in zwei Situationen vor:

Zum einen, falls die Kurve nicht injektiv ist (formell muss dabei ein Endpunkt ausgeschlossen werden, da $C(a) = C(b)$). Dann gibt es zwei innere Flächen, die nie verbunden werden und insgesamt somit mindestens drei Flächen, die je einen anderen Funktionswert in ihren Kreisen haben. Zum anderen kann es sein, dass ein „Tunnel“ in dem Hyperkantenumriss vorkommt, in dem keine Kontrollpunkte liegen. Dann beginnt in diesem auch kein Kreis, der sich nach dem Ausweiten auf „Tunnelhöhe“ in diesem zu beiden Seiten ausbreitet. Dadurch bleiben die beiden nur durch den Tunnel verbundenen Flächen eventuell unverbunden und der Algorithmus terminiert mit mindestens drei Mengen. In diesen Fällen kann der Algorithmus keine Entscheidung treffen. Die Injektivität einer periodischen NURBS-Kurve $C(u)$ auf dem Intervall $[a, b]$ zu prüfen bleibt ein offenes Problem.

Der Algorithmus wurde in der Implementierung um kleine Details erweitert. Diese wurden in der bisherigen Beschreibung nicht erwähnt, da sie einfach zu implementieren sind, jedoch den Algorithmus 5.1 unnötig verkompliziert hätten. Diese Erweiterungen sind:

(a) Diejenigen Knoten, welche die Bedingung in Zeile 37 verletzen, können gesammelt und zusätzlich zu dem Ergebnis, dass der Umriss nicht valide ist, markiert werden.

(b) Anstelle der Forderung, dass $\delta > 0$ erfüllt sein muss, ist die Variante implementiert, dass für einen Innenabstand δ_E der Hyperkante E (dessen Umriss geprüft wird) gilt: $\delta > \delta_E$. So bleibt zwischen den Knoten und dem Hyperkantenumriss ein gewisser Abstand bestehen.

(c) Die Abbruchbedingung der Schleife lässt sich dadurch verschärfen, dass abgebrochen wird, sobald für ein $v \in E$ und ein $u \notin E$ gilt, dass $f(u) = f(v)$. Dann ist der Umriss nicht valide.

Fazit

In den meisten Fällen terminiert der Algorithmus zur Validierung innerhalb weniger Schritte, vor allem, wenn der Hyperkantenumriss ein Kreis oder der konvexen Hülle sehr ähnlich ist. Bei sehr komplizierten Formen als Umriss benötigt der vorgestellte Algorithmus aufgrund der Entwicklung in verschiedenen Richtungen einige Zeit, um alle Bereiche zu erschließen. Das Ausschlusskriterium neuer Punkte ist dabei häufig die Ursache.

Insgesamt hängt die Laufzeit im wesentlichen von der Geschwindigkeit der Projektion ab, da diese für jeden Punkt, der aus M entfernt wird, aufgerufen wird.

Kapitel 6.

Implementierung

Die Implementierung basiert auf der in [5] begonnenen Implementierung eines Editors für Graphen. Dieser wurde im Rahmen der vorliegenden Diplomarbeit auf Hypergraphen erweitert. Grundlage ist also die Programmiersprache Java ab Version 1.5. Die grafische Oberfläche wurde mittels Java Swing realisiert. Das Programm ist plattformunabhängig, es werden jedoch einige spezielle Systemeigenschaften berücksichtigt. So sind etwa die Tastenkürzel und das „Look-and-Feel“ jeweils an das System angepasst, auf dem es gestartet wird.

Eine Kurzanleitung mit zwei Tutorien, einer Beschreibung der Dialoge und Auflistung der Interaktionen ist in Anhang A beigefügt.

In der Programmierung wurden verschiedene Konzepte umgesetzt: Zentrale Verwendung findet das Model-View-Control-Muster, in dem die Daten (Model) von der Darstellung (View) und diese beiden wiederum von der Behandlung der Eingabe (Control) klar getrennt werden. Dadurch kann eine dieser Komponenten ausgetauscht, erweitert oder intern verändert werden, ohne das die anderen Bereiche betroffen sind. Eine Erläuterung zur Struktur der Implementierung und einzelnen Klassen findet sich in Anhang C.

Dazu wird weiter das Observer-Muster eingesetzt. Dies ermöglicht einer Klasse, dem Observer (dt. Beobachter), eine andere zu beobachten, die beobachtbare Klasse. Ändert sich die beobachtete Klasse, so wird der Observer darüber informiert, und kann abhängig von der Art der Änderung reagieren.

Der Editor „Gravel“ ist Open Source, er ist veröffentlich unter der „GNU Public License Version 3“[1]. Sowohl das Programm selbst als auch der Quelltext können im Rahmen dieser Lizenz verwendet werden[2]. Details zu den Lizenzen der Software und in ihr verwendeter Komponenten finden sich in Anhang D.

[1] Die kompletten Lizenzbestimmungen finden sich unter `http://www.gnu.org/licenses/gpl-3.0.html`

[2] „Gravel“ ist verfügbar unter der Adresse `http://gravel.darkmoonwolf.de`

Interaktion mit NURBS-Kurven

NURBS-Kurven bieten viele Möglichkeiten zur Modifikation, haben jedoch auch einen großen Umfang an Daten. Bewegt man beispielsweise einen Kontrollpunkt auf die Kurve zu (in Richtung seines Projektionspunktes) und verringert entsprechend sein Gewicht, so bleibt die Kurve identisch und man erhält unterschiedliche Darstellungen derselben Kurve. Ist das Bewegen eines Kontrollpunktes in einer grafischen Oberfläche gut darstellbar, gilt dies für das Gewicht eines Punktes nicht direkt.

Um nun dieser Komplexität der NURBS-Kurven gerecht zu werden, ohne jedoch einen zu hohen Grad an Kompliziertheit in der Oberfläche zu erzeugen, wurde in der Implementierung auf die Darstellung der Kontrollpunkte bzw. des Kontrollpunktpolygons verzichtet. Stattdessen kann die Kurve an einem beliebigen Punkt durch Drag&Drop, also Anklicken und Ziehen, verändert werden. Dann wird jeweils derjenige Kontrollpunkt ermittelt, der auf den angeklickten Punkt den größten Einfluss hat. Dieser wird dann so weit verschoben, dass die Kurve durch die aktuelle Mausposition verläuft.

Außerdem lässt sich jede Manipulation direkt rückgängig machen und danach auch wieder herstellen. Dies fördert die Direktheit aus den Kriterien zur Dialoggestaltung (siehe Herczeg [17], Kapitel 8).

Speicherformat

Zur späteren Weiterverarbeitung können Graphen gespeichert werden. Das Dateiformat GraphML wurde von Brandes u. a. [9] entwickelt und ist ein XML-basiertes Dateiformat. Für GraphML steht ein XML-Schema zur Verfügung[3], das zur Überprüfung der Gültigkeit genutzt werden kann. Außerdem kann dieses Dateiformat um eigene Definitionen erweitert werden. Das Dateiformat von „Gravel“ nutzt diese Eigenschaft, um die Darstellungsinformationen von Knoten, Kanten und Hyperkanten in der XML-Datei zu speichern. Diese Erweiterung bleibt aufgrund der eigentlichen Definition von GraphML weiterhin ein GraphML-Format[4].

Ein XML-Format hat den großen Vorteil, dass es einerseits durch den logischen Aufbau für den Menschen verständlich bleibt, andererseits auch für Programme einfach zu verarbeiten ist, indem man einen XML-Parser verwendet. Ein validierender Parser kann zusätzlich die Angabe eines XML-Schemas im XML-Dokument erkennen und verarbeiten. Er prüft dann, ob

[3] verfügbar unter `http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd`

[4] die Erweiterungen befinden sich in der Datei `http://gravel.darkmoonwolf.de/xmlns/gravelml.xsd`

alle geforderten Felder vorhanden sind. So kann selbst nach eigenen Änderungen an der Datei dessen Gültigkeit geprüft werden bzw. eine ausführliche Meldung ausgegebene werden, welche Informationen gegebenenfalls nicht gültig sind.

Ein Nachteil der Angabe eines XML-Schemas ist, dass diese Angabe üblicherweise auf die .xsd-Datei im Internet verweist. Dadurch können Fehler im Schema behoben werden, die dann bei allen sofort wirksam sind. Allerdings ist dadurch während der Verarbeitung, also während des Ladevorgangs eines Graphen, eine Internetverbindung notwendig.

Ein ausführliches Beispiel mit Erläuterungen findet sich in Anhang B.

Kapitel 7.

Zusammenfassung und Ausblick

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde für die Darstellung von Hypergraphen in der Vorstellung einer Hyperkante als Teilmenge der Knoten ein Programm erstellt, das eine einfache Produktion von Darstellungen ermöglicht. Auf Grundlage der NURBS-Kurven, die in der Computergrafik für die Modellierung von Kurven weit verbreitete Anwendung finden, wurden periodische NURBS-Kurven und Algorithmen präsentiert, um eine interaktive Erzeugung und Bearbeitung von Hypergraphenbildern zu implementieren.

Dafür wurden der Begriff des Hyperkantenumrisses und dessen Gültigkeit eingeführt sowie ein Algorithmus, der für eine periodische NURBS-Kurve als Umriss die Gültigkeit prüft. Im implementierten Programm werden die Eigenschaften und vorgestellten Algorithmen, vor allem die Invarianz gegenüber affin-linearen Transformationen und die Projektion zur Manipulation der Kurve bzw. Teilen derselben angewandt. In der Interaktion des Benutzers mit dem entstandenen Editor für Graphen und Hypergraphen wurde dabei auf einfache Bedienung ein besonderer Schwerpunkt gelegt.

Der Editor „Gravel“ ist als Open Source Programm veröffentlicht, es sind also sowohl das Programm selbst als auch dessen Quelltext für eigene Anwendungen verwendbar, so die Lizenz eingehalten wird.

Ausblick

Basierend auf dem bereits existierenden Programm lassen sich verschiedene Erweiterungen erarbeiten:

Einerseits ist eine Programmierschnittstelle, kurz API (Application Programming Interface), denkbar, die auf Graphen und Hypergraphen agierende Algorithmen als Erweiterung des Programms möglich macht. Auf Basis von Java ließe sich dies so umsetzen, dass lediglich die Klassen, die den konkreten Algorithmus implementieren, in einem dafür vorgesehenen Verzeichnis liegen müssen, um vom Hauptprogramm erkannt, eingebun-

den und auf Graphen bzw. Hypergraphen angewandt zu werden. Dadurch entstünde die Möglichkeit „Gravel“ nach dem Prinzip der Plugins in seiner Funktionalität zu erweitern

Damit ließe sich der Editor mit Algorithmen anreichern, die beispielsweise Sachverhalte, Sätze und Algorithmen aus der Lehre nicht nur in einem Programm umsetzen, sondern auch interaktiv und erläuternd darstellen. Möglich ist dann etwa ein schrittweise geführter Ablauf, der mit Informationen angereichert ist und Eingriffe des Benutzers in den Ablauf zulässt. Dadurch können dann Lernende durch aktives Ausprobieren die praktische Seite eines formell vorgestellten Algorithmus erfahren. Denkbar wäre dies beispielsweise für den Färbealgorithmus, den Appel u. Haken [1] zu ihrem Beweis des Vierfarbensatzes einbrachten, oder die zum Beweis notwendigen Prozeduren der Entladung eines triangulierten planaren Graphen.

Neben diesem edukativen Algorithmen bietet sich andererseits auch auf dem bisherigen Fokus des Editors die Implementierung von Visualisierungsalgorithmen an. Dazu kann ebenso die API verwendet werden und. Über diese könnten die Algorithmen zur automatisierten Erzeugung und Manipulation von Graphenbildern, wie sie di Battista u. a. [3] zusammenfassend präsentieren, implementiert werden. Damit sind in der Erzeugung von Darstellungen weitere Hilfsmittel verfügbar. Ebenso ist es möglich, eine Betrachtung von Algorithmen zum automatisierten Positionieren der Knoten eines Hypergraphen zu vertiefen. Bisher ist dort nur ein Algorithmus veröffentlicht (siehe Bertault u. Eades [6]).

