Image Space Tensor Field Visualization using a LIC-like Method

Sebastian Eichelbaum Mario Hlawitschka[1] Gerik Scheuermann[2]

[1] Institute for Data Analysis and Visualization (IDAV), and Department of Computer Science, University of California, Davis

[2] Abteilung für Bild- und Signalverarbeitung, Institut für Informatik, Universität Leipzig

Gliederung

Gliederung

Zu visualisierende Daten

- Tensorfelder zweiter Ordnung
- Vornehmlich Diffusionstensorfelder
- Aber nicht beschränkt auf DTI Daten

Tensor

- Wichtiges Werkzeug vieler Natur- und Ingenieurwissenschaften
  - Trägheitstensor
  - Belastungstensor (stress tensor)
  - Diffusionstensor

Tensor

- Wichtiges Werkzeug vieler Natur- und Ingenieurwissenschaften
  - Trägheitstensor
  - Belastungstensor (stress tensor)
  - Diffusionstensor

1. Wir gehen von Tensoren 2. Stufe aus als Matrix

Tensor

- Wichtiges Werkzeug vieler Natur- und Ingenieurwissenschaften
  - Trägheitstensor
  - Belastungstensor (stress tensor)
  - Diffusionstensor
- Oft als Verallgemeinerung folgender Begriffe eingeführt
  - Skalar
  - Vektor
  - Matrix
- Hier nicht detailliert besprochen
  - Umfangreiche Einführung in die Tensorrechnung in [Ibe95]

- Wichtiges Werkzeug vieler Natur- und Ingenieurwissenschaften
  - Trägheitstensor
  - Belastungstensor (stress tensor)
  - Diffusionstensor
- Oft als Verallgemeinerung folgender Begriffe eingeführt
  - Skalar
  - Vektor
  - Matrix
- Hier nicht detailliert besprochen
  - Umfangreiche Einführung in die Tensorrechnung in [Ibe95]

1. Wir gehen von Tensoren 2. Stufe aus als Matrix

Diffusionstensor (Eigenschaften)

- Positiv Definit

- Positiv Definit

1. für uns wichtig, da Konzentration auf DTI Daten
2. Symmetrie ist praktisch für Speicherbedarf
3. Orthogonale Eigenvektoren im kartesischen Raum
4. paarweises Skalarprodukt orthogonaler Eigenvektoren ist 0

Diffusionstensor (Eigenschaften)

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung

1. für uns wichtig, da Konzentration auf DTI Daten
2. Symmetrie ist praktisch für Speicherbedarf
3. Orthogonale Eigenvektoren im kartesischen Raum
4. paarweises Skalarprodukt orthogonaler Eigenvektoren ist 0

Diffusionstensor (Eigenschaften)

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung
- Symmetrisch

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung
- Symmetrisch

1. für uns wichtig, da Konzentration auf DTI Daten
2. Symmetrie ist praktisch für Speicherbedarf
3. Orthogonale Eigenvektoren im kartesischen Raum
4. paarweises Skalarprodukt orthogonaler Eigenvektoren ist 0

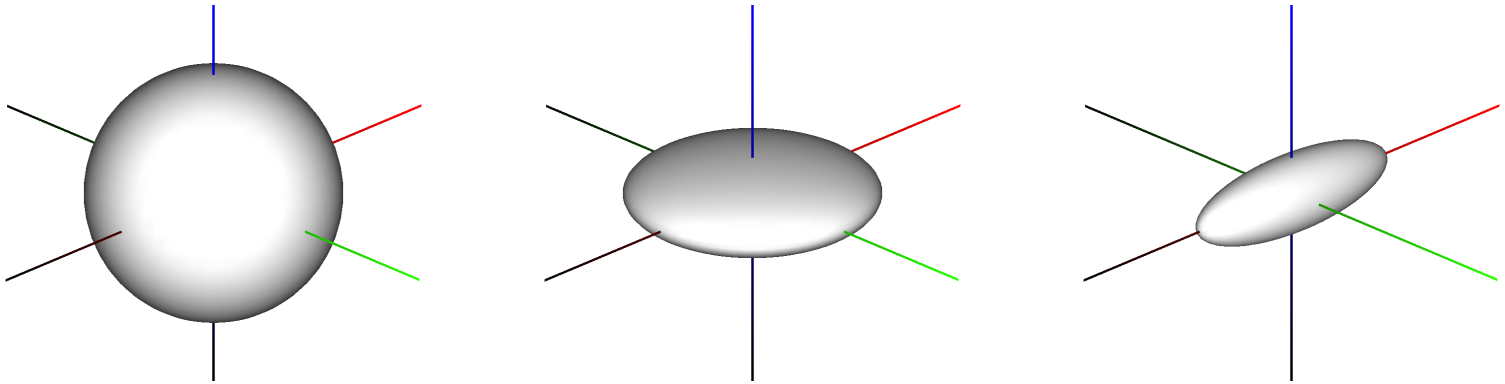
Diffusionstensor (Eigenschaften)

