Reconstruction of Surfaces from Unorganized Data Points

Richard Stotz

18. Januar 2011

1 Einführung

In Kapitel 13 des Buchs Level Set Methods and Dynamic Implicit Surfaces von Stanley Osher und Ronald Fedkiw wird die Rekonstruktion von Oberflächen aus einer gegebenen Menge von Punkten. Hierbei werden durch Level Set Methoden sehr gute Ergebnisse erzielt.

2 Theoretische Überlegungen

2.1 Problemstellung

Gegeben sei eine Menge $S \subset \mathbb{R}^3$ von Punkten auf einer Oberfäche eines Objekts. Gesucht wird eine Funktion f, welche die ursprüngliche Oberfäche möglichst exakt beschreibt.
Hierbei ergeben sich mehrere Probleme. Zum einen ist die Topologie des Objekts nicht bekannt, es könnte zusammenhängend sein, aber auch aus mehreren Teilen bestehen. Außerdem ist nichts über Glattheitseigenschaften der Oberfläche bekannt, sie könnte also scharfe Kanten enthalten, welche nur schwierig zu rekonstruieren sein werden. Des Weiteren muss eine numerische Implementierung der Algorithmen möglichst effizient arbeiten, da bei dreidimensionalen Objekten eine sehr hohe Anzahl gegebener Punkte zu erwarten ist.
Störungen in den Eingabewerten müssen gesondert betrachtet werden. Hier können die aus der Bildverarbeitung bekannten Algorithmen hilfreich sein.

2.2 Idee des Algorithmus

Die hier vorgestellte Methode basiert auf der Annahme, dass physikalische Objekte eine Minimierung der Oberflächenenergie anstreben. Definiere zunächst $d(\vec{x}) = dist(\vec{x}, S)$, die Abstandsfunktion zur Menge S. Dann ist die Oberflächenenergie:

$$E(\Gamma) = \left[\int_\Gamma d^p(\vec{x}) ds \right]^{\frac{1}{p}} \tag{1}$$

wobei Γ eine beliebige Oberfläche sei. Wir suchen nun einen lokalen Minimierer dieses Funktionals und erhalten den Gradientenfluss

$$\frac{d\Gamma}{dt} = - \left[\int_\Gamma d^p(\vec{x}) ds \right]^{\frac{1}{p}-1} d^{p-1}(\vec{x}) \left[\nabla d(\vec{x}) \cdot \vec{N} + \frac{1}{p} d(\vec{x}) \kappa \right] \vec{N} \tag{2}$$

Der Minimierer erfüllt die Euler-Lagrange Gleichung

$$d^{p-1}(\vec{x}) \left[\nabla d(\vec{x}) \cdot \vec{N} + \frac{1}{p} d(\vec{x}) \kappa \right] = 0 \tag{3}$$

Für alle weiteren Berechnungen wählen wir $p = 1$, da in diesem Fall die Oberflächenenergie die Dimension des Volumens hat und sich diese Wahl in der Praxis bewährt hat.

2.3 Konvektionsmodell

Das Konvektionsmodell bietet eine noch einfachere, dafür aber Berechnungsmethode für die Oberfäche:
Idee dieses Modells ist, die Abstandsfunktion d als konservatives Feld aufzufassen (auch Potentialfeld genannt). Daraus ergibt sich dann das Geschwindigkeitsfeld $\frac{d\Gamma}{dt} = \vec{v} = -\nabla d$
Die gesuchte Oberfläche ist dann ein lokaler Gleichgewichtszustand für das Feld.

2.4 Übertragung in eine Level-Set Gleichung

Wie in den vorherigen Kapiteln besprochen, bietet die Übertragung des Problems in eine Level-Set Gleichung viele Vorteile, vor allem ist damit eine Parametrisierung der Oberfläche unnötig. Die gesuchte Oberfläche wird als Null-Höhenlinie einer Level Set Funktion ϕ angenommen und ist also eine implizite Oberfläche.
Sei ϕ die Level-Set Funktion, dann erfült ϕ:

$$\Gamma(t) = \{\vec{x} | \phi(\vec{x}, t) = 0\} \tag{4}$$

$$\Rightarrow \qquad \nabla\phi \cdot \frac{d\Gamma}{dt} + \phi_t = 0 \tag{5}$$

Hieraus ergibt sich schließlich die Level Set Formulierung, folgende nichtlineare parabolische Differentialgleichung:

$$\frac{\partial\phi}{\partial\phi} = \left[\nabla d \cdot \frac{\nabla\phi}{|\nabla\phi|} + d\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} \right] |\nabla\phi| \tag{6}$$

