Approximation von Bézierkurven durch Polygonzüge

Karl kleine Kruse

14.11.2005

Inhaltsverzeichnis

1 Einleitung

Ich möchte in dieser Arbeit zum Thema „Approximation von Polygonzügen durch Bézierkurven“ eine Möglichkeit vorstellen, wie Polygone bestmöglich durch Bézierkurven angenähert werden können. Dies kann beispielsweise bei der Erstellung von Wetterkarten von Nutzen sein.

Grundlage dieser Karten sind Wetterdaten, die von einem Wetterdienst bereitgestellt wurden. Die Daten sind Werte von Messungen oder Voraussagen an Messpunkten die über ein Gebiet verteilt sind. Aus diesen Daten werden meist zur besseren Darstellung Isolinien berechnet, die aufgrund des Netzes, das die Messpunkte bilden, in Form von Polygonen oder Polygonzügen vorliegen. Diese Isolinien approximieren den wirklichen Verlauf, der durch stetige Kurven beschrieben werden kann. Um der realen Situation näher zu kommen und der Darstellung ein ästetischeres Aussehen zu verleihen, werden die berechneten Isolinien, also die Polygone, in stetige Kurven umgewandelt. Für diese Umwandlung kann das in dieser Arbeit vorgestellte und auf diesen Zweck ausgerichtete Verfahren genutzt werden.

Die berechneten Kurven sollen dazu in einem möglichst verbreitetem graphischen Format gespeichert werden, damit sie in Applikationen ohne größeren Aufwand verwendet werden können. Ein Format, welches das Speichern von Kurven unterstützt, ist durch die Vektorgraphiken gegeben. Mit diesen ist eine Umwandlung in viele Formate möglich. In Vektorgraphiken werden die Bilddaten nicht Pixel für Pixel gespeichert, sondern nur die Daten, die für das Zeichnen des gespeicherten Bildes relevant sind. Im Fall von Polygonen sind es nur die Eckpunkte, im Fall von Kurven die Kontrollpunkte. Bei einem Bild von 100 mal 100 Pixeln müssen somit nicht 10000 Farbwerte gespeichert werden, sondern nur die Daten der Eckpunkte oder Kontrollpunkte, die bedeutend weniger Speicher benötigen. Somit ermöglichen sie eine effektiven Speicherung der Daten. Darüber hinaus haben die Vektorgraphiken den Vorteil des verlustfreien Skalierens, so dass die Graphiken ohne Qualitätsverlust auf jede beliebige Größe gebracht werden können. Als Speicherformat habe

ich Skalierbare Vektorgraphiken (SVG)[1] gewählt. SVG ist ein offenes Format des W3C[2], das auf XML[3] basiert. Da es ein offenes Format ist, können Applikationen die Unterstützung für dieses Format ohne zusätzliche Lizenzkosten einsetzen.

Für die Approximation von Polygonen können Splines, Nurbs oder Bézierkurven verwendet werden, da SVG nur die Speicherung von Bézierkurven unterstützt, wird dieses Format für die Approximation verwendet. Bézierkurven werden auch von anderen Formaten unterstützt und somit ist eine verlustfreie Umwandlung in andere Formate möglich. Daraus ergibt sich für den Nutzer bei Bedarf die Möglichkeit ein anderes Format anstatt SVG zu verwenden. Die von SVG unterstützten Typen von Bézierkurven sind die quadratischen und die kubischen Bézierkurven[4]. In dieser Arbeit finden beide Formen gleichermaßen Berücksichtigung.

Um die Grundlage für die weiteren Überlegungen zu schaffen, stelle ich im ersten Teil meiner Arbeit einige zentrale Begriffe vor. Dazu gehört unter anderem die Erläuterung der Begriffe „Polygonzug“ und „Bézierkurve“. Im zweiten Teil, werde ich hergeleiten, wie eine sinnvolle Approximation von Polygonen und Polygonzügen durch quadratische Bézierkurven erfolgen kann. Dazu werden Formeln und Methoden erläutert, die zu diesen Bézierkurven führen. Im Anschluss daran werden die beschriebenen Methoden und Formeln auf die Approximation durch kubische Bézierkurven übertragen. Bezugnehmend auf diese Ausführungen wird in dem darauf folgendem Kapitel erläutert, welche Auswirkungen die Veränderung der Werte bestimmter Parameter auf das Aussehen der berechneten Bézierkurve hat. Auf diese Weise wird dem Nutzer die Möglichkeit gegeben die Bézierkurve den jeweiligen Ansprüchen anzupassen. Auf dieser Grundlage kann anschließend die Implementation der vorangegangenen Überlegungen in der Programmiersprache Java vorgestellt werden. Dazu werden zunächst alle Klassen und deren Eigenschaften dar-

[1] Vgl. Scalable Vector Graphics (SVG) http://www.w3.org/Graphics/SVG/

[2] Vgl. World Wide Web Consortium (W3C) http://www.w3.org.

[3] Vgl. Extensible Markup Language (XML) http://www.w3.org/XML/

[4] Die Erklärung einer quadratischen bzw. kubischen Bézierkurve erfolgt in Kapitel 2.3.

gestellt und anschließend der Ablauf der Berechnung der Bézierkurven bei Vorgabe eines Polygons erläutert. Den Schluss bildet die Darstellung der Nutzung der Implementation. In diesem Zusammenhang wird auch die möglichen Einbindungen der Implementation in andere Programme Berücksichtigung finden.

2 Grundlagen

In diesem Kapitel sollen die Grundlagen für die weiteren Überlegungen geschaffen werden. Dazu stelle ich im ersten Teil meiner Arbeit einige zentrale Begriffe vor. In diesem Zusammenhang werden die Begriffe „Polygon“, „Polygonzug“, „Bézierkurve“, „Konvex“ und der „Algorithmus von de Casteljau“ erläutert.

Da es in dieser Arbeit um die Approximation von Polygonzügen bzw. Polygonen durch Bézierkurven geht, gehören diese Begriffe zu den zentralen Elementen dieser Arbeit, so dass ihre Erläuterung als Basis für die weiteren Betrachtungen notwendig ist. Des Weiteren wird, für die in dieser Arbeit entwickelte Approximation, die Einteilung in konvexe und nicht konvexe Polygonzüge bzw. Polygone verwendet. Aus diesem Grund werden die Kriterien, nach denen die Einteilung erfolgt, verständlich gemacht. Um den Graphen einer Bézierkurve zu zeichnen, bietet sich der Algorithmus nach de Casteljau an. In den Gleichungen wird dieser Algorithmus verwendet um auf einfache Weise rekursiv die Formel für die Bézierkurve zu erhalten. Der Algorithmus wird hier vorgestellt, um die Aufstellung der Formeln transparent zu machen.

Bei diesen Erläuterungen beschränke ich mich auf den zweidimensionalen Raum, da die graphische Ausgabe und Speicherung ebenfalls in zweidimensionaler Form erfolgt. Der zweidimensionale Raum wird durch die Euklidischen Ebene, somit auf den $\mathbb{R}^2$ in Verbindung mit der Euklidischen Norm $\|x\| := \sqrt{x_1^2 + x_2^2}$, repräsentiert.

2.1 Polygon

Nach der freien online Enzyklopädie Wikipedia wird ein Polygon wie folgt definiert:

> Ein Polygon ist eine geschlossene Figur, die durch ein Tupel $(P_1, P_2, ..., P_n); P_i \in \mathbb{R}^m; 1 \leq i \leq n$ von n Punkten (die Eckpunkte genannt werden) eindeutig definiert wird[5].

[5] Vgl. Wikipedia [4]: „Polygon“

Ein Polygon besteht somit aus endlich vielen Punkten P_1 bis P_n die im $\mathbb{R}^m$ liegen. Die Zählung beginnt in der hier genutzten Form nicht bei 1 sondern bei 0 um die Erläuterung der Implementation zu vereinfachen. Des Weiteren wird die Dimension des Raumes, aus dem die Eckpunkte stammen, eingeschränkt auf die Euklidische Ebene, da die Speicherung im vektorgraphischen Format im Vordergrund steht.

Um Polygone darzustellen werden nicht nur die Punkte gezeichnet, sondern die Strecken zwischen den Punkten und zwar in der Reihenfolge, wie sie im Tupel festgelegt ist. Es werden somit auf Bildern, auf denen Polygone abgebildet sind, die Strecken $\overline{P_i P_{(i+1) \bmod n}}$ $(i = 0, ..., n-1)$ eingezeichnet[6]. Die Sammlung aller Strecken soll der Graph des Polygons genannt werden. Da dieser Graph geschlossen ist, ist ein Polygon entsprechend eine geschlossene Figur. Es ist ersichtlich, dass die Graphen zweier Polygone, bei denen das eine Polygon in das andere umgewandelt werden kann, indem die Punkte im Tupel beliebig zyklisch vertauscht werden oder die Reihenfolge umgekehrt wird, nicht unterscheidbar sind. Da das Approximieren von Polygonen für die graphische Darstellung verwendet werden soll, wird in den folgenden Überlegungen nicht zwischen solchen Polygonen unterschieden. Diese Polygone werden darum in Äquivalenzklassen zusammengefasst, wobei eine Klasse alle Polygone enthält die durch zyklische Vertauschung der Punkte oder Umkehrung der Reihenfolge ineinander überführt werden können. Wenn im Folgenden somit ein Polygon $\overline{P}$ genannt wird, ist die Äquivalenzklasse von P gemeint.

Aus den vorangegangenen Überlegungen ergibt sich folgende Definition eines Polygons die im Weiteren verwendet wird:

Definition 1 (Polygon)
Ein Polygon ist durch ein Tupel $(P_0, P_1, ..., P_{n-1})$ *;* $P_i \in \mathbb{R}^2; 0 \leq i \leq n-1$ *von n Punkten, die Eckpunkte genannt werden, eindeutig definiert.*

[6] **mod** soll als Operator fungieren. Es soll bei der ganzzahligen Division mit Rest den Rest zurückgeben.

Beispiel: $42 \bmod 10 = 2$

2.2 Polygonzug

Ein Polygonzug definiert im Vergleich zum Polygon keinen geschlossenen Linienverlauf[7]. Das bedeutet, dass die Strecke zwischen dem Endpunkt und dem Anfangspunkt entfällt. Aufgrund der engen Verwandtschaft ist die Darstellung von Polygonen und Polygonzügen ähnlich. Bei Polygonzügen werden ebenfalls die Stecken zwischen zwei aufeinander folgenden Punkten gezeichnet, wobei die Strecke zwischen dem letzten und dem ersten Punkt entfällt. Äquivalent zum Polygon soll die Sammlung dieser Strecken Graph des Polygonzuges genannt werden. Die im Folgenden verwendete Definition eines Polygonzuges, die sich an die des Polygons anlehnt, ist folgende:

Definition 2 (Polygonzug)
Ein Polygonzug ist durch ein Tupel
$(P_0, P_1, ..., P_{n-1})$; $P_i \in \mathbb{R}^2$; $0 \leq i \leq n-1$
von n Punkten die Eckpunkte genannt werden eindeutig definiert.

Bei den Polygonzügen werden ebenfalls Äquivalenzklassen gebildet, da zwei Polygonzüge den gleichen Graphen erzeugen, die durch Umkehrung der Reihenfolge der Punkte ineinander überführt werden können. Wenn im Folgenden ein Polygonzug $\overline{Q}$ genannt wird, ist also ebenso die Äquivalenzklasse von Q gemeint.

2.3 Bézierkurve

Bézierkurven sind Graphen spezieller Polynomfunktionen. Sie werden vielfach verwendet, um stetige Kurven in vektorgraphischen Formaten zu speichern. Eine Bézierkurve ist eine Funktion $C : \mathbb{R} \supset [0;1] \longrightarrow \mathbb{R}^m$. Bézierkurven können in allen Potenzen des Zahlenraumes $\mathbb{N}$ auftreten, wobei in Bézierkurven der Potenz n die Bernsteinpolynome vom Grad n der folgenden Form vorkommen.

[7] Vgl. Wikipedia [4]: „Polygonzug“

$$B_{(i,n)}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^{(n-i)}(1-t)^i \text{ mit } i \leq n; i, n \in \mathbb{N}; t \in [0;1]$$

Da die Speicherung in zweidimensionaler Form erfolgt, wird im weiteren Verlauf der Raum in den die Bézierkurve abbildet auf die Euklidische Ebene eingeschränkt. Des Weiteren werden Bézierkurven, deren Funktion als höchste Potenz den Grad zwei hat, quadratische Bézierkurven genannt, sowie die der Potenz drei kubische Bézierkurven.

Der Verlauf einer Bézierkurve vom Grad n in der Euklidischen Ebene ist festgelegt durch ein Tupel von $n+1$ Punkten die Kontrollpunkte genannt werden. Die entsprechende Funktion der Bézierkurve hat folgende Form:

$$C(x) = \sum_{i=0}^{n} P_i \cdot B_{(i,n)}(x) \text{ mit } x \in [0;1] \tag{1}$$

Die Graphen dieser Funktionen sollen im Folgenden als elementare Bézierkurven angesehen werden. In graphischen Applikationen wird der Name Bézierkurve meist für eine aus elementaren Bézierkurven zusammengesetzte Kurve verwendet, in der die elementaren Bézierkurven stetig ineinander übergehen. Diesem Unterschied soll durch die verschiedenen Bezeichnungen Rechnung getragen werden. Wenn im Folgenden Bézierkurven genannt werden, ist eine Sammlung von elementaren Bézierkurven gemeint, deren Graphen als Gesamtbild eine stetige Kurve in der Euklidischen Ebene erzeugen.

Innerhalb einer Bézierkurve lässt sich der stetige Übergang des Graphen einer elementaren Bézierkurve zum nächsten sich durch zwei Bedingungen erreichen. Zum einen müssen Endpunkt der einen elementaren Bézierkurve und Anfangspunkt der anderen identisch sein. Zum anderen müssen die Geraden, welche den Tangenten in den Endpunkten entsprechen, übereinstimmen, wobei die Tangenten in den Endpunkten durch die Kontrollpunkte festgelegt sind.

Für quadratische Bézierkurven sind die Tangenten durch den jeweiligen Endpunkt, also den jeweiligen Kontrollpunkt, und den mittleren Kontroll-

punkt gegeben. Für kubische Bézierkurven sind die Tangenten durch die Endpunkte und den jeweils benachbarten Kontrollpunkt gegeben. Weitere Informationen zu Bézierkurven können unter anderem bei Wikipedia [4] entnommen werden.

2.4 Algorithmus von de Casteljau

Die elementare quadratische Bézierkurve kann nach dem Algorithmus von de Casteljau in zwei Schritten berechnet werden. Zur Verdeutlichung soll das an einem konkreten Beispiel gezeigt werden (siehe Abbildung 1 auf Seite 11). Die elementare Bézierkurve wird definieren durch drei Punkte, Wodurch ebenfalls zwei Strecken definiert werden. Die erste führt vom ersten Punkt zum zweiten und die zweite vom zweiten Punkt zum dritten. Für die Berechnung des Punktes $C(0.25)$ werden die beiden Strecken zwischen den Punkten im Verhältnis 0.25 geteilt und dann die Strecke zwischen den erhaltenen Punkten wieder in dem Verhältnis. Der erhaltene Punkt ist der Wert der Bézierkurve an der Stelle 0.25.

Die elementare kubische Bézierkurve kann nach dem Algorithmus von de Casteljau in drei Schritten berechnet werden. Die drei Strecken zwischen den Kontrollpunkten werden wieder in dem gewünschten Verhältnis geteilt. Aus den zwei Strecken zwischen den erhaltenen drei Punkten ergibt sich dann $C(x)$, indem die drei Punkte wie die Kontrollpunkte einer quadratischen Bézierkurve behandelt werden.

Nach diesem Prinzip werden die Bézierkurven in den Gleichungen aufgestellt. Weitere Informationen zu diesem Algorithmus können unter anderem bei Wikipedia [4] entnommen werden.

2.5 Der Begriff „Konvex“

Eine Art den Begriff „Konvex“ zu definieren ist folgender Definition zu entnehmen:

> Ein Polygon oder Polygonzug ist konvex, wenn die Strecke zwi-

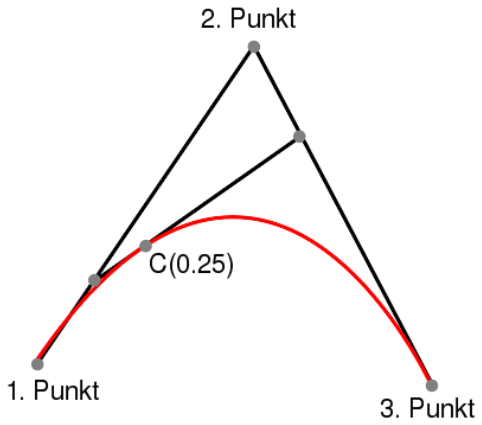


Abbildung 1: Berechnung der Punkte einer Bézierkurve nach de Casteljau.

schen zwei beliebigen Punkten, die innerhalb der vom Polygon oder Polygonzug begrenzten Fläche liegen, immer komplett innerhalb der vom Polygon oder Polygonzug begrenzten Fläche liegt.

In Abbildung 2 auf Seite 13 ist sowohl ein Beispiel für ein konvexes als auch für ein nicht konvexes Polygon gegeben. In Abbildung 3 auf Seite 13 sind Polygone gezeigt, die nach der gegebenen Definition als nicht Konvex angesehen werden. Da diese Polygone in dieser Arbeit ebenfalls als konvex angesehen werden sollen, da die vorgestellte Approximation diese nach dem gleichen Verfahren behandelt, benötigten diese eine andere Definition des Begriffes „Konvex“. Deswegen ist die in den folgenden Kapiteln angewendete Definition folgende.

Ein Polygon oder Polygonzug ist konvex, wenn der Graph des Polygons bzw. Polygonzuges durchgehend eine Linkskurve oder eine Rechtskurve beschreibt.

Mathematisch kann das durch folgende Formeln ausgedrückt werden:

Ein Polygon ist konvex, wenn eine von den folgenden Aussagen für alle i mit $0 \leq i \leq n-1$ zutrifft:

$$(P_i - P_{(i+1) \bmod n}) \otimes (P_{(i+2) \bmod n} - P_{(i+1) \bmod n}) \geq 0$$

oder

$$(P_i - P_{(i+1) \bmod n}) \otimes (P_{(i+2) \bmod n} - P_{(i+1) \bmod n}) \leq 0$$

Ein Polygonzug ist konvex, wenn eine von den folgenden Aussagen für alle i mit $0 \leq i \leq n-3$ zutrifft:

$$(P_i - P_{(i+1)}) \otimes (P_{(i+2)} - P_{(i+1)}) \geq 0$$

oder

$$(P_i - P_{(i+1)}) \otimes (P_{(i+2)} - P_{(i+1)}) \leq 0$$

Der Operator $\otimes$ hat dabei folgende Definition:

$$\begin{pmatrix} v_1^1 \\ v_2^1 \end{pmatrix} \otimes \begin{pmatrix} v_1^2 \\ v_2^2 \end{pmatrix} := (v_1^1 \cdot v_2^2) - (v_2^1 \cdot v_1^2)$$

Dieser Operator ergibt sich aus den nachfolgenden Überlegungen:

Jedem Vektor aus der Euklidischen Ebene kann ein Vektor im $\mathbb{R}^3$ zugeordnet werden, der durch ergänzen des Vektors um einen Eintrag mit dem Wert Null entsteht. Also

$$A : \mathbb{R}^2 \to \mathbb{R}^3$$
$$A\left(\begin{pmatrix} v_1^1 \\ v_2^1 \end{pmatrix}\right) = \begin{pmatrix} v_1^1 \\ v_2^1 \\ 0 \end{pmatrix}$$

Die Euklidische Ebene ist dann im $\mathbb{R}^3$ eingebettet. Das Spatprodukt dreier Vektoren v^1, v^2 und v^3 aus dem $\mathbb{R}^3$ ist definiert als

$$spat\left(v^1, v^2, v^3\right) = \left(\begin{pmatrix} v_1^1 \\ v_2^1 \\ v_3^1 \end{pmatrix} \times \begin{pmatrix} v_1^2 \\ v_2^2 \\ v_3^2 \end{pmatrix}\right) \bullet \begin{pmatrix} v_1^3 \\ v_2^3 \\ v_3^3 \end{pmatrix}$$

Das Vorzeichen dieses Spatproduktes gibt an ob es sich bei dem System von Vektoren um ein Links- oder Rechtssystem handelt. Dadurch kann mit

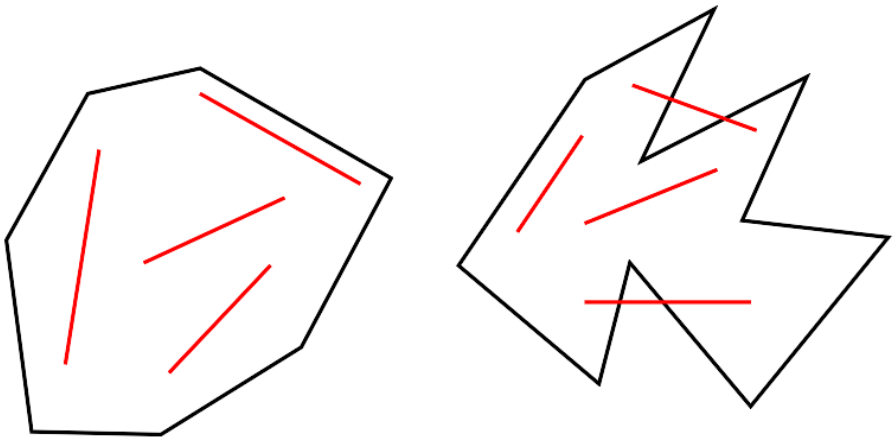


Abbildung 2: Ein konvexes Polygon und ein nicht konvexes Polygon.

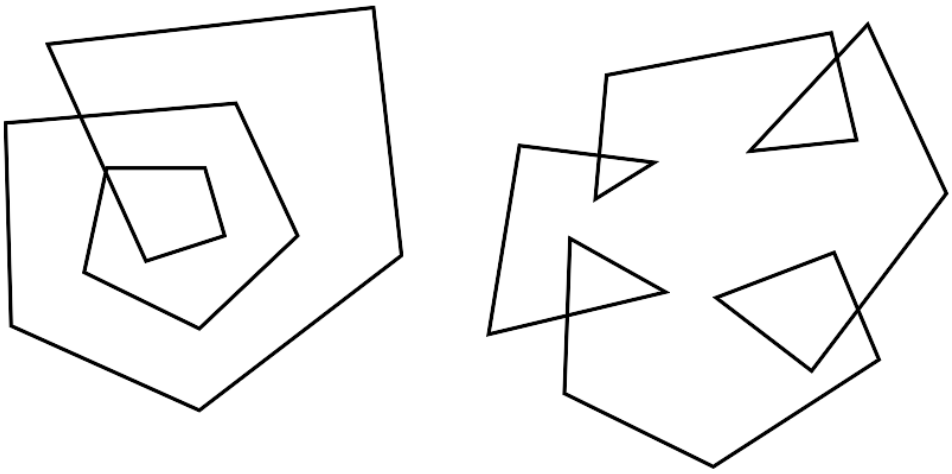


Abbildung 3: Beispiele für spezielle Polygone.

