Von Punkt zu Punkt

Georg-Johann Lay

4. Juli 2009

Für eine Anzahl vorgegebener Punkte $P_1, \ldots, P_n$ soll eine geschlossene Kurve gefunden werden, welche die Punkte in der gegebenen Reihenfolge approximiert. Vom letzten Punkt P_n soll eine Linie zum Anfangspunkt P_1 zurückführen, und zwischen den Punkten soll die Kurve einen möglichst „natürlichen“ Verlauf zeigen.

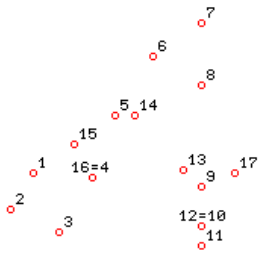
Bild 4 zeigt 17 Punkte, die wie bei einem Punkt-zu-Punkt-Bild miteinander zu verbinden sind, wobei der Linienschluß durch einen weiteren Punkt $P_{18} = P_1$ zustande käme.

1 Interpolation

Zunächst bestimmen wir eine Interpolante $\mathcal{I} : S_1 \to \mathbb{R}^d$ durch die n Vorgabepunkte $(P_k)_n \in \mathbb{R}^d$:

$$\mathcal{I}(t_k) = P_k \quad \text{für} \quad k = 1, \ldots, n$$

Bild 1



Als Interpolationsverfahren zur Berechnung von $\mathcal{I}$ bieten sich mehrere Standardverfahren an wie lineare Interpolation oder kubische Splines mit periodischer Anschlussbedingung. Das gewählte Interpolationsverfahren verwenden wir komponentenweise.

Tatsächlich hängt $\mathcal{I}$ nicht nur von dem gewählten Interpolationsverfahren und dem Parameter t ab, sondern auch von der Wahl der „Zeitpunkte“ t_k an denen $\mathcal{I}$ die Werte P_k annehmen soll:

$$\begin{aligned} \mathcal{I} :\ & S_1 \times S_1^n \to \mathbb{R}^d \\ & (t, (t_k)_n) \mapsto \mathcal{I}(t; t_1, \ldots, t_n) \end{aligned}$$

Zur Einfachheit schreiben wir die Interpolierte als $\mathcal{I}(t)$ wenn kein Bezug auf die Zeitpunkte $(t_k)_n$ genommem wird, ansonsten $\mathcal{I}_{(t_k)_n}(t)$ oder wie oben $\mathcal{I}(t; t_1, \ldots, t_n)$. Die Reihenfolge, in der die Zielpunkte durchlaufen werden, ergibt sich durch die Nebenbedingungen

$$t_k > t_{k-1} \quad \text{für} \quad k = 2, \ldots, n \tag{1}$$

1.1 Vorbesetzung der Zeitpunkte

Eine erste Belegung für die Zeitpunkte $(t_k)_n$ erhalten wir durch

$$t_k = \frac{\sum_{j=1}^{k} \delta(j)}{\sum_{j=1}^{n} \delta(j)} \quad \text{für} \quad k = 1, \ldots, n$$

etwa mit $\delta_\infty = \frac{1}{n}$ und damit $t_k = \frac{k}{n}$. Die Zeitpunkte werden dadurch in gleichen Abständen auf der Zeitachse angeordnet.

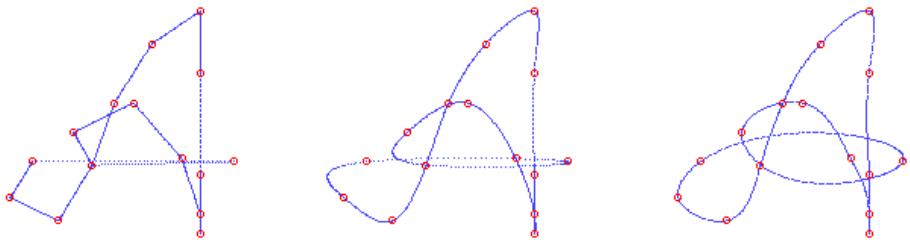
Eine weitere Vorbelegung ergibt sich mittels $\delta_0(j) = \|P_j - P_{j-1}\|$ mit $P_0 := P_n$, d.h. die Zeitabstände entsprechen den Abständen der Zielpunkte im Raum.