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung
- Symmetrisch
- Mehr Details dazu in [HJ05]

- Positiv Definit
- Drei Eigenvektoren (außer an kritischen Punkten)
  - Orthogonal
  - Besitzen keine Orientierung
- Symmetrisch
- Mehr Details dazu in [HJ05]

1. für uns wichtig, da Konzentration auf DTI Daten
2. Symmetrie ist praktisch für Speicherbedarf
3. Orthogonale Eigenvektoren im kartesischen Raum
4. paarweises Skalarprodukt orthogonaler Eigenvektoren ist 0

Diffusionstensor (Darstellung)



(a) Isotropie (b) Planare Anisotropie (c) Lineare Anisotropie

- Darstellung als Ellipsoid
- Radien entsprechen Diffusion in jeweiligen Richtungen

Diffusionstensor (Abgeleitete Metriken)

- Viele Metriken lassen sich aus Diffusionstensoren ableiten (vgl. Basser et al. [BP96] und Westin et al. [WPG+97])
  - Mean Diffusivity: $MD(D) = \frac{trace(D)}{3}$
  - Fractional Anisotropy: $FA(D) = \sqrt{\frac{3}{2}}\sqrt{\frac{\sum_i(\lambda_i - MD(D))^2}{\sum_i \lambda_i}} \in [0,1]$
  - Apparent diffusion coefficient: $ADC_v(D) = v^T D v$

- Viele Metriken lassen sich aus Diffusionstensoren ableiten (vgl. Basser et al. [BP96] und Westin et al. [WPG+97])
  - Mean Diffusivity: $MD(D) = \frac{trace(D)}{3}$
  - Fractional Anisotropy: $FA(D) = \sqrt{\frac{3}{2}}\sqrt{\frac{\sum_i(\lambda_i - MD(D))^2}{\sum_i \lambda_i}} \in [0,1]$
  - Apparent diffusion coefficient: $ADC_v(D) = v^T D v$

1. Baryzentrischen Shape Koordinaten erwähnen

Tensor Visualisierung

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung

1. um zu zeigen was mit eingeschränkter Vielfalt gemeint ist
   HyperLIC und Glyphs herausgepickt

Tensor Visualisierung

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung
- Stellen meist abgeleitete Werte des Diffusionstensor dar

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung
- Stellen meist abgeleitete Werte des Diffusionstensor dar

1. um zu zeigen was mit eingeschränkter Vielfalt gemeint ist
   HyperLIC und Glyphs herausgepickt

Tensor Visualisierung

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung
- Stellen meist abgeleitete Werte des Diffusionstensor dar
- Gängige Methoden:
  - Colormaps
  - HyperLIC (Zheng et al. [ZP03])
  - Tensor Glyphs (Superquadrics von Gordon Kindlmann [Kin04])
  - Direct Volume Rendering (Grundlagen in [Bli82, KVH84])
  - Advection Diffusion Tensorlines (Kindlmann et al. [WKL99])
  - Hyperstreamlines (Delmarcelle et al. [DH92])

- Viele Methoden übernommen au Skalar-/Vektorfeld Visualisierung
- Stellen meist abgeleitete Werte des Diffusionstensor dar
- Gängige Methoden:
  - Colormaps
  - HyperLIC (Zheng et al. [ZP03])
  - Tensor Glyphs (Superquadrics von Gordon Kindlmann [Kin04])
  - Direct Volume Rendering (Grundlagen in [Bli82, KVH84])
  - Advection Diffusion Tensorlines (Kindlmann et al. [WKL99])
  - Hyperstreamlines (Delmarcelle et al. [DH92])

1. um zu zeigen was mit eingeschränkter Vielfalt gemeint ist HyperLIC und Glyphs herausgepickt

Probleme

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen

1. hier sagen daß man keine Metrik heranziehen kann, die die Geometrie bestimmt auf der eine Methode laufen soll

Probleme

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten

1. hier sagen daß man keine Metrik heranziehen kann, die die Geometrie bestimmt auf der eine Methode laufen soll

Probleme

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten
- Eingeschränkt in Darstellungsvielfalt

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten
- Eingeschränkt in Darstellungsvielfalt

1. hier sagen daß man keine Metrik heranziehen kann, die die Geometrie bestimmt auf der eine Methode laufen soll