Hilfe dieser eingebetteten Euklidischen Ebene folgende Funktion aufgestellt werden.

$$
\begin{aligned}
f\left(v^1, v^2\right) &= spat\left(A\left(v^1\right), A\left(v^2\right), \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\right) \\
&= \left(\begin{pmatrix} v_1^1 \\ v_2^1 \\ 0 \end{pmatrix} \times \begin{pmatrix} v_1^2 \\ v_2^2 \\ 0 \end{pmatrix}\right) \bullet \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
&= \left(v_1^1 \cdot v_2^2\right) - \left(v_2^1 \cdot v_1^2\right) \\
&= \begin{pmatrix} v_1^1 \\ v_2^1 \end{pmatrix} \otimes \begin{pmatrix} v_1^2 \\ v_2^2 \end{pmatrix}
\end{aligned}
$$

In der Funktion f wurde das Spatprodukt der eingebetteten Vektoren und dem auf der eingebetteten Euklidischen Ebene senkrechten Einheitsvektor gebildet. Wenn die Reihenfolge der eingebetteten Vektoren in der Funktion vertauscht werden, ist das äquivalent zu einem Richtungswechsel. Das System der Vektoren wechselt dann ebenfalls die Richtung, wie das Vorzeichen des Spatproduktes. Somit kann die genannte Formel für die Bestimmung der Richtungswechsel genutzt werden.

Folgende Abbildungen sollen das Verfahren zum Erkennen eines Richtungswechsels verdeutlichen.

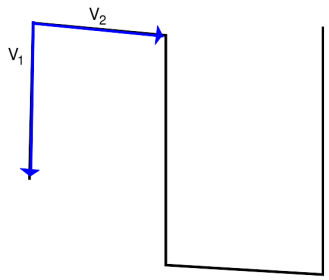


Abbildung 4: Der kleinere Winkel wird von v_1 zu v_2 gegen den Uhrzeigersinn eingeschlossen.

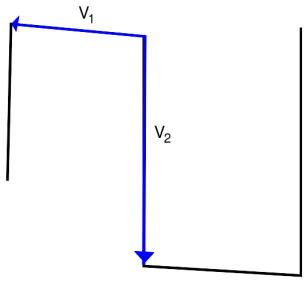


Abbildung 5: Der kleinere Winkel wird von v_1 zu v_2 ebenfalls gegen den Uhrzeigersinn eingeschlossen.

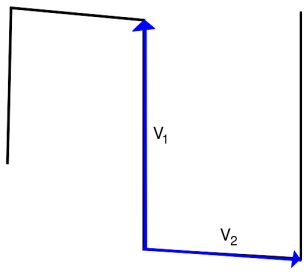


Abbildung 6: Der kleinere Winkel wird von v_1 zu v_2 mit dem Uhrzeigersinn eingeschlossen, da ein Richtungswechsel vorliegt.

3 Approximation durch quadratische Bézierkurven

In diesem Kapitel wird erläutert, wie beliebige Polygone und Polygonzüge durch quadratische Bézierkurven approximiert werden können. In diesem Zusammenhang werden zunächst Polygone und anschließend Polygonzüge betrachtet. In beiden Fällen wird eine Unterteilung in konvexe und nicht konvexe Polygone bzw. Polygonzüge vorgenommen.

Anhand eines konvexen Polygons werden die zur Approximation nötigen Formeln und Bedingungen hergeleitet. Dabei werden zunächst Kriterien entwickelt, welche als Maß für den Grad der Abweichung vom Polygon verwendet werden. Anschließend wird ein Verfahren zum Berechnen der Bézierkurve vorgestellt, welche das Polygon nach den hergeleiteten Bedingungen bestmöglich approximiert. Diese Erkenntnisse werden danach auf konvexe Polygonzüge übertragen. Dazu werden zusätzliche Bedingungen aufgestellt, um dem Umstand Rechnung zu tragen, dass die Polygonzüge im Vergleich zu den Polygonen keine geschlossenen Figuren sind. Die daraus gewonnenen Ergebnisse werden dann auf nicht konvexe Polygone und Polygonzüge übertragen.

3.1 Konvexe Polygone

Damit eine Bézierkurve ein konvexes Polygon approximiert, muss es bestimmte Kriterien erfüllen. Anhand dieser Kriterien werden anschließend Gleichungen erzeugt, welche die mathematische Repräsentation der Kriterien darstellen. Auf diese Weise werden die Kriterien bei der Berechnung der bestmöglichen Bézierkurve berücksichtigt. Da die Kriterien keine eindeutige Eingrenzung vornehmen, beschreiben die Gleichungen eine bestimmte Menge von in Frage kommenden Bézierkurven. In dem hier entwickelten Verfahren wird aus dieser Menge nun die Bézierkurve gewählt, welche das konvexe Polygon bestmöglich approximiert. Dazu wird ein weiteres Kriterium verwendet. Dieses Kriterium kann nicht in einer einfachen Gleichung ausgedrückt

werden, so dass es nicht in die obigen Gleichungen mit einfließen kann. Im Anschluss wird bewiesen, dass das Verfahren immer zu der bestmöglichen Bézierkurve führt.

3.1.1 Aufstellen von Kriterien

Für die folgenden Überlegungen, wie eine Bézierkurve verlaufen sollte, die ein konvexes Polygon approximiert, sei in Abbildung 7 ein Beispiel für ein solches Polygon gegeben, das im Folgenden mit $\overline{P}$ benannt wird.

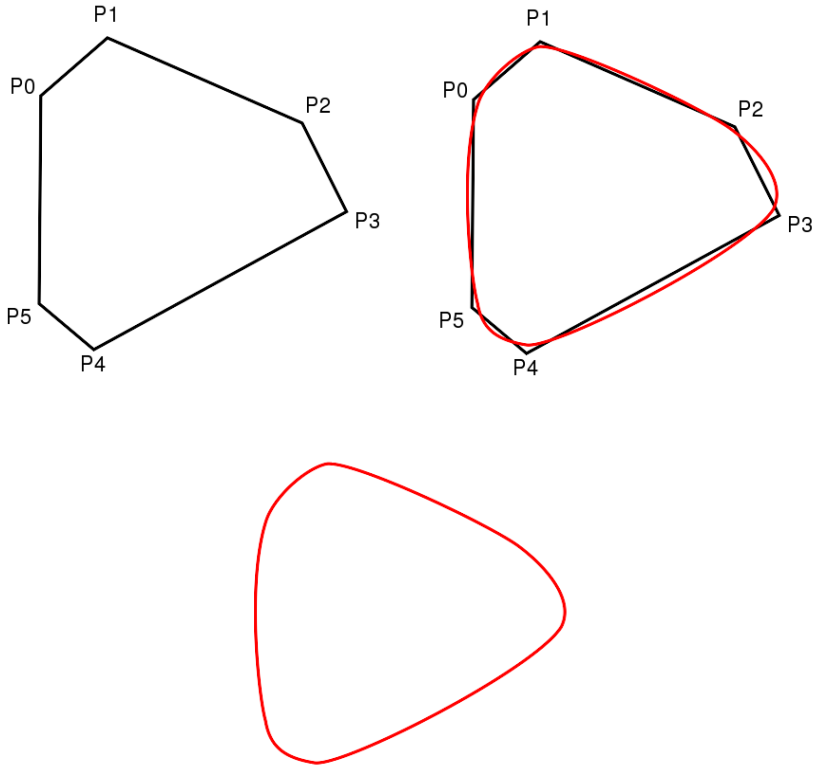


Abbildung 7: Polygon $\overline{P}$ mit 6 Punkten, $\overline{P}$ mit einer Bézierkurve B, sowie nur die Bézierkurve B.

$\overline{P}$ besitzt eine endliche Anzahl von Punkten, hier sechs, welche das Polygon eindeutig festlegen. Die Punkte sind im Uhrzeigersinn von $P0$ bis $P5$

nummeriert. Polygone werden im Folgenden immer im Uhrzeigersinn nummeriert, so dass der Graph des Polygons immer eine Rechtskurve beschreibt.

Eine Kurve die $\overline{P}$ approximiert, sei durch eine quadratische Bézierkurve B gegeben. B besitze eine endliche Anzahl von Kontrollpunkten, durch die es eindeutig definiert ist.

Da B das Polygon $\overline{P}$ approximieren soll, sollte B eine Fläche begrenzen dessen Flächeninhalt mit dem der Fläche übereinstimmt, die von $\overline{P}$ begrenzt wird. Die Qualität der Approximation kann davon abhängig gemacht werden, wieviel von der Fläche, die das Polygon einschließt von der Bézierkurve eingeschlossen wird und ob beide eingeschlossenen Flächen den gleichen Flächeninhalt haben. Wenn eine Bézierkurve exakt die selbe Fläche einschließt, hat sie automatisch den gleichen Flächeninhalt und der Graph der Bézierkurve wäre identisch mit dem Graphen des Polygons, aber somit auch nicht mehr stetig. Eine Vergrößerung der Differenz der Flächen und der Flächeninhalte führt dann zu einer schlechteren Approximation. Der Flächeninhalt und die Fläche können somit als Kriterien angesehen werden. Dabei ist allerdings zu beachten, dass die Berücksichtigung dieser Kriterien nicht zu einer Übereinstimmung der Bézierkurve mit dem Polygon führen darf.

Eine weiteres Kriterium für den Grad der Abweichung zwischen Bézierkurve und Polygon ist die Summe der Abstandsquadrate. Wenn die Summe der Abstandsquadrate den Wert Null hätte, für beliebige Orte der Abstandsmessung, wäre der Graph der Bézierkurve ebenfalls mit dem des Polygons oder Polygonzuges identisch. Da dies wiederum nicht erstrebenswert ist, muss mit diesem Kriterium wie mit den obigen Kriterien verfahren werden.

Da der wesentliche Unterschied zwischen $\overline{P}$ und B darin besteht, dass B keine Ecken besitzt, muss eine Umwandlung von $\overline{P}$ zu B die Abrundung der Ecken von $\overline{P}$ beinhalten. Wenn B den gleichen Flächeninhalt wie $\overline{P}$ haben soll, kann das nur durch einen Flächenverlust an den Ecken geschehen, der zwischen den Ecken kompensiert wird. Eine Flächenzunahme an den Ecken müsste durch eine Flächenabnahme zwischen den Ecken kompensiert werden und würde die Summe der Abstände im Allgemeinen erhöhen. Dazu ist in

Abbildung 8 ein Beispiel gegeben.

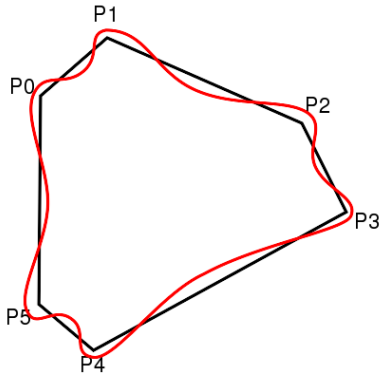


Abbildung 8: Polygon $\overline{P}$ mit einer Bézierkurve B.

Ein Polygon kann als Zusammensetzung von Kanten angesehen werden. $\overline{P}$ hat dann sechs Kanten, welche die Eckpunkte verbinden. Da ein Polygon beliebig viele Kanten aufweisen kann, ist es sinnvoll, wenn B so aus elementaren Bézierkurven zusammengesetzt ist, dass es für jede elementare Bézierkurve C_i von B eine Kante K_i von $\overline{P}$ gibt, die von C_i approximiert wird. So können Bedingungen für jedes C_i aufgestellt werden aus denen dann B hervorgeht.

Für jede Kante von $\overline{P}$ muss dann eine elementare quadratische Bézierkurve aufgestellt werden, so dass diese stetig ineinander übergehen, um eine nach den genannten Kriterien bestmögliche Bézierkurve zu erhalten.

Alternativ können Gruppen von Kanten durch eine elementare Bézierkurve approximiert werden. So würde die Komplexität der Bézierkurve verringert werden. Darauf soll in Kapitel 5.3 genauer eingegangen werden. Die einzige Einschränkung, die hier an das Polygon gestellt wird, ist dass keine drei benachbarten Punkte auf einer Geraden liegen und somit für den Graphen des Polygons der mittlere Punkt überflüssig ist.

Die Bézierkurve, die eine Kante eines Polygons approximieren soll, muss auch Kriterien erfüllen, die den Verlauf der Bézierkurve beschreiben. Diese Kriterien sollen nachfolgend vorgestellt werden.

Eine quadratische elementare Bézierkurve ist durch drei Kontrollpunkte definiert: Einen Anfangs- und einen Endpunkt, sowie einen Kontrollpunkt, der die Krümmung angibt und der durch den Schnittpunkt der Tangenten an den Endpunkten festgelegt ist. Eine Umwandlung einer Kante in eine entsprechende Bézierkurve durch Flächenabnahme an den Ecken könnte durch eine Verschiebung der Eckpunkte des Polygons in das Polygon beschrieben werden. Die Endpunkte der Kante sind dann, vor der Verschiebung ins Innere, den Endpunkten der elementaren Bézierkurve gleich zu setzen. Die Punkte müssen auf einem vorher beschriebenen Weg, abhängig von den Nachbarkanten, im einfachsten Fall einer Geraden, verschoben werden. Die Flächenzunahme zwischen den Eckpunkten des Polygons entspricht einem Verschieben des mittleren Kontrollpunktes vom Polygon weg. Je weiter die Eckpunkte des Polygons ins Inneren verschoben werden, desto mehr müssen die mittleren Kontrollpunkte nach Außen wandern, um den Flächenverlust auszugleichen. Eine mögliche Entwicklung vom Polygon zur Bézierkurve ist in der Abbildung 9 gezeigt.

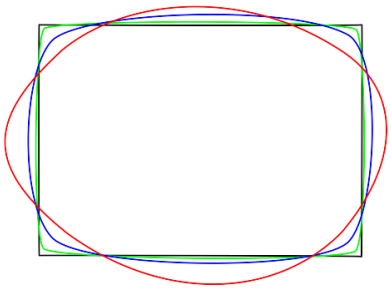


Abbildung 9: Eine möglich Entwicklung von einem Polygon zu einer Bézierkurve, welche das Polygon approximieren würde.

Die Verschiebung der Eckpunkte erfolgt auf einer Geraden, da sie die geringste Komplexität besitzt. Damit die Verschiebung für beliebige Winkel zwischen den Nachbarkanten ins Innere des Polygons führt, soll die Win-

kelhalbierende zwischen zwei Kanten als Richtungsvektor der Verschiebung des entsprechenden Eckpunktes gewählt werden. Die Winkelhalbierende lässt sich nicht ohne größeren Aufwand, wie zum Beispiel das Wurzelziehen in den reellen Zahlen, berechnen. Sie ist aber eine Gerade, die für jeden Winkel, den die Kanten einschließen ins Innere des Polygons führt und keine der Kanten bei der Verschiebung bevorzugt.

Es ist ein Tangentenvektor festzulegen, damit ein stetiger Übergang von einer Teilkurve C_i zur nächsten Teilkurve erfolgt, der für Kurvenstücke die ineinander übergehen einen stetigen Übergang erzeugt. In den verschobenen Eckpunkten, die den Übergang von einer elementaren Bézierkurve zur nächsten beschreiben, muss für einen stetigen Übergang der Tangentenvektor für beide elementaren Bézierkurven in dem Endpunkt gleich sein. Der Tangentenvektor ist weiterhin abhängig von den Kanten am jeweiligen Eckpunkt des Polygons, da sich die Kurve in dem jeweiligen Punkt beiden bestmöglichst anpassen sollte. Es gibt verschiedenen Möglichkeiten die Tangente in den Endpunkten zu definieren. Eine Möglichkeit ist den zur Winkelhalbierenden senkrechten Vektor als Tangentenvektor zu wählen (siehe Abbildung 10 auf Seite 22). In diesem Fall ist der Winkel, der mit jeder der Kanten eingeschlossen wird, gleich groß. Eine bessere Approximation der Kanten ist aber mit einem anderen Vektor möglich. Wenn der Tangentenvektor von den, dem Eckpunkt an dem die Bézierkurven zusammentreffen, benachbarten Eckpunkten definiert wird (siehe Abbildung 11 auf Seite 22), ergibt sich eine geringere Abweichung. Dies soll im Folgenden erläutert werden.

Ist eine Kante im Vergleich zu der Nachbarkante sehr viel länger, ist es von Vorteil, wenn der Tangentenvektor in den Ecken an den Bézierkurven diesem Umstand Rechnung trägt. Die Bézierkurve muss an einer langen Kante deutlich flacher verlaufen, um die Summe der Abstände niedrig zu halten. Ein zu steiler Winkel an den Eckpunkten würde zu einem großen Abstand zwischen Polygon und Bézierkurve führen. Wenn der Tangentenvektor nun durch die benachbarten Eckpunkte definiert wird (siehe Abbildung 12 auf Seite 23), führt eine lange Kante zu einer flachen Kurve, da der Tangentenvektor einen

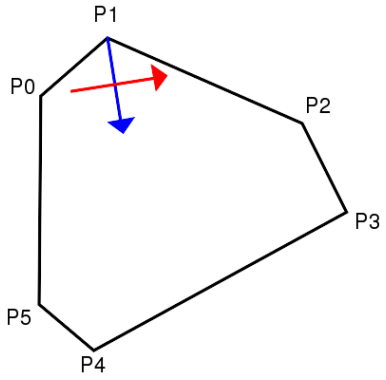


Abbildung 10: Richtungsvektor der Winkelhalbierenden in $P1$ (blau) und einen Richtungsvektor (rot) der zur Winkelhalbierenden senkrecht steht.

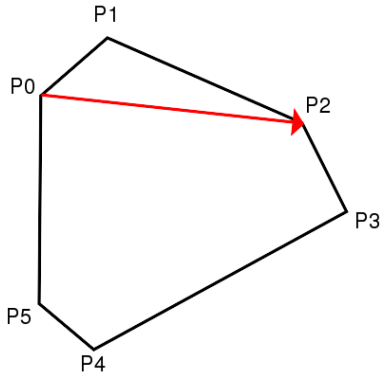


Abbildung 11: Vektor von $P0$ zu $P2$ der als Richtungsvektor für die Tangente im Endpunkt der Bézierkurve bei $P1$ dienen soll

kleinen Winkel mit der entsprechenden Kante einschließt und der Kontrollpunkt somit nahe an der Kante liegt. Bei kurzen Kanten führt dies zu einem gegenteiligen Effekt, da an diesen Stellen ein steiler Winkel entsteht. An den kurzen Kanten ist der Zuwachs an Abweichung aber im Gegensatz zur Abnahme an Abweichung an den langen Kanten geringer. Also wird der durch die benachbarten Eckpunkte definierte Richtungsvektor als Tangentenvektor der Bézierkurve an den Endpunkten verwendet.

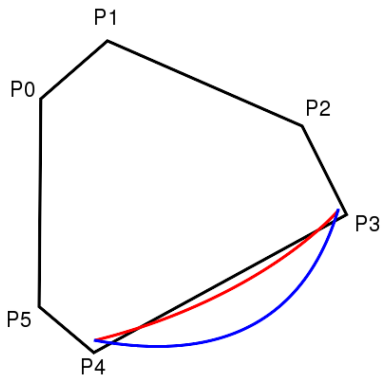


Abbildung 12: Je eine Bézierkurve mit dem Richtungsvektor der Senkrechten der Winkelhalbierenden als Tangentenvektoren (blau) und dem Richtungsvektor der durch die benachbarten Punkte gegeben ist.

3.1.2 Aufstellen eines Gleichungssystems

Aus den im vorherigen Kapitel genannten Bedingungen können Gleichungen aufgestellt werden, die den allgemeinen Verlauf der Bézierkurven im Verhältnis zu den Polygonecken beschreiben. Die in den Gleichungen verwendeten Variablen sollen hier definiert werden. Die Bedingungen für die elementaren Bézierkurven sollen an einem konkreten Polygonzug verdeutlicht werden. Dieser Polygonzug[8] besteht aus vier Punkten, wobei Gleichungen für die mittlere Kante aufgestellt werden sollen. Der Polygonzug ist konvex und

[8] Vgl. Abbildung 13 auf Seite 24.

kann für einen beliebigen Teil eines konvexen Polygons stehen. Die Eckpunkte des Polygonzuges sind von $P0$ bis $P3$ nummeriert. Da sich die Endpunkte der Bézierkurve durch die Verschiebung der Eckpunkte ergeben sollen, sind sie nach diesen benannt worden. Die Endpunkte der Bézierkurve lauten $Pt1$ und $Pt2$, wobei das „t“ im Namen verdeutlichen soll, dass der Punkt aus einem Eckpunkt hervorgeht, der um einen Parameter t verschoben ist.

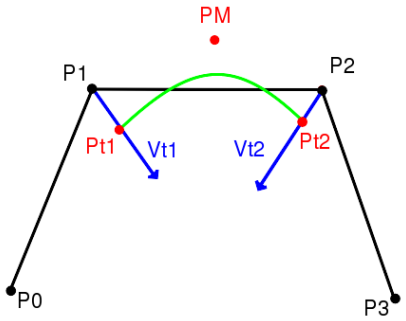


Abbildung 13: Benennung der Richtungsvektoren für die Verschiebung.

Der Vektor, der vom Eckpunkt $P1$ in Richtung Kontrollpunkt $Pt1$ der Bézierkurve zeigt, sei durch $Vt1$ gegeben. Der entsprechende Vektor für $P2$ sei mit $Vt2$ benannt. Der Vektor, der die Tangente an der Bézierkurve im Punkt $Pt1$ angibt, sei mit $V1$ benannt sowie der im Punkt $Pt2$ mit $V2$. Um alle Vektoren zu Vereinheitlichen, wird festgelegt, dass die Vektoren für die Verschiebung immer nach innen zeigen und die Richtungsvektoren der Tangenten immer in Richtung der Nummerierung, also in dem Teilstück von $P0$ zu $P2$ und von $P1$ zu $P3$[9]

$Pt1$ und $Pt2$ sind definiert als die Summe des Vektors zu $P1$ bzw $P2$ und dem entsprechenden Richtungsvektor der Winkelhalbierenden, der mit einem

[9]Die Gleichungen werden ohne explizite Angabe der entsprechenden Vektoren angegeben, obwohl diese in den vorherigen Kapiteln festgelegt wurden. Dies hat den Vorteil, dass die Gleichungen leichter, durch Austauschen der entsprechenden Formel für die Vektoren, entsprechend der jeweiligen Polygone angepasst werden können.