Diese beiden Belegungen lassen sich auch kombinieren, indem wir die d-dimensionalen Startpunkte in $S_1 \times \mathbb{R}^d$ einbetten und ihre (gleichen) Zeitabstände mit berücksichtigen. Dies ergibt schliesslich die Wahl $\delta_\sigma(j) = \|(\frac{\sigma}{n}, P_j - P_{j-1})\|$ mit einem Skalierungsfaktor σ. Die ersten beiden Wahlen für δ ergeben sich daraus für die Grenzfälle $\sigma \to \infty$ bzw. $\sigma = 0$.

Bild 2 zeigt den Einfluß des gewählten Interpolationsverfahrens und der Startbelegung auf $\mathcal{I}$.

Bild 2

Verschiedene Interpolationsverfahren und Startbelegungen für $(t_k)_n$



Linear und $\delta = \delta_\infty$ | Spline und $\delta = \delta_\infty$ | Spline und $\delta = \delta_0$

2 Approximation

Bei der Wahl der Interpolation $\mathcal{I}$ bleibt uns – unter Beachtung der Nebenbedingungen (1) – die freie Wahl der Zeitpunkte $(t_k)_n$. Da immer gilt $t_n = 1$, bleiben also $n-1$ Parameter, um die eingangs geforderte Eigenschaft der „natürlichen“ Approximation zu erfüllen. Wie oben zu sehen hat neben der Wahl des Interpolationsverfahrens auch die Wahl der Zeitpunkte einen starken Einfluß auf das Erscheinungsbild der erhaltenen Kurve.

Durch Variation der Zeitpunkte soll nun eine Approximation berechnet werden. Die Idee ist, die diskrete Fourier-Transformation (DFT) der Interpolanten als Bewertungskriterium zu verwenden. Dazu verwenden wir m Stützstellen $(\tau_j)_m$ mit

$$\tau_j = \frac{j}{m} \quad \text{für} \quad j = 1, \ldots, m \quad \text{und} \quad m \geqslant n$$

Das durch die Fourier-Transformation erhaltene trigonometrische Polynom $\mathcal{F}\{\mathcal{I}\}$ hat dann die Eigenschaft

$$\mathcal{F}\{\mathcal{I}\}(\tau_j) = \mathcal{I}(\tau_j) \quad \text{für} \quad j = 1, \ldots, m$$

und liefert damit eine Approximation der Vorgabepunkte:

$$\mathcal{F}\{\mathcal{I}\}(t_k) \approx P_k \quad \text{für} \quad k = 1, \ldots, n$$

Die Fourier-Transformierte berechnen wir komponentenweise für die d Koordinaten

$$\mathcal{F}\{\mathcal{I}\} = (\mathcal{F}_i\{\mathcal{I}\})_d \quad \text{mit} \quad \mathcal{F}_i\{\mathcal{I}\} = A_{i,0} + \sum_{k=1}^{m/2} A_{i,k} \cdot \cos\left(2\pi k t + \varphi_{i,k}\right) \tag{2}$$

und mit einer Zweierpotenz für m. Durch die Wahl einer Zweierpotenz für m lässt sich die diskrete Fourier-Transformation effizient durch das Verfahren der schnellen Fourier-Transformation (FFT) bestimmen.

2.1 Bewertung

Eine Approximation bewerten wir mit einer Bewertungsfunktion anhand der Amplituden in den DFTs ihrer d Komponenten:

$$\begin{aligned} \mu: \quad & S_1^n \to \mathbb{R}^{\geqslant 0} \\ & (t_k)_n \mapsto \sum_i \sum_k A_{i,k}^{\alpha} k^{\beta} \end{aligned}$$

mit zwei Parametern α und β. Gute Ergebnisse ergeben sich mit $\alpha \approx 0.9$ und $\beta = 1$. Durch die Wahl $\alpha < 1$ erhält der A-Terme eine Rechtskrümmung, so daß Amplituden, die ohnehin schon klein sind, zu Null tendieren. Für große Amplituden hingegen fällt eine Änderung weniger stark ins Gewicht. Zusammen mit dem k-Term, der zur Verteurung hoher Frequenzen führt, tendiert die Bewertung zu niedrigfrequenten Approximationen.