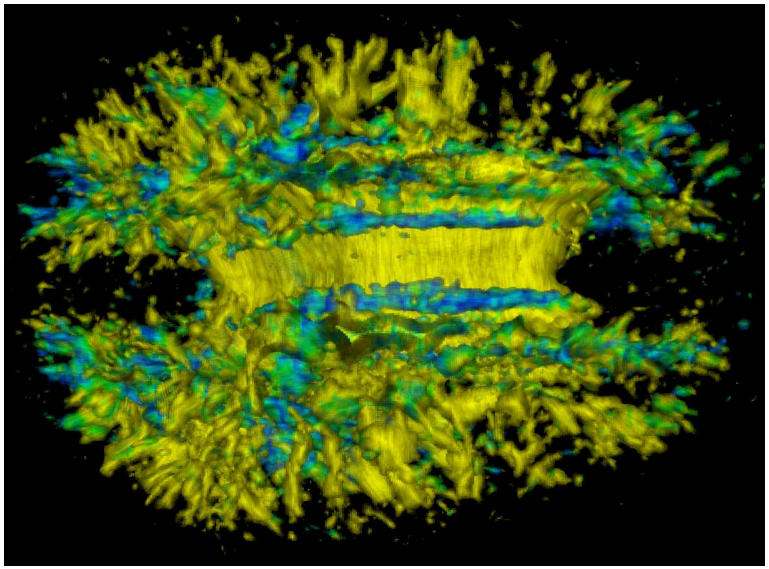
Probleme

- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten
- Eingeschränkt in Darstellungsvielfalt
- Performance oft nicht für Benutzer- Interaktion geeignet

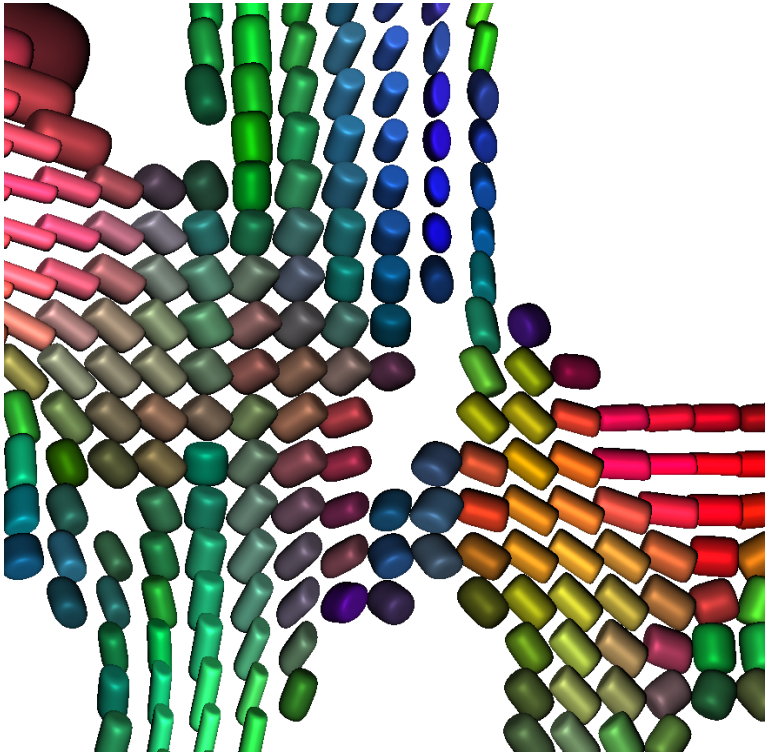
- Lokale oder *globale* Darstellung
  - Auf kleinen, lokalen Ausschnitt der Daten beschränkt
  - Oder für globale Strukturen
- Einschränkungen in der Menge der Daten
- Eingeschränkt in Darstellungsvielfalt
- Performance oft nicht für Benutzer- Interaktion geeignet

1. hier sagen daß man keine Metrik heranziehen kann, die die Geometrie bestimmt auf der eine Methode laufen soll