Auf Basis des Algorithmus zur Validierung ließe sich ein ähnlicher, ebenfalls iterativ arbeitender Algorithmus zur Positionierung der Knoten implementieren. Im Bereich des Hypergraphenzeichnens liegen also sowohl ein großes Potential an weiteren Forschungen als auch interessante offene Fragestellungen.

Danksagung

Besonderer Dank gilt vor allem PD Dr. Hanns-Martin Teichert. Seine hervorragende Betreuung ließ mir genügend Freiheiten in der Festlegung der Schwerpunkte, gab in den kreativen Gesprächen viele Anregungen und schuf insgesamt eine sehr angenehme und motivierende Arbeitsatmosphäre.

Weiterer Dank gilt Julian Bäume für die Ideen und Tipps in der Ausarbeitung des Klassendiagramms, David Gregorczyk und Richard Mietz für die zeitintensive Zusammenarbeit während der Diplomprüfungen sowie Albert Krewinkel, dem ich einige gute Einfälle bei den Algorithmen verdanke. Nicht zu vergessen seien dabei noch all jene, die als Lektoren halfen oder mich zeitweise auf andere Gedanken brachten.

Literaturverzeichnis

[1] Appel, Kenneth ; Haken, Wolfgang: *Contemporary Mathematics*. Bd. 98: *Every Planar Map is Four Colorable*. American Mathematical Society, 1989. – ISBN 0821851039

[2] Bangert, Claudia ; Prautzsch, Hartmut: Circle and Sphere as rational splines. In: *Neural, Parallel & Scientific Computations* 5 (1997), S. 153–162

[3] Battista, Guiseppe di ; Eades, Peter ; Tamassia, Roberto ; Tollis, Ioannis G.: *Graph Drawing: Algorithms for the visualization of graphs*. first Edition. Prentice Hall, Inc., 1999. – ISBN 0–13–301615–3

[4] Berge, Claude: *Hypergraphs: Combinatorics of finite sets*. North-Holland, 1989. – ISBN 0–444–87489–5

[5] Bergmann, Ronny: *Gravel - ein Editor für Graphen*. Universität zu Lübeck, Institut für Mathematik, Oktober 2007. – Studienarbeit

[6] Bertault, François ; Eades, Peter: Drawing Hypergraphs in the Subset Standard (Short Demo Paper). In: *GD '00: Proceedings of the 8th International Symposium on Graph Drawing*. London, UK : Springer-Verlag, 2001. – ISBN 3540415548, S. 164–169

[7] Bloch, Ethan D.: *A First Course in Geometric Topology and Differential Geometry*. Birkhäuser Verlag, Boston, 1997. – ISBN 0–8176–3840–7

[8] Boor, Carl de: *A Practical Guide to Splines*. Springer Verlag, Berlin, 1978. – ISBN 0–540–90356–9

[9] Brandes, Ulrik ; Eiglsperger, Markus ; Herman, Ivan ; Himsolt, Michael ; Marshall, M. S.: GraphML Progress Report, Structural Layer Proposal. In: Mutzel, P. (Hrsg.) ; Junger, M. (Hrsg.) ; Leipert, S. (Hrsg.): *Graph Drawing - 9th International Symposium, GD 2001 Vienna Austria,*. Heidelberg : Springer Verlag, 2001, S. 501–512

[10] Chen, Xiao-Diao ; Yong, Jun-Hai ; Wang, Guozhao ; Paul, Jean-Claude ; Xu, Gang: Computing the minimum distance between a point and a NURBS curve. In: *Computer-Aided Design* 40 (2008), Nr. 10 – 11, S. 1051 – 1054

[11] Cohen, J.E.: *Interval graphs and food webs: a finding and a problem*. 1968. – Rand Corporation Document 17696-PR, Santa Monica, CA

[12] Danos, Vincent ; Laneve, Cosimo: Formal molecular biology. In: *Theoretical Computer Science* 325 (2003), S. 69–110

[13] Diestel, Reinhard: *Graphentheorie*. dritte Auflage. Springer Verlag, 2006. – ISBN 3–540–21391–0

[14] Farin, Gerald: *Kurven und Flächen im Computer Aided Geometric Design*. 2. Auflage. Braunschweig, Wiesbaden : Vieweg, 1994. – ISBN 3–528–16542–1

[15] Gallo, Giorgio ; Longo, Giustino ; Pallottino, Stefano: Directed Hypergraphs and Applications. In: *Discrete Applied Mathematics* 42 (1993), Nr. 2, S. 177–201

[16] Gansner, Emden R. ; North, Stephen C.: An open graph visualization system and its applications to software engineering. In: *Software — Practice and Experience* 30 (2000), Nr. 11, S. 1203–1233

[17] Herczeg, Michael: *Software-Ergonomie. Grundlagen der Mensch-Computer-Kommunikation*. 2., vollst. überarbeitete Auflage. Oldenbourg, 2005. – ISBN 9783486250527

[18] Himsolt, Michael: Graphlet: design and implementation of a graph editor. In: *Software – Practice and Experience* 30 (2000), Nr. 11, S. 1303–1324

[19] Kaufmann, Michael ; Kreveld, Marc J. ; Speckmann, Bettina: Subdivision Drawings of Hypergraphs. In: Tollis, Ioannis G. (Hrsg.) ; Patrignani, Maurizio (Hrsg.): *Graph Drawing* Bd. 5417, Springer, 2008 (Lecture Notes in Computer Science). – ISBN 978–3–642–00218–2, S. 396–407

[20] Mäkinen, Erkki: How to draw a hypergraph. In: *International Journal of Computer Mathematics* 34 (1990), Nr. 3, S. 177–185

[21] Mørken, Knut: Some Identities for Products and Degree Raising of Splines. In: *Constructive Approximation* 7 (1991), S. 195 – 209

[22] Opfer, Gerhard: *Numerische Mathematik für Anfänger: Eine Einführung für Mathematiker, Ingenieure und Informatiker*. 5. Auflage. Vieweg + Teubner, Wiesbaden, 2008. – ISBN 3834804134

[23] Piegl, Les ; Tiller, Wayne: *The NURBS Book*. 2nd Edition. Springer Verlag, Berlin, 1997. – ISBN 3–540–61545–8

[24] Piegl, Leslie ; Tiller, Wayne: A Menagerie of Rational B-Spline Circles. In: *IEEE Comput. Graph. Appl.* 9 (1989), Nr. 5, S. 48 – 56. – ISSN 0272–1716

[25] Rowe, Lawrence A. ; Davis, Michael ; Messinger, Eli ; Meyer, Carl ; Spirakis, Charles ; Tuan, Allan: A Browser for Directed Graphs. Berkeley, CA, USA : University of California at Berkeley, 1986. – Forschungsbericht

[26] Schiermeyer, Ingo ; Sonntag, Martin ; Teichert, Hanns-Martin: Structural properties and hamiltonicity of neighborhood graphs. Schriftenreihe der Institute für Informatik/Mathematik, Serie A, Universität zu Lübeck, 2009. – Forschungsbericht. – Zur Veröffentlichung eingereicht

[27] Sonntag, Martin ; Teichert, Hanns-Martin: Competition hypergraphs. In: *Discrete Appl. Math.* 143 (2004), Nr. 1-3, S. 324–329. – ISSN 0166–218X

[28] yWorks GmbH: *yEd Graph Editor*. `http://www.yworks.com/de/products_yed_about.html`, Abruf: 11. August. 2009

Abbildungsverzeichnis

Algorithmenverzeichnis

Anhang

Anhang A.

Kurzanleitung

A.1. Einleitung

Diese Anleitung stellt die grundlegenden Möglichkeiten des Editors „Gravel“ vor und geht im Verlauf der beiden Tutorien auf übliche Arbeitsabläufe ein.

Die Screenshots wurden auf Mac OS X 10.5 Leopard erstellt. Auf anderen System weicht die Optik ein wenig ab, die Positionierung der Elemente ist jedoch bis auf die Position des Menüs identisch. Während auf den anderen Systemen das Menü oben im Fenster platziert wird, ist es bei Mac OS oben in der Menüzeile und ist daher nicht auf den Screenshots abgebildet. Außerdem unterscheiden sich auf den Systemen die Tastenkombinationen. Bei Mac OS werden diese stets durch die Befehls-Taste (englisch Command-Key, auf älteren Systemen auch die „Apfel“-Taste) und einen Buchstaben aktiviert. Unter Windows und Linux ist stattdessen die Taste Steuerung (Strg bzw. Crtl) üblich. Ansonsten unterscheiden sich die Tastaturbefehle nicht. Der Einheitlichkeit wegen verwenden wir den Begriff „Menütaste“.

In der Maussteuerung besteht ebenfalls ein kleiner Unterschied in der Verwendung: Während unter Windows die rechte Maustaste häufig verwendet wird, ist (zumindest wieder bei älteren Macintosh-Systemen) stattdessen die Kombination von Strg und der linken Maustaste das Pendant. Bei Macintosh-Systemen werden außerdem häufig Eintastenmäuse eingesetzt. Deren eine Taste ist identisch mit der linken Maustaste. Sowohl die rechte Maustaste als auch Strg in Verbindung mit der linken Maustaste haben im Programm „Gravel“ den gleichen Effekt. Der Einfachheit halber wird im Folgenden von der linken bzw. rechten Maustaste gesprochen.

An Systemvorraussetzungen gibt es lediglich, dass Java in einer Version von mindestens 1.5.0[1] installiert sein muss. Für das Laden von Graphen ist einer Internetverbindung notwendig.

Diese Anleitung beginnt mit zwei Tutorien, eines erstellt einen Graphen (Abschnitt A.2), auf diesem Wissen aufbauend erstellt ein zweites einen

[1] Also mindestens von Sun das JDK 5

Hypergraphen (Abschnitt A.3). Danach werden in Abschnitt A.4 die Elemente des Programms kurz erläutert. Abschließend werden in Abschnitt A.5 noch Tastenkürzel und Mausaktionen in tabellarischer Form aufgeführt.

Das Hauptfenster

Zum Starten des Programms genügt auf den meisten Systemen ein Doppelklick auf die Datei „gravel.jar“ (bzw. bei Mac OS auf die Anwendung „Gravel“). Auf einigen wenigen Systemen muss das Programm noch explizit über Java gestartet werden. Dazu öffnet man entweder ein Terminal oder wählt unter Windows Start → Ausführen und gibt dann `java –jar <Pfad>\gravel.jar` ein. Dabei ist der `<Pfad>` dasjenige Verzeichnis, in dem „Gravel“ entpackt bzw. kopiert worden ist.

Beim ersten Start sieht das Programm aus, wie in Abbildung A.1 und gliedert sich in 3 Bereiche:

- Den (noch leeren) Bereich der Darstellung des Graphen
- Die Toolbar am oberen Rand mit der Einstellung des Maßstabs
- Die rechte Seite mit einer Liste aller Elemente des Graphen und einer Statistik

Nach dem Start beginnen wir nun mit dem ersten Tutorial.

A.2. Tutorial I: Der Petersen-Graph

Ziel: In diesem Tutorial wird am Beispiel des Petersen-Graphen demonstriert, wie man einen Graphen erstellt und in seinen Eigenschaften verändert.

Ist das Programm gestartet worden, so ist zu Beginn ein leerer Graph geladen. Ist dies nicht der Fall, etwa weil bereits vorher mit dem Programm gearbeitet worden ist, so wählt man im Menü Datei → Neu und erstellt einen neuen Graphen.

Zum anfänglichen Erzeugen eines Graphen gibt es den sogenannten *One-Click-Modus*. Dieser lässt sich aktivieren unter Ansicht → Modus → One-Click (kurz: Menütaste+5). In diesem Modus erstellt jeder Mausklick einen Knoten an der Position des Mauszeigers, durch Ziehen (Drag) lassen sich Kanten erzeugen. Alle Aktionen lassen sich durch Bearbeiten → Widerrufen (oder kurz: Menütaste+Z) rückgängig machen. Bearbeiten → Wiederholen (kurz: Menütaste+Shift+Z) stellt die letzten zurückgenommenen Aktionen wieder her.