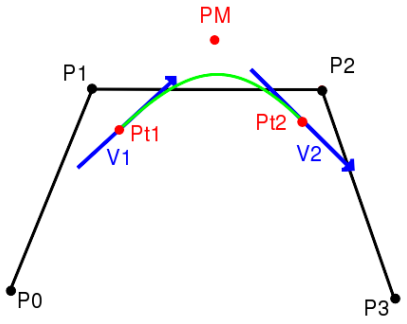


Abbildung 14: Benennung der Richtungsvektoren der Tangenten.

entsprechenden Parameter t_1 oder t_2 multipliziert wird. Da die Punkte somit von den Parametern abhängig sind, können sie als Funktion angesehen. Also:

$$Pt1(t_1) := P1 + t_1 \cdot Vt1 \tag{2}$$

$$Pt2(t_2) := P2 + t_2 \cdot Vt2 \tag{3}$$

Die Nummerierungen der Bezeichnungen der Parameter, soll die Zugehörigkeit zu einem Punkt verdeutlichen. So gehört zum Beispiel der Parameter t_1 zum Punkt $P1$.

Da eine Verschiebung ins Innere des Polygonzuges definiert und eine Verschiebung nach Außen ausgeschlossen werden soll, weil dies einer schlechteren Approximation entsprechen würde, ergibt sich, dass die t_i keinen negativen Wert annehmen dürfen.

Die Vektoren Vti sollen auf die Länge 1 normiert sein, denn dann werden die Abstände zwischen den Endpunkten der elementaren Bézierkurve und denen der Polygonkante, durch die Parameter t_1 und t_2 wiedergegeben. Der kürzeste Abstand zwischen der Bézierkurve und dem Polygon wird aber nicht immer durch die Werte der Parameter t_1 und t_2 wiedergegeben, sondern sie repräsentieren den Abstand zwischen zwei bestimmten Punkten. Die Parameter können negative Werte abnehmen, was aber nicht erwünscht ist. Der Ausschluss dieser negativen Werte führt so zu einer besseren Approximation

und schließt eventuell auftretende Sonderfälle aus.

Die Parameter t_1 und t_2 werden als Indikatoren für die Abweichung der Bézierkurve vom Polygon festgesetzt, da sie die tatsächliche Verschiebung des Eckpunktes und somit einen Wert für die Veränderung des Polygons wiedergeben. Die kürzeste Strecke zwischen Eckpunkt und Bézierkurve ließe sich nicht durch eine lineare Gleichung beschreiben und würde ein komplett anderes Verfahren, welches auf quadratischen Gleichungen beruht, erfordern. Es ist zudem auch nicht notwendig die kürzeste Stecke zu verwenden, da der Wert der Verschiebung einen gleichwertigen Indikator darstellt, da durch ihn ein Verfahren ermöglicht wird, das auf linearen Gleichungen basiert und somit bei weitem nicht so komplex ist wie ein Verfahren das quadratischen Gleichungen beinhaltet.

Damit die elementaren Bézierkurven stetig ineinander übergehen, muss der 2. Kontrollpunkt, im Folgenden PM genannt, als Schnittpunkt der Tangenten der Endpunkte festgelegt werden. Dafür sind zwei weitere Parameter s_1 und r_2 erforderlich. PM ist durch die Parameter folgendermaßen festgelegt:

$$PM := Pt1(t_1) + s_1 \cdot V1 \tag{4}$$

$$PM = Pt2(t_2) - r_2 \cdot V2 \tag{5}$$

Als weiterer Parameter für die Abweichung wird ein Abstand d_i von einem Punkt der entsprechenden elementaren Bézierkurve zur Kante K_i berechnet. Dazu wird der Parameter x der Bézierkurve $C_i(x)$ festgelegt und von dem daraus resultierenden Punkt der Abstand zur Kante berechnet. Daraus ergibt sich der Vorteil, dass dieser Abstand zur Kante von jedem Punkt der Bézierkurve aus durch eine lineare Gleichung beschrieben werden kann, während es bei einem Abstand, der von einem Punkt auf der Kante ausgehend angegeben wird, zu quadratischen Gleichungen kommt.

Die elementare Bézierkurve wird nach dem Algorithmus von de Casteljau

wie folgt festgelegt:

$$B1(x) := Pt1(t_1) + x \cdot s_1 \cdot V1 \quad (6)$$

$$B2(x) := PM + x \cdot r_2 \cdot V2 \quad (7)$$

$$C(x) := B1(x) + x \cdot (B2(x) - B1(x)) \quad (8)$$

Für die Gleichung, welche die Abstandsmessung definiert, wird zum einen eine Matrix D benötigt. D ist diejenige Matrix, durch die ein Vektor um 90° im Uhrzeigersinn gedreht wird. Zum anderen die Funktion $N(v)$, die einen Vektor auf die Länge 1 normiert, wobei v der zu normierende Vektor ist. Die Gleichung für d_1 an der Stelle $C(x_1)$ lautet:

$$P1 + e_1 \cdot (P2 - P1) = C(x_1) + d_1 \cdot D \cdot N(P2 - P1) \quad (9)$$

Es ist an dieser Stelle zu betonen, dass durch die Gleichungen keine Veränderung der Polygonzugkante beschrieben wird, die zu einer Bézierkurve führt, die das Polygon bestmöglich approximieren würde. Sondern es werden aufgrund der Lage der Eckpunkte des Polygons Gleichungen aufgestellt, welche die Lage der Kontrollpunkte der Bézierkurve beschreiben. Durch die Gleichungen sind alle Kontrollpunkte einer Bézierkurve nur noch in einem bestimmten Rahmen variabel. Der Graph der Bézierkurve kann nur in bestimmten Bereichen, welche durch die Gleichungen vorgegeben sind, verändert werden.

Die in den Gleichungen vorkommenden Parameter sollen, wie die Parameter t_1 und t_2, auf positive Werte eingeschränkt werden. Die Einschränkung, dass d_i nur positive Werte annehmen darf, führt dazu, dass die Bézierkurve nicht beliebig weit ins Innere des Polygonzuges geschoben werden kann. Negative Werte von d_1 könnten zum Beispiel bei einem konvexen Polygon zu einem vollständigen Verlauf einer Bézierkurve innerhalb eines Polygons führen (siehe Abbildung 15 auf Seite 28.). Dies wurde keiner Approximation eines Polygons, nach den genannten Kriterien entsprechen. In dieser Arbeit wird die Qualität der Approximation eines Polygons durch eine Bézierkurve, anhand der Summe der freien Parameter t_i und d_i gemessen. Dabei gilt, dass

die bestmöglichste Bézierkurve die geringste Summe der Parameter in den Gleichungen aufweist. Zudem führt ein negativer Parameter zu einer ungültigen Approximation.

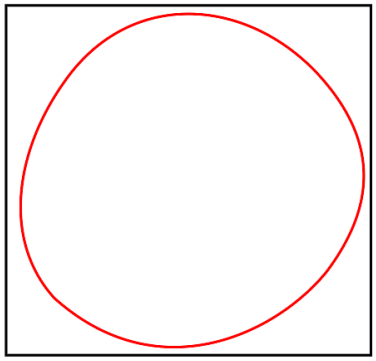


Abbildung 15: Bézierkurve, die innerhalb eines Polygons verläuft.

Für jede Kante von $\overline{P}$ werden Gleichungen aufgestellt, die elementare Bézierkurven beschreiben, die diese Kante approximieren. Diese werden dann zu einem linearen Gleichungssystem zusammengefasst, das quadratische Bézierkurven beschreibt, die das gesamte Polygon approximieren. Diese Gleichungen sollen als Parameter nur noch t_i und d_i enthalten, die zur Bewertung der Approximation herangezogen werden. Diese Gleichungen werden nachfolgend vorgestellt. Die Bennennung erfolgt wiederum anhand des Polygonzuges aus Abbildung 13 auf Seite 24.

Zur Berechnung und Lösung von Gleichungen und Gleichungssystemen wurde das Computer Algebra System (CAS) Derive verwendet. Ein Ausdruck der jeweiligen Eingaben ist der beigelegten CD-ROM im PDF-Format beigefügt.

Nach Eingabe der Gleichungen 2 bis 9 in das CAS Derive, kann aus den Gleichungen 5 und 9 ein lineares Gleichungssystem aufgestellt werden, und dieses nach s_1, e_1, r_2 und d_1 gelöst werden. Das CAS Derive liefert als Ergebnis vier Gleichungen[10]. In den Gleichungen werden s_1, e_1, r_2 und d_1 jeweils

[10] Die Formeln werden hier nicht im Text eingefügt, da diese den Lesefluss erheblich

in Abhängigkeit von t_1, t_2 und x beschrieben. Der Parameter x ist zwar frei wählbar, ist hier aber keine freie Variable. Der Wert von x soll zu Beginn der Approximation eines Polygons gewählt werden und geht somit nicht als freier Parameter in die Gleichungen ein, sondern als Konstante. Die Auswirkungen der Wahl des Wertes des Parameters werden in Kapitel 5.1 erläutert, da dies anhand der dann erstellten Bézierkurven einfacher ist.

Die vierte Gleichung (siehe Kapitel A.1.4) spielt in den folgenden Überlegungen eine zentrale Rolle. In dieser Gleichung wird das Verhältnis zwischen der Verschiebung der Eckpunkte $P1$ und $P2$ ins Innere des Polygonzuges und dem Abstand der Bézierkurve B zur Polygonkante beschrieben. Es soll mit Hilfe einer geringen Verschiebung t_1 und t_2 ein geringer Abstand d_1 erreicht werden. Diese Gleichung wird für jede Polygonkante eines konvexen Polygons mit n Kanten aufgestellt. Daraus ergeben sich n Gleichungen, die alle Kriterien für eine quadratische Bézierkurve enthalten, und ein Gleichungssystem bilden. Eine solche Gleichung für eine Kante hat die allgemeine Form

$$a_i^{(1)} \cdot d_i + a_i^{(2)} \cdot t_i + a_i^{(3)} t_{i+1} = a_i^{(4)} \tag{10}$$

Wenn für jede Kante nur ein Parameter d_i in das Gleichungssystem eingeht, kann es zu einer ungleichmäßigen Approximation kommen. In Abbildung 16 auf Seite 31 ist ein Beispiel für eine solche ungleichmäßige Approximation gezeigt. Um gleichmäßigere Abstände zu erhalten, wird eine weitere Gleichung für jede Kante in das Gleichungssystem aufgenommen. In dieser Gleichung wird der Wert des Parameters x auf einen anderen wiederum vorher festgesetzten Wert gesetzt[11]. Dadurch ergibt sich eine gleichmäßige Approximation (siehe Abbildung 17 auf Seite 31) und ein weiterer Punkt auf der Bézierkurve C_i an dem der Abstand zur Kante berechnet wird. Damit die zwei Parameter, die den Abstand zwischen Bézierkurve und Kante angeben nicht verwechselt

stören würden. Die Formeln sind dem jeweilig angegeben Unterkapitel zu entnehmen. Auf die ersten drei Gleichungen (siehe Kapitel A.1.1 bis A.1.3) wird zudem in späteren Überlegungen kein Bezug genommen, deswegen werden sie in dieser Arbeit nicht genauer erläutert.

[11] Die Auswirkungen der Wahl des Wertes der Parameter wird in Kapitel 5.1 erläutert

werden, wird ein weiterer Index an den Parameter angefügt. Der erste Abstand an der Kante zwischen $P1$ und $P2$ lautet $d_{1,1}$ und der zweite $d_{1,2}$. Für jede Kante werden also zwei Gleichungen aufgestellt, mit zwei verschiedenen Parametern x_1 und x_2, welche den Ort der Abstandsmessung an der Bézierkurve angeben. Folglich gehen zwei Gleichungen der folgenden Form für jede Kante in ein Gleichungssystem ein.

$$a_{i,j}^{(1)} \cdot d_{i,j} + a_{i,j}^{(2)} \cdot t_i + a_{i,j}^{(3)} t_{i+1} = a_{i,j}^{(4)} \tag{11}$$

3.1.3 Suchen der optimalen Bézierkurve

Die Lösung eines linearen Gleichungssystem ist nach der linearen Algebra ein linearer Unterraum (im Weiteren LU genannt). Jeder Punkt in diesem LU beschreibt eine Bézierkurve, aber nicht jeder Punkt in diesem LU beschreibt nach den bis jetzt genannten Kriterien eine gültige Bézierkurve. Im LU können die Parameter negative Werte annehmen, die hier ungültig sind. Der optimale Punkt im LU, der die Parameter für die Bézierkurve beschreibt, hat die Eigenschaft, dass alle Einträge positiv sind. Des Weiteren sollte die Abweichung der Bézierkurve vom Polygon minimal sein und somit auch die Summe aller Einträge des Punktes. Dieser Punkt kann durch Umformung des Gleichungssystems ermittelt werden.

Dazu muss vorab geklärt werden, ob es überhaupt für jedes Polygon eine Bézierkurve gibt, die alle Kriterien erfüllt. Andernfalls würde ein Punkt gesucht werden, der nicht im LU vorhanden ist. Im Falle eines konvexen Polygons existiert allerdings immer ein solcher Punkt, der eine gültige Bézierkurve beschreibt. Die Bézierkurve dessen Graph durch die Eckpunkte des Polygons verläuft, erfüllt alle Kriterien. Die Werte der Parameter t_i sind Null, sowie die der Parameter $d_{i,j}$ größer Null, da die Tangenten in den Eckpunkten die Kontrollpunkte vorgeben. In Abbildung 18 auf Seite 32 ist ein Beispiel gezeigt. Unabhängig davon wie ein konvexes geschlossenes Polygon verläuft, die Bézierkurve durch die Eckpunkte ist eine gültige Lösung nach

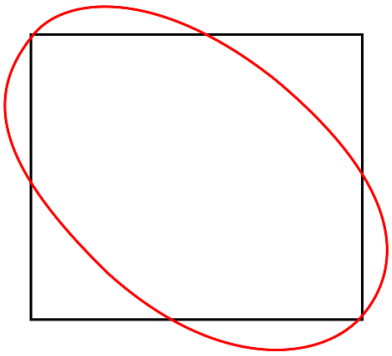


Abbildung 16: Unausgeglichene Approximation: Die Abstände sind nicht gleichmäßig auf alle Eckpunkte verteilt.

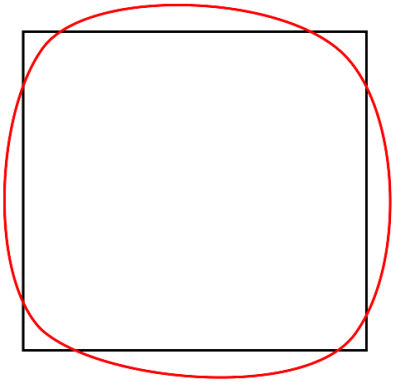


Abbildung 17: Ausgeglichene Approximation: Die Abstände sind gleichmäßig auf alle Eckpunkte verteilt.

diesen Kriterien. Da also immer eine Lösung vorhanden ist, muss durch Wahl eines geeigneten Suchverfahrens der optimale Punkt im LU ermittelt werden.

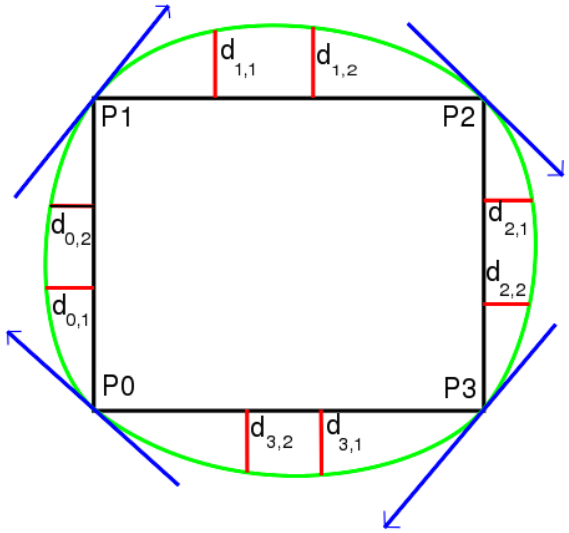


Abbildung 18: Gültige Anfangslösung.

Ein Gleichungssystem kann nach dem Gauß'schen Eliminationsverfahren[12] vereinfacht werden, so dass eine Hauptdiagonale entsteht. Dazu wird wie folgt vorgegangen: Es werden die Gleichungen von oben im Gleichungssystem angefangen, also von der 1-sten Zeile, der Reihe nach umgeformt, also bis zur $2 \cdot n$-ten Zeile. Dabei weist das zum Gleichungssystem gehörige Polygon n Kanten auf. Zuerst wird die entsprechende Zeile, angenommen die j-te, normiert, so dass der j-te Eintrag in der j-ten Zeile auf Eins normiert wird. Das wird erreicht indem die j-te Zeile durch den Eintrag $a_{j,j}$ dividiert wird, sofern $a_{j,j} \neq 0$. Anschließend werden die Zeilen unterhalb der j-ten Zeile umgeformt, so dass in der j-ten Spalte unterhalb des j-ten Eintrags nur noch Nullen vorhanden sind, indem von den unteren Zeilen die j-te Zeile mit einem passendem Faktor abgezogen wird. Diese Schritte sind an folgendem

[12] Vgl. Trapp [1], Seite 67ff., Satz 9.2

Gleichungssystem gezeigt. Das Gleichungssystem enthält nur 3 Gleichungen, da es nur die Schritte der Umformung veranschaulichen soll.

$$\left|\begin{array}{cccccc|c} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & b_1 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & b_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & b_3 \end{array}\right|$$

1. Schritt: Normieren der 1. Zeile ($a_{j,j} \neq 0$)

$$\left|\begin{array}{cccccc|c} 1 & a'_{1,2} & a'_{1,3} & a'_{1,4} & a'_{1,5} & a'_{1,6} & b'_1 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & b_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & b_3 \end{array}\right|$$

2. Schritt: Erzeugen der Nullen

$$\left|\begin{array}{cccccc|c} 1 & a'_{1,2} & a'_{1,3} & a'_{1,4} & a'_{1,5} & a'_{1,6} & b'_1 \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} & a'_{2,6} & b'_2 \\ 0 & a'_{3,2} & a'_{3,3} & a'_{3,4} & a'_{3,5} & a'_{3,6} & b'_3 \end{array}\right|$$

Restlichen Zeilen analog

$$\left|\begin{array}{cccccc|c} 1 & a'_{1,2} & a'_{1,3} & a'_{1,4} & a'_{1,5} & a'_{1,6} & b'_1 \\ 0 & 1 & a''_{2,3} & a''_{2,4} & a''_{2,5} & a''_{2,6} & b''_2 \\ 0 & 0 & 1 & a''_{3,4} & a''_{3,5} & a''_{3,6} & b''_3 \end{array}\right|$$

Um die restlichen Einträge zu dem Wert Null umzuformen, wird analog vorgegangen. Diesmal wird von der $2 \cdot n$-ten Zeile beginnend nach oben und von der $2 \cdot n$-ten Spalte nach links umgeformt. Das Normieren entfällt und die Einträge in der j-ten Spalte über dem j-ten Eintrag werden zum Wert Null umgeformt, indem die j-te Zeile mit dem entsprechenden Faktor von der jeweiligen Zeile abgezogen wird. Dies wird wiederum am Gleichungssystem, das zu einem Reduzierten System[13] umgeformt wird, verdeutlicht.

Die oberen Nullen erzeugen

$$\left|\begin{array}{ccccccc} 1 & 0 & 0 & c_{1,4} & c_{1,5} & c_{1,6} & \tilde{b}_1 \\ 0 & 1 & 0 & c_{2,4} & c_{2,5} & c_{2,6} & \tilde{b}_2 \\ 0 & 0 & 1 & c_{3,4} & c_{3,5} & c_{3,6} & \tilde{b}_3 \end{array}\right|$$

[13] Vgl. Trapp [1], Seite 68, Satz 9.4

Beim Erzeugen der Hauptdiagonalen ist Folgendes zu berücksichtigen. An der Stelle $a_{j,j}$ muss zum Einen nicht immer ein Eintrag sein, dessen Wert ungleich Null ist. Somit könnte dann keine Normierung durchgeführt werden. Das Gleichungssystem wird aber so aufgestellt, dass die Einträge auf der Hauptdiagonalen immer ungleich Null sind. Das Gleichungssystem für ein konvexes Polygon mit n Kanten soll folgende Form haben.

$d_{1,1}$	$d_{2,1}$	$\cdots$	$\cdots$	$d_{n,2}$	t_1	t_2	$\cdots$	$\cdots$	t_n	
$a_{1,1}$	$a_{1,2}$	$\cdots$	$\cdots$	$a_{1,2\cdot n}$	$a_{1,2\cdot n+1}$	$a_{1,2\cdot n+2}$	$\cdots$	$\cdots$	$a_{1,3\cdot n}$	b_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Das Gleichungssystem wird so aufgebaut, dass die Gleichungen entsprechend der Reihenfolge der Parameter in das Gleichungssystem eingetragen werden. Folglich steht in der ersten Zeile die Gleichung mit $d_{1,1}$, dann die mit $d_{1,2}$ usw., so dass die Gleichung für $d_{n,2}$ in der $2\cdot n$ ten Zeile steht. Aus diesem Aufbau ergibt sich eine Hauptdiagonale, die nur noch normiert werden muss. Dies folgt aus der Tatsache, dass in jeweils einer Gleichung nur ein Parameter $d_{i,j}$ und die Parameter t_i und t_{i+1} auftreten (siehe Gleichung 11).

Das obige System kann nur Verwendung finden, wenn die Anzahl der freien Variablen höher ist als die der Gleichungen. Da dies bei konvexen Polygonen der Fall ist, kann diese Vorgehensweise bei diesen zur Anwendung kommen. Bei einem Polygon mit n Punkten, gibt es n Kanten die zu approximieren sind. Das bedeutet, das für jede Kante zwei Gleichungen aufgestellt werden müssen und das Gleichungssystem $2 \cdot n$ Gleichungen enthält. Für jeden Eckpunkt existiert ein freier Parameter t_i und für jede Kante zwei Parameter $d_{i,j}$. Daraus folgt, dass es $3 \cdot n$ freie Variablen gibt, also n mehr als Gleichungen.