Beispiele I



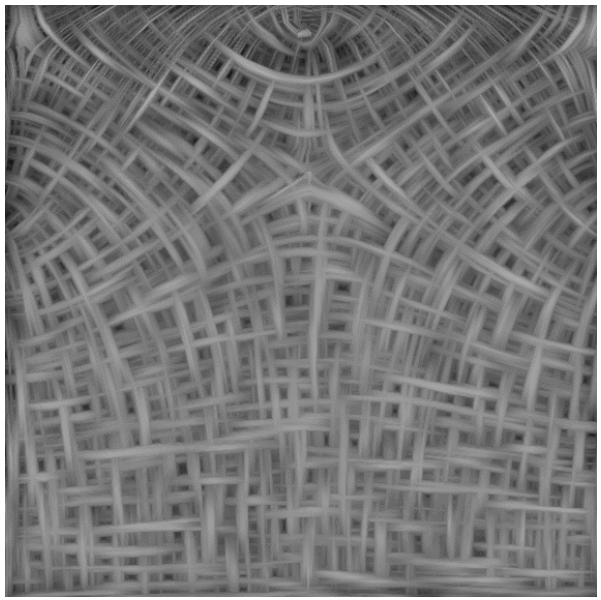
(a) HyperLIC



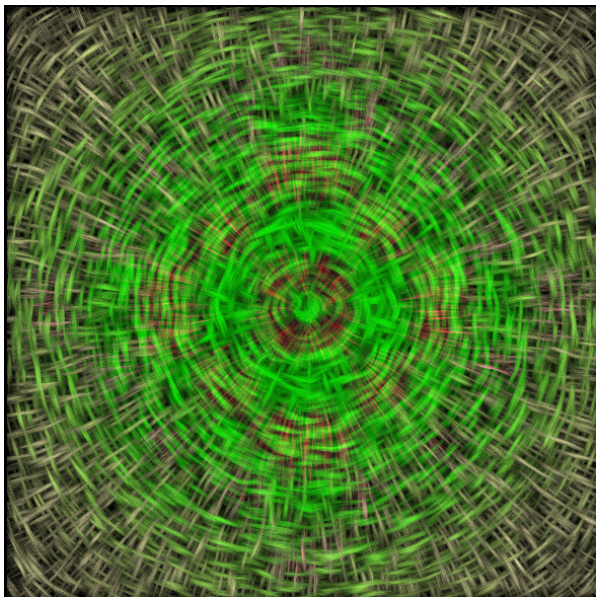
(b) Superquadrics

Abbildung: HyperLIC und Superquadric Tensor Glyps

Beispiele II



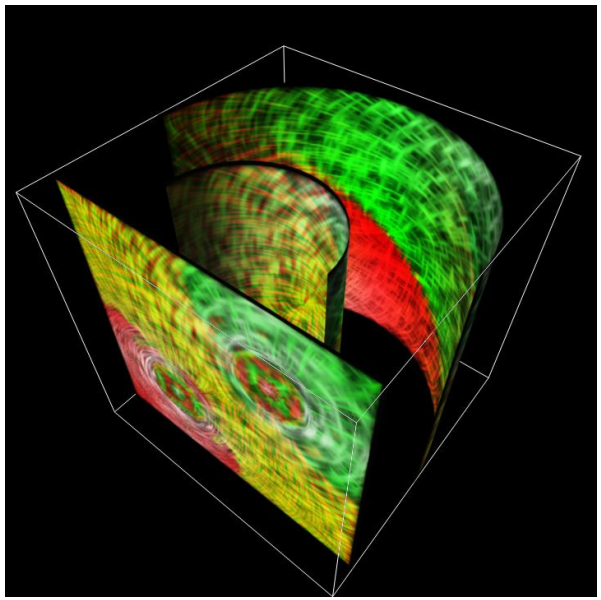
(a) YZ-Slice, variable Punktgröße und Dichte entsprechend Eigenwerten



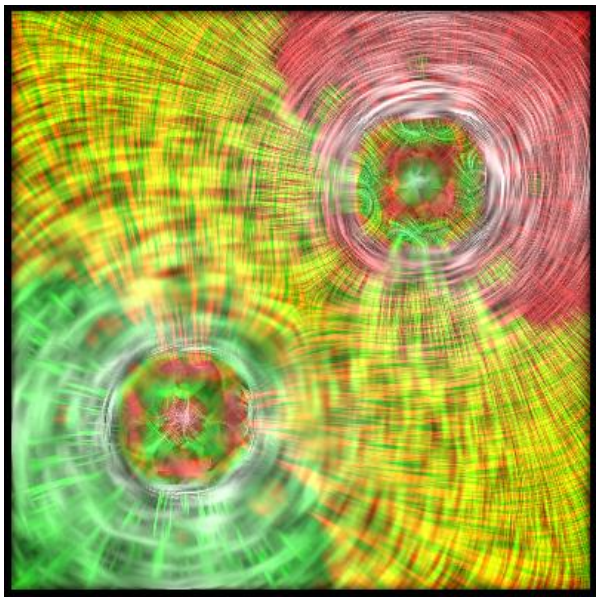
(b) XY-Slice, orthogonal zur Kraft, mit Colormapping

Abbildung: Aus: Hotz et al. [HFH+06]. Single Top Load Datensatz mit Kraftwirkung in Z Richtung.

Beispiele III



(a) XZ-Slice



(b) Verschiedene Ebenen im Datesatz

Abbildung: Aus: Hotz et al. [HFHJ09]. Single Top Load Datensatz mit Kraftwirkung in Z Richtung.

