Level Set Methods

Constructing Signed Distance Functions
Extrapolation in the Normal Direction

André Gripshöfer

03.05.2007

1 Constructing Signed Distance Functions

- Introduction
- Reinitialization
- Crossing Times
- The Reinitialization Equation
- The Fast Marching Method

2 Extrapolation in the Normal Direction

- One-Way Extrapolation
- Two-Way Extrapolation
- Fast Marching Method

Kap.7: Constructing SDFs

Falls ϕ eine Signed Distance Function (SDF) ist, kann man einige Vereinfachungen machen
$\rightarrow$ Ziel: numerische Verfahren zur Konstruktion von SDFs

Problem: Sobald sich die Oberfläche ändert, ist ϕ keine SDF mehr

Lösung: periodische Anwendung der Verfahren, damit ϕ eine SDF bleibt

Eine gute Wahl für ϕ ist:
$\phi = +\text{d}$, in Ω^+ $\quad$ $\phi = -\text{d}$, in Ω^- $\quad$ $\phi = 0$, auf $\delta\Omega$
wobei d die Abstandsfunktion ist

Um numerische Fehler zu vermeiden, reinitialisiere die level set function periodisch.

Nur die $\phi = 0$ Isokontur interessant
$\rightarrow$ Die anderen Isokonturen können jederzeit zurückgesetzt werden, damit ϕ wieder eine SDF ist

Verfahren:

- Lokalisiere und diskretisiere die $\phi = 0$ Isokontur mit einem *contour plotting algorithm*
- Miss die Abstände von $\phi = 0$

Problem: langsam

Lösung: Verwende nur Punkte in einer kleinen Umgebung von $\phi = 0$

$\longrightarrow$ 1.Version der *local level set* Methode

Was bringt uns die Reinitialisierung?

Numerisches Standard-Verfahren:

startet mit Anfangswerten; Annahme, dass die numerische Lösung gutartig bleibt

Reinitialisierung:

Nur die $\phi = 0$ Isokontur muss gutartig bleiben; Anderswo auftretende Probleme werden beim Reinitialisieren gelöscht

$\longrightarrow$ Erhöhung der Flexibilität beim Algorithmusdesign, da Probleme außerhalb von $\phi = 0$ vernachlässigt werden können.

Schwierigkeit beim Berechnen von SDFs: Lokalisierung und Diskretisierung der Oberfläche
Verfahren, um das zu vereinfachen:

- Wähle einen Punkt $\overrightarrow{x} \in \Omega^+$, $\phi(\overrightarrow{x}) = +\mathrm{d}$
- Benutze Gl.(6.1): $\phi_t + \mathrm{a}\,|\nabla\phi| = 0$, $\mathrm{a} := 1$, um die Oberfläche mit Geschwindigkeit 1 in Normalenrichtung zu bewegen
- Bestimme zu jedem Zeitpunkt das Vorzeichen von ϕ, bis es sich ändert
- Bestimme durch zeitliche Interpolation den exaten Zeitpunkt t_0, an dem $\phi = 0$ galt; $t_0 :=$ *crossing time*
- Da $\mathrm{a} = 1$, gilt: $t_0 = \mathrm{d}$

Für Punkte $\overrightarrow{x} \in \Omega^-$ verfahre analog, aber mit a = -1

Numerische Methode, um $|\nabla\phi| = f(\vec{x})$ zu lösen:

Addiere $f(\vec{x})$ zur rechten Seite von (6.1) und setze a = 1:

$$\phi_t + |\nabla\phi| = f(\vec{x}) \qquad (7.1)$$

steady-state: $\phi_t = 0 \quad \Rightarrow \quad |\nabla\phi| = f(\vec{x})$

(Trivialfall $f(\vec{x}) = 1$: Lösung ϕ ist eine SDF)

Zur Reinitialisierung der level set function ist (7.1) unbrauchbar, da sich die $\phi = 0$ Isokontur möglicherweise bewegt.

$\rightarrow$ Vermeidbar durch Berechnung der SDF für alle Punkte, die mit der Oberfläche benachbart sind.

$$\phi_t + |\nabla \phi| = 1 \qquad (7.2)$$

kann dann in Ω^+ gelöst werden, um ϕ zu aktualisieren. Dabei werden die mit der Oberfläche benachbarten Punkte als Randbedingungen verwendet.

Falls aber ein 2-Punkt breites Band in Ω^- initialisiert ist, kann man HJ WENO anwenden.