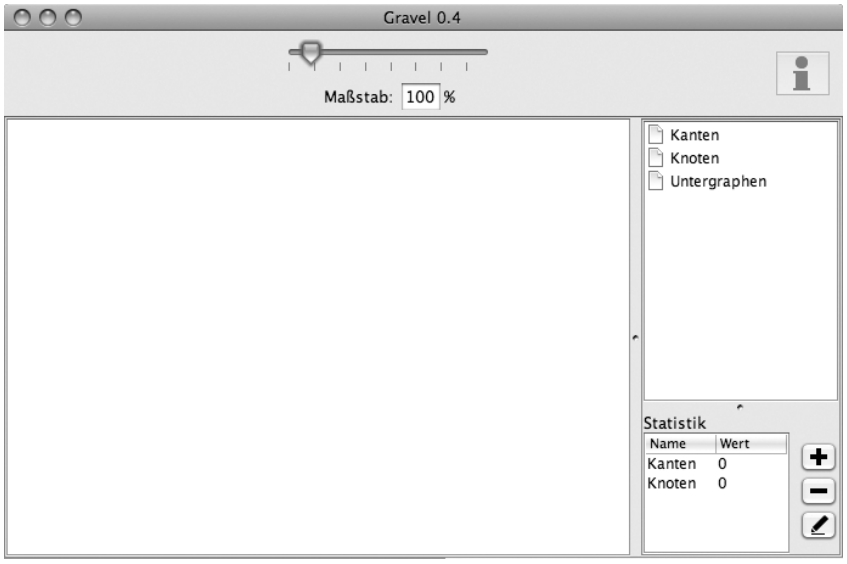
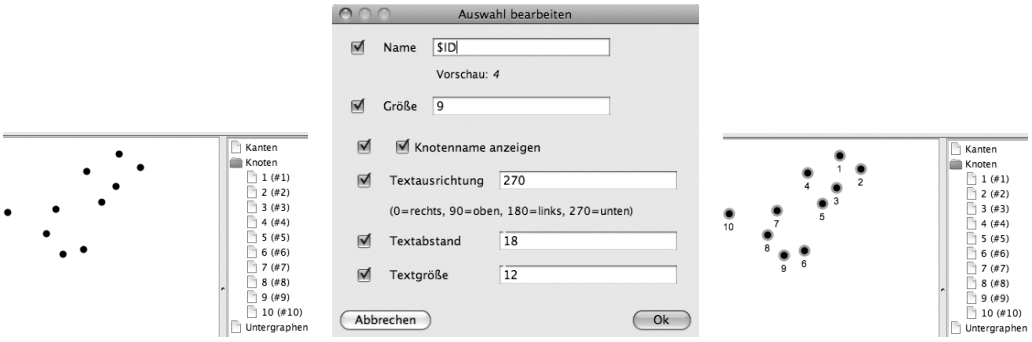


Abbildung A.1: Das Hauptfenster mit den 3 Bereichen: Toolbar (oben), Darstellung (Mitte) und dem Bereich der Liste und Statistik (rechts)

Erstellen und Anordnen der Knoten

Erstellen wir zunächst 10 beliebige Knoten durch Klicken auf den Hauptbereich. Das Ergebnis kann dann etwa so aussehen wie in Abbildung A.2(a). Nun sind deren Namen noch nicht zu sehen[2]. Also markieren wir alle Knoten mit der Maus, entweder durch Aufziehen eines Rechteckes, das alle Knoten enthält oder durch einzelnes Anklicken bei gedrückter Shift-Taste und wählen dann im Menü Bearbeiten → Auswahl bearbeiten (kurz: Menütaste+M). Es öffnet sich ein Fenster, in dem man die Werte eintragen kann, die alle Knoten erhalten sollen. Bereits gemeinsame Werte (hier also alle) sind bereits ausgefüllt. Der vordere Auswahlkasten gibt an, ob dieser Wert bei allen ausgewählten Knoten gesetzt werden soll, dahinter stehen die eigentlichen Werte. Wir setzen also bei „Knotenname anzeigen“ einen Haken im zweiten Auswahlkasten (siehe Abbildung A.2(b)) und erhalten eine Anzeige der Knotenindizes wie in Abbildung A.2(c). Einzelne Knoten lassen sich auch bearbeiten, indem man diese mit der rechten Maustaste anklickt und „Eigenschaften“ auswählt oder sie in der Liste auswählt und auf den Informations-Button („i“) darüber in der Toolbar klickt.

[2]Die Namen werden eventuell angezeigt, wenn man in den Einstellungen die Anzeige für neue Knoten aktiviert hat.



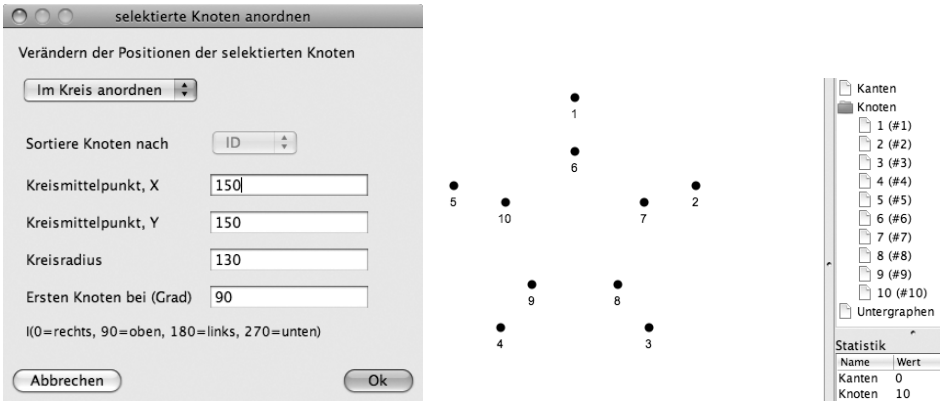
(a) 10 Knoten (b) Auswahl bearbeiten (c) 10 Knoten mit Namen

Abbildung A.2: Erstellen von 10 Knoten und Bearbeiten der Auswahl

Da die Knoten beim Petersen-Graphen in zwei Kreisen angeordnet werden sollen, wählen wir zunächst die Knoten 1 bis 5 aus. Dann wählen wir im Menü Bearbeiten den Eintrag „Auswahl anordnen…“. In dessen Dialog geben wir für die Anordnung im Kreis die Daten für den Mittelpunkt und Radius an, im Beispiel ist dies der Punkt $\begin{pmatrix}150 & 150\end{pmatrix}^T$ mit dem Radius 130, siehe Abbildung A.3(a). Durch die 90 im letzten Datenfeld wird der erste Knoten oben platziert und alle nachfolgenden im Uhrzeigersinn auf dem Kreis gleichmäßig verteilt. Die Anordnung ist stets nach dem Index sortiert. Analog verfahren wir mit den zweiten 5 Knoten und erhalten, so man als kleineren Radius 75 wählt, die Anordnung der Knoten aus Abbildung A.3(b).

Erzeugen der Kanten

Nach der (bei anderen Graphen vielleicht auch nur groben) Positionierung der Knoten werden wir nun die Kanten angeben. Dazu bewegt man die Maus auf einen Knoten und zieht die Kante zum zweiten Knoten hinüber. Alternativ wählt man alle Knoten aus, die zu einem Knoten adjazent werden sollen (mittels Shift nacheinander anklicken), etwa für den Knoten 2 die Knoten 1,7 und 3 und wählt dann nach einem Klick mit der rechten Maustaste auf Knoten 2 „Kanten zu ausgewählten Knoten“ aus. Der Menüeintrag darunter bewirkt für ungerichtete Graphen die gleiche Kantenmenge. Formt man einen ungerichteten Graphen in einen gerichteten Graphen um, ist die Erzeugungsrichtung entscheidend. Neben diesen beiden Möglichkeiten der Erzeugung, die in Abbildung A.4 dargestellt sind, gibt es noch eine weitere Möglichkeit: Wählen wir alle Knoten aus, von denen eine Kante zu einem speziellen Knoten v erzeugt werden soll, so



(a) Dialog zum Anordnen (b) Zwei Knotenkreise

Abbildung A.3: Anordnen im Kreis mit den Parameters aus (a) für den äußeren Kreis erhalten wir mit einem Radius von 75 für den inneren Kreis die Darstellung in (b)

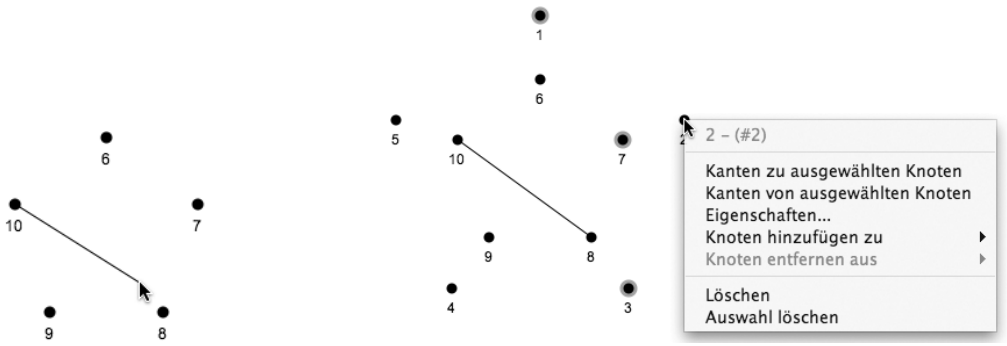
können wir durch Ziehen einer Kante von einem ausgewählten zu v alle gewünschten Kanten erzeugen, indem wir dabei die Shift-Taste gedrückt halten.

Nachdem wir also alle 15 Kanten des Petersen-Graphen erzeugt haben, erhalten wir eine erste Rohfassung, wie sie in Abbildung A.5(a) dargestellt ist.

Feinschliff I: Optimieren für TeX

In dieser Rohfassung sind die Beschriftungen nicht sonderlich gut platziert. Dies können wir auf zwei Arten ändern: Zum einen durch Aufrufen der Eigenschaften, dort finden sich im Tab „Ansicht“ die Werte, welche die Anzeige des Textes beeinflussen: Ausrichtung, Abstand und Textgröße. Wir ändern hier einmal für den Knoten 2 die Ausrichtung auf 0, dann wird dessen Name rechts vom Knoten angezeigt. Diese Methode eignet sich sehr gut, wenn wir explizit eine spezielle Ausrichtung oder einen speziellen Abstand erzwingen möchten.

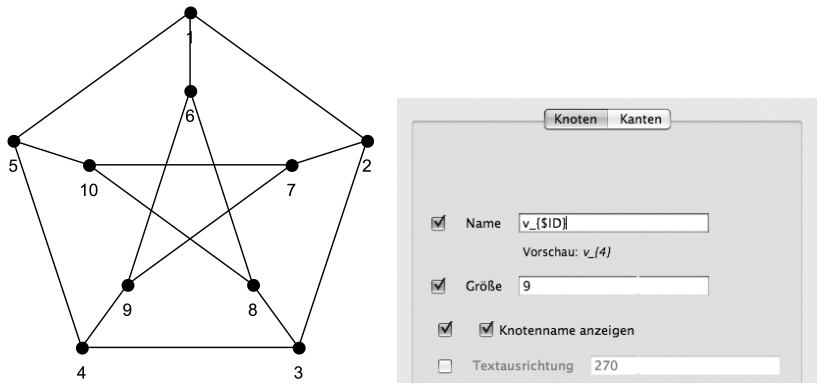
Die zweite Methode basiert wieder auf der Interaktion mit der Maus, genauer dem Ziehen. Dies können wir gut anwenden, wenn eine explizite Angabe der genannten Werte nicht sofort möglich ist und wir lieber herumprobieren möchten, wie beispielsweise beim Knoten 1. Beginnen wir einen Drag auf dem Knoten 1 und halten dabei die Alt-Taste gedrückt, so ändert eine Bewegung der Maus in horizontaler Richtung die Ausrichtung um den Knoten herum. Eine vertikale Mausbewegung ändert den Abstand. Mit ei-



(a) Ziehen einer Kante

(b) Zwei Knotenkreise

Abbildung A.4: Zwei Möglichkeiten zum Erstellen von Kanten: (a) Ziehen oder (b) erstellen mehrerer Kanten über das Menü.