Motivation und Ziele I

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode

1. Physikalisch Basierte Methoden stellen physikalische Eigenschaften direkt dar, wie Diffusionsverläufe
2. Bei uns möglich Metrik zu nutzen um Deformationen zu erzeugen
3. Bei groberen Strukturen Detail der Info anpassen

Motivation und Ziele I

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode
- Lokale und globale Strukturen gleichermaßen darstellbar (Zooming)

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode
- Lokale und globale Strukturen gleichermaßen darstellbar (Zooming)

1. Physikalisch Basierte Methoden stellen physikalische Eigenschaften direkt dar, wie Diffusionsverläufe
2. Bei uns möglich Metrik zu nutzen um Deformationen zu erzeugen
3. Bei groberen Strukturen Detail der Info anpassen

Motivation und Ziele I

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode
- Lokale und globale Strukturen gleichermaßen darstellbar (Zooming)
- Anwendbarkeit auf beliebiger Geometrie
  - beliebige Metriken nutzbar um physikalische Strukturen als Geometrie definieren (vgl. PBM)
  - darauf Diffusionsstrukturen abbilden

- Nutzung der einfach verständlichen und akzeptierten LIC-Methode
- Lokale und globale Strukturen gleichermaßen darstellbar (Zooming)
- Anwendbarkeit auf beliebiger Geometrie
  - beliebige Metriken nutzbar um physikalische Strukturen als Geometrie definieren (vgl. PBM)
  - darauf Diffusionsstrukturen abbilden

1. Physikalisch Basierte Methoden stellen physikalische Eigenschaften direkt dar, wie Diffusionsverläufe
2. Bei uns möglich Metrik zu nutzen um Deformationen zu erzeugen
3. Bei groberen Strukturen Detail der Info anpassen

Motivation und Ziele II

- Minimale oder keine Anforderungen an die Eingabedaten

- Minimale oder keine Anforderungen an die Eingabedaten

1. Animieren des Rauschens möglich bei LIC
2. räumliche Wahrnehmung auf Geometrie in 3D gerade bei LIC schwer

Motivation und Ziele II

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen

1. Animieren des Rauschens möglich bei LIC
2. räumliche Wahrnehmung auf Geometrie in 3D gerade bei LIC schwer

Motivation und Ziele II

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen
- Diffusionstensorfeld- Struktur deutlicher wahrnehmbar machen (vgl. LIC)

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen
- Diffusionstensorfeld- Struktur deutlicher wahrnehmbar machen (vgl. LIC)

1. Animieren des Rauschens möglich bei LIC
2. räumliche Wahrnehmung auf Geometrie in 3D gerade bei LIC schwer

Motivation und Ziele II

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen
- Diffusionstensorfeld- Struktur deutlicher wahrnehmbar machen (vgl. LIC)
- Echtzeitfähigkeit

- Minimale oder keine Anforderungen an die Eingabedaten
- Kontinuierlichkeit der gerenderten Struktur bei Transformationen
- Diffusionstensorfeld- Struktur deutlicher wahrnehmbar machen (vgl. LIC)
- Echtzeitfähigkeit

1. Animieren des Rauschens möglich bei LIC
2. räumliche Wahrnehmung auf Geometrie in 3D gerade bei LIC schwer

Gliederung

Überblick I

- Verschieben der Aufgabe in den Bildraum
  - Ausnutzung moderner Grafikhardware

Überblick I

- Verschieben der Aufgabe in den Bildraum
  - Ausnutzung moderner Grafikhardware
- Aufteilung in kleine, parallelisierbare Schritte
  - GPU bietet hohe Parallelität
  - GPU kann nur lokal schreiben
  - GPU kann nur Daten aus Texturen global lesen

Überblick I

- Verschieben der Aufgabe in den Bildraum
  - Ausnutzung moderner Grafikhardware
- Aufteilung in kleine, parallelisierbare Schritte
  - GPU bietet hohe Parallelität
  - GPU kann nur lokal schreiben
  - GPU kann nur Daten aus Texturen global lesen
- Implementierung mittels OpenGL, GLSL und Framebuffer Objects
  - Kleinere Float Präzision
  - Viele Einschränkungen und Limits