Ein Punkt des LU kann berechnet werden, indem alle Parameter bis auf die ersten $2 \cdot n$ des Gleichungssystems mit $2 \cdot n$ Gleichungen auf Null gesetzt und die restlichen dann berechnet werden. Die so berechnete Lösung wird Basislösung des Gleichungsystems genannt[14]. Der Lösungsvektor entspricht

[14] Vgl. Felgenhauer [3], Seite 23, Definition 2.2

der b-Spalte[15] die um Null-Einträge ergänzt worden ist. Da nach dem vorher genannten Aufbau alle Werte der t_i zu Beginn auf Null gesetzt wurden, entspricht die Anfangslösung einer gültigen Lösung. Die Basislösung muss aber nicht die bestmögliche Bézierkurve wiedergeben. Dazu werden nun Überlegungen durchgeführt, wie von dieser Basislösung zu einer besseren Basislösung übergegangen werden kann.

Ein Gleichungssystem, mit dessen Hilfe ein Lösungsvektor berechnet wurde, hat nach Bildung einer Hauptdiagonalen folgende Gestalt.

a_1	a_2	$\cdots$	$\cdots$	$a_{2\cdot n}$	c_1	c_2	$\cdots$	$\cdots$	c_n	
1	0	$\cdots$	$\cdots$	0	$c_{1,2\cdot n+1}$	$c_{1,2\cdot n+2}$	$\cdots$	$\cdots$	$c_{1,3\cdot n}$	b_1
0	1			0	$c_{2,2\cdot n+1}$	$c_{2,2\cdot n+2}$	$\cdots$	$\cdots$	$c_{2,3\cdot n}$	b_2
$\vdots$	0	$\ddots$		$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\vdots$	$\vdots$
$\vdots$	$\vdots$		1	0	$\vdots$	$\vdots$		$\ddots$	$c_{2\cdot n-1,3\cdot n}$	$\vdots$
0	0	$\cdots$	0	1	$c_{2\cdot n,2\cdot n+1}$	$c_{2\cdot n,2\cdot n+2}$	$\cdots$	$c_{2\cdot n,3\cdot n-1}$	$c_{2\cdot n,3\cdot n}$	$b_{2\cdot n}$

Da der Lösungsvektor berechnet wird, indem die Werte der Variablen c_1 bis c_n auf Null gesetzt werden, ist die Summe der Einträge der Basislösung gleich der Summe der Einträge der b-Spalte. Im Folgenden soll mit *Min-Schritt* eine Umformung im Gleichungssystem bezeichnet werden, die eine Verringerung der Summe der Einträge der b-Spalte zur Folge hat, aber nicht zu einer ungültigen Lösung führt. Dieses Verfahren basiert auf dem Simplex-Algorithmus. Der angewendete *Min-Schritt* ist dem Austauschschritt des Simplex-Algorithmus ähnlich. In diesem Verfahren, wird aber auf die Zielfunktion verzichtet, da immer die Summe aller Einträge berechnet werden soll.

Ein *Min-Schritt* besteht aus den folgenden, beispielhaft an einem kleinen Gleichungssystem gezeigten, Schritten[16] :

- Wählen eines geeigneten Eintrags $c_{i,j}$

[15] Als b-Spalte soll die letzte Spalte im Gleichungssystem benannt werden.

[16] Für die Variablen gelte im Folgenden: $i, j, l \in \mathbb{N} \setminus \{0\}$ und $l \neq j$

Eintrag $c_{1,6}$ wird gewählt.

$$\left(\begin{array}{cccccc|c} 1 & 0 & 0 & c_{1,4} & c_{1,5} & \mathbf{c_{1,6}} & b_1 \\ 0 & 1 & 0 & c_{2,4} & c_{2,5} & c_{2,6} & b_2 \\ 0 & 0 & 1 & c_{3,4} & c_{3,5} & c_{3,6} & b_3 \end{array}\right)$$

- Tauschen der Spalten j und i

  Tauschen der Spalten 6 und 3.

$$\left(\begin{array}{cccccc|c} \mathbf{c_{1,6}} & 0 & 0 & c_{1,4} & c_{1,5} & 1 & b_1 \\ c_{2,6} & 1 & 0 & c_{2,4} & c_{2,5} & 0 & b_2 \\ c_{3,6} & 0 & 1 & c_{3,4} & c_{3,5} & 0 & b_3 \end{array}\right)$$

- Umformen um eine Hauptdiagonale zu erstellen

$$\left(\begin{array}{cccccc|c} 1 & 0 & 0 & c'_{1,4} & c'_{1,5} & c'_{1,6} & b'_1 \\ 0 & 1 & 0 & c'_{2,4} & c'_{2,5} & c'_{2,6} & b'_2 \\ 0 & 0 & 1 & c'_{3,4} & c'_{3,5} & c'_{3,6} & b'_3 \end{array}\right)$$

Zu klären ist nun, welchen Kriterien der Eintrag $c_{i,j}$, der zu einer Verringerung der Summe führen soll, entsprechen muss. Je nach Wahl des Eintrags $c_{i,j}$, ergibt sich nach Durchführung des *Min-Schrittes*, eine entsprechende b-Spalte und somit auch eine entsprechende Basislösung. Um die Kriterien, die an den Eintrag $c_{i,j}$ gestellt werden, zu bestimmen, wird der dritte Schritt genauer untersucht, da der erste und der zweite Schritt keine Veränderung der b-Spalte erzeugen und somit auch keine Veränderung der Basislösung.

Der dritte Schritt kann wiederum in zwei Teile unterteilt werden. Im ersten Teil wird die Zeile durch den Eintrag $c_{i,j}$ dividiert. Wenn der Eintrag $c_{i,j}$ negativ war, ergibt sich eine nicht gültige Basislösung, da mit einer gültigen Anfangslösung begonnen wurde. Der Eintrag b_j war daher vorher positiv und wird durch das Dividieren mit einem negativen Eintrag $c_{i,j}$ negativ. Die erste Einschränkung für $c_{i,j}$ besteht somit darin, dass dieser Eintrag positiv sein muss, um bei einer korrekten Basislösung zu bleiben. Darüber hinaus

darf der Eintrag $c_{i,j}$ nicht den Wert Null haben, um den Austausch der Spalten durchführen zu können. Wenn der Eintrag Null wäre, könnte durch den beschriebenen *Min-Schritt* keine Hauptdiagonale entstehen.

Im zweiten Teil wird dann von den übrigen Zeilen das $c_{i,l}$-te der j-ten Zeile von der l-ten Zeile abgezogen, wobei gilt $l \neq j$. Da dies zu Veränderungen in der b-Spalte führt, ist darauf zu achten, dass die Einträge in der b-Spalte nicht negativ werden oder dass es zu einer Vergrößerung der Summe kommt und somit dann zu einer schlechteren Approximation.

Um die Bedingungen für den Eintrag $c_{i,j}$ weiter einzugrenzen, werden die Veränderungen in der b-Spalte betrachtet. Nachdem ein *Min-Schritt* durchgeführt wurde, ergibt sich die neue b-Spalte aus der alten in folgender Weise. Die neuen Einträge sind durch eine Tilde über dem Buchstaben kenntlich gemacht.

$$\begin{aligned} \tilde{b}_j &= \frac{b_j}{c_{i,j}} \\ \tilde{b}_l &= b_l - \frac{c_{i,l} \cdot b_j}{c_{i,j}} \text{ mit } l \neq j \end{aligned}$$

Da die Summe der Einträge der b-Spalte verringert werden soll, wird nun die Veränderung dieser in den Blick genommen.

$$\begin{aligned} \sum_{l=1}^{n} \left(\tilde{b}_l\right) &= \sum_{\substack{l=1 \\ l \neq j}}^{n} \left(b_l - \frac{c_{i,l} \cdot b_j}{c_{i,j}}\right) + \frac{b_j}{c_{i,j}} \\ &= \sum_{\substack{l=1 \\ l \neq j}}^{n} (b_l) - \sum_{\substack{l=1 \\ l \neq j}}^{n} \left(\frac{c_{i,l} \cdot b_j}{c_{i,j}}\right) + \frac{b_j}{c_{i,j}} \\ &= \underbrace{\sum_{\substack{l=1 \\ l \neq j}}^{n} (b_l)}_{1.Summand} \underbrace{-\frac{b_j}{c_{i,j}} \cdot \sum_{\substack{l=1 \\ l \neq j}}^{n} (c_{i,l})}_{2.Summand} + \underbrace{\frac{b_j}{c_{i,j}}}_{3.Summand} \end{aligned}$$

Der Wert des ersten Summanden ist größer Null, da mit einer gültigen Basislösung begonnen wurde und die b-Spalte somit nur positive Einträge hatte.

Der dritte Summand ist ebenfalls positiv, da der Eintrag b_j der vorherigen Basislösung ebenfalls positiv war, sowie der Eintrag $c_{i,j}$. Daraus folgt, dass der zweite Summand einen Wert kleiner Null haben muss, um die Summe der Einträge zu verringern. Damit der Wert des zweiten Summanden kleiner Null ist, muss die Summe der Einträge in der i-ten Spalte ohne $c_{i,j}$ größer Null sein. Dies ergibt sich aus der Wahl des Eintrags $c_{i,j}$, der bereits nach den oben genannten Voraussetzung positiv sein soll. Eine weitere Einschränkung für die Wahl von $c_{i,j}$ ist somit, dass die Summe der Einträge der Spalte, in der das $c_{i,j}$ gewählt wird, ohne $c_{i,j}$ einen Wert größer Null haben muss, sowie der Eintrag b_j einen ungleich Null, da ansonsten der zweite Summand den Wert Null hätte. Die Spalte in der das $c_{i,j}$ gewählt werden soll, kann sogar noch weiter eingeschränkt werden. Die folgenden Umformungen zeigen, dass die Summe ohne $c_{i,j}$ sogar größer als Eins sein muss, um die Summe zu verringern.

$$
\begin{aligned}
& -\tfrac{b_j}{c_{i,j}} \cdot \sum_{\substack{l=1 \\ l \neq j}}^{n} (c_{i,l}) + \tfrac{b_j}{c_{i,j}} && < 0 \\
\iff \quad & -\sum_{\substack{l=1 \\ l \neq j}}^{n} (c_{i,l}) + 1 && < 0 \mid \text{da } b_j \text{ und } c_{i,j} \text{ grösser Null} \\
\iff \quad & \sum_{\substack{l=1 \\ l \neq j}}^{n} (c_{i,l}) - 1 && > 0 \\
\iff \quad & \sum_{\substack{l=1 \\ l \neq j}}^{n} (c_{i,l}) && > 1
\end{aligned}
$$

Ein Eintrag $c_{i,j}$, der diese Voraussetzungen erfüllt, garantiert noch nicht, dass nach dem *Min-Schritt* mindestens ein Eintrag in der b-Spalte negativ ist. Somit muss, nachdem ein Eintrag nach diesen sehr einfachen Kriterien gefunden ist, geprüft werden, ob sich ein ungültiges Ergebnis ergibt.

Damit die minimale Summe der b-Spalte und somit das bestmögliche Ergebnis gefunden wird, muss so oft ein *Min-Schritt* durchgeführt werden, bis kein dafür geeignetes Element mehr vorhanden ist. Durch Einsetzen der

im Gleichungsystem stehenden Einträge in die genannten Formeln[17], können die entsprechenden Kontrollpunkte berechnet werden.

3.1.4 Zusammenfassung

Die Vorgehensweise, die in den vorherigen Kapiteln erarbeitet wurde, wird hier zusammengefasst.

Die Kriterien, welche an die Bézierkurve gestellt werden sollen, die eine Kante eines Polygons approximieren soll, wurden in Gleichungen zusammengefasst. Aus diesen Gleichungen ergaben sich vier Gleichungen, wobei eine von diesen die Verhältnisse der zur Approximation verwendeten Parameter wiedergibt. Für jede Kante eines konvexen Polygons werden zwei solcher Gleichungen aufgestellt, die aus den Endpunkten der Kante und den Nachbarpunkten berechnet werden können. Aus diesen Gleichungen wird ein lineares Gleichungssystem erstellt, durch das nach dem vorgestellen Algorithmus der Punkt im Lösungsraum berechnet wird, der die bestmögliche Bézierkurve wiedergibt. Aus diesem Punkt wird dann die entsprechende Bézierkurve berechnet.

3.1.5 Beweise

In den vorangegangenen Kapiteln wurden die Beweise, dass der vorgestellte *Min-Schritt* die bestmögliche Bézierkurve berechnet, nicht aufgeführt, um den Lesefluss nicht zu stören. Daher sollen sie in diesem Kapitel Berücksichtigung finden.

Die Lösungsmenge eines linearen Gleichungssystems ist ein linearer Unterraum[18]. Durch die Einschränkung, dass die Einträge der Lösungen alle positiv und beschränkt sind, wird die Lösungsmenge auf einen Polyeder eingeschränkt[19]. Dies soll im Folgenden erläutert werden.

[17] Vgl. Gleichungen 2 bis 5.
[18] Vgl. Trapp [1], Seite 78, Satz 11.6
[19] Vgl. Felgenhauer [3], Seite 21

Es sei eine gültige elementare Bézierkurve an der Kante eines Polygonzuges gegeben, welche die Punkte $P1$ und $P2$ verbindet (siehe Abbildung 19). Wenn der Punkt $Pt1$ auf der Winkelhalbierenden $Vt1$ nun soweit ins Innere des Polygonzuges verschoben wird, bis die Tangente $V1$ in diesem Punkt durch den Punkt $P2$ verläuft, muss der mittlere Kontrollpunkt auf der Tangente liegen und ist mit dem Punkt $P2$ identisch. Da nun alle drei Kontrollpunkte auf einer Geraden liegen, liegt die Bézierkurve ebenfalls auf der Geraden. Der Parameter d_1 müsste nun negativ sein, um die Gleichung 9 zu erfüllen, was keiner korrekten Approximation entsprechen würde. Somit muss t_1 beschränkt sein.

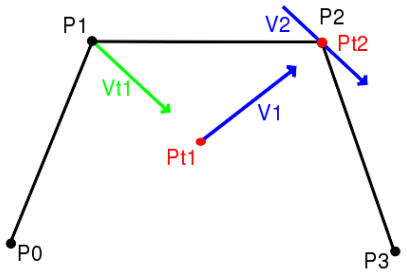


Abbildung 19: t_1 muss beschränkt sein.

Für den Parameter d_1 gilt Folgendes. Um den Abstand zwischen Bézierkurve und Polygonzugkante zu vergrößern, muss der Kontrollpunkt von der Kante entfernt werden. Die maximale Entfernung ist diejenige, bei der die Parameter t_1 und t_2 den Wert Null annehmen. Da der mittlere Kontrollpunkt durch den Schnittpunkt der beiden Tangenten definiert ist, würde ein weiteres Entfernen des mittleren Kontrollpunktes von der Kante, dazu führen, dass der Parameter t_1 oder t_2 negativ wird, was wiederrum zu einer nicht gültigen Approximation führt. Damit kann der Parameter d_1 nicht größer werden, und ist somit beschränkt (siehe Abbildung 20 auf Seite 41). Da die Parameter alle positiv sein sollen, sind alle Parameter beschränkt und die

gültige Lösungsmenge ist ein Polyeder.

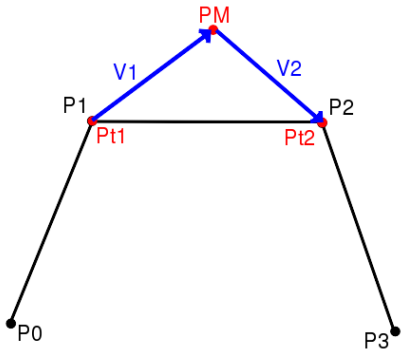


Abbildung 20: d_1 muss beschränkt sein.

Die Summe der Einträge ist eine Zielfunktion und das Minimum wird in einer Ecke des Polyeders eingenommen[20]. Jede Ecke des Polyeders entspricht dabei einer Basislösung. Der hier angewendete *Min-Schritt* ist ein spezieller Austauschschritt, in dem von einer Ecke zu einer günstigeren Ecke, also von einer Basislösung zu einer besseren Basislösung, übergegangen wird[21]. Ein solcher Übergang kann, wenn eine Degenerierung des Gleichungssystems eingetreten ist, nach den oben genannten Kriterien nicht immer durchgeführt werden.

Eine Degenerierung tritt ein, wenn ein Eintrag in der b-Spalte zu dem Wert Null umgeformt wird[22]. Wenn nun zu keiner günstigeren Ecke übergegangen werden kann, das Gleichungssystem aber degeneriert ist, müssen alle möglichen Formen, die das Gleichungssystem an dieser Ecke haben kann, erzeugt werden und es muss geprüft werden, ob ein *Min-Schritt* von diesem Gleichungssystem ausgehend möglich ist. Ist von keiner dieser Gleichungssysteme ein *Min-Schritt* möglich, ist das Minimum erreicht[23].

[20] Vgl. Felgenhauer [3], Seite 21, Satz 2.2
[21] Vgl. Felgenhauer [3], Seite 27
[22] Vgl. Felgenhauer [3], Seite 27
[23] Vgl. Felgenhauer [3], Seite 29f.

3.2 Konvexe Polygonzüge

In diesem Kapitel sollen die Kriterien für das Approximieren von konvexen Polygonzügen entwickelt werden. Diese Kriterien müssen eingeführt werden, um festzulegen, wie die quadratischen Bézierkurven an den Enden des Polygonzuges verlaufen sollen. Bei konvexen Polygonen wurden die Kriterien für die elementare Bézierkurve einer Kante durch die beiden Nachbarkanten beschrieben. Eine Kante, die am Anfang oder Ende eines Polygonzuges liegt, hat aber nur eine Nachbarkante. Deswegen müssen für diese Kante neue Überlegungen angestellt werden.

Für die Lage der Endpunkt der Bézierkurve sind zwei verschiedene Möglichkeiten denkbar. Die erste Möglichkeit ist, dass der Endpunkt, wie die anderen Eckpunkte in das Innere des Polygonzuges verschoben wird (siehe Abbildung 21). Wenn nun aber die Endpunkte eines Polygonzuges auf dem Rand einer Graphik liegen, kann dies dazu führen, dass eine Lücke entsteht und somit die Bézierkurve im Gegensatz zum Polygonzug keine Fläche mehr begrenzt. Um einen solchen Fall zu vermeiden, werden die Endpunkte der Bézierkurve auf die Endpunkte des Polygonzuges gesetzt.

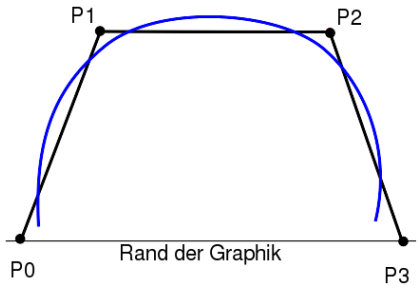


Abbildung 21: Die Endpunkte sind ins Innere des Polygonzuges verschoben.

Die Kriterien werden an einem Polygonzug (siehe Abbildung 22 auf Seite 44) mit vier Punkten vorgestellt, wobei die linke Seite den Anfang des

Polygonzuges beschreiben soll.

Als Tangente der Bézierkurve am Punkt $P0$ wurde der Vektor von $P0$ zu $P1$ gewählt. Der Endpunkt des Polygonzuges wurde nicht ins Innere des Polygonzuges verschoben, deswegen ist kein Ausgleich der Fläche notwendig. Ein Vektor, um den $P0$ verschoben wird, muss somit nicht aufgestellt werden. Für die Tangente in $P1$ muss, damit ein stetiger Übergang zu der nächsten elementaren Bézierkurve erreicht wird, der Vektor zwischen $P0$ und $P2$ gewählt werden. Die elementare quadratische Bézierkurve ist somit bereits festgelegt. Der erste Kontrollpunkt ist der Endpunkt, der dritte Kontrollpunkt ist der Punkt $Pt1$, und der zweite Kontrollpunkt ist der Schnittpunkt der Tangenten mit der 1. Kante.

Die daraus resultierende Gleichung kann äquivalent zu den vorherigen Gleichungen aufgestellt werden. Sie wird aber nicht in das Gleichungssystem aufgenommen, da aus folgenden Gründen keine Optimierung dieser elementaren Bézierkurve möglich ist. Um einen stetigen Übergang von der elementaren Bézierkurve, welche die erste Kante beschreibt, zu der nächsten elementaren Bézierkurve zu erreichen, darf der Parameter t_1 nicht den Wert Null annehmen, da ansonsten der Übergang an dem Eckpunkt nicht stetig wäre. Der 2. Kontrollpunkt PM wäre dann mit dem Eckpunkt $P1$ identisch. Beim Suchen der bestmöglichen Basislösung wird aber ein Teil der Parameter auf den Wert Null gesetzt, um eine Basislösung zu erhalten. Um zu verhindern, dass das Gleichungssystem eine Basislösung wiedergibt, die den Parameter t_1 auf den Wert Null setzt, wird von vornherein der Parameter $d_{1,1}$ auf den Wert Null gesetzt.

Dieses Vorgehen führt dazu, dass der Parameter $d_{1,1}$ den niedrigsten Wert annimmt, der möglich ist und somit zu einer guten Approximation führt. Des Weiteren kann der Parameter t_1 nicht den Wert Null annehmen, da ansonsten eine ungültige Basislösung erreicht wird.

In dem Fall, dass t_1 und $d_{1,1}$ den Wert Null annehmen, verläuft die elementare Bézierkurve der Kante, aufgrund der Tangente in $P1$, wie in der Abbildung 23 auf Seite 44 zu sehen. Dies setzt vorraus, dass der Parameter

$d_{1,2}$ negativ sein muss, was keiner gültigen Basislösung entspricht.

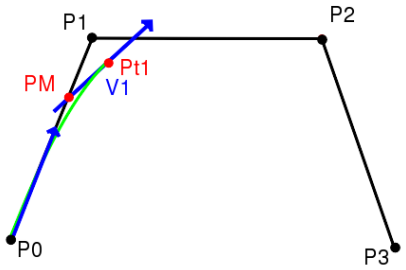


Abbildung 22: Benennung an der ersten Kante.

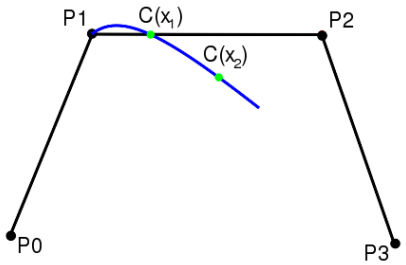


Abbildung 23: Ungültige Approximation wenn $d_{1,1}$ und t_1 den Wert Null annehmen.