(a) Eine erste Rohfassung (PNG-Export)

(b) Modifizieren von Knoten und Kanten

Abbildung A.5: Zum Feinschliff: (a) Eine erste Rohfassung mit schlechten Beschriftungen und (b) Ändern des Knotennamens. Hier sehen wir, dass die Knotennamen zum Zeitpunkt der Modifikation schon nicht mehr alle unten ausgerichtet sind, das Feld Textausrichtung ist ausgegraut.

nem Klick auf einen Knoten bei gedrückter Alt-Taste lässt sich die Anzeige des entsprechenden Knotennamens umschalten. Diese Mausaktionen bei gedrückter Alt-Taste funktionieren auch analog für die Kanten bzw. Hyperkanten. So können wir nun alle Beschriftungen anordnen.

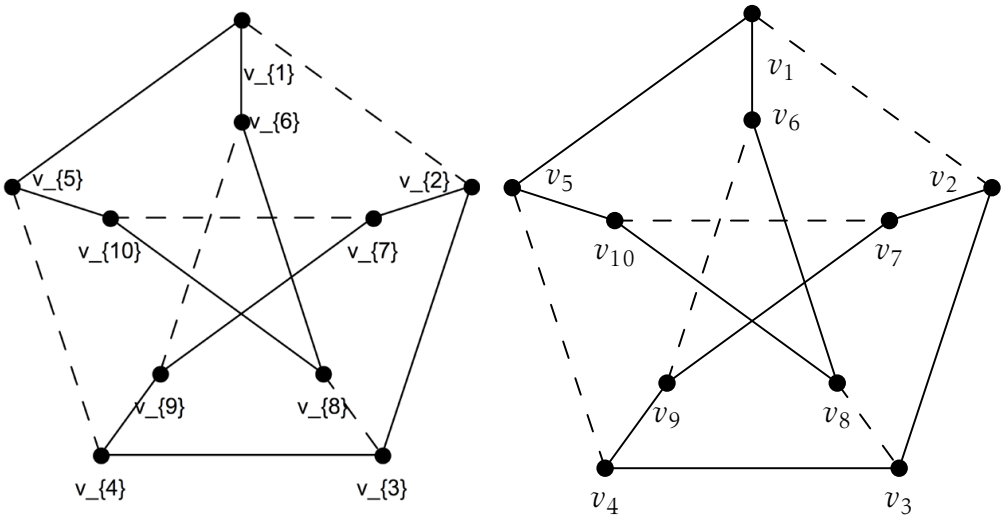
Nun wollen wir außerdem noch einen 1-Faktor hervorheben. Dazu klicken wir bei gedrückter Shift-Taste die entsprechenden Kanten an (hier die Kanten 1-2, 5-4, 7-10,6-9 und 3-8) und wählen wieder den Menüpunkt Bearbeiten → Auswahl bearbeiten. Dort öffnet sich ein ähnlicher Dialog wie vorhin bei den Knoten. Der Dialog enthält 3 Untertabs im Bereich der Kanten; jeweils ein Unterbereich zur Angabe der Werte für Linienart, Beschriftung und (bei gerichteten Graphen) Pfeilspitze. Im Tab „Linienart“ ändern wir den Linienstil auf gestrichelt. Die restlichen Werte entsprechen dem momentan eingestellten Standard, denn jede Kante (auch mit durchgezogenem Strich) wird mit diesen initialisiert.

Eine letzte Änderung betrifft nochmals die Knoten. Da wir diese besser mit v_i, $i = 1,\ldots,10$ bezeichnet wollen, markieren wir wiederum (mindestens) alle Knoten. Starten wir nun „Auswahl bearbeiten“ aus (Menütaste+M), so erhalten wir zwei Tabs (s. Abbildung A.5(b)), wenn Knoten und Kanten ausgewählt sind. Um nun alle Namen auf den gewünschten Namen zu setzen, tragen wir „v_{$ID}“ in das entsprechende Feld ein. Darunter erscheint eine Vorschau (stets für den Index 4), denn der Term $ID wird für jeden Knoten durch seinen Index ersetzt. Die Schreibweise ist TEX-Code und wird in dieser Art auch in der Darstellung erscheinen. In einem Export für TEX steht dann jedoch die jeweilige Nummer des Knotens im Index. Dazu brauchen wir auch die geschweiften Klammern, sonst steht bei Knoten 10 nur die 1 im Index.

Das Ergebnis dieser beiden Veränderungen entspricht in der Darstellung dann der Abbildung A.6(a). Hier sieht man einen kleinen Nachteil des Editors: Er ist nicht in der Lage, TEX-Code zu interpretieren, so unterscheidet sich das Resultat des Exports in Abbildung A.6(b) ein wenig von der Darstellung im Editor. Im verwendeten TikZ-Export wird die Schriftgröße an die des Dokumentes angepasst. Basierend auf den Paketen epic und eepic ist jedoch auch ein Export möglich, der die Textgrößen beibehält. Dieser Export verwendet etwas ältere TEX-Pakete und ist somit auf den latex-Kompiler (bzw. pdflatex im DVI-Modus) angewiesen.

Feinschliff II: Eine bunte TikZ-Grafik

Im letzte Feinschliff betrachten wir die Möglichkeit zur Farbgebung. Diese lassen sich nicht im älteren TEX-Export ausgeben, sondern sind nur für den Export in eine TikZ-Grafik verfügbar.



(a) Das Ergebnis als PNG-Export (b) Das Ergebnis als TikZ-Export

Abbildung A.6: Gegenüberstellung der Exporte in PNG und TEX(TikZ): (a) PNG-Export mit den Indizes wie im Editor (b) TikZ-Export (Vektorgrafik) in dem die Knotennamen als mathematische Formel gesetzt, damit die Form „v_i“ entsteht.

Ausgehend von der eben erstellten Fassung, wählen wir wieder den 1-Faktor aus und setzen die Linienart dieser Kanten zurück auf den normalen durchgezogenen Stil. Dann klicken wir auf den Hintergrund und wählen den Menüpunkt „Neuer Untergraph…“. In diesem Dialog sind nun in der einen Liste die Kanten des 1-Faktors ausgewählt. Ebenso wäre dies mit einer Auswahl an Knoten in deren Liste. Als Farbe wählen wir einen Blauton. Zusätzlich können wir analog einen zweiten 1-Faktor auswählen (etwa den, der nur die Kante 1-2 mit dem Ersten gemeinsam hat) und wählen für diesen Untergraphen die Farbe grün. In der Kante 1-2 werden die Farben gemischt, da diese zu beiden Untergraphen gehört.

Der daraus resultierende Export in eine TikZ-Grafik ist in Abbildung A.7 dargestellt und bildet den Abschluss des ersten Tutorials. In der TikZ-Datei werden zu Beginn die Farben der einzelnen Untergraphen definiert. So ist es möglich, die Farben auch im Nachhinein noch zu verändern. Dazu ist jedoch ein Vorwissen in den Farbdefinitionen nach dem xcolor-Paket notwendig.

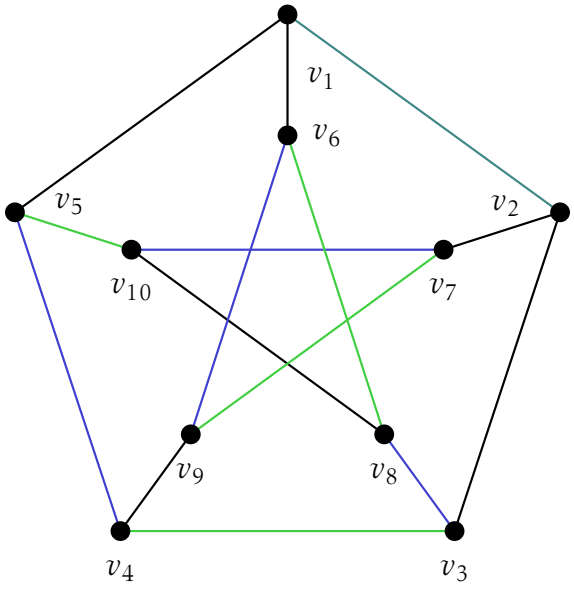


Abbildung A.7: Der Petersen-Graph mit zwei hervorgehobenen 1-Faktoren: Einem in blau und einem in grün.

A.3. Tutorial II: Ein Konkurrenzhypergraph

Ziel: Dieses Tutorial zeigt die Erzeugung von Hypergraphen. Dabei wird zu Beginn eine weitere Methode dargestellt, Knoten zu platzieren. Ausgehend davon werden die verschiedenen Arten erläutert, wie man Hyperkantenumrisse erzeugt.

Erzeugen der Knoten

Wir beginnen wieder im One-Click-Modus und erzeugen 11 beliebige Knoten. Dann öffnen wir die „Raster-Einstellungen “ aus dem Menü „Ansicht“ (kurz: Menütaste+R). Zunächst aktivieren wir den obersten Punkt „Raster aktivieren“. Wir stellen dann auf „synchron“ und bewegen einen Regler auf 50. Zusätzlich setzen wir noch einen Haken vor „Knoten am Raster ausrichten“. Diese Ausrichtung der Knoten ist jedoch lediglich im Standard-Modus aktiv, wir wechseln also in diesen unter Ansicht → Modus → Standard (kurz: Menütaste+4). Durch Ziehen kann nun jeder Knoten bewegt werden. Beim Loslassen der Maustaste wird der jeweilige Knoten am Raster ausgerichtet. Wir positionieren die Knoten also wie in Abbildung A.8.

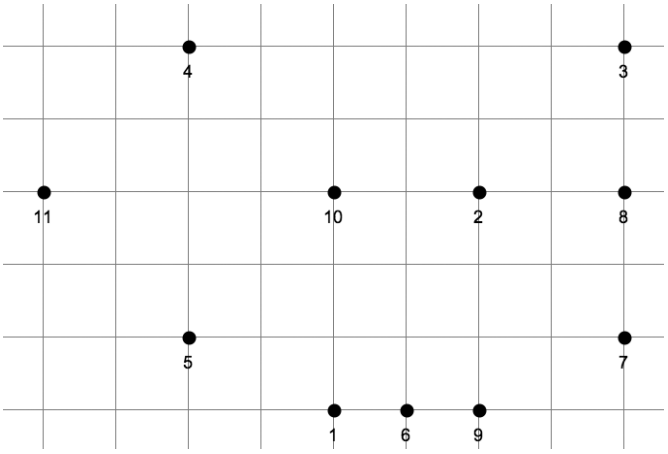


Abbildung A.8: Ausrichtung der Knoten am Raster

Erzeugen der Hyperkanten und ihrer Umrisse

Zum Erzeugen eines Umrisses stellt „Gravel“ drei Möglichkeiten zur Verfügung. Diese schauen wir uns an drei Kanten einmal genauer an.

Ein Kreis. Wir markieren die Knoten 4,5,10 und 11, die links in der Darstellung platziert sind und klicken mit der Rechten Maustaste auf den Hintergrund. In dem Menü wählen wir „Neue Hyperkante...“ und es öffnet sich ein Dialog zum Erstellen derselben. Dort sind die eben markierten Knoten bereits ausgewählt. Um eine Hyperkante mit den Standardwerten zu erzeugen ist also lediglich ein Klick auf „Hyperkante erstellen“ notwendig.

Damit hat der Hypergraph eine erste Hyperkante, die jedoch noch keinen Umriss besitzt. Dazu klicken wir mit der rechten Maustaste in der Liste auf die entstandene Hyperkante „E_{1}“ (im Ordner Hyperkanten) und wählen in dem Kontextmenü „Umriss bearbeiten...“. Dann ändern sich die Darstellung im Hauptfenster sowie die Elemente auf der rechten Seite: In der Anzeige werden alle bisher erstellten Hyperkanten und alle nicht an der aktuellen Hyperkante beteiligten Knoten ausgeblendet. Auf der rechten Seite werden Parameter angezeigt, die im Folgenden beschrieben werden.