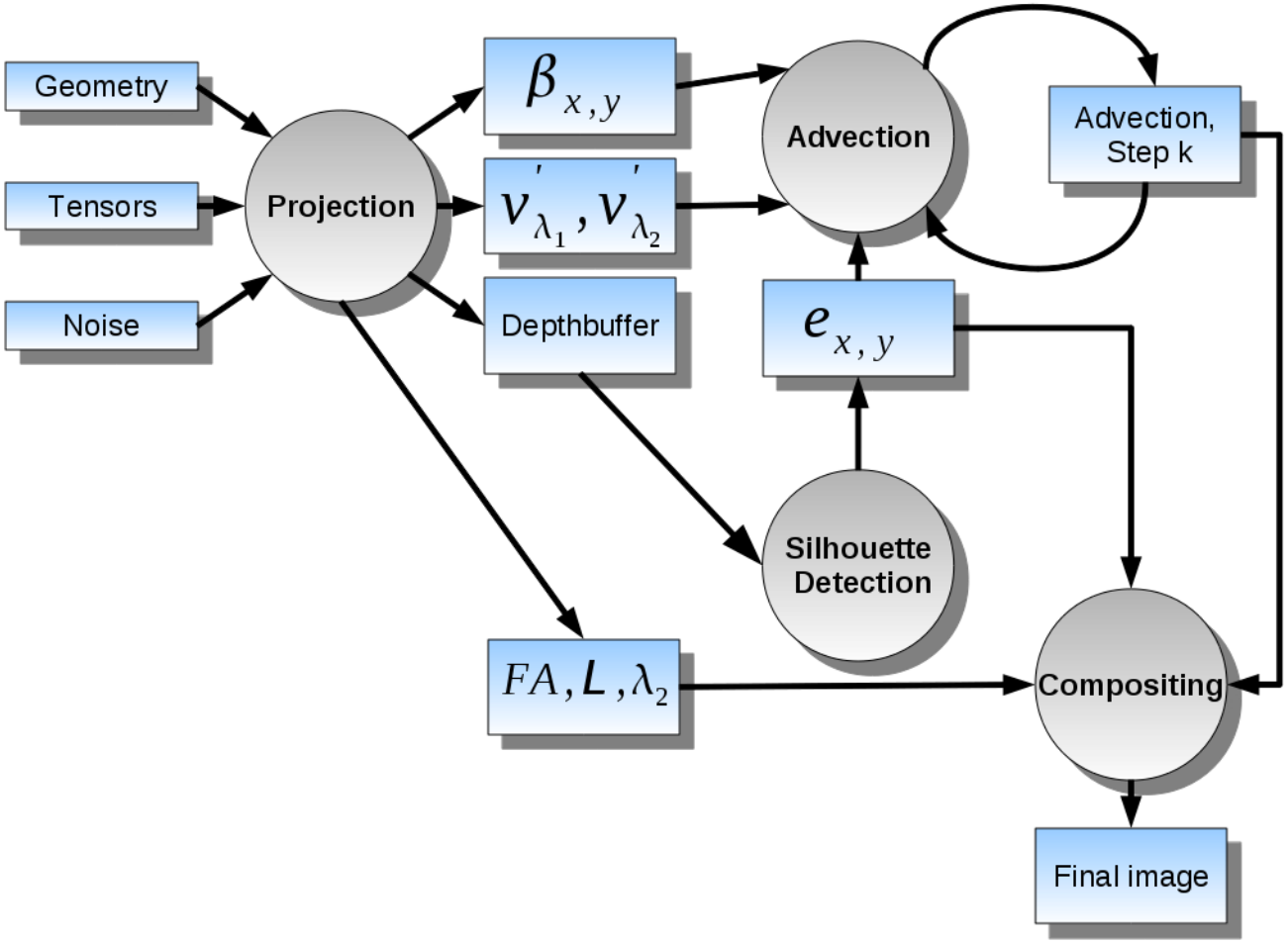
Überblick I

- Verschieben der Aufgabe in den Bildraum
  - Ausnutzung moderner Grafikhardware
- Aufteilung in kleine, parallelisierbare Schritte
  - GPU bietet hohe Parallelität
  - GPU kann nur lokal schreiben
  - GPU kann nur Daten aus Texturen global lesen
- Implementierung mittels OpenGL, GLSL und Framebuffer Objects
  - Kleinere Float Präzision
  - Viele Einschränkungen und Limits
- Aufteilung des Algorithmus in 4 Schritte (4 Renderingdurchläufe)

Überblick I

- Verschieben der Aufgabe in den Bildraum
  - Ausnutzung moderner Grafikhardware
- Aufteilung in kleine, parallelisierbare Schritte
  - GPU bietet hohe Parallelität
  - GPU kann nur lokal schreiben
  - GPU kann nur Daten aus Texturen global lesen
- Implementierung mittels OpenGL, GLSL und Framebuffer Objects
  - Kleinere Float Präzision
  - Viele Einschränkungen und Limits
- Aufteilung des Algorithmus in 4 Schritte (4 Renderingdurchläufe)
- Nutzung von Texturen als Transportmedium