Leider ist für eine derartige partielle Differentialgleichung kein effizientes Lösungsverfahren bekannt, denn die Zeitschrittweite ist beschränkt durch $\triangle t = \mathcal{O}(\triangle x^2)$. Im Konvektionsmodell ergibt sich als Level Set Gleichung

$$\frac{\partial\phi}{\partial\phi} = \nabla d(\vec{x}) \cdot \nabla\phi \tag{7}$$

Diese Gleichung ist nicht parabolisch und kann mit $\triangle t = \mathcal{O}(\triangle x)$ gelöst werden.
In der Praxis werden beide Modelle schließlich kombiniert, um eine gute Balance zwischen Laufzeit und Genauigkeit zu erhalten.

3 Numerische Umsetzung

Die numerische Umsetzung dieser Überlegungen erfolgt in 3 Schritten:

1. Berechnung der Abstandsfunktion $dist(\vec{x}, S)$ für alle Punkte auf dem Gitter
2. Berechnung einer Startoberfläche
3. Numerisches Lösen der Gleichungen (6) und (7)

Im Folgenden werden wir auf die ersten beiden Punkte näher eingehen. Numerische Lösungsverfahren für die Gleichungen (6) und (7) wurden bereits in den vorangegangenen Kapiteln besprochen.

3.1 Berechnung der Abstandsfunktion

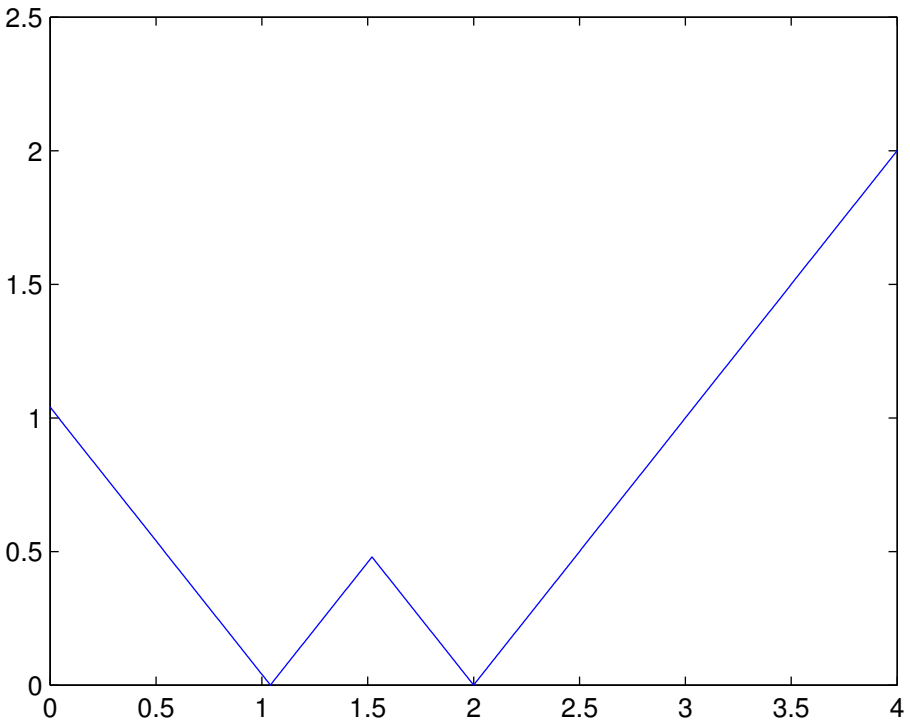
Für die Berechnung der Oberflächenenergie muss zunächst der Abstand jedes Punktes zur Menge S berechnet werden:

$$d(\vec{x}) := dist(\vec{x}, S) := min\{|\vec{x} - \vec{s}||\vec{s} \in S\} \tag{8}$$

Einem naiven Ansatz folgend könnte man für jeden Punkt $\vec{x}$ auf dem Gitter den Abstand zu jedem Punkt $s \in S$ berechnen, und dann das Minimum über diese Werte bestimmen. Der Aufwand bei diesem Ansatz wäre allerdings (bei N Gitterpunkten und M Punkten in S) in $\mathcal{O}(M \cdot N)$, und somit viel zu hoch.
Eine effizientere Methode ergibt sich aus der Beobachtung, dass $|\nabla d(\vec{x})| = 1$ gilt, d also eine sogenannte *Eikonalgleichung* erfüllt. Anschaulich wird die am Bild einer eindimensionalen Abstandsfunktion zu $S = \{1.0, 2\}$:

Hongkai Zhao schlägt für die Berechnung der Abstandsfunktion den Fast Sweeping Algorithmus vor. Dieser hat bei N Gitterpunkten eine Laufzeit von $\mathcal{O}(N)$ und ist auch in mehrdimensionalen Räumen leicht zu implementieren.
Ich werde nun eine vereinfachte Version des Algorithmus in 2D angeben. Weiterführende Informationen, sowie eine theoretische Analyse finden sich unter anderem in [1].