Das Bearbeiten eines Umrisses besteht aus zwei Schritten: Zunächst wird eine Grundform erzeugt. Diese wird im zweiten Schritt denn verändert, entweder global oder lokal. Für den Kreis genügt der erste Schritt. Dieser ist zu Beginn als Grundform bereits ausgewählt und lässt sich in der Darstellung einfach „aufziehen“. Seine Werte werden dann rechts eingetragen und können auch dort verändert werden.

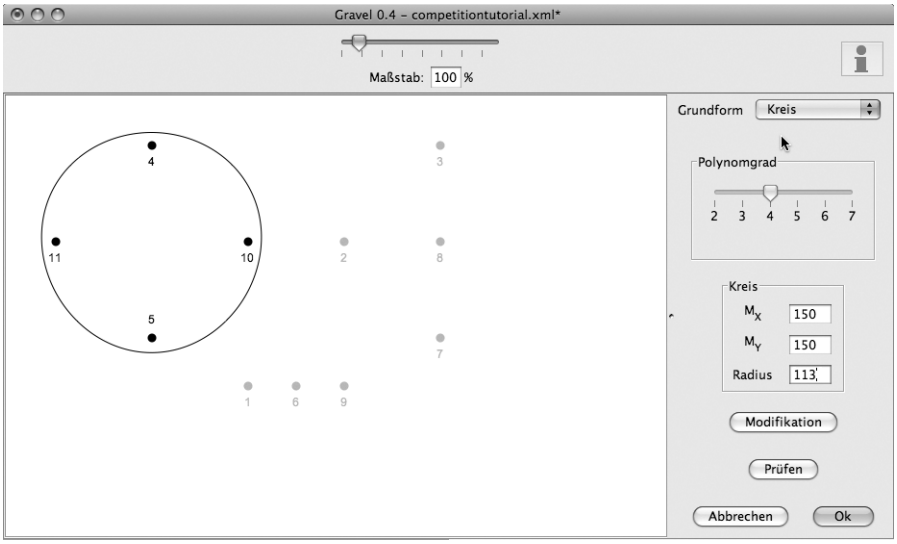


Abbildung A.9: Erzeugen eines Hyperkantenumrisses: Der Kreis. Er ist stets vom Grad 4 und lässt sich in seinen Daten rechts noch verändern

Wir erstellen einen Kreis mit Mittelpunkt $\begin{pmatrix}150 & 150\end{pmatrix}^T$ und einem Radius von 113, wie er in Abbildung A.9 zu sehen ist. Dort sind auch die restlichen Buttons des ersten (erzeugenden) Modus zu sehen: Der „Prüfen"-Button startet die Prüfung der Validität des momentanen Umrisses. „Modifikation" wechselt in den zweiten Modus, dazu später mehr. Da der Kreis als Umriss für uns schon das passende Resultat darstellt, setzen wir diesen als Umriss der Hyperkante E_1, indem wir auf „Ok" klicken.

Interpolation. Für die nächste Hyperkante der Knoten 4,5 und 2 eignet sich die Interpolation sehr gut. Wir erstellen also mit diesen Knoten die Hyperkante „E_{2}" und klicken wieder auf „Umriss bearbeiten…". Allerdings wählen wir nun die Grundform „Interpolation" in der ersten Auswahl und setzen danach den Polynomgrad auf 3. Damit sind mindestens 6 Interpolationspunkte notwendig, um einen Umriss berechnen zu können. Eine gute Anzahl sind etwa 2-3 Interpolationspunkte für jede Krümmung. Ein Klick erzeugt einen neuen Interpolationspunkt, ein Drag (Ziehen) ebenso.

Dabei werden in der Einstellung „Interpolationspunkte am Ende hinzufügen" die Punkte genau in der Reihenfolge von der interpolierenden Kurve durchlaufen, wie sie erzeugt wurden. Bis zum ersten berechneten Umriss ist dies sinnvoll. Danach wird bei einem Drag die Kurve ohne den

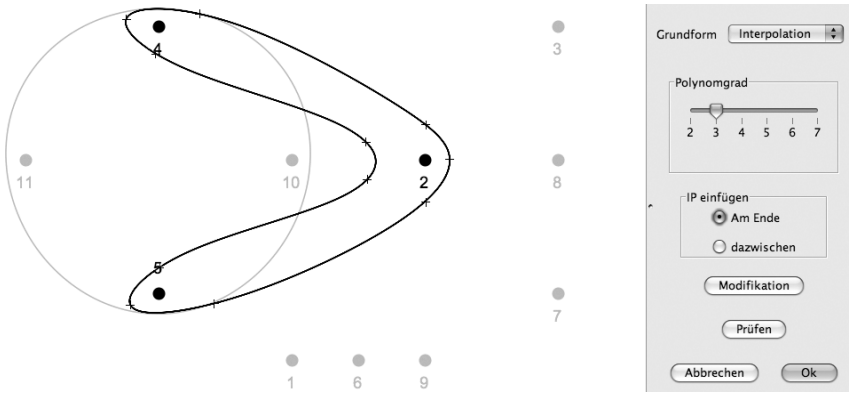


Abbildung A.10: Erzeugen eines Hyperkantenumrisses: Die Interpolation. Die Kurve wird dabei so berechnet, dass sie alle gegebenen blauen Kreuze interpoliert. Diese können interaktiv erstellt, verschoben und gelöscht werden.

neuen Punkt weiterhin schwarz, die neue in Blau gezeichnet. So kann ein Interpolationspunkt gut platziert werden. Ebenso können existierende Interpolationspunkte verschoben werden, wenn man sie mit der Maus hin- oder herzieht.

In der Einstellung, Interpolationspunkte „dazwischen“ einzufügen, wird bei einem Klick die Position als neuer Interpolationspunkt zwischen zwei bestehenden eingefügt, nämlich jenen, bei denen dieser neue Punkt am Nächsten an der Verbindungsstrecke liegt. Ein Klick mit der rechten Maustaste auf einen Interpolationspunkt löscht diesen.

Eine mögliche Kurve aus Interpolationspunkten für die Kante findet sich in Abbildung A.10. In dieser Abbildung sind die Interpolationspunkte durch „+“-Symbole dargestellt, in „Gravel“ sind diese blau. Eine Kurve mit weniger Punkten sieht im Allgemeinen besser aus.

Konvexe Hülle. Für die Kante der Knoten 2,3,7,8 und 10 nutzen wir die dritte Möglichkeit zur Erzeugung: Die konvexe Hülle. Bei ihr muss lediglich ein Polynomgrad vorgegeben werden. Der Umriss wird dann aus den Knoten und dem Innenabstand (bei einer Hyperkante unter „Einstellungen“ im Tab „Ansicht“) berechnet. In diesem Beispiel ist der Innenabstand auf 14 anstelle des Standardwertes von 8 eingestellt. Mit einem Polynomgrad von 3 erhält man zunächst den Umriss wie in Abbildung A.11

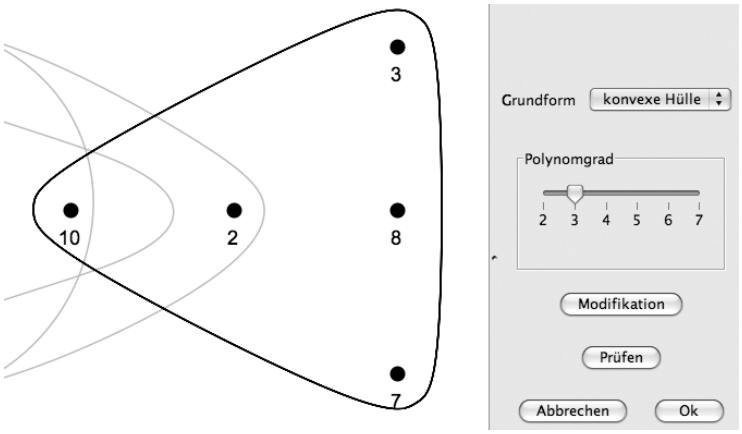


Abbildung A.11: Erzeugen eines Hyperkantenumrisses: Die konvexe Hülle. Die Kurve wird dabei um die konvexe Hülle herum unter Einhaltung des Innenabstandes erzeugt.

Modifikation einer existenten Kurve

Nun gefällt uns diese Form noch nicht ganz, aber schon fast. Wir klicken auf den Button „Modifikation“ und das Programm wechselt in den zweiten Modus, welcher der Modifikation eines Umrisses dient. Hat eine Hyperkante bereits einen Umriss und wählt man – entweder in der Liste oder per rechtem Mausklick auf die Kante – „Umriss bearbeiten…“ aus, so gelangt man ebenfalls in diesen Modus.

Die gesamten Buttons, die nun rechts aufgetaucht sind, werden in Abschnitt A.4 erläutert. Solange keiner von ihnen aktiv (also angeklickt worden) ist, kann die Kurve verändert werden, indem wir sie durch Ziehen (begonnen auf der Kurve) in ihrer Form verändern. Da der Umriss eine NURBS-Kurve ist, bleibt der Einfluss stets lokal. Dieser Einfluss ist bei der konvexen Hülle sehr weitreichend, daher klicken wir einmal auf den „+“-Button. Dann werden auf der Kurve die Stellen angezeigt, an denen die Knoten (der Kurve) liegen. Wir fügen auf den beiden Schrägen je einen Knoten etwa in der Mitte hinzu, indem wir die Kurve dort anklicken. Es entstehen zwei weitere hellblaue Punkte, wie sie in Abbildung A.12 dargestellt sind.

Wir verlassen diesen Modus durch erneutes Anklicken des „+“-Buttons und bewegen die beiden Schrägen durch Anklicken und Ziehen nach innen. Alle weiteren Modifikationen, die man mit dem Umriss vornehmen kann finden sich im Abschnitt A.4. Wir verlassen die Bearbeitung durch Anklicken von „Ok“.

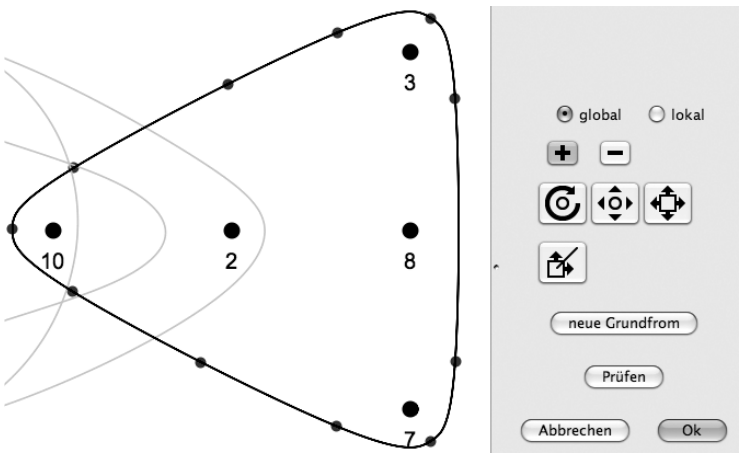
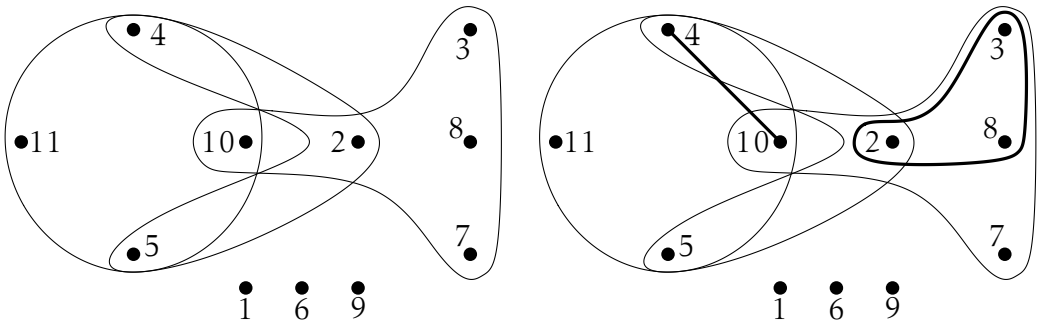


Abbildung A.12: Im „+"-Modus wurden durch Anklicken der Kurve oberhalb und unterhalb des (Graphen-)Knotens 2 zwei weitere Knoten auf der Kurve eingefügt.