Überblick II



Eingabetextur

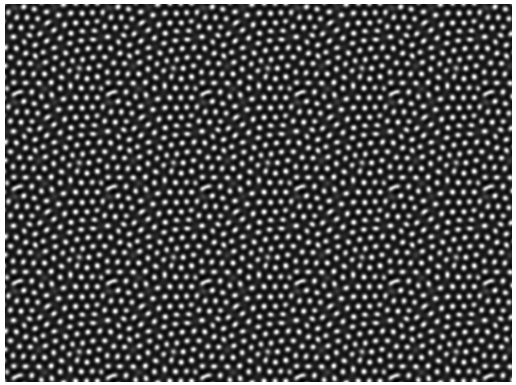


Abbildung: Gekachelte 100 mal 100 Reaction Diffusion Textur mit $D_a = 0.125$ und $D_b = 0.031$.

- Erzeugung der Eingabetextur
  - Reaction Diffusion ([Tur52])
- da Reaction Diffusion rechenaufwendig: Tiling
- Einzige Berechnung auf CPU

Reaction Diffusion

Definition (Diffusion Reaction Scheme)

Seien $a_{i,j}$ und $b_{i,j}$ zwei diskrete Felder, indiziert durch i und j und D_a bzw. D_b deren zugehörige Diffusionskonstanten.

$$\Delta a_{i,j} = F(i,j) + D_a \cdot (a_{i+1,j} + a_{i-1,j} + a_{i,j+1} + a_{i,j-1} - 4 \cdot a_{i,j}),$$
$$\Delta b_{i,j} = G(i,j) + D_b \cdot (b_{i+1,j} + b_{i-1,j} + b_{i,j+1} + b_{i,j-1} - 4 \cdot b_{i,j}),$$

mit den Funktionen F und G:

$$F(i,j) = s(16 - a_{i,j} \cdot b_{i,j}) \text{ und } G(i,j) = s(a_{i,j} \cdot b_{i,j} - b_{i,j} - \beta_{i,j}).$$

Definition (Diffusion Reaction Scheme)

Seien $a_{i,j}$ und $b_{i,j}$ zwei diskrete Felder, indiziert durch i und j und D_a bzw. D_b deren zugehörige Diffusionskonstanten.

$$\Delta a_{i,j} = F(i,j) + D_a \cdot (a_{i+1,j} + a_{i-1,j} + a_{i,j+1} + a_{i,j-1} - 4 \cdot a_{i,j}),$$
$$\Delta b_{i,j} = G(i,j) + D_b \cdot (b_{i+1,j} + b_{i-1,j} + b_{i,j+1} + b_{i,j-1} - 4 \cdot b_{i,j}),$$

mit den Funktionen F und G:

$$F(i,j) = s(16 - a_{i,j} \cdot b_{i,j}) \text{ und } G(i,j) = s(a_{i,j} \cdot b_{i,j} - b_{i,j} - \beta_{i,j}).$$

1. Laplace Operator

Geometrie und Tensoren

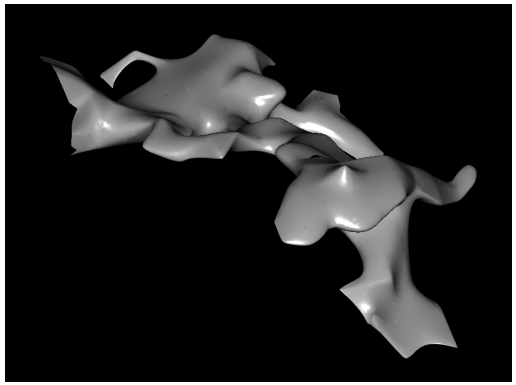


Abbildung: Eingabegeometrie mit Beleuchtung nach Phong.

- Erzeugt durch beliebigen Algorithmus
- Tensoren als zwei 3D Texturkoordinaten auf die GPU geladen
- Voraussetzung an Geometrie:
  - Glatte Normalen
  - Geometrie sollte nicht selbstschneidend sein