Initialisierung: Weise d bei allen Gitterpunkten nahe S exakte Werte zu. Bei allen



weiteren Gitterpunkten wird d ein sehr hoher, positiver Wert zugewiesen.
Iteration: An jedem Gitterpunkt, dessen Abstand nicht bereits bei der Initialisierung festgelegt wurde, berechne die Lösung $d_{i,j}$ von folgender Gleichung:

$$d_{i,j} = \begin{cases} min(x_{min}, y_{min}) + h, & falls \quad |x_{min} - y_{min}| \geqslant h \\ \frac{x_{min}+y_{min}+\sqrt{2h^2-(x_{min}-y_{min})^2}}{2}, & falls \quad |x_{min} - y_{min}| \leqslant h \end{cases} \tag{9}$$

mit h als Schrittweite im Gitter,

$$x_{min} = min(d_{i-1,j}, d_{i+1,j}) \quad y_{min} = min(d_{i,j-1}, d_{i,j+1}) \tag{10}$$

Falls $d_{i,j}$ kleiner als der bisher gespeicherte Abstand ist, so wird $d_{i,j}$ als aktualisierter Abstand gespeichert. Wir durchlaufen mit dieser Methode das Gitter nacheinander über folgende Ordnungen:

$$\begin{array}{ll} i = 1:n, j = 1:n & i = 1:n, j = n:1 \\ i = n:1, j = 1:n & i = n:1, j = n:1 \end{array} \tag{11}$$

3.2 Berechnung der Startoberfäche

Die Berechnung einer sinnvollen Startoberfläche ist für die Effizienz unserer Methode entscheidend. Die Startoberfläche sollte also eine möglichst gute Näherung an die tatsächliche Oberfläche sein, aber auch sehr schnell zu berechnen sein. Stanley Osher und Ronald Fedkiw schlagen in ihrem Buch folgenden iterativen Algorithmus der Laufzeit $\mathcal{O}(N \cdot logN)$ vor:

1. Starte mit einer beliebigen Region R, die S komplett enthält.
2. Markiere alle Gitterpunkte in R als innere Gitterpunkte, alle anderen als äußere Gitterpunkte.
3. Markiere nun alle inneren Gitterpunkte mit einem außere"Nachbarpunkt als temporäre Grenzpunkte.
4. Erstelle einen Heapsort Binärbaum mit den temporären Grenzpunkten, sortiert nach ihrer Distanz zu S.
5. Nimm den obersten temporären Grenzpunkt g (mit dem größten Abstand zu S) und prüfe, ob einer seiner inneren Nachbarn weiter als er selbst von S entfernt ist.
   - Falls Ja, so ist g finaler Grenzpunkt. Entferne g aus dem Baum und gehe zu Schritt 4.
   - Falls Nein, so markiere g als äußeren Gitterpunkt, entferne g aus dem Baum und markiere alle inneren Nachbarn von g als temporäre Grenzpunkte. Gehe zu Schritt 4.

   Iteriere diesen Prozess so lange, bis alle temporären Grenzpunkte nah genug an S liegen. Markiere dann die verbleibenden temporären Grenzpunkte als finale Grenzpunkte.

[2] zeigt anhand verschiedener Beispiele, wie stark die Startoberfläche das Endergebnis beeinflusst.

4 Zusammenfassung

Level Set Methoden bieten sehr attraktive Möglichkeiten in der Oberflächenrekonstruktion, die bei geschickter numerischer Implementierung auch sehr gute Laufzeiten erreichen. Vor allem geschickte Vorbereitungsrechnungen, wie die Berechnung einer Startoberfläche, führen zu erstaunlichen Verbesserungen in Ergebnis und Laufzeit.

Literatur

[1] Hongkai Zhao: *A Fast Sweeping Method For Eikonal Equations.* http://www.math.uci.edu/?zhao/publication, 2003, preprint.

[2] Pengye Xia Dong Yang Congmin Zhang: *3D Surface Reconstruction Using Level Set Method.* http://www.xiapengye.com/uploads/4/0/0/8/4008137/surrec.pdf, project report for COMP641O: Image Based Modeling, May 2010