(a) Der erste Konkurrenzhypergraph $\mathcal{CH}(D_1)$

(b) Der zweite Konkurrenzhypergraph $\mathcal{CH}(D_2)$

Abbildung A.13: Die zwei Hypergraphen, die in diesem Tutorial entstanden sind. Die ursprünglichen Hypergraphen sind von Sonntag u. Teichert [27].

Nun sind die Umrisse fertig und nach ein wenig Feinschliff an den Knotennamen und deren Ausrichtung erhält man die TikZ-Grafik aus Abbildung A.13(a).

Für eine weitere Hyperkante mit den Knoten 4 und 10 erzeugt die konvexe Hülle die direkte Verbindung, auch wenn das keinem Umriss mehr entspricht. Möchte man stattdessen einen richtigen Umriss haben, nutzt man entweder den Kreis als Ausgangsgrundform oder die Interpolation. Beide Arten der Darstellung werden in der Überprüfung akzeptiert. Mit der Hyperkante E_5 bestehend aus den Knoten 3,2 und 8 entsteht schließlich der zweite Hypergraph Aus Abbildung A.13(b).

A.4. Die grafische Oberfläche

Das Hauptfenster

Das Hauptfenster, wie es in Abbildung A.1 dargestellt ist, gliedert sich in drei Bereiche. Diese werden im Folgenden ausführlich dargestellt.

Hauptbereich: Darstellung des Graphen. Der Hauptbereich stellt den aktuellen Graphen bzw. Hypergraphen dar, wobei eventuell nur ein Teil zu sehen ist, wenn der Graph größer ist als das Fenster. Dann sind Scrollbalken aktiv. In dieser Darstellung gibt es zwei Modi zur Interaktion, den Standard- und den One-Click-Modus. Deren Tastenkombination und Mausinteraktionen finden sich tabellarisch in Abschnitt A.5.

rechte Seite: Liste und Statistik. In der Liste sind alle Elemente des Graphen aufgeführt, Knoten und Kanten bzw. Hyperkanten sowie die Untergraphen. Jeder Bereich ist sortiert nach dem Index der Elemente. Klickt man ein Element mit der linken Maustaste an, so wird es in der Liste und im Graphen ausgewählt. Analog wird bei genau einem ausgewählten Element im Graphen dieses auch in der Liste hervorgehoben.

Mit der rechten Maustaste angeklickt, öffnet sich bei jedem Element ein Menü, über das die Eigenschaften erreichbar sind, dieses Element gelöscht oder (bei Hyperkanten) der Umriss bearbeitet werden kann. Das angeklickte Element wird dabei nicht ausgewählt.

In der Statistik lassen sich verschiedene Werte eintragen. Dazu gibt es eine Übersicht der berechneten Werte in der Hilfe von „Gravel“. Aus diesen lassen sich arithmetische Ausdrücke angeben, die dann in der Statistik für den Graphen den jeweils aktuellen Wert anzeigen. Die drei Buttons zum

Hinzufügen, Entfernen und Bearbeiten von Einträgen befinden sich rechts von der Statistik.

Toolbar. Oberhalb der beiden eben genannten Bereiche befindet sich die Toolbar. Diese enthält mittig den Maßstab, in dem der Graph im Hauptfenster dargestellt wird. Der Maßstab ist frei wählbar im Bereich von 15%-800%. Am rechten Rand der Toolbar befindet sich der Informations-(oder „i"-)Button. Ist im Graphen exakt ein Element ausgewählt, lassen sich über diesen Button die Eigenschaften des Elementes aufrufen.

Eigenschaften-Dialoge. Ruft man für ein Element des Graphen, also Knoten, (Hyper-)Kanten oder Untergraphen, die Eigenschaften auf, so besteht der Dialog aus zwei Tabs: Der erste enthält die mathematischen, der zweite die visuellen Eigenschaften. Dabei gehört die Größe bzw. Breite noch zu den mathematischen Eigenschaften. Eine gerichtete Kante hat zusätzlich ein weiteres Tab mit den Eigenschaften des Pfeils.

Der Dialog „Auswahl bearbeiten…" fasst alle Eigenschaften Dialoge zusammen, es werden jedoch auch einige Eigenschaften weggelassen, da sich diese nicht allgemein setzen lassen (etwa die Indizes). Zusätzlich erhält in diesem Dialog jedes Datenfeld eine weitere Auswahlkasten, der angibt, ob dieser Wert in allen ausgewählten Elementen geändert werden soll. Bei Werten, die zu Beginn des Dialoges in allen ausgewählten Elementen identisch sind, wird dieser Auswahlkasten gesetzt.

Umriss einer Hyperkante

Der Umriss einer Hyperkante wird in zwei Schritten erzeugt: Zunächst wird eine Grundform erstellt, die dann modifiziert werden kann. Wird der Dialog für den Umriss mit einer Hyperkante aufgerufen, die bereits einen Umriss besitzt, so beginnt gleich die Modifikation. Beide Modi haben drei gemeinsame Buttons: „Prüfen" zur Validierung des Umrisses sowie „Abrechen" und „Ok" zum Verwerfen bzw. Übernehmen des aktuellen Umrisses. Zum Übernehmen muss ein nichtleerer Umriss existieren. In der Darstellung werden alle bisherigen Hyperkantenumrisse sowie alle nicht zur aktuellen Hyperkante gehörenden Knoten kaum sichtbar gezeichnet, außer, wenn Knoten nach dem Prüfen im Umriss liegen. Anstelle der Liste auf der rechten Seite werden Parameter eingeblendet.

Erzeugen eines Umrisses Das Erzeugen eines Umrisses ist der erste Schritt zu einem Umriss. Dieser Modus wird auch verwendet, wenn ein Teil des

Umrisses ersetzt werden soll, dann steht allerdings lediglich die Interpolation zur Verfügung und deren Grad ist durch den bisherigen Umriss gegeben. Für die drei Grundformen gibt es jeweils verschiedene Parameter, um die Erzeugung zu beeinflussen:

Kreis Hierbei wird der Polynomgrad auf 4 festgelegt. Dann kann durch Angabe eines Mittelpunktes und eines Radius der Kreis angegeben werden. Alternativ kann er auch mit der Maus aufgezogen werden, die Felder auf der rechten Seite werden dann mit den entsprechenden Werten gefüllt.

Interpolation Neben dem Polynomgrad kann noch spezifiziert werden, wo neu erzeugte Punkte im Vektor der Interpolationspunkte eingefügt werden sollen, am Ende oder zwischen den zwei aufeinanderfolgenden Interpolationspunkten, bei denen der neue Punkt am nächsten an der Verbindungsstrecke dran liegt. Jeder linke Mausklick fügt einen neuen Punkt ein, ein Rechtsklick auf einen Punkt entfernt diesen.

konvexe Hülle Hier ist lediglich der Polynomgrad notwendig, basierend auf den Knotenpositionen, dem Innenabstand und diesem Grad wird dann der Umriss berechnet.

Mit dem Button „Modifikation“ gelangt man in den zweiten Modus zur Modifikation des erstellten Umrisses.

Modifizieren eines Umrisses Da es in diesem Modus viele Buttons gibt, schauen wir uns diese im Folgenden einmal einzeln an:

global und lokal Mit diesem Optionsfeld wechselt man zwischen dem globalen und dem lokalen Bearbeitungsmodus. Im globalen Modus werden die Aktionen auf den gesamten Umriss angewandt. Im lokalen Modus muss zunächst ein Bereich des Umrisses gewählt werden, alle Modifikationen wirken dann nur auf diesem Bereich.

\+ Dieser Button ist nur im globalen Modus verfügbar. Er schaltet die Anzeige des Knotenvektors an; es werden dann die Positionen auf der Kurve angezeigt, an denen die Knoten auf der Kurve liegen. Durch Anklicken der Kurve wird an der Stelle ein Knoten eingefügt.

\- Analog zum „+“-Button wird hier der Modus zum Entfernen von Knoten aktiviert. Kann ein Knoten nicht entfernt werden, passiert nichts,

andernfalls wird er entfernt. Dabei kann sich die Kurve (unter Umständen sogar stark) verändern. Man kann jedoch jede Aktion stets rückgängig machen.

Jede der folgenden Aktionen verläuft durch Ziehen, wobei der Ursprungspunkt p des Ziehens und der Richtungsvektor v vom Ursprung zur aktuellen Mausposition jeweils in der Berechnung verwendet werden:

Rotiert den Umriss (bzw. den lokalen Bereich) um den Punkt p und zwar um den Winkel, der zwischen der positiven X–Achse im Uhrzeigersinn zur Richtung v vorliegt. v wird dabei auch eingezeichnet, um dies zu verdeutlichen.

Skalieren der Kurve (bzw. des lokalen Bereichs) mit dem Fixpunkt p. Die Skalierung ergibt sich aus dem Verhältnis von v zur Größe der ursprünglichen Kurve. Ist also der Mauszeiger nahe der Kurve, ergibt sich etwa ein Faktor von 1. Die Richtung wird dabei ignoriert.

Skalieren mit Richtung. Verläuft analog zur Skalierung (dem vorherigen Icon), nur dass lediglich in Richtung v skaliert wird. Die Richtung orthogonal zu v bleibt unverändert. So können beispielsweise aus Kreisen Ellipsen geformt werden.

Verschieben. Der Umriss (bzw. lokale Bereich) wird um v verschoben.

Ist keiner dieser Modi aktiv, so kann die Kurve verändert werden, indem direkt auf der Kurve mit dem Ziehen begonnen wird, dann wird dieser Punkt an die aktuelle Mausposition verschoben. Aufgrund der lokalen Eigenschaften der NURBS-Kurven hat dies ebenfalls nur lokalen Einfluss.

Im lokalen Modus gibt es drei weitere Icons, die zur Auswahl der Teilkurve dienen:

Setzen des Startpunktes für den lokalen Bereich. Start- und Ende sind nicht direkt unterscheidbar, die Darstellung auf den Icons entspricht folgender Situation:

Ist die Kurve im Uhrzeigersinn definiert – was nur bei der Interpolation anders passieren kann – so verläuft die Auswahl ebenso im Uhrzeigersinn. Dann entspricht die Darstellung auf dem Startpunkt-Icon der Situation, dass der Startpunkt auf einer horizontalen Linie liegt, die von links nach rechts durchlaufen wird (im Parameter u gedacht).

Jeder Mausklick wird dabei auf die Kurve projiziert und dieser Punkt als neuer Startpunkt verwendet.


Invertieren. Die Punkte Start und Ende werden vertauscht, was in der Auswahl genau des Kurvenanteils resultiert, der vorher nicht der lokale Bereich war.


Setzen des Endpunktes. Analog zu den Ausführungen des Startpunktes, nur bezeichnet dies dann das andere Ende des Teilstücks

Ist hier keiner der Modi aktiv, so wird im lokalen Modus bei jedem Klick bzw. Ziehen abwechselnd Start und Ende der Kurve gesetzt. Der Button „neue Grundform“ erstellt eine neue Grundform wie folgt:

im globalen Fall gelangt man zu dem zurück, was bereits vorgestellt worden ist: Der Erzeugung einer kompletten neuen Grundform.

im lokalen Fall wird eine neue Grundform des lokalen Bereichs erzeugt, so dieser existiert und groß genug ist (man benötigt einen gewissen zu ersetzenden Teil). Die Interpolationspunkte definieren dann eine Kurve, die den lokalen Bereich ersetzt.

A.5. Tabellarischer Überblick

Mausgesten

Es gibt drei Modi im Hauptbereich von „Gravel“. Der Modus zur Bearbeitung von Hyperkantenumrissen ist bereits in Abschnitt A.4 behandelt worden, dort gibt es meist nur wenige Mausaktionen, dafür mehrere Modifikationsmöglichkeiten über die Buttons.