2.2 Variation der Zeitpunkte

Einen besseren Satz an Zeitpunkten erhalten wir durch Trial-and-Error, also durch ein Probierverfahren. Alles andere hat sich als nicht praktikabel erwiesen. Das liegt unter anderem daran, daß die Nebenbedingungen (1) weiterhin erfüllt sein müssen und μ aufgrund der Exponenten $\alpha < 1$ eine rauhe Struktur hat.

Zu Beginn des Suchverfahrens werden die Zeitpunkte gruppenweise verschoben. Die Größe der Gruppe wird ebenso wie die maximale Verschiebung, die die Gruppe erfährt, langsam reduziert. Nach jedem Schritt wird für den so erhaltenen Satz an Zeitpunkten $(t_k)_n$ seine Bewertung bestimmt und der Satz übernommen, wenn er sich als besser erweist. Wir suchen also eine Lösung für

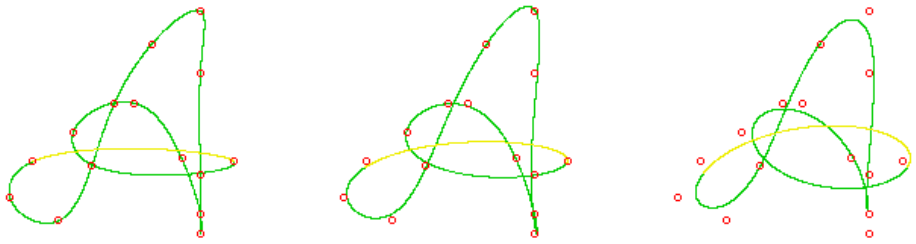
$$\mu(\mathcal{F}\{\mathcal{I}_{(t_k)_n}\}) \stackrel{!}{=} \text{Minimum}$$

Nachdem alle Terme mit $|A| < 0.06$ aus der DFT entfernt wurden, erhalten wir das Endergebnis.

Zunächst scheint die Minimierung von μ kaum einen Einfluß auf das Bild zu haben. Nach der Optimierung sieht es praktisch genauso aus wie das Startbild, das ganz rechts in Bild 2 zu sehen ist.

Bild 3

Phasen der Approximation für $m = 64$, $\alpha = 0.95$ und $\beta = 1$



μ-Minimum bestimmt mit Vorbelegung δ_0 und Splines wie in Bild 2

Kleine DFT-Terme vernachlässigt, mit μ-Optimierung

Kleine DFT-Terme vernachlässigt, ohne μ-Optimierung

Der Effekt der Optimierung zeigt sich erst nachdem kleine Amplitudenterme vernachlässigt wurden. Zwar bleiben sowohl mit der Optimierung als auch ohne sie noch 8 cos-Terme übrig,

aber die verbleibenden Terme der μ-optimierten Version zeigen eine deutlich bessere Approximation der Ausgangsdaten. Dies ermöglicht eine Verminderung der Datenmenge im Sinne (verlustbehafteter) Datenkompression.

2.3 Ergebnis

Nachdem alle Terme mit $|A| < 0.06$ aus der DFT entfernt wurden, bleiben noch 8 cos-Terme übrig. Das Punkt-zu-Punkt-Bild 4, das aus 17 Datenpaaren besteht, ist nun durch eine Kurve $f(t) \approx \mathcal{F}\{\mathcal{I}\}(t)$ mit noch 18 Koeffizienten angenähert.

Als Parametrisierung für unser Beispiel eines Punkt-zu-Punkt-Bildes des Buchstaben $\mathcal{A}$ erhalten wir

$$\begin{aligned} f_x(t) = \; & 0.20 + 0.45\cos(\tau + 3.09) + 0.44\cos(2\tau + 1.54) + 0.36\cos(3\tau + 2.06) \\ & + 0.20\cos(4\tau - 3.12) + 0.09\cos(5\tau - 2.86) \\ f_y(t) = \; & -0.20 + 0.22\cos(\tau - 2.08) + 0.46\cos(2\tau + 2.78) + 0.46\cos(3\tau + 0.55) \end{aligned}$$

mit $\tau = 2\pi t$ und $\tau \in [0, 5.55]$. Die Koeffizienten wurden auf die zweite Nachkommastelle gerundet. Der Linienschluss, also der gelb gezeichnete Rückstrich in den obigen Bildern, wird durchlaufen für $\tau \in [5.55, 2\pi]$.