Somit muss der Parameter t_1 einen positiven Wert annehmen und die Tangente in $Pt1$ nach Innen verschieben. Der mittlere Kontrollpunkt der ersten elementaren Bézierkurve muss zudem immer zwischen $P0$ und $P1$ liegen. Wenn der Abstand $d_{1,1}$ zwischen $P1$ und $P2$ angesetzt wird ist dies immer der Fall, da die Tangente in $Pt1$ passend gewählt wurde. Somit wird immer eine gültige elementare Bézierkurve für die erste Kante erreicht.

Also werden nach dem obigen Verfahren zunächst alle Gleichungen für die Kanten aufgestellt. Dabei werden die Anfangs- und Endkante vorerst vernachlässigt. Dann wird für die Anfangskante der Parameter $d_{1,1}$ auf den Wert Null gesetzt, sowie für die Endkante der entsprechende Parameter $d_{n-1,2}$. Das Gleichungssystem hat dann folgende Gestalt.

$d_{1,2}$	$d_{2,1}$	$\cdots$	$\cdots$	$d_{n-1,1}$	t_1	t_n	t_2	$\cdots$	t_{n-1}	
$a_{1,1}$	$a_{1,2}$	$\cdots$	$\cdots$	$a_{1,2\cdot n}$	$a_{1,2\cdot n+1}$	$a_{1,2\cdot n+2}$	$\cdots$	$\cdots$	$a_{1,3\cdot n}$	b_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Die Gleichungen, in denen die Parameter $d_{1,1}$ und $d_{n-1,2}$ auf den Wert Null gesetzt wurden, sind in die letzten beiden Zeilen des Gleichungsystems eingetragen. Da t_1 und t_n positiv sind und die Hauptdiagonale in dem Gleichungssystem bis t_n reicht, wird ein negativer Eintrag in der b-Spalte verhindert. Durch diesen Aufbau wird der Rechenaufwand verringert, da die Hauptdiagonale nur noch in der $2 \cdot n - 1$ ten und $2 \cdot n$ ten Spalte erzeugt werden muss. Die Gleichungen werden mittels des Verfahrens, welches innerhalb der Approximation von konvexen Polygonen erläutert wurde, nach Bildung einer Hauptdiagonalen, äquivalent umgeformt und die Basislösung berechnet, welche die bestmögliche Bézierkurve wiedergibt.

Wegen der Wahl der Tangenten gibt es immer eine Bézierkurve, die der gültigen Anfangslösung entspricht. In dieser Lösung haben die Parameter t_2 bis t_{n-1}, $d_{1,1}$ und $d_{n-1,2}$ jeweils den Wert Null. Die restlichen Parameter müssen dann, aufgrund der Tangenten an den Eckpunkten, positiv sein. Die einzige Ausnahme bilden Polygonzüge mit weniger als vier Eckpunkten. Polygonzüge mit drei Eckpunkten können ohne Optimierung folgendermaßen approximiert werden. Die zwei Kanten der Polygonzüge werden durch eine quadratische Bézierkurve approximiert. Der erste Kontrollpunkt ist der erste Eckpunkt, der mittlere Kontrollpunkt ist der mittlere Eckpunkt und der letzte Kontrollpunkt ist der letzte Eckpunkt. Dieses Verfahren wiederspricht zwar dem Anspruch den Flächeninhalt möglichst gleich zu halten, aber die Einhaltung dieser Forderung würde eine Bézierkurve aus mindestens zwei

elementaren Bézierkurven erfordern, was den Aufwand zur Darstellung für diesen Polygonzug verdoppeln würde. Des Weiteren wird später ein Vorteil daraus gezogen, dass die Tangente am Ende der Bézierkurve, der Geraden durch die letzten zwei Eckpunkte entspricht. Um also die Komplexität gering zu halten und diesem Spezialfall eine nicht zu hohe Bedeutung beizumessen, wird eine ungenauere Approximation durch eine elementare Bézierkurve verwendet. In der Implementation wird dieser Fall gesondert behandelt und kann unter Verwendung der aufwändigeren Methode, falls erwünscht, schnell ausgetauscht werden.

Bei einem Polygonzug bestehend aus zwei Eckpunkten, ist der Polygonzug selber, also die Kante, die beste Approximation und wird somit nicht durch das Verfahren optimiert.

3.3 Nicht konvexe Polygone und Polygonzüge

In den vorangegangenen Kapiteln wurden ausschließlich konvexe Polygone und Polygonzüge berücksichtigt. Daher sollen nun nicht konvexe Polygone und Polygonzüge betrachtet werden. Diese lassen sich durch Zerlegen in die konvexen Teilsegmente auf die konvexen Polygonzüge zurückführen. In Abbildung 24 auf Seite 47 ist ein Beispiel für einen nicht konvexen Polygonzug gegeben. Der Graph des Polygonzuges beschreibt bis zum Punkt $P5$ eine Rechtskurve. An der Kante zwischen $P4$ und $P5$ vollzieht sich ein Richtungswechsel, so dass der Graph vom Punkt $P4$ bis zum Ende eine Linkskurve vollzieht. Wenn der Polygonzug nun in zwei Polygonzüge aufgeteilt wird, so dass der Polygonzug in der Mitte der Kante zwischen $P4$ und $P5$ am Punkt PZ geteilt wird, ergeben sich zwei konvexe Polygonzüge. Diese können durch das in Kapitel 3.2 erläuterte Verfahren approximiert werden.

Hier wird ein großer Vorteil aus der Tangentenwahl bei den Endpunkten der konvexen Polygonzüge deutlich. Die Bézierkurven, welche die einzelnen Teilsegmente approximieren, passen stetig aneinander. Das selbe Verfahren kann für nicht konvexe Polygone angewendet werden. Erst werden diese in die konvexen Teilsegmente zerlegt und anschließend werden die berechneten

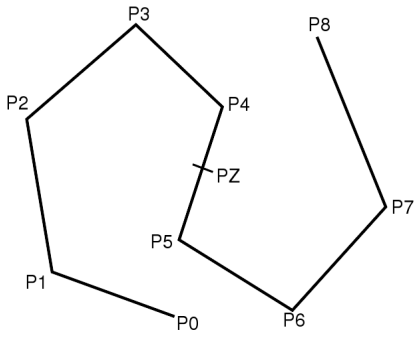


Abbildung 24: Zerlegen eines nicht konvexen Polygonzuges am Punkt PZ.

Bézierkurven wieder zusammengefügt.

Alternativ könnten Bedingungen aufgestellt werden, die ein Gleichungssystem zur Folge haben, das eine Bézierkurve beschreibt, die einen solchen Übergang von einer Rechts- in eine Linkskurve approximieren kann. Dies würde eine globale Approximation eines nicht konvexen Polygons oder Polygonzuges ermöglichen. Allerdings sollten diese Bedingungen einerseits mit zwei oder drei quadratischen Bézierkurven für eine solche Kante auskommen und gleichzeitig keine Sonderfälle erzeugen, die wiederrum durch komplexe weitere Bedingungen abgefangen werden müssen. Das Aufstellen solcher Bedingungen ist schwer möglich. Daher wird von dieser Vorgehensweise Abstand genommen.

Auf der anderen Seite stellt die Approximation durch das Aufteilen eine durchaus akzeptable Möglichkeit dar. Eine Kante, an der der Übergang stattfindet wird bei den quadratischen Bézierkurven zwar durch zwei elementare Bézierkurven approximiert, da mit einer quadratischen elementaren Bézierkurve kein Richtungswechsel dargestellt werden kann. Trotzdem wurde das Minimum an Komplexität, das bei der Approximation solcher Kanten möglich ist, erreicht.

4 Approximation durch kubische Bézierkurven

In den vorangegangenen Kapiteln erfolgte die Approximation von Polygonzügen nur durch quadratische Bézierkurven. In diesem Kapitel soll die Approximation von Polygonen und Polygonzügen durch kubische Bézierkurven betrachtet werden. Diese wird auf die Approximation durch quadratische Bézierkurven zurückgeführt.

In Abbildung 25 ist eine Polygonkante mit den Kontrollpunkten und den Strecken zwischen den Kontrollpunkten einer quadratischen Bézierkurve gezeigt. Da für die kubischen Bézierkurven ebenfalls die gleichen Bedingungen für die Tangenten in den Endpunkten $Pt1$ und $Pt2$ gelten sollen, wie für die quadratischen Bézierkurven, müssen sich die zwei Kontrollpunkte $PM1$ und $PM2$ der kubischen Bézierkurve auf der Strecke zwischen den Kontrollpunkten einer entsprechenden quadratischen Bézierkurve befinden, welche die selbe Kante approximieren würde. Die Verschiebung der Punkte auf dieser Strecke ist ein Freiheitsgrad, der in den Gleichungen für jeden mittleren Kontrollpunkt hinzukommt.

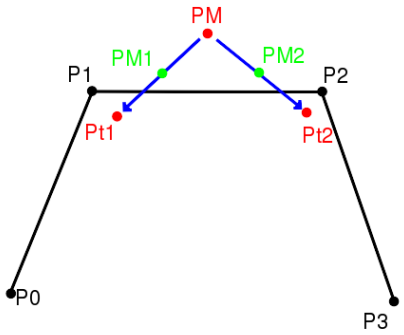


Abbildung 25: Die Kontrollpunkte einer kubischen und einer quadratischen Bézierkurve.

Wenn sich die beiden mittleren Kontrollpunkte nahe der Kontrollpunkte am Ende der elementaren Bézierkurve befinden (siehe Abbildung 26), ergibt

sich eine sehr flache Bézierkurve, die sich sehr nahe an der Polygonkante befindet und somit den Polygonzug besser approximiert. Dieser Freiheitsgrad stellt ein gravierendes Problem dar, wenn er in die Gleichungen aufgenommen wird. Da der Algorithmus auf Minimierung der Abstände ausgerichtet wurde, wird er die mittleren Kontrollpunkte mit den Kontrollpunkten an den Enden gleichsetzen, wenn ihm die Wahl überlassen wird, und somit die Abstände auf den Wert Null bringen, aber dadurch keine stetige Bézierkurve erzeugen. Ein weiteres Problem ist, dass je weiter sich die mittleren Kontrollpunkte den Endpunkten näheren, desto enger wird die Kurve von einer elementaren Bézierkurve zur nächsten.

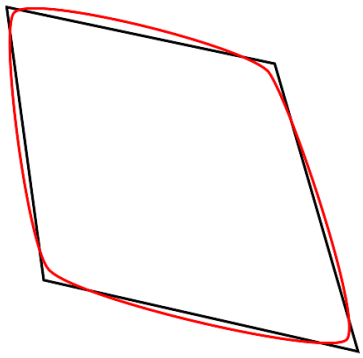


Abbildung 26: Kontrollpunkte nahe der Ecken.

Um diesen Freiheitsgrad am besten nutzen zu können, wurde die Entscheidung auf den Benutzer übertragen[24] . Dazu wird ein Parameter k eingeführt, der festlegt, an welcher Stelle die Kontrollpunkte für die kubische Bézierkurve liegen sollen. Der Parameter beschreibt des Verhältnis, in das die Strecke zwischen Endpunkt und Kontrollpunkt einer quadratischen Bézierkurve durch den Kontrollpunkt einer entsprechenden kubischen Bézierkurve geteilt wird. Je kleiner das Verhältnis ist, desto näher liegen die Kontrollpunkte der kubischen Bézierkurve an den Endpunkten, desto besser ist die Approximation

[24] Weitere Erläuterungen können dem später folgendem Kapitel 5.2 entnommen werden.

und desto enger ist die Kurve an den Übergängen.

Für die kubischen Bézierkurven muss eine Änderung in den Gleichungen vorgenommen werden. Die Gleichungen für die quadratische Bézierkurve werden durch folgende Gleichungen der kubischen Bézierkurve ausgetauscht.

$$B01(x) := Pt1(t_1) + x \cdot k \cdot s_1 \cdot V1 \tag{12}$$

$$B02(x) := (1-x) \cdot (Pt1(t_1) + k \cdot s_1 \cdot V1) + x \cdot (Pt2(t_2) - k \cdot r_2 \cdot V2) \tag{13}$$

$$B03(x) := Pt2(t_2) - k \cdot r_2 \cdot V2 + x \cdot k \cdot r_2 \cdot V2 \tag{14}$$

$$B11(x) := B01(x) + x \cdot (B02(x) - B01(x)) \tag{15}$$

$$B12(x) := B02(x) + x \cdot (B03(x) - B02(x)) \tag{16}$$

$$C(x) := B11(x) + x \cdot (B12(x) - B11(x)) \tag{17}$$

Die übrigen Gleichungen bleiben bestehen. Alle Gleichungen können äquivalent zu den Gleichungen bezüglich der quadratischen Bézierkurven umgeformt werden und es ergeben sich wieder vier entsprechende Gleichungen. Die vierte Gleichung wird wiederum in den Gleichungssystemen verwendet. Die Kriterien, die bei den konvexen Polygonen für die Endkanten aufgestellt wurden, können ebenfalls ohne Änderung verwendet werden. Die Werte der Parameter $d_{1,1}$ und $d_{n-1,2}$ werden ebenso wie bei der Approximation durch quadratische Bézierkurven auf Null gesetzt, nur der Parameter k wird bei der Berechnung der elementaren Bézierkurven der Endkanten berücksichtigt.

Alternativ wäre eine Neuaufstellung von Bedingungen für die Endkanten denkbar, so dass die Parameter $d_{1,1}$ und $d_{n-1,2}$ nicht auf den Null gesetzt werden müssen. Dies folgt daraus, dass die kubischen Bézierkurven, im Gegensatz zu den quadratischen, für Richtungswechsel genutzt werden können. Somit ist es möglich, Bedingungen aufzustellen, die einer Bézierkurve entsprechen würden, welche die Endkanten approximiert, wie es in Abbildung 27 auf Seite 51 zu sehen ist. Wenn nun aber die nicht konvexen Polygone unter Verwendung dieser Bedingungen approximiert werden, führt dies zu dem Auftreten von oszillierendem Verhalten an den Übergängen (siehe Abbildung 28 auf Seite 51). Dies ergibt aber keine gute Approximation.

Die Polygonzüge, die aus drei Punkten bestehen, werden durch eine kubische Bézierkurve approximiert, indem die mittleren Kontrollpunkte beide

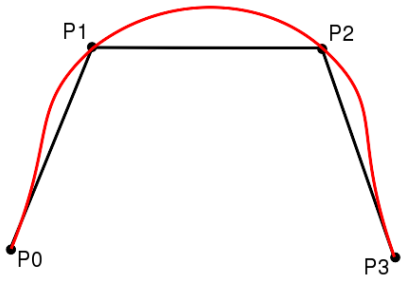


Abbildung 27: Variante des Approximierens der Endkanten.

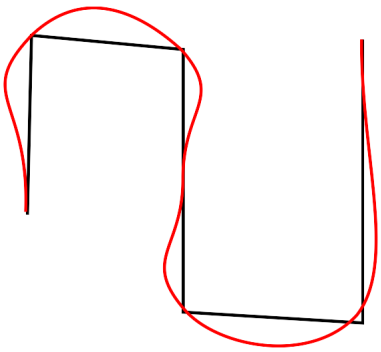


Abbildung 28: Oszillierendes Verhalten am Übergang.

mit dem mittleren Punkt gleichgesetzt werden. Die kubische Bézierkurve ist dadurch näher an den Polygonkanten, als die quadratische Bézierkurve mit den gleichen Vorteilen und es müssen keine zusätzlichen Berechnungen durchgeführt werden (siehe Abbildung 29).

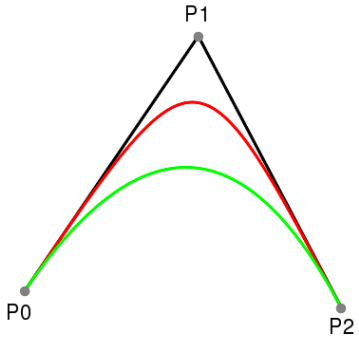


Abbildung 29: Kubische Bézierkurve (blau) und quadratische Bézierkurve (grün), die einen Polygonzug mit drei Eckpunkten approximieren.

Die nicht konvexen Polygone und Polygonzüge werden bei der Approximation auf die konvexen Polygonzüge zurückgeführt. Beim Zusammensetzen der einzelnen konvexen Segmente gibt es im Gegensatz zu der Approximation durch quadratische Bézierkurven eine Alternative. Da die kubischen Bézierkurven zur Darstellung eines Richtungswechsels herangezogen werden können, muss eine Kante, die einen solchen Übergang enthält, nicht durch zwei kubische Bézierkurven approximiert werden. In Abbildung 30 auf Seite 53 ist eine solche Kante mit den entsprechenden Kontrollpunkten gegeben. Die zwei elementaren Bézierkurven können zu einer elementaren Bézierkurve zusammengefasst werden, indem die mittleren drei Kontrollpunkte entfernt werden. In Abbildung 31 auf Seite 53 ist das Ergebnis gezeigt. Dies führt zu einer Verminderung der Komplexität und des Aufwands der Darstellung. Der Grad der Approximation wird dabei aber nicht wesentlich verschlechtert. Trotz dieser Vorteile wird dieses Verfahren in dieser Arbeit aber nicht verwendet, da es in Kombination mit der Vorgehensweise bei der Approxima-

tion von Polygonzügen die nur drei Kontrollpunkte besitzen, nicht verwendet werden kann.

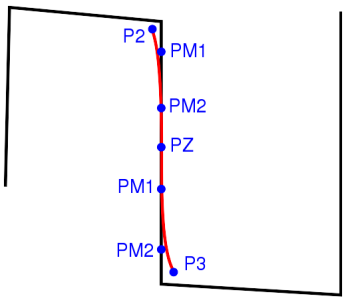


Abbildung 30: Übergang mit allen Kontrollpunkten.

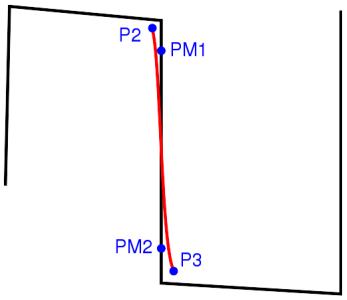


Abbildung 31: Übergang mit entfernten Kontrollpunkten.

Nachdem die Approximation von Polygonen und Polygonzügen nun sowohl durch quadratische als auch durch kubische Bézierkurven gezeigt wurde, soll ein Vergleich dieser beiden Möglichkeiten durchgeführt werden. Die Verwendung von kubischen Bézierkurven führt zu einem harmonischerem Aussehen der berechneten Bézierkurven. Damit geht allerdings ein erheblich größerer Aufwand bei der Darstellung einher. Dies mag bei einzelnen Polygonen keine deutlich negativen Auswirkungen haben, doch wenn die Anzahl an Polygonen größer ist, kann die Rechenkapazität schnell an ihre Grenzen

stoßen. Zur Verdeutlichung soll hier ein praktisches Beispiel gegeben werden. Eine Fläche der Größe der Bundesrepublik Deutschland, soll auf dem Bildschirm dargestellt werden. Angenommen ein Raster an Messpunkten, das auf diesen Gebiet liegt, aus deren Werten Isolinien berechnet wurden, hat eine Dichte von 10 mal 10 Kilometern. Sollten die Polygone, welche dann die Isolinien wiedergeben, alle dargestellt werden, summiert sich der Rechenaufwand pro elementarer Bézierkurve, also pro Kante der Polygone, schnell zu einem immensen Wert. Bei der Approximation durch quadratische Bézierkurven ist der Aufwand pro Kante geringer. Allerdings sind sie nicht in der Lage die Polygone und Polygonzüge so gut zu approximieren. Die Wahl zwischen den beiden Möglichkeiten der Approximation sollte also genau abgewogen und auf die jeweilige Situation abgestimmt sein. Die Approximation vieler Polygone durch kubische Bézierkurven ist nur dann möglich, wenn die Rechenkapazität, die zur Darstellung nötig ist, bereitgestellt werden kann. Darüber hinaus sollten auch die nötige Genauigkeit der Darstellung berücksichtigt werden. Wenn keine besondere Genauigkeit notwendig ist, bietet sich die Verwendung von quadratischen Bézierkurven an.

5 Die Auswirkungen der Parameter

In diesem Abschnitt der Arbeit sollen die Auswirkungen, die sich aus der Wahl des Wertes der einzelnen Parameter ergeben in den Blick genommen werden. Diese Werte müssen vor der Approximation festgesetzt werden, da sie die Grundlage der Approximation der Kurve bilden.

5.1 Die Orte der Abstandsmessung

Eine elementare Bézierkurve ist hier eine Funktion $C : [0; 1] \rightarrow \mathbb{R}^2$. Ein Punkt dieser Bézierkurve ist somit festgelegt durch einen Parameter $x \in [0; 1]$ (siehe Gleichung 1). Für die Berechnung einer elementaren Bézierkurve C_i, werden zwei Abstände ermittelt, deren Orte der Messung durch zwei Parameter x_1 und x_2 festgelegt sind, die für jede Kante K_i zwei Punkte $C_i(x_1)$ und $C_i(x_2)$ festsetzen. Da die zugehörigen Parameter $d_{i,1}$ und $d_{i,2}$, die den Abstand zur Kante K_i wiedergeben, nicht negativ sein dürfen, hat die Wahl der Werte der Parameter x_1 und x_2 folgenden Einfluss auf die Bézierkurve C_i.

In den Abbildungen auf Seite 56 sind vier Polygone und vier verschiedene Möglichkeiten der Wahl der Parameter x_1 und x_2 gegeben.

In der ersten Abbildung haben die Parameter x_1 und x_2 den Wert Eins und Null. Das hat zur Folge, dass die Punkte $C_i(x_1)$ und $C_i(x_2)$ mit den Endpunkten der Bézierkurve identisch sind. Da die Parameter $d_{i,1}$ und $d_{i,2}$ nicht negativ werden dürfen, müssen sich diese Punkte mindestens auf der Kante K_i befinden oder außerhalb des Polygons[25], was aber aufgrund der Parameter t_i und t_{i+1} nicht möglich ist, da diese nicht negativ sein dürfen. Somit ist die Bézierkurve, die berechnet wurde, die Anfangslösung des Optimierungsverfahrens. Dies kann aber nur bei konvexen Polygonen zu einer korrekten Approximation führen. Bei einem Polygonzug führt dies an den Endkanten zu einem nicht stetigen Übergang, da die Parameter t_i in den Eckpunkten den Wert Null annehmen, was an den Endkanten nicht auftreten darf[26]. Für Po-

[25] Vgl. Gleichung 9

[26] Vgl. Kapitel 3.2.