Die folgenden Tabellen stellen für den Standard- und den One-Click-Modus eine Übersicht über deren Mausaktionen dar. Alle Aktionen werden, so sie mit gedrückter Shift-Taste begonnen werden und bezogen auf eine ausgewählten (Hyper-)Kante oder einen ausgewählten Knoten, auch auf alle ausgewählten (Hyper-)Kanten bzw. Knoten angewendet.

Die gemeinsamen Aktionen beider Modi sind in Tabelle A.1 dargestellt, diejenigen des Standard-Modus in Tabelle A.2 und die des One-Click-Modus in Tabelle A.3. Bei aktivem Raster mit Ausrichtung an demselben werden nur einzeln bewegte Knoten im Standard-Modus ausgerichtet.

Tastenkombination

Die Tastenkombination dienen der schnellen Ausführung von Menüpunkte. Daher ist in der Tabelle A.4 das komplette Menü aufgeführt. Die Tastenkombination wird unter Mac OS durch die Command- oder Apfeltaste erreicht, auf anderen Systemen, etwa Linux oder Windows, über die Strg-Taste.

Mausgeste	(begonnen) auf	Taste	Kommentar
Klick	Graphenelement		wählt das Element aus
Klick	Graphenelement	Shift	Ändert die Auswahl des Elements
Klick	Knoten	Alt	Anzeige des Namens de- bzw. reaktivieren
Ziehen	Knoten	Alt	Ändern der Namensposition
Klick	(Hyper-)Kante	Alt	Anzeige des Namens de- bzw. reaktivieren
Ziehen	(Hyper-)Kante	Alt	Ändern der Namensposition
Rechtsklick	allem		Kontextmenü
Ziehen	Hintergrund		Auswahl aller Elemente im aufgezogenen Rechteck
Ziehen	Hintergrund	Shift	Erweitert die Auswahl um das Rechteck
Ziehen	Hintergrund	Alt	Verringert die Auswahl um das Rechteck

Tabelle A.1: gemeinsame Mausaktionen beider Modi

Mausgeste	(begonnen) auf	Taste	Kommentar
Ziehen	Kantenkontrollpukt		Bewegen des Punktes
Ziehen	Kante (Gerade)		Umformen von gerader zu gebogener Kante
Ziehen	Knoten		Bewegen des Knotens
Ziehen	sel. Knoten	Shift	Bewegen der selektierten Knoten

Tabelle A.2: zusätzliche Mausaktionen im Standard-Modus

Mausgeste	(begonnen) auf	Taste	Kommentar
Klick	Hintergrund		Erzeugen eines neuen Knotens
Ziehen	Knoten		Erzeugen einer Kante, so die Aktion auf einem Knoten endet
Ziehen	sel. Knoten	Shift	Erzeugen von Kanten aller sel. Knoten zum Zielknoten

Tabelle A.3: zusätzliche Mausaktionen im One-Click-Modus

Menüpunkt	Taste	Kommentar
Gravel		nur u. Mac OS
Über Gravel	n.v.	
Einstellungen...	,	
Beenden	Q	
Datei bzw. **Ablage** (Mac OS)		
Neuer Graph	N	
Öffnen…	O	
Speichern	S	
Speichern unter…	Shift+S	
Einstellungen…	P	nur u. Windows
Export…	E	zu PNG, SVG oder TEX
Beenden	Q	nur u. Windows
Bearbeiten		
Widerrufen	Z	
Wiederholen	Shift+Z	
Auswahl löschen	Entf.	Mac OS: Rücklöschtaste
Auswahl bearbeiten…	M	
Auswahl anordnen…	n.v.	
Ansicht		
Graph umformen (**zu**) (bei Graphen)		
(un)gerichtet	D	
mit/ohne Schleifen	L	
mit/ohne Mehrfachkanten	n.v.	
Modus		
Standard	4	
One-Click	5	
Hyperkantenumriss…	6	bei Hypergraphen
Raster-Einstellungen…	R	
Kontrollpunkte anzeigen	A	
Zoom		
50%	1	
100%	2	
200%	3	
Hilfe		
Index	F1	
Über Gravel	n.v.	nur u. Windows

Tabelle A.4: Menüstruktur von Gravel mit den Tastenkombinationen

Anhang B.

GraphML

Die Definition des Dateiformates für „Gravel“ über XMLSchema bietet, wie in Kapitel 6 erwähnt den Vorteil der Validierung. Hier soll die Definition eines Elementes anhand zweier Beispiele erläutert werden.

B.1. Allgemeine Angabe der Attribute

Um Attribute für die einzelnen Elemente der Datei, also Graph, (Hyper)Kante und Knoten, verwenden zu können, müssen sie zunächst definiert werden. GraphML stellt dazu bereits die einfachen Datentypen `int, boolean, string` bereit. So wird etwa der Name eines Knotens als Attribut definiert, indem innerhalb des Wurzelelements `<graphml>` ein Datenschlüssel spezifiziert wird:

```
1 <graphml>
2     [...] <!-- vorherige key-Spezifikationen -->
3     <key id='nodename' for='node' attr.name='node.name'
4         attr.type='string'>
5         <default>v_{$ID}</default>
6     </key>
7     [...]<!-- nachfolgende keys und der Graph -->
8 </graphml>
```

Dabei gibt id='nodename' eindeutig einen Referenznamen an, der Datenschlüssel ist nur für Knoten (for='node') definiert und ein string. Zusätzlich wird ein Standardwert innerhalb von <default> angegeben.

Der Graph folgt in der Datei nach den Schlüsselangaben (Zeile 7) innerhalb eines `<graph>`-Elementes. Dort werden Knotennamen wie folgt angegeben:

```
1 <node id='1'></node>
2 <node id='2'>
3     <data key='nodename'>u</data>
4 </node>
```

Das Attribut `key='nodename'` verweist auf die ID der Schlüsseldefinition. Dadurch ist die Angabe in Zeile 4 als Name des Knotens 2 festgelegt. Da der erste Knoten keinen expliziten Namen erhält, wird ihm v_{$ID} zugewiesen, wobei innerhalb von „Gravel“ $ID ersetzt wird durch dessen Index. Der Name von Knoten 1 lautet dann v_{1}.

Definition komplexer Datentypen

Für komplexere Datentypen werden in der Erweiterung des GraphML-XMLSchemas für Gravel[1] spezielle komplexe Typen definiert, die als Datenschlüssel verwendet werden dürfen. Für die Darstellung eines Knotens etwa

```
1  <xs:simpleType name='node.form.type.type'>
2      <xs:restriction base='xs:string'>
3          <xs:enumeration value='Circle'/>
4      </xs:restriction>
5  </xs:simpleType>
6  <xs:complexType name='node.form.type'>
7      <xs:attribute name='type' type='node.form.type.type' use='
           required'/>
8      <xs:attribute name='x' type='xs:integer' use='required'/>
9      <xs:attribute name='y' type='xs:integer' use='required'/>
10     <xs:attribute name='size' type='xs:nonNegativeInteger'/>
11 </xs:complexType>
```

Der Typ der Knotenform node.form.type.type (Zeile 1 bis 5) ist bisher lediglich ein Kreis, daher wird auf Basis des Strings (Zeile 2) der Typ als Aufzählung von nur einer Möglichkeit definiert.

Der komplexe Typ der gesamten Knotenform node.form.type wird nun durch 4 Attribute definiert (Zeile 6 bis 11)

- type der eben erwähnte Formtyp, bei „Gravel“ ausschließlich der Kreis. Die Angabe des Typs ist stets Pflicht (use='required')
- x als ganze Zahl für die horizontale Position, ebenso Pflicht
- y als vertikale Position analog zu x
- size als die Größe des Knotens, hier der Durchmesser des Kreises und optional, da „Gravel“ einen Standardwert vorhält

[1] verfügbar unter `http://gravel.darkmoonwolf.de/xmlns/gravelml.xsd`

Zur Verwendung im Dokument muss nun ein weiterer Schlüssel im XML-Dokument angegeben werden, die derjenigen aus des einfachen Datentyps ähnelt:

```
<key id='nodeform' for='node' attr.name='node.form' attr.
   complexType='node.form.type'>
    <default>
        <form type='Circle' x='0' y='0' size='9'/>
    </default>
</key>
```

anstelle des attr.type wird nun mit attr.complexType der eben erwähnte Knotenformtyp angegeben. Die Angabe des Standards legt eine Knotengröße von 9 fest, die Position ist der Knotenform Pflicht, wird aber in „Gravel“ bezüglich des Standards ignoriert.

Im Graphen lassen sich dann Knoten wie folgt positionieren:

```
<node id='2'>
    <data key='nodename'>u</data>
    <data key='nodeform'>
        <form type='Circle' x='274' y='110'/>
    </data>
</node>
<node id='3'>
    <data key='nodeform'>
        <form type='Circle' x='23' y='42' size='12'/>
    </data>
</node>
```

Der Knoten 2 hat keine Größenangabe, er erhält die Größe 9, die im vorherigen Beispiel als Standard angegeben wurde, da keine explizite Größe angegeben worden ist. Der Knoten 3 erhält die Größe 12.

B.2. Ausführliches Beispiel: Der Petersen-Graph

Der im Abschnitt A.2 entstandene innere Stern des Petersen-Graphen ist im folgenden Quelltext wiedergegeben. Der Quelltext ist ein valides GraphML-Dokument mit entsprechenden Datenschlüsseln, lässt sich in „Gravel“ jedoch nicht laden. Dies liegt daran, dass „Gravel“ weitere Datenschlüssel im Dokument voraussetzt.

Der innere Stern des Petersen-Graphen

```
<?xml version='1.0' encoding='UTF-8'?>
<graphml xmlns='http://graphml.graphdrawing.org/xmlns'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xsi:schemaLocation='http://graphml.graphdrawing.org/xmlns
   http://gravel.darkmoonwolf.de/xmlns/gravelml.xsd'>
   <!-- Graphen-Atribute -->
   <key id='graphtype' for='graph' attr.name='graph.type' attr.type='string'/>
   <key id='graphloops' for='graph' attr.name='graph.allowloops'
         attr.type='boolean'>
      <default>true</default>
   </key>
   <key id='graphmultipleedges' for='graph' attr.name='graph.allowmultiple'
         attr.type='boolean'>
      <default>true</default>
   </key>
   <!--visuelle Attribute -->
   <key id='edgewidth' for='edge' attr.name='edge.width' attr.type='int'>
      <default>2</default>
   </key>
   <key id='edgeline' for='edge' attr.name='edge.line'
         attr.complexType='edge.line.type'>
      <default><edgeline length='12' distance='8' type='solid'/></default>
   </key>
   <key id='nodeform' for='node' attr.name='node'
         attr.complexType='node.form.type'>
      <default><form type='Circle' x='0' y='0' size='9'/></default>
   </key>
   <!-- Beginn des Graphens -->
   <graph id='G' edgedefault='undirected'><!-- Ein ungerichteter Graph... -->
      <data key='graphtype'>visual graph</data><!-- mit Darstellung -->
      <data key='graphloops'>true</data><!-- in den Schleifen erlaubt sind -->
      <data key='graphmultipleedges'>true</data><!-- und Mehrfachkanten -->
      <node id='6'>
         <data key='nodeform'><form type='Circle' x='150' y='75'/></data>
      </node>
      <node id='7'>
         <data key='nodeform'><form type='Circle' x='221' y='127'/></data>
      </node>
      <node id='8'>
         <data key='nodeform'><form type='Circle' x='194' y='211'/></data>
      </node>
      <node id='9'>
         <data key='nodeform'><form type='Circle' x='106' y='211'/></data>
      </node>
      <node id='10'>
         <data key='nodeform'><form type='Circle' x='79' y='127'/></data>
      </node>
      <edge id='16' source='9' target='7'>
         <data key='edgeline'><edgeline type='dashed'/></data>
      </edge>
      <edge id='17' source='7' target='10'/>
      <edge id='18' source='10' target='8'/>
      <edge id='19' source='8' target='6'>
         <data key='edgeline'><edgeline type='dashed'/></data>
      </edge>
      <edge id='20' source='6' target='9'/>
   </graph>
</graphml>
```

Anhang C.