Neben der guten Approximation der Vorgabepunkte, dem harmonischen Erscheinungsbild der Parametrisierung und der Verringerung der Datenmenge, bietet sie weitere Vorteile wie Skalierbarkeit, einfache Auswertung ihrer Funktionswerte und der exakten Berechnung von Tangenten und Steigungen durch eine geschlossene Formel.

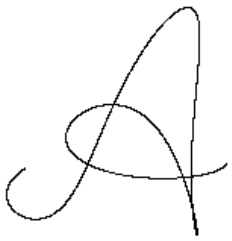
Die Bestimmung der Bounding-Box, also des kleinsten, die Figur überdeckenden Rechtecks mit Seiten parallel zu den Koordinatenachsen, kann jedoch i.A. nur näherungsweise geschehen.

Der Abstand der Kurve zu den Originalpunkten lässt sich in guter Näherung angeben durch

$$\inf_{t \in S^1} \|P_k - f(t)\| \approx \|P_k - f(t_k)\|$$

was als Kriterium dafür verwendet werden kann, ob die Vernachlässigung bestimmter Terme der DFT im Rahmen einer voregegbenen Abweichung akzeptabel ist. Eine analoge Näherung gilt natürlich auch für $\mathcal{F}\{\mathcal{I}\}$.

Bild 4



Ergebnis

Falls die Spline-Darstellung der Kurve praktikabler ist, kann natürlich auch diese verwendet werden. Neben den n Punkten sind dann zusätzlich noch die $n-1$ Zeitpunkte $t_1, \ldots, t_{n-1}$ zu speichern (t_n ist immer 1).

3 Weblinks

GSL – GNU Scientific Library — Quelle

Eine frei verfügbare C-Bibliothek mit Algorithmen zu Spline-Interpolation, Scheller Fourier-Transformation und Wavelet-Transformation, etc. Für Microsoft Windows gibt es Binaries der GSL unter GSL for Windows. Die Binärdateien haben evtl. eine ältere Version als die aktuelle GSL, und daher sind nicht alle Funktionalitäten, die in der GSL-Dokumentation beschrieben sind, damit verfügbar.

Die Windows-Version der GSL 1.12 habe ich als Archiv verfügbar gemacht (tgz, 6.2 MB). Das Archiv wird einfach nur in ein Verzeichnis extrahiert und der Pfad im Makefile des Projekts bzw. bei den gcc-Optionen angegeben.

GNU Compiler Collection — Software

Der C- und C++-Compiler. Wer den Compiler nicht aus den Quellen erzeugen möchte, der findet zahlreiche generierte Versionen im Netz. So zum Beispiel das MinGW – Minimalist GNU for Windows oder zusammen mit der integrierten Entwicklungsumgebung Code::Blocks.

ImageMagick — Anwendung

Eine mächtige, freie Software zur Bildmanipulation und -bearbeitung. Sie enthält u.a. C-Bibliotheken und Werkzeuge für die Kommandozeile wie `convert`, mit dem die Grafiken für dieses Dokument von PPM zum gebräuchlicheren PNG konvertiert wurden.

„Wie Parametrisierung für Kurve finden?“ — Foren-Beitrag

Beitrag in einem Internet-Diskussionsforum zu diesem Thema.

Von Punkt zu Punkt — C-Projekt

Ein kleines C-Projekt, das das hier vorgestellte Verfahren implementiert, und mit dem auch die Grafiken in diesem Dokument erstellt wurden. Als Software werden benötigt GCC, make und convert.

Von Punkt zu Punkt — Internet-Artikel

Internet-Version dieses Artikels.

$$\begin{pmatrix} 12\sin(t)-4\sin(3t) \\ 13\cos(t)-5\cos(2t)-2\cos(3t)-\cos(4t) \end{pmatrix}$$

Inhaltsverzeichnis