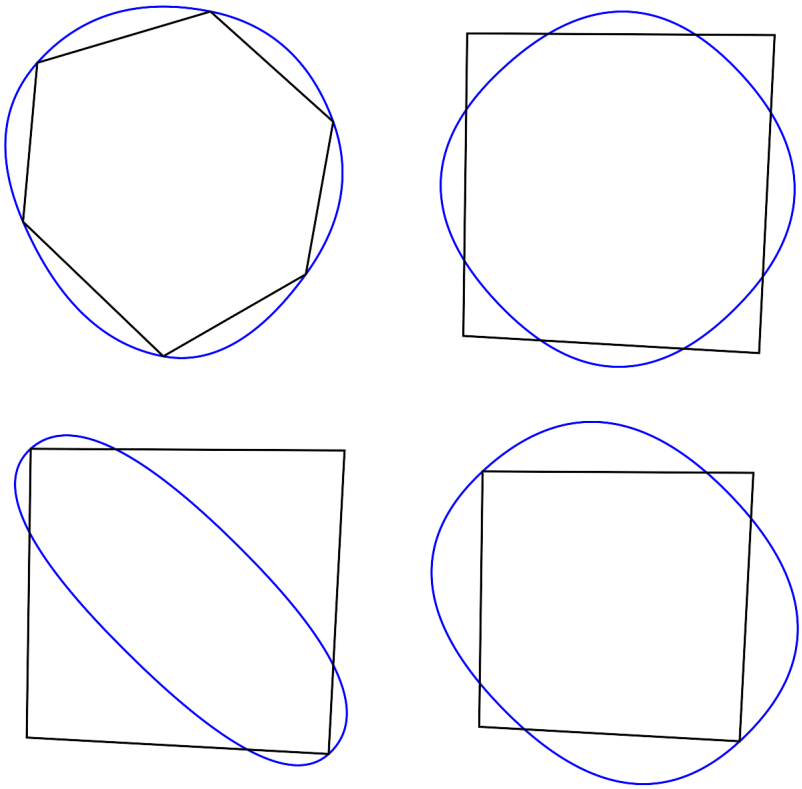


Abbildung 32: Beispiele für die Wahl der Parameter x_1 und x_2

lygonzüge müssen die Parameter deshalb unbedingt ungleich Eins und Null sein.

In der zweiten Abbildung wurden die Werte der Parameter weiter in die Mitte des Intervalls $[0;1]$ verschoben. Die berechnete Bézierkurve befindet sich nun weiter im Inneren des Polygons. Die Einschränkung, dass die Parameter $d_{i,j}$ nicht negativ sein dürfen, führt hier zu einer gleichmäßigen Approximation des Polygons. Je weiter die Werte der Parameter x_1 und x_2 in Innere des Intervalls verschoben werden, desto mehr kann die Bézierkurve ins Innere des Polygons verschoben werden.

Die dritte Abbildung zeigt die sich ergebende Bézierkurve, wenn die Werte der Parameter gleich gewählt werden. Das Polygon wurde im Vergleich zu den vorhergehenden Bézierkurven nicht gleichmäßig approximiert. Die Parameter verlieren bei der gleichmäßigen Verteilung der Abstände an Einfluss. Die Summe der Abstände ist hier geringer, aber die Varianz der Abstände ist wesentlich höher. Auf den Fall der Approximation von Isolinien übertragen, zeigt sich Folgendes. Liegen die berechneten Polygone, nahe aneinander, kann eine hohe Varianz zur Überschneidung von Bézierkurven führen (siehe Abbildung 33 auf Seite 58). Eine scheinbar schlechtere Approximation, die eine geringere Varianz besitzt, verringert das Risiko einer solchen Überschneidung. Andererseits kann eine Wahl der Parameter, die zu nahe an den Grenzen liegt zu anderweitigen Überschneidungen führen (siehe ebenfalls Abbildung 33).

In der vierten Abbildung ist gezeigt, wie sich eine Wahl der Parameter auswirkt, bei der beide Werte nahe dem Wert Null sind. Da die Parameter x_1 und x_2 Einfluss auf die Parameter t_i nehmen, führt diese Wahl der Werte der Parameter x_1 und x_2 dazu, dass der Wert des Parameters t_i klein bleibt. Als Konsequenz ergibt sich, da die Werte nicht gleichmäßig auf dem Intervall $[1;0]$ verteilt sind, dass die gleichmäßige Approximation verloren geht.

Anhand dieser Beispiele muss der Benutzer die Wahl der Parameter auf die gegebene Situation abstimmen. Dabei ist zwischen geringer Abweichung und geringer Varianz zu wählen. Im Test der Implementation, haben sich die Werte $\frac{1}{5}$ und $\frac{4}{5}$, bewährt.

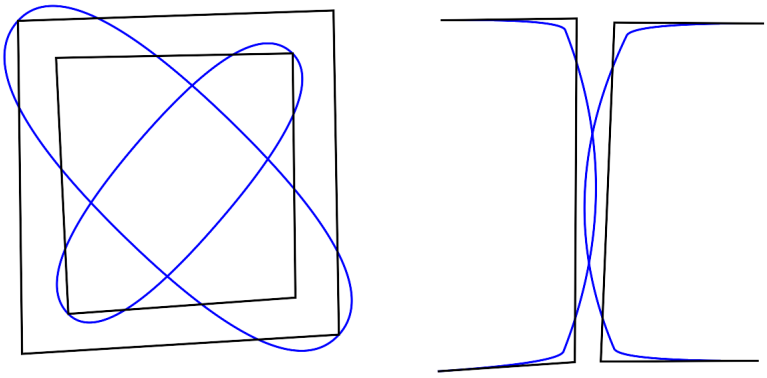


Abbildung 33: Hohe Varianz der Abstände

5.2 Modifikator der kubischen Bézierkurven

Der Parameter $k \in [0; 1]$ beschreibt[27], wie nahe sich die beiden mittleren Kontrollpunkte der kubischen Bézierkurve an den Endpunkten der Bézierkurve befinden. In der Abbildung 34 auf Seite 59 sind Beispiele für die möglichen Auswirkungen der Wahl des Parameters gezeigt.

In der ersten Abbildung wurde der Wert des Parameters auf Eins gesetzt. Daraus ergibt sich eine Bézierkurve, die weit von den Kanten und Eckpunkten des Polygons entfernt ist und das Polygon somit schlecht approximiert. In der folgenden Abbildungen wurde ein niedrigerer Wert für k gewählt. Die Bézierkurve nähert sich dann dem Polygon weiter an, so dass die Kurve in den Eckpunkten einen engeren Bogen vollzieht. Dadurch kommt es bei der berechneten Kurve zu dem Eindruck, dass die Bézierkurve den Verlauf des Polygons sehr genau wiedergibt und somit das Polygon gut approximiert (siehe zweite Abbildung).

Der Benutzer muss sich also zwischen einer größeren Abweichung und einer gleichmäßig gekrümmten Bézierkurve oder einer geringeren Abweichung und gegebenenfalls sehr engen Kurven entscheiden. Diese engen Kurven können das harmonische Aussehen der Bézierkurve deutlich verringern. Dies gilt

[27] Vgl. Kapitel 4

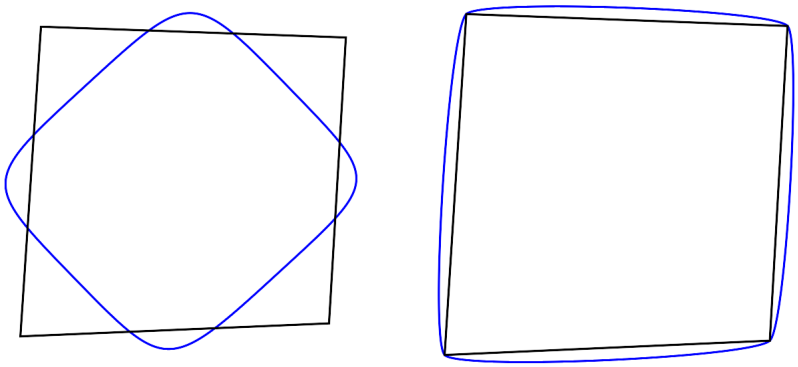


Abbildung 34: Beispiele für die Wahl des Parameters k

besonders für Kanten, die einen spitzen Winkel einschließen.

5.3 Die Vereinfachung von Polygonen

In Kapitel 3 wurde erläutert, dass zur Approximation eines Polygons oder Polygonzuges Kanten zu Gruppen zusammengefasst werden können, so dass für jede Gruppe eine elementare Bézierkurve verwendet wird. Auf diese Weise wird die Komplexität der gesamten Bézierkurve verringert. Die Gruppen müssen so gebildet werden, dass der Grad der Approximation sich durch das Bilden der Gruppen nicht wesentlich verschlechtert. Stattdessen sollte er sich sofern möglich sogar verbessert.

Die Zusammenfassung zu Gruppen kann vor der Approximation durch Zusammenfassen von geeigneten Nachbarkanten oder nach der Approximation durch vereinfachen der Bézierkurve, durch Ersetzen von benachbarten elementaren Bézierkurven durch eine elementare Bézierkurve, erfolgen. Das Zusammenfassen nach der Approximation hat den Vorteil, dass der Graph der Bézierkurve festgelegt ist und somit eine Vergrößerung der Abweichung nach der Zusammenfassung von Teilen der berechneten Bézierkurve gut festgestellt werden kann. Die Wahl der zusammengefassten Kanten kann dann, entsprechend dieser Erkenntnisse verändert werden. Der Nachteil ist allerdings, dass

das Optimierungsverfahren auf alle elementaren Bézierkurven angewendet wird. Dies kann bei einem konvexen Teilsegment mit vielen Kanten sehr aufwendig werden, wobei es anschließend zu einem Zusammenfassen von vielen elementaren Bézierkurven kommen kann. Dieser Aufwand kann durch das Zusammenfassen der Kanten vor der Approximation verhindert werden. Der Nachteil dieser Möglichkeit ist, dass sich das Zusammenfassen kaum auf den Verlauf der Bézierkurve beziehen lässt, da diese erst nach dem Optimieren feststeht. Somit können mögliche Auswirkungen auf die neue Bézierkurve durch das Bilden von Gruppen vorab nicht berücksichtigt werden.

Hier werden die Kanten vor der Approximation zu Gruppen zusammengefasst, um die Laufzeit der Approximation gering zu halten. Die Laufzeit und der Bedarf an Speicherplatz für die Berechnungen wachsen, bezogen auf die Anzahl der Kanten aufgrund des Gleichungssystems mindestens exponentiell, was im Fall von vielen kurzen Kanten zu einer nicht zu bewältigenden Datenmenge führen kann. Daher hat das Verringern der Abweichung durch das Zusammenfassen von Kanten, im Gegensatz zu der Verringerung des benötigten Speicherplatzes, eine untergeordnete Bedeutung.

Für die Art und Weise der Zusammenfassung von Kanten gibt es ebenfalls zwei Möglichkeiten. Bei der ersten Möglichkeit werden die Kanten so zu Gruppen zusammengefasst, dass für jede Gruppe eine elementare Bézierkurve aufgestellt werden kann, wobei jede Kante als Bedingung in die elementare Bézierkurve der Gruppe eingeht. Dieses Verfahren kann allerdings ebenfalls zu einer übermäßig großen Datenmenge führen. Bei der zweiten Möglichkeit werden die Kanten so zu Gruppen zusammengefasst, dass für die Approximation nur die äußeren beiden Punkte der Gruppe verwendet werden. Die Gruppe von Kanten muss dann so gewählt werden, dass die mittleren Punkte für die Approximation vernachlässigt werden können. Da die Verringerung des Speicherplatzes im Vordergrund steht, wird hier die zweite Möglichkeit verwendet.

Das zu approximierende Polygon kann mit dem gewählten Verfahren auf folgende Weise nach und nach vereinfacht werden. Für jeden Eckpunkt des

Polygons wird geprüft, ob er für die Approximation nötig ist, falls dies nicht der Fall ist, wird er vernachlässigt. Andernfalls wird zum nächsten Punkt übergegangen. Dieses Vernachlässigen entspricht der Bildung einer Gruppe aus den Kanten, zu denen der Eckpunkt gehört. Nachdem dieses Verfahren abgeschlossen ist, bleibt ein Polygon übrig, dass keine Punkte enthält, die nach vorgegebenen Kriterien überflüssig sind.Diese Kriterien sollen im Folgenden erläutert werden.

Ein Eckpunkt ist für die Approximation unnötig, wenn eine der zugehörigen Kanten zu kurz ist. Dies ist relativ zur gesamten Bildgröße, sowie zur gewünschten Auflösung des Bildes zu dem die berechnete Bézierkurve gehört, zu sehen. Angenommen die zu approximierenden Polygone sind auf einer Abbildung dargestellt mit 500 mal 500 Bildpunkten. Darüber hinaus sollen die Polygone Kanten besitzen, die nur 5 Bildpunkte lang sind. Eine Bézierkurve, die für diese Polygone berechnet wurde, die in der gleichen Auflösung dargestellt werden soll, ist an dieser Kante kaum von der Kante zu unterscheiden. Wenn die Kanten an den entsprechenden Punkt einen Winkel von ca. 180° einschließt, ändert sich der Graph des Polygons kaum, wenn der Punkt nicht berücksichtigt wird. Die an den Punkt gestellten Kriterien sind somit die Länge der zugehörigen Kanten, der Winkel im Eckpunkt und die verwendete Auflösung bei der Darstellung.

Der Wert, der herangezogenen wird, um zu prüfen, ob ein Polygon oder Polygonzug konvex ist, gibt auch Aufschluss darüber, ob diese Kriterien in einem Eckpunkt erfüllt werden. Dieser Wert (siehe Seiten 11ff) entspricht dem Flächeninhalt der Fläche, der von den Vektoren aufgespannt wird. Wenn dieser Flächeninhalt sehr gering ist, schließen die Vektoren einen kleinen Winkel ein oder bzw. und haben eine geringe Länge. Aus einem komplexen Polygon, das viele kurze Kanten besitzt kann durch das Vernachlässigen derjenigen Eckpunkte, deren Kanten einen Grenzwert unterschreiten, ein einfacheres Polygon berechnet werden, dessen Bézierkurve keine große Abweichung vom Originalpolygon aufweist.

Auf Seite 37 sind drei verschiedene Beispiele für die Wahl des Parameters

gezeigt.

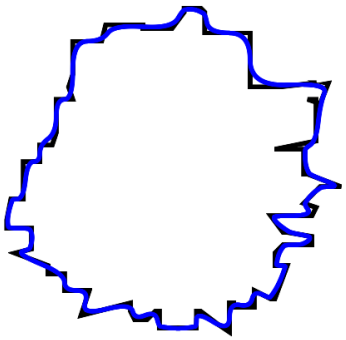


Abbildung 35: Bild mit 65x60 Pixeln, Wert des Parameters 10.

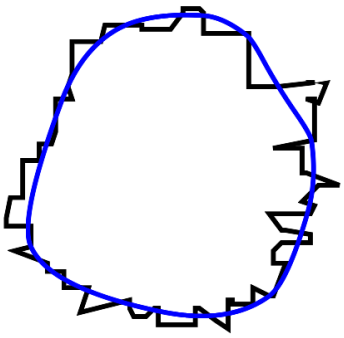


Abbildung 36: Bild mit 65x60 Pixeln, Wert des Parameters 100.

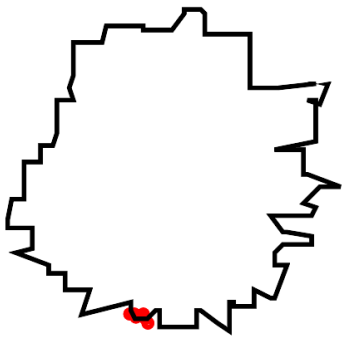


Abbildung 37: Bild mit 65x60 Pixeln, Wert des Parameters 1000, fast alle Punkte (rot) wurden entfernt.

6 Die Implementation

6.1 Beschreibung der Klassen

Die Klassen, welche eine Implementation darstellen, die zu einem gegebenen Polygon oder Polygonzug die zugehörige Bézierkurve berechnet, sind in der Programmiersprache Java[28] geschrieben. Die Strukturierung in Java wird durch Klassen und Pakete vorgenommen. Die Implementation umfasst zwei Pakete. Das erste Paket `poly2bezier` enthält zwei Klassen. Die eine approximiert Polygone durch quadratische Bézierkurven, die andere approximiert sie durch kubische Bézierkurven. Das zweite Paket mit Namen `poly2bezier.material` enthält Klassen, die primär Hilfsmittel für die Klassen im Paket `poly2bezier` sind. Zuerst sollen nun die Hilfsklassen und anschließend die Hauptklassen der Implementation vorgestellt werden[29].

6.1.1 `poly2bezier.material.Punkt`

Die Klasse `poly2bezier.material.Punkt` ist zur Verknüpfung mit anderen Klassen erforderlich. Die vorhandenen Hauptklassen benötigen als Eingabe die Punkte des Polygonzuges, der approximiert werden soll. Die Punkte müssen in Form von Referenzen auf Instanzen dieser Klasse übergeben werden.

Eine Instanz der Klasse `Punkt` stellt einen Punkt in der Euklidischen Ebene dar. Die Instanz besitzt als Instanzvariablen zwei Gleitkommazahlen, welche die x- und die y-Koordinate des Punktes wiedergeben. Ein Punkt kann seine Position in der Euklidischen Ebene nicht ändern, deswegen gibt es keine Methode die Koordinaten zu verändern.

Die Implementation eines Punktes umfasst zwei Instanzmethoden, die spätere Berechnungen erleichtern. Eine Instanz der Klasse `Punkt` kann den Abstand, definiert durch die Euklidische Norm, zwischen dem Punkt, den diese repräsentiert, und einem anderen Punkt berechnen und wiedergeben.

[28] Version 1.5.0, siehe auch http://java.sun.com/j2se/1.5.0/

[29] Weiterführende Informationen können der Java Dokumentation, die auf der beigefügten CD-ROM vorliegt.

Die andere Methode berechnet einen Punkt auf der Geraden, die durch den Punkt und einen übergebenen Punkt geht. Der Rückgabewert wird durch folgende Formel beschrieben, wobei der Parameter x an die Methode neben dem Punkt P_{other} übergeben werden muss.

$$P_{this} + x \cdot (P_{other} - P_{this})$$

In der Klasse `Punkt` wurden drei wichtige Methoden implementiert, um eine Textrepräsentation eines Punktes zu erhalten. Die erste Methode gibt die Koordinaten des Punktes durch ein Komma getrennt zurück. Die zweite Methode rundet die Koordinaten vor der Ausgabe auf eine ganze Zahl und die dritte Methode rundet die Koordinaten auf eine feste Anzahl von Stellen hinter dem Komma. In der hier vorgestellten Implementation ist die Anzahl an Stellen auf 5 gesetzt.

6.1.2 `poly2bezier.material.Converter`

Die Gleichungen, die zur Approximation hergeleitet wurden, sind in Abhängigkeit von den Punkten des Polygonzuges aufgestellt worden. Im Laufe der Entwicklung der Implementation, sollten die Formeln, welche sehr lang und komplex sind, einfach und schnell auszutauschen sein, um flexibel auf Änderungen reagieren zu können. Die Formeln wurden aufgrund der Komplexität durch das CAS Derive berechnet. Derive hat die Möglichkeit, die Formeln in einer lesbaren Form in einer Textdatei zu speichern, so dass die Formeln auf einfache Weise durch Kopieren in ein Java Programm aufgenommen werden können. Die Klasse `Converter` wurde implementiert, um diese Formeln auszulesen. Bei den Gleichungen, die in das Gleichungssystem eingetragen werden, müssen die Koeffizienten, die vor den Parametern stehen in Abhängigkeit von den Punkten berechnet werden. Dabei ändert sich bei jeder Gleichung die Formel für die Berechnung nicht[30].

Diese Klasse kann aus übergebenen Formeln, die in einem Character-Array gespeichert sind, den Wert zurückgeben, den diese Formel repräsen-

[30] siehe Gleichung 11

tiert. Voraussetzung dafür ist natürlich, dass der Wert aller Variablen übergeben wurde. Des Weiteren verfügt diese Klasse über eine Methode, die zum Runden von Gleitkommazahlen verwendet wird. Der `Converter` beherrscht außerdem die vier Grundrechenarten, sowie das Berechnen aller Potenzen. Das Umwandeln der Formeln erfolgt durch einen rekursiven Aufruf innerhalb der Klasse, um die Komplexität der Berechnung zu verringern. Der rekursive Aufruf wird für jeden Teil der Formel, der in Klammern zusammengefasst ist, gestartet.

6.1.3 `poly2bezier.material.BezierFormeln`

Diese Klasse ist eine Sammlung aller Formeln, die von den Hauptklassen benötigt werden. Jede Formel kann hier, sofern nötig, zentral geändert werden oder, falls umfangreiche Änderungen erforderlich sind, komplett ausgetauscht werden. Die vom CAS Derive berechneten Formeln können fast ohne Änderung in diese Klasse übernommen werden, so dass ein Austausch ohne Probleme vollzogen werden kann. Die einzige Änderung, die in den Formeln vorgenommen werden muss, ist der Austausch des Namens der Funktion `SQRT` durch den Namen `S`.

Die Formeln sind in Klassenvariablen gespeichert. Für jede Formel wurde zur Vereinfachung eine eigene Klassenmethode implementiert, der nur die benötigten Daten übergeben werden. Diese Daten werden entsprechend der Formel für den `Converter` konvertiert und dann an den `Converter` mitsamt der Formel übergeben, um den angeforderten Wert berechnen zu können.

Besondere Aufmerksamkeit ist der richtigen Benennung der Variablen zu schenken, da eventuelle Fehler in Formeln nicht automatisch korrigiert werden können. Eine augenscheinlich richtige Formel muss nicht in allen Fällen ein gewünschtes Ergebnis zurückgeben und kann starke Nebeneffekte verursachen, was u. a. beim Suchen der optimalen Lösung zu Problemen führen kann.

6.1.4 poly2bezier.material.Gleichungssystem

Diese Klasse bildet das Herzstück der gesamten Optimierung. Die Gleichungen, welche die Bézierkurven beschreiben, werden in Instanzen dieser Klasse eingetragen, um die bestmögliche Bézierkurve berechnen zu können.

Diese Instanzen besitzen alle nötigen Methoden, um den Punkt zu finden, welcher dieser Bézierkurve entspricht. Ein Gleichungssystem kann dazu über eine entsprechende Methode Eintrag für Eintrag gefüllt werden. Nachdem alle Einträge gesetzt wurden, kann über eine weitere Methode der Suchalgorithmus begonnen werden. Der Aufruf führt zu einer Reihe von Umformungen und Austauschschritten, welche das gesamte Gleichungssystem verändern. Die Berechnung erfolgt wie in den vorherigen Kapiteln[31] beschrieben.