Alternativ: Initialisiere ein 3-Punkt breites Band in Ω^- und benutze dies, um alle Punkte in Ω^+ zu aktualisieren.

Analog kann man mit einem 3-Punkt breites Band in Ω^+ alle Werte in Ω^- aktualisieren, in dem man

$$\phi_t - |\nabla \phi| = -1 \qquad (7.3)$$

bis zu einem *steady-state* löst.

Fasse (7.2) und (7.3) zur *Reinitialisationsgleichung* zusammen:

$$\phi_t + S(\phi_o)(|\nabla\phi| - 1) = 0 \qquad (7.4)$$

wobei $S(\phi_o) = 1$ in Ω^+, -1 in Ω^- und 0 auf der Oberfläche ist.

Man braucht hiermit keine Punkte mehr zu initialisieren, die man als Randbedingungen verwendet:

Die der Oberfläche benachbarten Punkte in Ω^+ benutzen die Punkte in Ω^- als Randbedingungen und umgekehrt.
Falls sich diese zirkuläre Abhängigkeit ausbalanciert, erhält man eine SDF.

Problem: Falls ϕ nicht glatt ist, kann sich die Oberfläche durch die zirkuläre Abhängigkeit von seiner Ausgangsposition wegbewegen.

Bei der Diskretisierung von (7.4) wird der $S(\phi_o)|\nabla\phi|$ Term als Bewegung in Normalenrichtung behandelt (siehe Kap.6). Bessere Ergebnisse, wenn

$$S(\phi_o) = \frac{\phi_o}{\sqrt{\phi_o^2 + (\Delta x)^2}} \qquad (7.5)$$

Aber noch besser:

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla\phi|^2(\Delta x)^2}} \qquad (7.6)$$

vor allem, wenn ϕ_o nicht nahe an einer SDF ist.

Idealerweise bleibt die Oberfläche während der Reinitialisierung konstant, aber durch numerische Fehler wird sie sich ein Stückchen bewegen.

$\rightarrow$ Verbesserung des Standard-Reinitialisierungsverfahrens: Betrachte inkompressiblen 2-Phasen-Fluss:

Oberfläche ändert sich nicht: Die Fläche wird erhalten
Oberfläche ändert sich: man kann die Fläche erhalten, in dem man (1.15) benutzt: $\int f(\overrightarrow{x})H(\phi(\overrightarrow{x}))d\overrightarrow{x}$, mit $f(\overrightarrow{x}) = 1$, um die Fläche in jeder Zelle durch

$$A_{i,j} = \int_{\Omega_{i,j}} H(\phi(\overrightarrow{x}))d\overrightarrow{x} \qquad (7.7)$$

zu approximieren. $\Omega_{i,j}$: jeweilige Gitterzelle, H: smeared-out Heavyside function (siehe 1.22)

Implementierung: Addiere Korrektur-Term zur r. S. von (7.4):

$$\phi_t + S(\phi_o)(|\nabla\phi| - 1) = \lambda\delta(\phi)|\nabla\phi| \qquad (7.8)$$

mit der multidim. δ-Funktion: $\widehat{\delta} = \delta(\phi)|\nabla\phi|$ aus (1.19).

Die Bedingung $(A_{i,j})_t = 0$ ist äquivalent zu:

$$\int_{\Omega_{i,j}} H'(\phi)\phi_t d\overrightarrow{x} = 0 \qquad (7.9)$$

Benutze (7.8) und $H'(\phi) = \delta(\phi)$ aus (1.18):

$$\Rightarrow \int_{\Omega_{i,j}} \delta(\phi)(-S(\phi_o)(|\nabla\phi|-1)+\lambda\delta(\phi)|\nabla\phi|)d\overrightarrow{x} = 0 (7.10)$$

Definiere eine Separable λ in jeder Zelle mit (7.10):

$$\lambda_{i,j} = -\frac{\int_{\Omega_{i,j}} \delta(\phi)(-S(\phi_o)(|\nabla\phi| - 1)d\overrightarrow{x}}{\int_{\Omega_{i,j}} \delta^2(\phi)|\nabla\phi|d\overrightarrow{x}} \qquad (7.11)$$

Verwende (7.4), um ϕ^{n+1} aus ϕ^n zu berechnen:

$$\lambda_{i,j} = -\frac{\int_{\Omega_{i,j}} \delta(\phi)(\frac{\phi^{n+1}-\phi^n}{\Delta t})d\overrightarrow{x}}{\int_{\Omega_{i,j}} \delta^2(\phi)|\nabla\phi|d\overrightarrow{x}} \qquad (7.12)$$

Alle der Oberfläche benachbarten Punkte seien mit ihren exakten Abstandswerten versehen

Ziel: Von diesen Anfangspunkten ausgehend wollen wir jedem Gitterpunkt einen Abstandswert zuordnen.