Programmstruktur und Bedeutung einzelner Klassen

Als Projekt, das in einer objektorientierten Programmiersprache implementiert ist, folgt „Gravel“ dem Model-View-Control-Modell (kurz: MVC). Dabei werden die drei Programmbereiche Daten (Model), Anzeige (View) und Behandlung von Benutzereingaben (Control) klar voneinander getrennt. Diese Trennung ermöglicht, unabhängig voneinander die einzelnen Komponenten zu entwickeln und zu einem späteren Zeitpunkt einzelne Bereiche auszutauschen, ohne dass der andere Teile betroffen sind. Zur besseren Übersicht sind bei der Programmiersprache Java die einzelnen Klassen in Paketen organisiert.

Der Schwerpunkt liegt bei mathematischen Programmen in den Daten, danach wird in diesem Kapitel auf die beiden anderen Bereiche kurz eingegangen sowie auf weitere Peripherie. In der Implementierung der einzelnen Klassen wurde auf eine Kommentierung des Quelltextes und der Beschreibung der einzelnen Methoden viel Wert gelegt.

C.1. Model

Das Paket model enthält das gesamte Modell sowohl des Graphen als auch seiner visuellen Daten. Die mathematischen und visuellen Daten sind dabei konzeptuell getrennt.

Der mathematische Graph

Die grundlegenden Klassen MNode.java, MSubgraph.java und MEdge.java bzw. MHyperEdge.java enthalten je einen Index, einen Namen sowie mathematische Eigenschaften, etwa beim Untergraphen die Indizes beteiligter Knoten und (Hyper-)Kanten. Die Kante bzw. Hyperkante enthält neben ihren inzidenten Knoten noch ein Gewicht.

Aufbauend auf diesen Klassen werden Mengen dieser Objekte definiert und mit ihrer Funktionalität befinden sich diese in MNodeSet.java, MSubgraphSet.java, MEdgeSet.java bzw. MHyperEdgeSet.java. Diese Mengen haben die Mög-

lichkeit, andere Mengen zu „beobachten“. Dies ist beispielsweise notwendig, wenn ein Knoten in einem Objekt der Klasse MNodeSet.java gelöscht wird, da dann die Kanten- bzw. Hyperkantenmenge alle inzidenten Kanten ebenso löschen muss und die Untergraphen den Löschvorgang ebenso vollziehen müssen.

Durch diese vier Mengen ist dann ein Graph (MGraph.java) bzw. Hypergraph (MHypergraph.java) definiert, der im Kern aus drei der vier Mengen besteht und Funktionen enthält, die den ganzen Graphen betreffen, wie das Erzeugen einer Kopie des Graphen (die clone()-Funktion). Die gemeinsamen Funktionen von Graph und Hypergraph sind in dem Interface – zu deutsch Schnittstelle – MGraphInterface.java aufgelistet. Beide Klassen implementieren dieses Interface. Dadurch kann etwa das Erzeugen einer Kopie veranlasst werden, ohne auf den expliziten Graphentyp einzugehen. Die Kopie wird umfassend verwendet bei der Speicherung der letzten Zustände, auf deren Basis das „Widerrufen“ implementiert wird.

Der visuelle Graph

Ganz analog zu den Klassen des mathematischen Graphen existieren die Klassen für die visuellen Informationen erneut. Der VGraph.java enthält allerdings intern einen mathematischen Graphen. Durch diese Modellierung ist es möglich, die Darstellung eines Graphen zu verwerfen und neu zu beginnen, etwa mit einem Visualisierungsalgorithmus, die jedoch bisher nicht implementiert wurden.

Die visuellen Informationen sind jedoch so umfangreich, dass sie in je einer Klasse unübersichtlich geworden wären. So gibt es gesonderte Klassen für die Linienart (VEdgeLinestyle.java) und die Texteigenschaften (VEdgeText.java). Dies hat außerdem den Vorteil, dass die Klassen sowohl in der Kante (VEdge.java bzw. deren Spezialisierungen) als auch der Hyperkante (VHyperEdge.java) verwendet werden. Für einen neuen Linienstil muss so nur die Klasse VEdgeLinestyle.java angepasst werden, die visuelle (Hyper-)Kante bleibt unverändert.

Der Hyperkantenumriss

Der Umriss einer Hyperkante ist ebenso in einer gesonderten Klasse implementiert: Die Klassen NURBSShape.java enthält die Daten einer (eventuell periodischen) NURBS-Kurve, also den Knoten- und den Kontrollpunktvektor; letzteren mit seinen Gewichten bzw. in homogenen Koordinaten. Auf Basis dieser Daten sind in der Klasse die Auswertung (CurveAt(double u)),

die Ableitungen, sowie Knoten- und Kontrollpunktmanipulationen implementiert.

Zusätzlich gibt es drei weitere Klassen, die einzelne Algorithmen implementieren. Diese Klassen folgen dabei (mit Ausnahme der Factory) dem Dekorationsmuster. Das bedeutet, dass die Klassen jeweils auf Basis der Klasse NURBSShape.java aufbauen und diese um Funktionen erweitern. Bei Bedarf lässt dich diese Dekoration einfach „abstreifen“.

NURBSShapeFactory.java dient der Erzeugung von Kurven. Dabei wird neben einigen Daten, wie etwa den Interpolationspunkten oder dem Kreismittelpunkt und -radius der Typ der zu erzeugenden Kurve mitgegeben. Die „Fabrik“ liefert dann, so die Daten korrekt sind, eine NURBS-Kurve zurück. Diese Erzeugung umfasst auch das Ersetzen eines lokalen Kurvenbereiches durch eine neue Kurve.

NURBSShapeFragment.java erweitert eine Kurve um die Auswahl des lokalen Bereiches und der Extraktion desselben als eigenständige Kurve.

NURBSShapeProjection.java ist die Implementierung der Projektion aus Abschnitt 4.3. Nach der Initialisierung wird die Projektion automatisch berechnet und das Ergebnis zur Verfügung gestellt, entweder der Parameter oder der Punkt auf der Kurve.

NURBSShapeValidator.java beinhaltet die Implementierung des Validierungsalgorithmus aus Abschnitt 5.4. Auch hier kann das Ergebnis unterschiedlich aus der Klasse bezogen werden, etwa nur die Antwort oder zusätzlich noch die Knoten, welche die Validität verletzen.

Das Paket model.messages

Im Modell gibt es ein weiteres Unterpaket an Klassen, die Nachrichten. Diese werden zwischen Objekten der Klassen versandt, beispielsweise den oben erwähnten Mengen an Knoten, Untergraphen bzw. (Hyper-)Kanten. Auch zur Benachrichtigung anderer Klassen werden die Nachrichten verwendet. Dabei enthält eine Nachricht immer die Veränderungen, die gerade am Objekt vorgenommen worden sind, das diese verschickt. Versendet werden die Nachrichten meist vom (Hyper-)Graphen, wenn an ihm Änderungen vorgenommen worden sind.

Empfänger solcher Nachrichten sind alle jene Objekte, die sich über das Observable-Interface dazu eingetragen haben. So ist es beispielsweise mög-

lich, dass auf die Erzeugung eines Knotens sowohl die Anzeige des Graphen, die Anzeige der Liste und Statistik als auch das Menü reagieren (beim ersten Knoten wird so der Menüpunkt „Widerrufen“ aktiviert).

C.2. View und Control

View

Die Anzeige-Elemente befinden sich in zwei Paketen: dialog und view. Im ersteren Paket sind sämtliche Dialoge implementiert: Die Eigenschaften der Elemente eines Graphen, die allgemeinen Einstellungen und die Einstellungen beim Export. Die Dialoge enthalten dabei auch den Anteil an Verarbeitung, der die neuen Daten auf Korrektheit prüft und diese an das Modell weiterreicht bzw. im Fehlerfalle eine aussagekräftige Fehlermeldung ausgibt.

Im Paket View finden sich alle Elemente des Hauptfensters, wie sie in der Kurzanleitung im Anhang A beschrieben wurden. Auch diese Objekte „beobachten“ stets den Graphen und reagieren auf dessen Änderungen. Zusätzlich sind einigen dieser Elemente (etwa dem Hauptbereich aus der Klasse VGraphic.java) bestimmte Kontroll-Objekte zugeordnet, welche die Mausinteraktionen realisieren. Diese Trennung von Darstellung und Eingabebehandlung hat den großen Vorteil, dass auch zur Laufzeit sehr einfach und schnell die Eingabebehandlung gewechselt werden kann. Dies passiert beispielsweise beim Umschalten zwischen Standard- und One-Click-Modus.

Control

Das Paket control enthält alle Eingabebehandlungen, wobei diese gruppiert sind nach Aktionen mit Klick bzw. Ziehen sowie danach, ob sie beim Graphen, Hypergraphen oder bei beiden Anwendung finden können. Eine solche Behandlung der Eingabe wird stets für einen speziellen Graphen initialisiert, auf den die Eingaben angewandt werden.

Das Paket nurbs innerhalb des control-Paketes enthält die gesamten Eingabebehandlungen bei der Bearbeitung von NURBS-Kurven.

C.3. Peripherie

Zusätzlich zu den drei Hauptpaketen gibt es einige Pakete, die ausgelagert wurden. Von diesen sollen hier die zwei Wichtigsten vorgestellt werden.

Das io-Paket

Das Paket io enthält die Ein- und Ausgabe von Graphen vom bzw. in das Dateisystem. Zum Laden dient die Klassen GraphMLReader.java (bei älteren Dateien GravelMLReader.java), um diese zu schreiben GraphMLWriter.java. Außerdem gibt es noch die Klasse GeneralPreferences.java zum Laden und Speichern der allgemeinen Einstellungen. Diese ist die gesamte Zeit, die das Programm läuft, im Hintergrund aktiv und gibt auf Anfrage bestimmte Werte aus den Einstellungen zurück.

Die restlichen Klassen realisieren je einen Export in die Formate PNG, SVG und TEX.

Das history-Paket

Im Paket history befindet sich die Modellierung einer allgemeinen Aktion, wie sie auf einem Graphen bzw. Hypergraphen stattfinden kann. Diese werden dann im CommonGraphHistoryManager behandelt, der entweder für einen Graphen oder einen Hypergraphen dessen Veränderungen aufzeichnet. Für die aufgezeichneten Aktion ist darin auch deren Revidierung bzw. Wiederausführung implementiert.

Anhang D.

Lizenzen

Die entstandene Software „Gravel“ steht unter der GNU Public License Version 3. Der vollständige Text der Lizenz liegt dem Programm bei oder kann unter http://www.gnu.org/licenses/gpl-3.0.html abgerufen werden. Zusätzlich verwendet „Gravel“ einige Bibliotheken bzw. Elemente anderer Projekte, deren Lizenzen im Folgenden genannt sind.

PiCoL

Die verwendeten Piktogramme sind aus der Bibliothek der *Pictorial Communication Language* von *Melih Bilgil*. Die Bibliothek ist verfügbar unter www.picol.org und steht unter der Creative Commons Lizenz BY-SA. Diese Lizenz ist kompatibel zur GPL3 und verfügbar unter http://creativecommons.org/licenses/by-sa/3.0/deed.de

GraphML

Das XML-Format *GraphML* (siehe http://graphml.graphdrawing.org) dient in „Gravel“ der Speicherung von Graphen. Es wurde von der *GraphML Working Group* unter der Creative Commons-Lizenz CC-BY veröffentlicht, siehe dazu http://creativecommons.org/licenses/by/3.0/deed.de.

Apache Xerces

Zum Laden von Graphen verwendet „Gravel“ den Apache Xerces DOM-Parser. Dieser steht under der *Apache License, Version 2.0* verfügbar unter http://www.apache.org/licenses/LICENSE-2.0.

ANTLR

Die Bibliothek ANTLR von Terence Parr (siehe http://www.antlr.org) wird in Gravel verwendet, um arithmetische Ausdrücke zu verarbeiten. Die verwendete Version 2.7.6 steht unter der public domain und kann frei verwendet werden.