6.1.5 poly2bezier.material.PolyMaterial

In dieser Klasse sind kleine Methoden gesammelt, die von beiden Hauptklassen verwendet werden. Falls zu den beiden Hauptklassen weitere Klassen hinzugefügt werden sollten, müssen diese Methoden nicht neu implementiert werden, sondern können von allen verwendet werden. Da es sich um Klassenmethoden handelt, ist eine Sammlung der Methoden in einer abstrakten Klasse, von der die Hauptklassen erben nicht von großem Vorteil.

Eine Methode dieser Klasse zerlegt nicht konvexe Polygone und Polygonzüge in ihre konvexen Teilstücke, so dass diese gesondert approximiert werden können. Eine weitere Methode prüft, ob ein Punkt für das Approximieren des Polygons berücksichtigt werden muss oder nicht. Des Weiteren ist eine Methode implementiert worden, die entscheidet, ob zwei Kanten eine Links- oder eine Rechtskurve beschreiben. Die letzte Methode kopiert Punkte aus einem Vektor in ein Array.

[31] Vgl. Kapitel 3.1.3

6.1.6 Die Hauptklassen

Die beiden Klassen `Punkte2KubBezier` und `Punkte2QuadBezier` die im Paket `poly2bezier` sind, stellen die beiden Hauptklassen der Implementation dar. Diese Klassen beinhalten Klassenmethoden, welchen die Punkte eines Polygons oder Polygonzuges übergeben werden. Aus diesen werden dann die Kontrollpunkte der optimalen Bézierkurve berechnet.

Diese Berechnung erfolgt durch Aufrufen einer Klassenmethode, welche den Methodenaufruf an die entsprechende Methode weiterleitet, welche ein konvexes Polygon oder ein konvexen Polygonzug approximiert. In den beiden Methoden werden die jeweiligen Gleichungsysteme erzeugt, die Optimierung gestartet, sowie die Kontrollpunkte der jeweiligen quadratischen oder kubischen Bézierkurve berechnet. Die Klassen beinhalten zudem die Methode `public void main()`, die es ermöglicht die Datei direkt auf der Kommandozeile zu starten. Dazu wird die Datei eingelesen, in der die Koordinaten der Punkte eines Polygons stehen und eine implementierte Methode erzeugt eine SVG Datei mit der berechneten Bézierkurve[32].

[32]Die Spezifikationen in der die Bézierkurve in der SVG Datei gespeichert wurde sind dem Buch von Eisenberg [2] entnommen.

6.2 Der Ablauf in der Implementation

Im vorherigen Kapitel wurde eine allgemeine Beschreibung der Klassen gegeben. Im Folgenden soll der Ablauf des Programms innerhalb der Klassen, wenn ein Polygon oder Polygonzug approximiert wird, am Beispiel eines nicht konvexen Polygons, das durch eine kubische Bézierkurve approximiert werden soll, erläutert werden. Die Abbildung 38 auf der Seite 70 veranschaulicht die Aufrufe der Methoden.

Die Punkte des Polygons müssen zunächst in Instanzen der entsprechenden Klasse `Punkt`[33] gespeichert werden. Diese sollen dann in einer Instanz der Klasse `Vector`[34], in der vorgegebenen Reihenfolge vorliegen. Um dieses Polygon zu approximieren, wird die Methode `poly2Kurve()`[35] aufgerufen. Diese Methode benötigt das `Vector`[36] Objekt[37], in dem die Punkte gespeichert sind, drei Gleitkommazahlen, sowie einen Booleschen Wert, der angibt, ob es sich um einen Polygonzug oder ein Polygon handelt. Die ersten beiden Gleitkommazahlen geben die Stellen der Bézierkurve an, an denen die Abstände zur Kante gemessen werden sollen, die für die Optimierung benötigt werden. In den Gleichungen wurden die Werte x_1 und x_2 genannt. Die dritte Gleitkommazahl gibt an, wie nahe sich die mittleren Kontrollpunkte an den äußeren Kontrollpunkten befinden. In den Gleichungen wurde dieser Wert k genannt.

In dieser Methode wird nun geprüft, welche Punkte in dem übergebenen `Vector` für die Approximation überflüssig sind. Alle Eckpunkte, deren Winkel und deren zugehörige Kanten einen bestimmten Faktor unterschreiten, werden bei der Approximation nicht berücksichtigt. Dazu wird mithilfe der Methode `canDelete()`[38] geprüft, ob ein Eckpunkt berücksichtigt werden

[33] Aus dem Paket `poly2bezier.material`

[34] Aus dem `Paket java.util`

[35] In der Klasse `poly2bezier.Punkte2KubBezier`

[36] Aus dem Paket `java.util`

[37] Es muss eigentlich die „Referenz auf dieses `Vector` Objekt“ heißen. Auf diese Feinheiten wurde u. a. auf Grund der Lesbarkeit des Textes verzichtet.

[38] In der Klasse `poly2bezier.material.PolyMaterial`

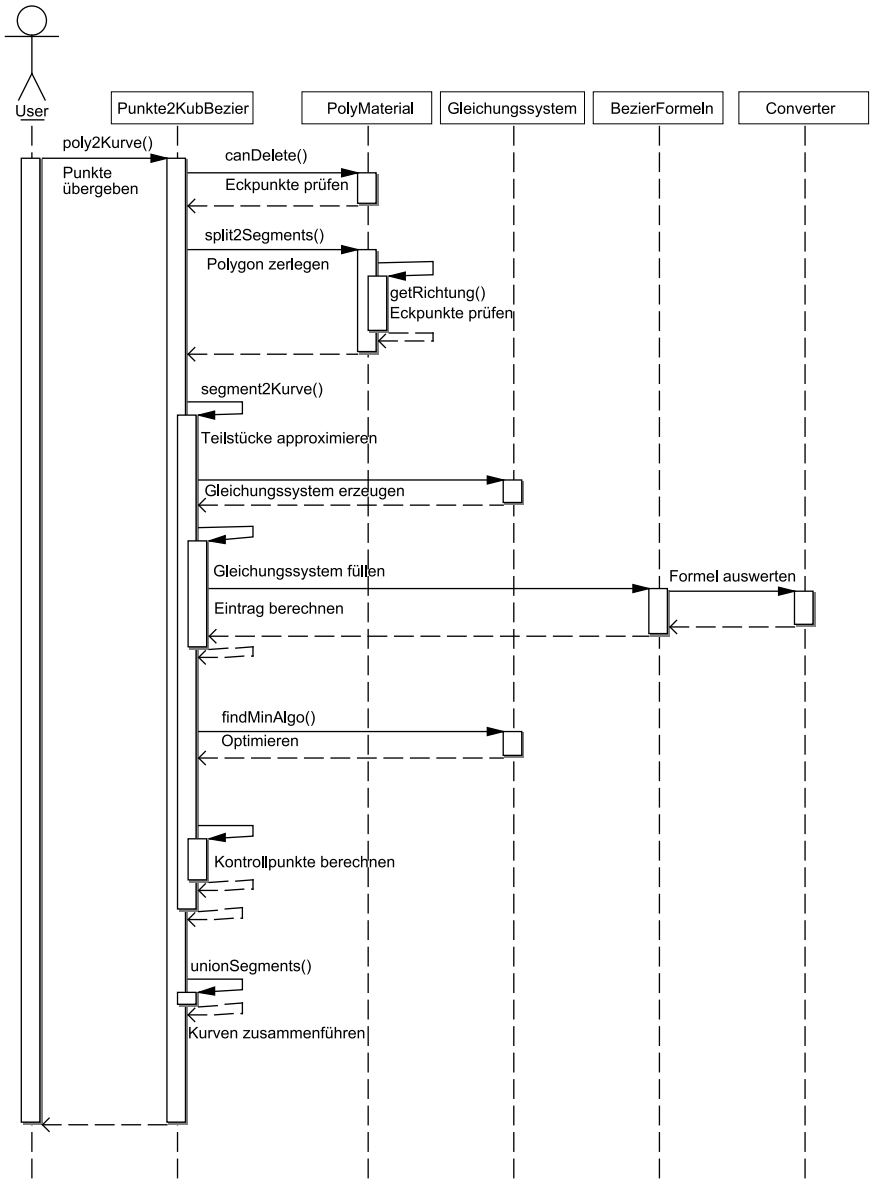


Abbildung 38: Sequenzdiagramm zum Ablauf der Implementation. Aus Gründen der Übersichtlichkeit unvollständig dargestellt.

muss. Dazu werden lediglich der Eckpunkt, sowie die beiden Nachbarpunkte benötigt.

Anschließend wird das Polygon in seine konvexen Polygonzüge zerlegt. Dies geschieht mithilfe der Methode `split2Segments()`[39]. Dafür wird das `Vector` Objekt, sowie die Information, ob es sich um ein Polygon oder Polygonzug handelt, benötigt. Diese Information ist bedeutsam, weil Polygone an der zusätzlichen Kante, welche den ersten und den letzten Punkt verbindet, ebenfalls geteilt werden können.

Ein Richtungswechsel wird daran erkannt, dass der Rückgabewert der Methode `getRichtung()`[40], welche von der vorherigen Methode aufgerufen wird, das Vorzeichen ändert. Um die Richtung zu bestimmen, wird von der Methode lediglich ein Eckpunkt und die beiden benachbarten Eckpunkte benötigt. Die Richtung wird nach der Gleichung auf Seite 11 bestimmt.

An die Methode `segment2Kurve()`[41] werden dann die konvexen Polygonzüge einzeln übergeben, welche die Kontrollpunkte der berechneten Bézierkurve zurückgibt. Durch diese Methode wird zu Beginn geprüft, ob die Punkte im Uhrzeigersinn verlaufen, also einer Rechtskurve folgen. Falls dies nicht der Fall sein sollte, wird die Reihenfolge umgekehrt, um eine Rechtskurve zu erzeugen. Der Sonderfall, dass es sich um einen Polygonzug mit weniger als 4 Eckpunkten handelt, wird abgefangen und gesondert bearbeitet. Die übrigen Fälle werden wie vorher theoretisch erläutert optimiert.

Dazu wird ein Gleichungssystem erzeugt und für jede Kante die entsprechende Gleichung eingetragen. Die einzelnen Einträge in der Matrix werden mithilfe der Klasse `BezierFormeln`[42] berechnet. Dazu werden lediglich die Werte x_i und k sowie die vier Eckpunkte, welche die zur Gleichung gehörige Kante und die beiden Nachbarkanten definieren, benötigt. Die Klasse `BezierFormeln`[43] berechnet die Einträge mit Hilfe der Klasse `Converter`[44],

[39] In der Klasse `poly2bezier.material.PolyMaterial`
[40] In der Klasse `poly2bezier.material.PolyMaterial`
[41] In der Klasse `poly2bezier.Punkte2KubBezier`
[42] Aus dem Paket `poly2bezier.material`
[43] Aus dem Paket `poly2bezier.material`
[44] Aus dem Paket `poly2bezier.material`

welche die Formeln verarbeitet. Die zwei Parameter $d_{1,1}$ und $d_{n-1,2}$, die für die korrekte Approximation der Endkanten sorgen, entfallen, da sie den Wert Null haben.

Um den Punkt im Polyeder zu berechnen, welcher die optimale Bézierkurve wiedergibt, wird bei der Instanz des erzeugten Gleichungssystems die Methode `findMinAlgo()`[45] aufgerufen. Diese Methode gibt dann den Punkt des Polyeders zurück, der die Parameter enthält, welche die optimale Bézierkurve wiedergeben.

Diese Methode formt das Gleichungssystem wie in Kapitel 3.1.3 beschrieben solange um, bis es den optimalen Punkt im Polyeder gefunden hat. Wenn der optimale Punkt im Polyeder gefunden wurde, wird dieser in Form eines Arrays zurückgegeben.

In der Methode `segment2Kurve()`[46] wird aus den Parametern, die in dem Array vorliegen, die Bézierkurve berechnet. Das geschieht ebenfalls unter Zuhilfenahme der Klasse `BezierFormeln`[47], welche die Formeln zur Berechnung der Kontrollpunkte aus den Parameter enthält.

Nachdem alle konvexen Teilstücke approximiert wurden, werden diese mit Hilfe der Methode `unionSegments()`[48] wieder zusammengefügt. Die entstandene Bézierkurve kann nun einfach in einem vektorgraphischen Format gespeichert werden.

6.3 Anleitung zur Nutzung der Implementation

An dieser Stelle soll erläutert werden, wie die Implementation genutzt werden kann. Dazu gibt es drei Möglichkeiten. Die erste Möglichkeit ist die graphische Oberfläche, die durch die Klasse `PunktBezierHersteller` zur Verfügung gestellt wird, zu nutzen [49]. Diese Klasse ist in den vorherigen Kapiteln nicht

[45] In der Klasse `poly2bezier.material.Gleichungssystem`

[46] In der Klasse `poly2bezier.Punkte2KubBezier`

[47] Aus dem Paket `poly2bezier.material`

[48] In der Klasse `poly2bezier.Punkte2KubBezier`

[49] Die graphische Oberfläche ist ebenfalls im Internet als Applet, mit eingeschränkten Funktionen, aufrufbar. http://www-lehre.inf.uos.de/ kkleinek/Bachelor/Applet.html

beschrieben worden, da sie keine Methoden bereitstellt, die zur Optimierung dienen. Sie ist lediglich implementiert worden, um die Optimierung zu verdeutlichen und zu testen. Die zweite Möglichkeit besteht darin die `main()`-Methode der Hauptklassen zu nutzen und Daten aus einer Datei auszulesen und anschließend die berechnete Bézierkurve in eine SVG Datei zu schreiben. Als dritte Möglichkeit kann die Methode `poly2Kurve()` der entsprechenden Hauptklasse direkt genutzt werden. Diese gibt die berechneten Kontrollpunkte der berechneten Bézierkurve zurück.

6.3.1 Die graphische Oberfläche

Diese Klasse kann über ihre Methode `main()` mit folgendem Aufruf auf der Kommandozeile gestartet werden:

```
java poly2bezier.PunkteBezierHersteller
```

Die Methode erzeugt eine graphische Oberfläche, die in Abbildung 39 zu sehen ist. Auf dieser sind die Eingabefelder und die den Feldern zugeordneten Parameter gezeigt. Die Funktion der einzelnen Felder wird nun, unter Zuhilfenahme der Abbildung, erläutert.

Die Oberfläche ist in einen unteren und in einen oberen Teil eingeteilt. Im unteren Teil befindet sich auf der linken Seite die Zeichenfläche, auf der die zu approximierenden Polygone eingegeben werden. Auf der rechten Seite werden die berechneten Bézierkurven dargestellt. Die Darstellung erfolgt durch die Klassen des Apache-Batik Projektes[50] (hier im weiterenn Batik-Viewer genannt). Die Darstellung kann durch folgende Befehle angepasst werden.

- STRG + Linke Maustaste: Zoom-Fenster aufziehen bei Mausbewegungen
- STRG + Rechte Maustaste: Drehen des Bildes bei Mausbewegungen
- Shift + Linke Maustaste: Verschieben des Bildes bei Mausbewegungen

[50] vgl. http://xml.apache.org/batik

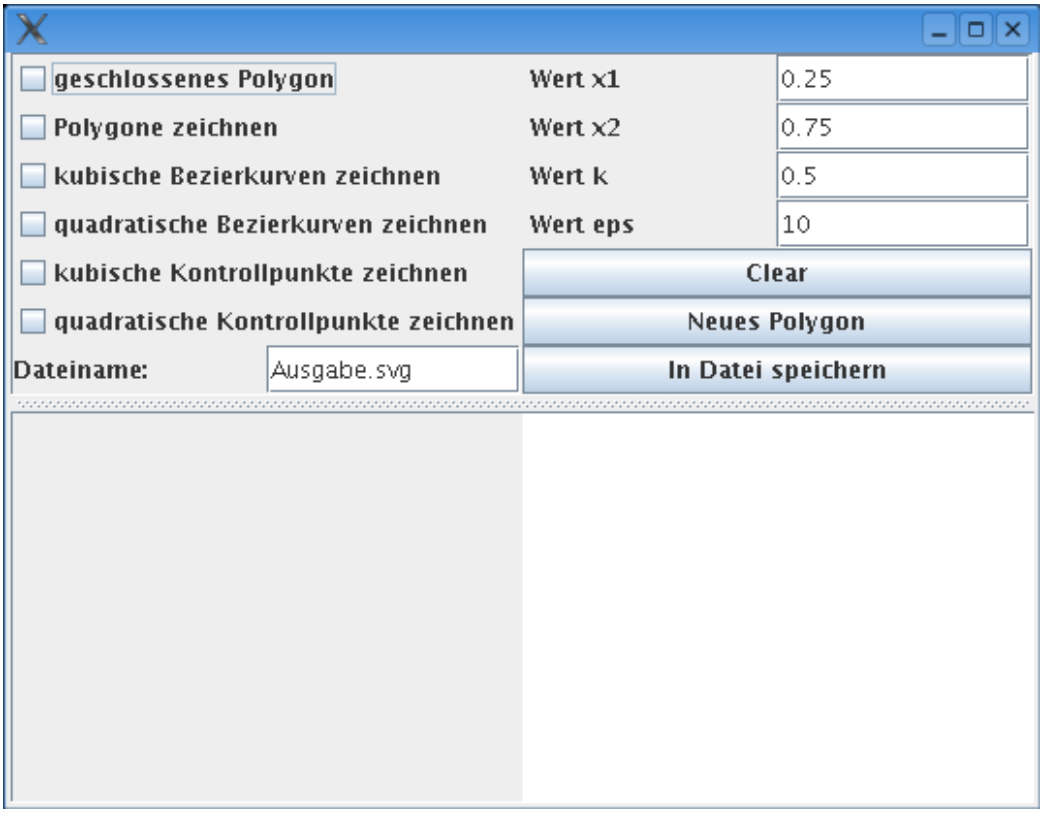


Abbildung 39: Die graphische Oberfläche

- Shift + Rechte Maustaste + Mausbewegung nach oben –> Herauszoomen
- Shift + Rechte Maustaste + Mausbewegung nach unten –> Hineinzoomen

Im oberen Bereich der graphischen Oberfläche befinden sich auf der linken Seite die Checkboxen, anhand derer dieElementee ausgewählt werden können, die im Batik-Viewer dargestellt werden sollen. Dabei kann mit der obersten Checkbox „geschlossenes Polygon“ausgewählt werden, ob die eingegebenen Polygonzügen geschlossen sein sollen, also Polygone darstellen. Auf der rechten Seite befinden sich die Eingabefelder für die Parameter, die für die Approximation benötigt werden. Darunter befinden sich drei Knöpfe. Der erste ist zum zurücksetzen der Zeichenfläche und des Batik-Viewers. Mit demKlickk des zweiten Buttons wird auf der Eingabefläche ein neues Polygon begonnen, das zusammen mit den vorherigen Polygonen approximiert wird. Der dritte Button schreibt dieSVG-Dateii, die im Batik-Viewer angezeigt wird, in eine Datei. Der Dateiname kann im Eingabefeld, das links daneben liegt eingegeben werden.

6.3.2 Daten aus Dateien auslesen

In dem Fall, dass die Polygone bzw.Polygonzüge in einer Datei vorliegen, können diese Daten direkt ausgelesen werden, vorausgesetzt, dass die Datei einen bestimmten Aufbau hat. Die Approximation der Polygone kann dann, durch folgenden Aufruf gestartet werden. Sei der Dateiname, der Datei in der die Polygone gespeichert sind, `Punkte.txt` und der der SVG Datei `Bezier.svg`, dann ergeben sich folgende Aufrufe auf der Kommandozeile[51].

```
java poly2bezier.Punkte2QuadBezier Punkte.txt Bezier.txt
```

oder für kubische Bézierkurven:

```
java poly2bezier.Punkte2KubBezier Punkte.txt Bezier.txt
```

[51]Der Implementation sind vier Beispieldateien beigefügt. Sie tragen die Endung `.dat`.

Die Datei in der die Eckpunkte gespeichert sind, muss dazu folgenden Aufbau haben.

1. In der ersten Zeile stehen die Parameter, die zur Approximation nötig sind, in folgender Reihenfolge, durch einen Doppelpunkt (‚: ‘) getrennt.
   (a) Das Wort `closed`, falls es sich um Eckpunkte von Polygonen handelt, oder `open`, falls es sich um Polygonzüge handelt.
   (b) Den Wert des Parameters x_1
   (c) Den Wert des Parameters x_2
   (d) Die Höhe des Bildes in der SVG Datei
   (e) Die Breite des Bildes in der SVG Datei
   (f) Den Wert des Parameters t
   (g) Den Wert des Parameters k (Nur wenn kubische Bézierkurven erzeugt werden sollen!)
2. In der zweiten Zeile muss die Auflistung der Koordinaten der Eckpunkte beginnen. In jeder Zeile darf nur ein Eckpunkt stehen und die Koordinaten müssen durch ein Komma getrennt sein. Der Beginn der Eckpunkte eines neuen Polygons oder Polygonzugs wird durch eine Leerzeile angezeigt.
3. Dezimalzahlen werden mit einem Punkt geschrieben und nicht, wie in Deutschland üblich, mit einem Komma.

6.3.3 Die direkte Übergabe der Punkte

Jede der Hauptklassen beinhaltet eine Methode `poly2Kurve()`. Wenn die Applikation in ein anderes Programm eingebunden werden sollte, ist es sinnvoll, wenn diese Methode direkt aufgerufen wird und keines der vorherigen Verfahren genutzt wird. Die Punkte, die an die Implementation übergeben werden, müssen Instanzen der in Kapitel 6.1.1 beschriebene Klasse sein und

werden für jedes Polygon und jeden Polygonzug separat der Reihenfolge entsprechend in einer Instanz der Klasse `Vector` gespeichert. Nun müssen alle `Vector`-Objekte, die ein Polygon oder einen Polygonzug darstellen getrennt in zwei `Vector`-Objekten gespeichert werden. Diese beiden Objekte können dann jeweils an die Methode `poly2Kurve()` der entsprechenden Klasse übergeben werden. Die genaue Methodensignatur kann der Java Dokumentation der Implementation entnommen werden.

7 Schluss

Ich habe in dieser Arbeit zum Thema „Approximation von Polygonzügen durch Bézierkurven“ eine Möglichkeit vorgestellt, wie Polygone bestmöglich durch Bézierkurven angenähert werden können. Die von mir entwickelte Möglichkeit ist ein Optimierungsverfahren, das iterativ die bestmögliche Lösung berechnet. Eine Alternative wäre es eine Bézierkurve aufgrund der Eckpunkte direkt zu berechnen und diese dann als die Bézierkurve zu deklarieren, die das Polygon bestmöglich approximiert. Im Laufe der Entwicklung der Implementation hat sich aber gezeigt, dass auf diese Weise zwar eine Bézierkurve schnell aufgestellt werden kann, die sich dem Polygon gut annähert, diese Bézierkurve dann mit großer Wahrscheinlichkeit nicht das Optimum darstellt.