Verfahren:

- Berechne für jeden Punkt, der mit den Anfangspunkten benachbart ist, einen vorläufigen Abstandswert
- Füge den Punkt mit dem kleinsten Wert dem Band der Anfangspunkte hinzu
- Die meisten Punkte haben schon einen vorl. Abstandswert. Berechne also nur für die Punkte, die mit dem Hinzugefügten benachbart sind vorläufige Abstandswerte
- Wiederhole den Prozess

$\longrightarrow$ alle Gitterpunkte in Ω^+ werden mit einem Abstandswert versehen

Problem: es ist aufwendig, unter allen Abstandswerten den kleinsten zu finden

Lösung: Benutze einen Baum der Struktur:

- Jeder Punkt besitzt einen Abstandswert, der jeweils kleiner ist als die zwei unter ihm
- Neue Punkte werden von unten zugefügt

Berechnung des kleinsten Abstands:

$\phi_{i,j,k}$ soll der Punkt sein, der der Oberfläche am nächsten ist.

$\overrightarrow{\theta} = (\theta_1, \theta_2, \theta_3)$ bezeichne die charakteristische Richtung, in der $\phi_{i,j,k}$ liegt;
mit $\sum_{s=1}^{3} \theta_s = 1$, $\theta_s > 0$

Suche in jedem Quadranten nach dem minimalen Wert für $\phi_{i,j,k}$, vergl. alle Werte, um den kleinsten zu ermitteln.

Den kleinsten Wert in einem Quadranten erhält man, in dem man

$$\phi_{i,j,k} = \tau(\overrightarrow{\theta}) + \theta_1\phi_1 + \theta_2\phi_2 + \theta_3\phi_3 \qquad (7.13)$$

für alle $\overrightarrow{\theta}$ minimiert, wobei

$$\tau(\overrightarrow{\theta}) = \sqrt{(\theta_1\Delta x_1)^2 + (\theta_2\Delta x_2)^2 + (\theta_3\Delta x_3)^2}$$

der zurückgelegte Abstand und $\sum \theta_s\phi_s$ der Startpunkt im jeweiligen Quadranten ist.

8 mögliche Quadranten mit Startpunkten:

$$\phi_1 = \phi_{i\pm1,j,k} \qquad \phi_2 = \phi_{i,j\pm1,k} \qquad \phi_3 = \phi_{i,j,k\pm1}$$

Füge einen Lagrange-Multiplikator zur Gl. (7.13) hinzu:

$$\phi_{i,j,k} = \tau(\overrightarrow{\theta}) + \theta_1\phi_1 + \theta_2\phi_2 + \theta_3\phi_3 + \lambda(1 - \sum \theta_s) \qquad (7.15)$$

Bilde part. Ableitung für jedes θ_s und setze gleich Null:

$$\frac{\theta_s(\Delta x_s)^2}{\tau(\overrightarrow{\theta})} + \phi_s - \lambda = 0 \qquad (7.16)$$

Löse (7.16) für jedes ϕ_s und setze die Ergebnisse in (7.13) ein

$$\Rightarrow \qquad \phi_{i,j,k} = \lambda$$

Bestimme λ:

$$(7.16) \Rightarrow \left(\frac{\lambda-\phi_s}{\Delta x_s}\right)^2 = \frac{\theta_s^2(\Delta x_s)^2}{\tau(\vec{\theta})^2} \qquad (7.17)$$

Summier über alle Dimensionen:

$$\left(\frac{\lambda-\phi_1}{\Delta x_1}\right)^2 + \left(\frac{\lambda-\phi_2}{\Delta x_2}\right)^2 + \left(\frac{\lambda-\phi_3}{\Delta x_3}\right)^2 = 1 \qquad (7.18)$$

$\rightarrow$ Den kleinsten Wert von $\phi_{i,j,k}$ erhält man durch Lösen der quadratischen Gleichung:

$$\left(\frac{\phi_{i,j,k}-\phi_1}{\Delta x_1}\right)^2 + \left(\frac{\phi_{i,j,k}-\phi_2}{\Delta x_2}\right)^2 + \left(\frac{\phi_{i,j,k}-\phi_3}{\Delta x_3}\right)^2 = 1 \qquad (7.19)$$

Minimierung von $\phi_{i,j,k}$ über alle Quadranten:

Halte ϕ_2 und ϕ_3 fest, dann erhält man den kleineren Wert von $\phi_{i,j,k}$ mit $\phi_1 = \min(\phi_{i-1,j,k}, \phi_{i+1,j,k})$

Analog für ϕ_2 und ϕ_3.

(7.19) mit diesen Definitionen der ϕ_s zu lösen ist äquivalent zur Approximierung der Ableitungen von ϕ mit dem Vor- bzw. Rückwärtsdifferenzen-Verfahren.

ϕ_s kann unendlich sein, falls in der entsprechenden Dimension die benachbarten Punkte nicht im Band liegen
$\rightarrow$ Setze $\theta_s = 0$

Da aber mind. ein benachbarter Punkt im Band liegt, können max. 2 der 3 Terme verschwinden.

1.Fall: 2 Terme gleich Null

(7.19) hat dann die Form:

$$(\frac{\phi_{i,j,k} - \phi_s}{\Delta x_s})^2 = 1 \qquad (7.20)$$

Lösung: $\phi_{i,j,k} = \phi_s \pm \Delta x_s$

Wir benutzen den '+' -Term, da der Abstand größer wird, je weiter der Algorithmus voranschreitet.

2.Fall: 1 Terme gleich Null

(7.19) hat dann die Form:

$$(\frac{\phi_{i,j,k} - \phi_{s_1}}{\Delta x_{s_1}})^2 + (\frac{\phi_{i,j,k} - \phi_{s_2}}{\Delta x_{s_2}})^2 = 1 \qquad (7.21)$$

Diese Gleichung kann keine, eine oder zwei Lösungen haben.

$$P(\phi_{i,j,k}) := (\frac{\phi_{i,j,k} - \phi_{s_1}}{\Delta x_{s_1}})^2 + (\frac{\phi_{i,j,k} - \phi_{s_2}}{\Delta x_{s_2}})^2 \qquad (7.22)$$

a) $P(\max(\phi_{s_1}, \phi_{s_2})) > 1$

$\phi_{i,j,k} < \max(\phi_{s_1}, \phi_{s_2})$ (falls eine Lösung ex.)
$\rightarrow$ streiche Term mit dem größeren ϕ_s und verfahre dann wie im 1.Fall

b) $P(\max(\phi_{s_1}, \phi_{s_2})) \leqslant 1$

(7.21) hat dann 2 Lösungen. Verwende die Größere.

3.Fall: alle Terme ungleich Null

$$P(\phi_{i,j,k}) := (\frac{\phi_{i,j,k} - \phi_1}{\Delta x_1})^2 + (\frac{\phi_{i,j,k} - \phi_2}{\Delta x_2})^2 + (\frac{\phi_{i,j,k} - \phi_3}{\Delta x_3})^2$$

a) $P(\max(\phi_s)) > 1$

streiche den Term mit dem größten ϕ_s und verfahre weiter wie im 2.Fall

b) $P(\max(\phi_s)) \leqslant 1$

verwende die größere Lösung.

Noch ausstehend: Die Initialisierung der mit der Oberfläche benachbarten Punkte (initialization routine):

- betrachte alle Koordinatenrichtungen unabhängig
- falls ϕ in einer Richtung das Vorzeichen wechselt, bestimme durch Interpolation den exakten Übergangspunkt
- Setze ϕ dann gleich dem kleinsten Wert

Zusammenfassung:

Sowohl die initialization routine, als auch der Fast Marching Algorithmus sind von der Genauigkeit 1. Ordnung

$\longrightarrow$ es wäre besser, die Reinitialisierungs-Gl. zu verwenden, da sie von höherer Genauigkeit ist

Aber: Reinitialisierungs-Gl. zeitaufwendig und funktioniert nicht gut, wenn ϕ anfangs nicht nahe an einer SDF ist

$\longrightarrow$ in den meisten Fällen wird der Algorithmus bevorzugt

Erreichen von Genauigkeit höherer Ordnung:

Falls genügend Punkte im Band liegen, ersetze die Vor- bzw. Rückwärtsdifferenzen-Verfahren 1. Ordnung in (7.19) durch entsprechende Verfahren 2. Ordnung.