Die durch meine Implementation berechneten Bézierkurven zeichnen sich dadurch aus, dass sie innerhalb meiner aufgestellten Kriterien das Optimum darstellen. Ein Ziel meiner Implementation war eine möglichst genaue Approximation zu erreichen. Dies kann zu sehr engen Kurven der Bézierkurven führen, so dass das harmonische Aussehen darunter leiden kann. Ich hätte ein Programm entwickeln können, das Bézierkurven erzeugt, die ein sehr harmonisches Bild erzeugen, aber das hätte zu großen Abstrichen in der Genauigkeit der Approximation geführt und Genauigkeit gehört für mich zu den Kernelementen einer guten Approximation.

Um ein harmonisches Aussehen mit einer genauen Approximation zu verbinden, wäre u.a. eine deutlich höhere Anzahl an elementaren Bézierkurven nötig. Daraus folgt allerdings unweigerlich eine Erhöhung der Komplexität der Bézierkurve, die wiederum zu einem hohen Rechenaufwand bei der Darstellung führt. Eine optimale Approximation ist meiner Meinung nach die beste Näherung mit geringster Komplexität, so dass ich mich auch gegen diese Möglichkeit entschieden habe.

Meine Implementation bietet zudem durch die zu wählenden Parameter eine effektive Möglichkeit auf das Aussehen der Bézierkurve Einfluss zu nehmen. Der Nutzer kann die Bézierkurve berechnen lassen, die für seine

Zwecke am geeignetsten ist. Die Implementation mag außerdem durch die aufwendige Berechnung den Anschein erwecken, dass sie die Bézierkurven zu aufwendig berechnet. Das Finden der besten Bézierkurve ist diesen größeren Aufwand meines Erachtens aber Wert, da man sich durch eine bestmögliche Approximation gegen starke Nebeneffekte schützt. Diese, wie zum Beispiel das Überschneiden von Bézierkurven, können mit anderen Bézierkurven in einer Abbildung auftauchen und das Gesamtbild stark schädigen.

Meine Implementation verbindet also eine geringe Komplexität mit möglichst großer Genauigkeit. Das Auftreten von Nebeneffekten wird verhindert und dem Nutzer wird die Möglichkeit eingeräumt die Approximation den jeweiligen Anforderungen anzupassen. Somit wird dem Nutzer mit dieser Implementation ein flexibles Werkzeug in die Hand gegeben.

A Gleichungen

A.1 Quadratische Bezierkurven

A.1.1 Erste Gleichung

$$
\begin{aligned}
&e1 \cdot ((p11^2 - 2 \cdot p11 \cdot p21 + p12^2 - 2 \cdot p12 \cdot p22 + p21^2 + p22^2) \\
&\cdot (v12 \cdot v21 - v11 \cdot v22)) \\
&= \\
&(x^2 \cdot (p11^2 \cdot (v11 \cdot v22 + v12 \cdot v21) - p11 \cdot (2 \cdot p12 \cdot (v11 \cdot v21 \\
&- v12 \cdot v22) + 2 \cdot p21 \cdot (v11 \cdot v22 + v12 \cdot v21) + 2 \cdot p22 \cdot (v12 \cdot v22 \\
&- v11 \cdot v21) + t1 \cdot (v11 \cdot (2 \cdot v21 \cdot vt12 - v22 \cdot vt11) \\
&- v12 \cdot v21 \cdot vt11) + t2 \cdot (v12 \cdot v21 \cdot vt21 - v11 \cdot (2 \cdot v21 \cdot vt22 \\
&- v22 \cdot vt21))) - p12^2 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&+ p12 \cdot (2 \cdot p21 \cdot (v11 \cdot v21 - v12 \cdot v22) + 2 \cdot p22 \cdot (v11 \cdot v22 \\
&+ v12 \cdot v21) - t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v21 \cdot vt12 \\
&- 2 \cdot v22 \cdot vt11)) + t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v21 \cdot vt22 \\
&- 2 \cdot v22 \cdot vt21))) + p21^2 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&- p21 \cdot (2 \cdot p22 \cdot (v11 \cdot v21 - v12 \cdot v22) + t1 \cdot (v12 \cdot v21 \cdot vt11 \\
&- v11 \cdot (2 \cdot v21 \cdot vt12 - v22 \cdot vt11)) + t2 \cdot (v11 \cdot (2 \cdot v21 \cdot vt22 \\
&- v22 \cdot vt21) - v12 \cdot v21 \cdot vt21)) - p22 \cdot (p22 \cdot (v11 \cdot v22 \\
&+ v12 \cdot v21) - t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v21 \cdot vt12 \\
&- 2 \cdot v22 \cdot vt11)) + t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v21 \cdot vt22 \\
&- 2 \cdot v22 \cdot vt21)))) - 2 \cdot x \cdot (p11 \cdot v11 + p12 \cdot v12 - p21 \cdot v11 \\
&- p22 \cdot v12) \cdot (p11 \cdot v22 - p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 \\
&+ t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) + t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21)) \\
&+ t1 \cdot (p11 \cdot vt11 + p12 \cdot vt12 - p21 \cdot vt11 \\
&- p22 \cdot vt12) \cdot (v11 \cdot v22 - v12 \cdot v21))
\end{aligned}
$$

A.1.2 Zweite Gleichung

$$
\begin{aligned}
& s1 \cdot (v12 \cdot v21 - v11 \cdot v22) \\
& = \\
& (p11 \cdot v22 - p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 \\
& + t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) + t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21))
\end{aligned}
$$

A.1.3 Dritte Gleichung

$$
\begin{aligned}
& r2 \cdot (v11 \cdot v22 - v12 \cdot v21) \\
& = \\
& (p11 \cdot v12 - p12 \cdot v11 - p21 \cdot v12 + p22 \cdot v11 \\
& + t1 \cdot (v12 \cdot vt11 - v11 \cdot vt12) \\
& + t2 \cdot (v11 \cdot vt22 - v12 \cdot vt21))
\end{aligned}
$$

A.1.4 Vierte Gleichung

$$
\begin{aligned}
&d1 \cdot (\sqrt{(p11^2 - 2 \cdot p11 \cdot p21 + p12^2 - 2 \cdot p12 \cdot p22 + p21^2 + p22^2)} \\
&\cdot (v12 \cdot v21 - v11 \cdot v22)) \\
&= \\
&(x^2 \cdot (2 \cdot p11^2 \cdot v12 \cdot v22 - p11 \\
&\cdot (2 \cdot p12 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&+4 \cdot p21 \cdot v12 \cdot v22 - 2 \cdot p22 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&+t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v21 \cdot vt12 - 2 \cdot v22 \cdot vt11)) \\
&-t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v21 \cdot vt22 - 2 \cdot v22 \cdot vt21))) \\
&+2 \cdot p12^2 \cdot v11 \cdot v21 + p12 \cdot (2 \cdot p21 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&-4 \cdot p22 \cdot v11 \cdot v21 + t1 \cdot (v11 \cdot (2 \cdot v21 \cdot vt12 - v22 \cdot vt11) \\
&-v12 \cdot v21 \cdot vt11) + t2 \cdot (v12 \cdot v21 \cdot vt21 - v11 \cdot (2 \cdot v21 \cdot vt22 \\
&-v22 \cdot vt21))) + 2 \cdot p21^2 \cdot v12 \cdot v22 - p21 \cdot (2 \cdot p22 \cdot (v11 \cdot v22 \\
&+v12 \cdot v21) - t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v21 \cdot vt12 \\
&-2 \cdot v22 \cdot vt11)) + t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \\
&\cdot (v21 \cdot vt22 - 2 \cdot v22 \cdot vt21))) + p22 \cdot (2 \cdot p22 \cdot v11 \cdot v21 \\
&+t1 \cdot (v12 \cdot v21 \cdot vt11 - v11 \cdot (2 \cdot v21 \cdot vt12 - v22 \cdot vt11)) \\
&+t2 \cdot (v11 \cdot (2 \cdot v21 \cdot vt22 - v22 \cdot vt21) - v12 \cdot v21 \cdot vt21))) \\
&-2 \cdot x \cdot (p11 \cdot v12 - p12 \cdot v11 - p21 \cdot v12 + p22 \cdot v11) \\
&\cdot (p11 \cdot v22 - p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 \\
&+t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) + t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21)) \\
&+t1 \cdot (p11 \cdot vt12 - p12 \cdot vt11 - p21 \cdot vt12 + p22 \cdot vt11) \\
&\cdot (v11 \cdot v22 - v12 \cdot v21))
\end{aligned}
$$

A.2 Kubische Bezierkurven

A.2.1 Erste Gleichung

$$
\begin{aligned}
&e1 \cdot ((p11^2 - 2 \cdot p11 \cdot p21 + p12^2 - 2 \cdot p12 \cdot p22 + p21^2 + p22^2) \\
&\cdot (v11 \cdot v22 - v12 \cdot v21)) \\
&= \\
&(x^3 \cdot (p11^2 - p11 \cdot (2 \cdot p21 - t1 \cdot vt11 + t2 \cdot vt21) + p12^2 - p12 \cdot (2 \cdot p22 \\
&- t1 \cdot vt12 + t2 \cdot vt22) + p21^2 + p21 \cdot (t2 \cdot vt21 - t1 \cdot vt11) + p22 \cdot (p22 \\
&- t1 \cdot vt12 + t2 \cdot vt22)) \cdot (3 \cdot k - 2) \cdot (v11 \cdot v22 - v12 \cdot v21) - 3 \cdot x^2 \\
&\cdot (k \cdot (p11^2 \cdot (2 \cdot v11 \cdot v22 - v12 \cdot v21) - p11 \cdot (p12 \cdot (v11 \cdot v21 \\
&- v12 \cdot v22) + 2 \cdot p21 \cdot (2 \cdot v11 \cdot v22 - v12 \cdot v21) + p22 \cdot (v12 \cdot v22 \\
&- v11 \cdot v21) + t1 \cdot (v11 \cdot (v21 \cdot vt12 - 2 \cdot v22 \cdot vt11) + v12 \cdot v21 \cdot vt11) \\
&- t2 \cdot (v11 \cdot (v21 \cdot vt22 - 2 \cdot v22 \cdot vt21) + v12 \cdot v21 \cdot vt21)) + p12^2 \\
&\cdot (v11 \cdot v22 - 2 \cdot v12 \cdot v21) + p12 \cdot (p21 \cdot (v11 \cdot v21 - v12 \cdot v22) \\
&+ 2 \cdot p22 \cdot (2 \cdot v12 \cdot v21 - v11 \cdot v22) + t1 \cdot (v11 \cdot v22 \cdot vt12 \\
&+ v12 \cdot (v22 \cdot vt11 - 2 \cdot v21 \cdot vt12)) - t2 \cdot (v11 \cdot v22 \cdot vt22 \\
&+ v12 \cdot (v22 \cdot vt21 - 2 \cdot v21 \cdot vt22))) + p21^2 \cdot (2 \cdot v11 \cdot v22 - v12 \cdot v21) \\
&- p21 \cdot (p22 \cdot (v11 \cdot v21 - v12 \cdot v22) - t1 \cdot (v11 \cdot (v21 \cdot vt12 \\
&- 2 \cdot v22 \cdot vt11) + v12 \cdot v21 \cdot vt11) + t2 \cdot (v11 \cdot (v21 \cdot vt22 \\
&- 2 \cdot v22 \cdot vt21) + v12 \cdot v21 \cdot vt21)) + p22 \cdot (p22 \cdot (v11 \cdot v22 \\
&- 2 \cdot v12 \cdot v21) - t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v22 \cdot vt11 \\
&- 2 \cdot v21 \cdot vt12)) + t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v22 \cdot vt21 \\
&- 2 \cdot v21 \cdot vt22)))) + (p11^2 - p11 \cdot (2 \cdot p21 - t1 \cdot vt11 + t2 \cdot vt21) + p12^2 \\
&- p12 \cdot (2 \cdot p22 - t1 \cdot vt12 + t2 \cdot vt22) + p21^2 + p21 \cdot (t2 \cdot vt21 - t1 \cdot vt11) \\
&+ p22 \cdot (p22 - t1 \cdot vt12 + t2 \cdot vt22)) \cdot (v12 \cdot v21 - v11 \cdot v22)) \\
&+ 3 \cdot k \cdot x \cdot (p11 \cdot v11 + p12 \cdot v12 - p21 \cdot v11 - p22 \cdot v12) \cdot (p11 \cdot v22 \\
&- p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 + t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) \\
&+ t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21)) + t1 \cdot (p11 \cdot vt11 + p12 \cdot vt12 - p21 \cdot vt11 \\
&- p22 \cdot vt12) \cdot (v12 \cdot v21 - v11 \cdot v22))
\end{aligned}
$$

A.2.2 Zweite Gleichung

$$
\begin{aligned}
& s1 \cdot (v12 \cdot v21 - v11 \cdot v22) \\
& = \\
& (p11 \cdot v22 - p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 \\
& + t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) + t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21))
\end{aligned}
$$

A.2.3 Dritte Gleichung

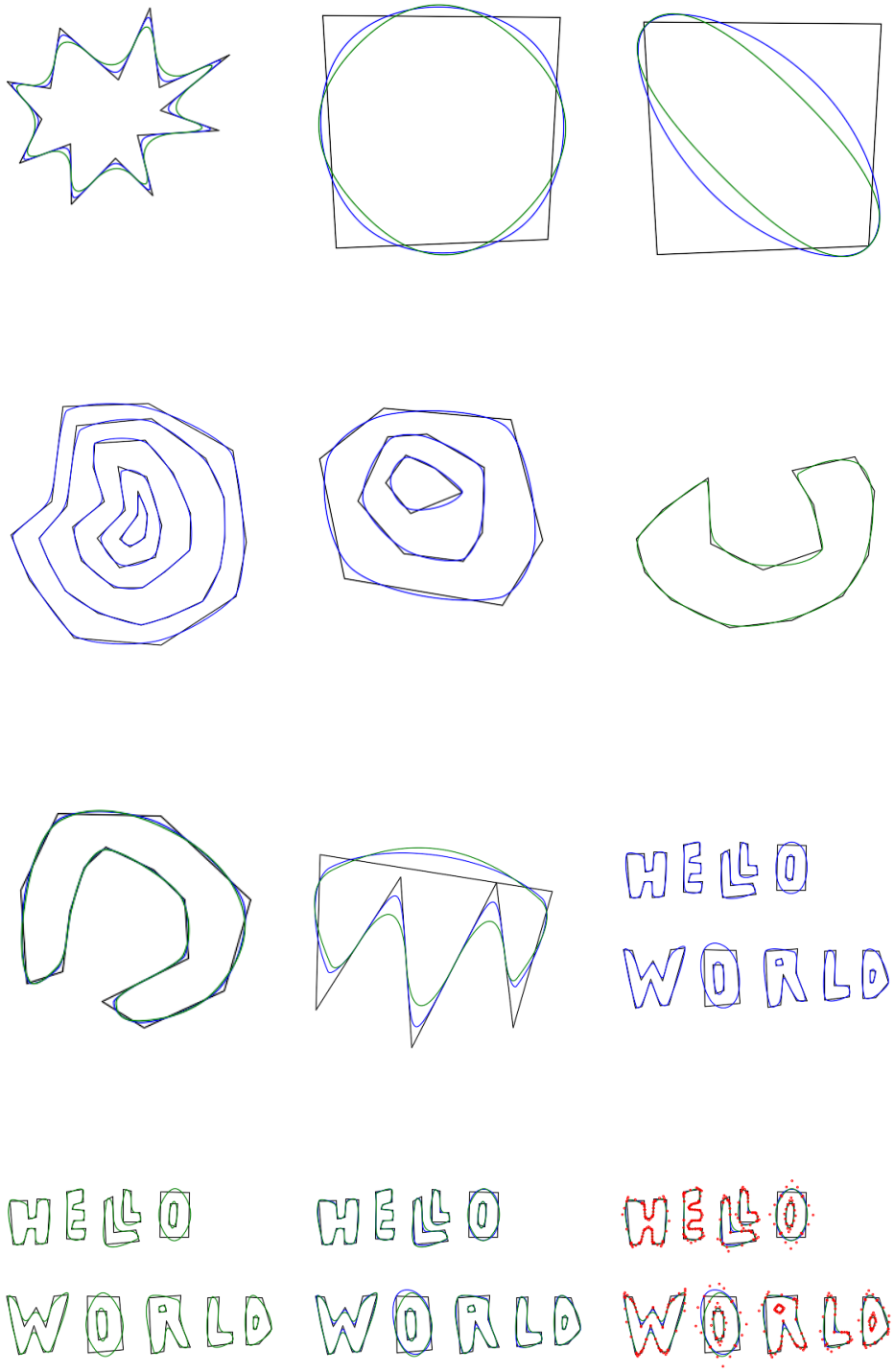
$$
\begin{aligned}
& r2 \cdot (v11 \cdot v22 - v12 \cdot v21) \\
& = \\
& (p11 \cdot v12 - p12 \cdot v11 - p21 \cdot v12 + p22 \cdot v11 \\
& + t1 \cdot (v12 \cdot vt11 - v11 \cdot vt12) + t2 \cdot (v11 \cdot vt22 - v12 \cdot vt21))
\end{aligned}
$$

A.2.4 Vierte Gleichung

$$
\begin{aligned}
&d1 \cdot (\sqrt{(p11^2 - 2 \cdot p11 \cdot p21 + p12^2 - 2 \cdot p12 \cdot p22 + p21^2 + p22^2)} \\
&\cdot (v11 \cdot v22 - v12 \cdot v21)) \\
&= \\
&(x^3 \cdot (3 \cdot k - 2) \cdot (p11 \cdot (t1 \cdot vt12 - t2 \cdot vt22) \\
&+p12 \cdot (t2 \cdot vt21 - t1 \cdot vt11) + p21 \cdot (t2 \cdot vt22 - t1 \cdot vt12) \\
&+p22 \cdot (t1 \cdot vt11 - t2 \cdot vt21)) \cdot (v11 \cdot v22 - v12 \cdot v21) \\
&-3 \cdot x^2 \cdot (k \cdot (p11^2 \cdot v12 \cdot v22 - p11 \cdot (p12 \cdot (v11 \cdot v22 \\
&+v12 \cdot v21) + 2 \cdot p21 \cdot v12 \cdot v22 - p22 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&-t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v22 \cdot vt11 - 2 \cdot v21 \cdot vt12)) \\
&+t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v22 \cdot vt21 - 2 \cdot v21 \cdot vt22))) \\
&+p12^2 \cdot v11 \cdot v21 + p12 \cdot (p21 \cdot (v11 \cdot v22 + v12 \cdot v21) \\
&-2 \cdot p22 \cdot v11 \cdot v21 + t1 \cdot (v11 \cdot (v21 \cdot vt12 - 2 \cdot v22 \cdot vt11) \\
&+v12 \cdot v21 \cdot vt11) - t2 \cdot (v11 \cdot (v21 \cdot vt22 - 2 \cdot v22 \cdot vt21) \\
&+v12 \cdot v21 \cdot vt21)) + p21^2 \cdot v12 \cdot v22 - p21 \cdot (p22 \cdot (v11 \cdot v22 \\
&+v12 \cdot v21) + t1 \cdot (v11 \cdot v22 \cdot vt12 + v12 \cdot (v22 \cdot vt11 \\
&-2 \cdot v21 \cdot vt12)) - t2 \cdot (v11 \cdot v22 \cdot vt22 + v12 \cdot (v22 \cdot vt21 \\
&-2 \cdot v21 \cdot vt22))) + p22 \cdot (p22 \cdot v11 \cdot v21 - t1 \cdot (v11 \cdot (v21 \cdot vt12 \\
&-2 \cdot v22 \cdot vt11) + v12 \cdot v21 \cdot vt11) + t2 \cdot (v11 \cdot (v21 \cdot vt22 \\
&-2 \cdot v22 \cdot vt21) + v12 \cdot v21 \cdot vt21))) + (p11 \cdot (t1 \cdot vt12 - t2 \cdot vt22) \\
&+p12 \cdot (t2 \cdot vt21 - t1 \cdot vt11) + p21 \cdot (t2 \cdot vt22 - t1 \cdot vt12) \\
&+p22 \cdot (t1 \cdot vt11 - t2 \cdot vt21)) \cdot (v12 \cdot v21 - v11 \cdot v22)) \\
&+3 \cdot k \cdot x \cdot (p11 \cdot v12 - p12 \cdot v11 - p21 \cdot v12 + p22 \cdot v11) \cdot (p11 \cdot v22 \\
&-p12 \cdot v21 - p21 \cdot v22 + p22 \cdot v21 + t1 \cdot (v22 \cdot vt11 - v21 \cdot vt12) \\
&+t2 \cdot (v21 \cdot vt22 - v22 \cdot vt21)) + t1 \cdot (p11 \cdot vt12 - p12 \cdot vt11 \\
&-p21 \cdot vt12 + p22 \cdot vt11) \cdot (v12 \cdot v21 - v11 \cdot v22))
\end{aligned}
$$

B Beispiele

Auf den folgenden Seiten folgen einige Beispiele von Berechnungen des Programms. Die Bilder wurden alle mit der graphischen Oberfläche erzeugt. Ein besserer Überblick kann aber durch das Applet auf der Seite http://www-lehre.inf.uos.de/ kkleinek/Bachelor/Applet.html gewonnen werden. Bei den Beispielen handelt es sich um Approximationen durch kubische (blau) und quadratische (grün) Bézierkurven.


HELLO
WORLD
HELLO
WORLD
HELLO
WORLD

Literatur

[1] **Trapp, Heinz Wilhelm (1996):**
Einführung in die Algebra: Vektorrechnung, algebraische Grundbegriffe, lineare Algebra
(Osnabrücker Studien zur Mathematik)
2. durchgesehene und verbesserte Auflage,
Osnabrück (Universitätsverlag Rasch)

[2] **Eisenberg, J. David (2002):**
SVG Essentials: Producing Scalable Vector Graphics with XML
First Edition,
Sebastopol (O'Reilly Media, Inc.)

[3] **Felgenhauer, Ursula:**
Kapitel 2: Lineare Optimierung
(PDF Dokument: Kap2.pdf)
Quelle:
http://www.math.tu-cottbus.de/~felgenh/lehre/so05/Kap2.pdf
(Stand: 11. August 2005)

[4] **Freie Online Enzyklopädie (Stand: 11. August 2005):**
de.Wikipedia.org
Verwendete Quellen:
http://de.wikipedia.org/wiki/Polygon
http://de.wikipedia.org/wiki/Polygonzug
http://de.wikipedia.org/wiki/Bezierkurve
http://de.wikipedia.org/wiki/De_Casteljau-Algorithmus