Man kann auch Verfahren noch höherer Ordnung verwenden, solange genügend Punkte vorhanden sind.

Kap.8: Extrapol. in the Normal Direction

Die Hamilton-Jacobi-Gleichung:

$$S_t + \overrightarrow{N} \cdot \nabla S = 0 \qquad (8.1)$$

extrapoliert S senkrecht zur Oberfläche.

Da $H(\nabla S) = \overrightarrow{N} \cdot \nabla S$, wähle $H_1 = n_1, H_2 = n_2, H_3 = n_3$ und löse die Gleichung mit den aus Kapitel 5 (Ralfs Vortrag) bekannten Verfahren.

Bestimmung der Normalenrichtung:

- An einem Punkt $\overrightarrow{x_{i,j,k}}$ sei $\phi_{i,j,k}$ die Levelset Funktion
- Betrachte $\phi_{i-1,j,k}$ und $\phi_{i+1,j,k}$, um ϕ_x zu bestimmen:

  1.Fall: einer der Werte kleiner als $\phi_{i,j,k}$
  Benutze den kleineren Wert, um eine 'one-sided difference' zu berechnen

  2.Fall: beide Werte größer als $\phi_{i,j,k}$
  Setze $\phi_x = 0$

- Verfahre analog mit ϕ_y und ϕ_z
- Benutze (1.2): $\overrightarrow{N} = \frac{\nabla\phi}{|\nabla\phi|}$ um die Normale zu bestimmen

Angenommen, S ist anfangs nur in Ω^- definiert und man möchte S auf Ω^+ ausweiten:

Löse (8.1) in Ω^+, in dem man die Werte in Ω^- als Randbedingungen betrachtet.

$\longrightarrow$ S wird über die Oberfläche hinaus extrapoliert und ist konstant in Normalenrichtung

Ist umgekehrt S anfangs nur in Ω^+ definiert, löse die Gleichung

$$S_t - \overrightarrow{N} \cdot \nabla S = 0 \qquad (8.2)$$

in Ω^- und benutze die Werte in Ω^+ als Randbedingungen.

Verwende in diesem Fall bei der Berechnung der Normalen den größeren Wert der benachbarten Punkte anstatt des kleineren.

Durch Kombination von (8.1) und (8.2) erhält man:

$$S_t + V(\phi)\overrightarrow{N} \cdot \nabla S = 0 \qquad (8.3)$$

Die Normale berechnet man, in dem man die kleineren Werte von ϕ in Ω^+ und die größeren in Ω^- verwendet. $V(\phi)$ ist eine Vorzeichenfunktion:
$V(\phi) = 1$ in Ω^+ , $V(\phi) =$ -1 in Ω^-

Bsp.: Löse die Gleichung (8.3) für jede Komponente des Geschwindigkeitfeldes einzeln
$\rightarrow$ man erhält ein Geschwindigkeitsfeld, welches senkrecht zur Oberfläche konstant ist

Diese Extrapolation der Geschwindigkeit ist nützlich, wenn die Geschwindkeit nur nahe der Oberfläche bekannt ist.

Die FMM kann auch verwendet werden, um S zu extrapolieren:

- Berechne die Normale mit Hilfe der kleineren Werte von ϕ (wie oben)
- Da ϕ in schon ganz Ω^+ definiert ist, kann man direkt alle Punkte verwenden und ordnen
- Wähle den kleinsten Wert von ϕ und berechne einen zugehörigen Wert für S
- Finde den nächstkleineren Wert von ϕ und verfahre analog

Berechnung von S:

In jedem Gitterpunkt wird der Wert von S mit Hilfe der Werte von S in den benachbarten Punkten bestimmt. Da S senkrecht zur Oberfläche konstant sein soll, setzte

$$\overrightarrow{N} \cdot \nabla S = 0 \Leftrightarrow \nabla \phi \cdot \nabla S = 0,$$

da $\overrightarrow{N}$ und $\nabla \phi$ in dieselbe Richtung zeigen.

$$\Rightarrow \phi_x S_x + \phi_y S_y + \phi_z S_z = 0 \qquad (8.4)$$

Setze

$$S_x = D^- S_{i,j,k} \text{ , falls } \phi_x > 0 \text{ ,}$$
$$S_x = D^+ S_{i,j,k} \text{ , falls } \phi_x < 0$$

Analog für S_y und S_z