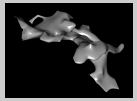
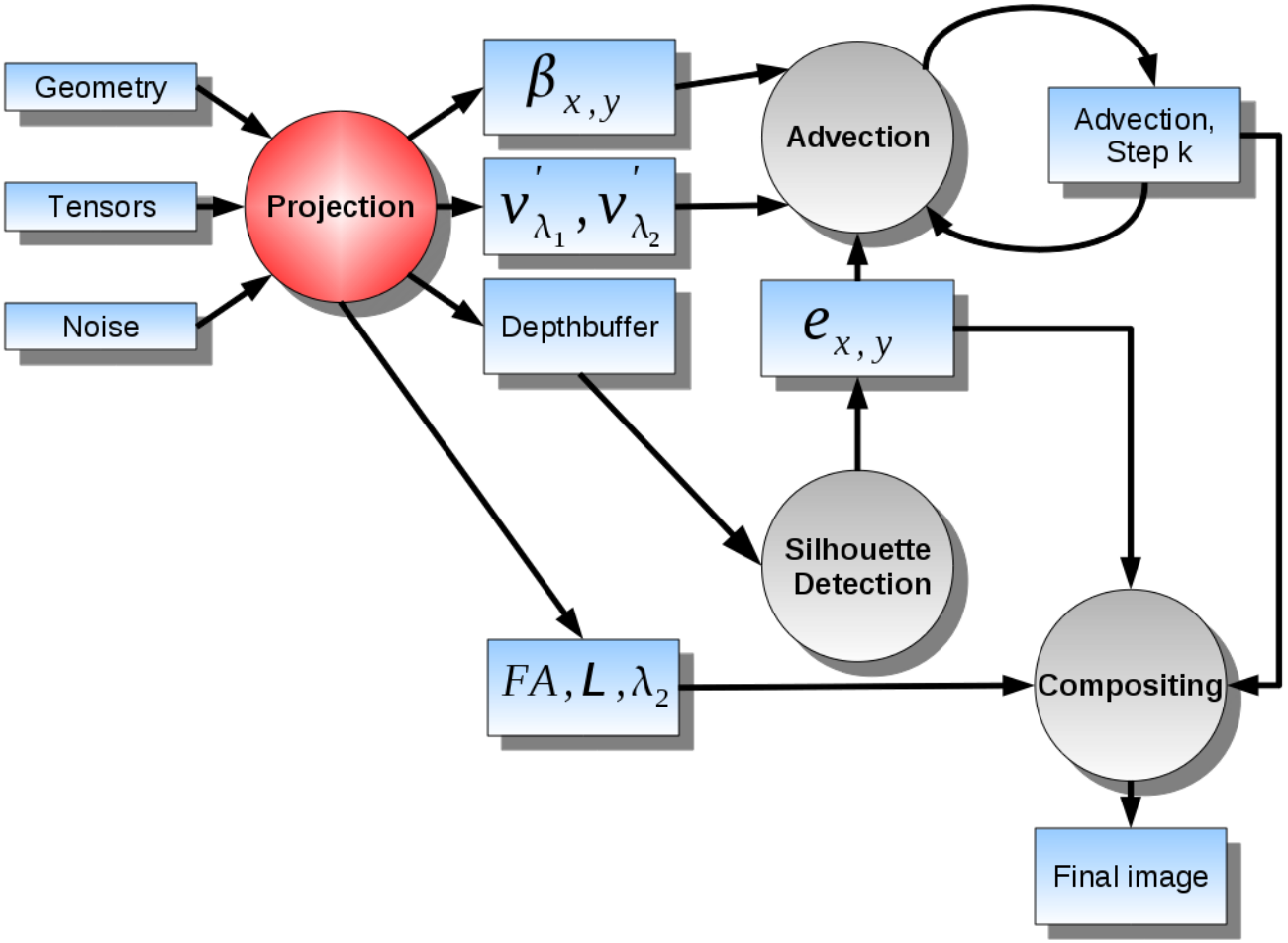


Abbildung: Eingabegeometrie mit Beleuchtung nach Phong.

- Erzeugt durch beliebigen Algorithmus
- Tensoren als zwei 3D Texturkoordinaten auf die GPU geladen
- Voraussetzung an Geometrie:
  - Glatte Normalen
  - Geometrie sollte nicht selbstschneidend sein

1. nur 2 mal 3 Werte für Tensor nötig da symmetrisch

Schritt 1: Projektion in den Bildraum



Tensorprojektion

- Tensor durch GPU komponentenweise interpoliert

- Tensor durch GPU komponentenweise interpoliert

1. nur 2 Eigenvektoren, da 3. Eigenvek = Normale

Tensorprojektion

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:

$$T' = P \cdot T \cdot P^T \text{ mit } P = \begin{pmatrix} 1 - n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1 - n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1 - n_z^2 \end{pmatrix}.$$

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:

$$T' = P \cdot T \cdot P^T \text{ mit } P = \begin{pmatrix} 1-n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1-n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1-n_z^2 \end{pmatrix}.$$

1. nur 2 Eigenvektoren, da 3. Eigenvek = Normale

Tensorprojektion

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:

$$T' = P \cdot T \cdot P^T \text{ mit } P = \begin{pmatrix} 1 - n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1 - n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1 - n_z^2 \end{pmatrix}.$$

- Eigenwertzerlegung nach Hasan et al. [HBPA01]
  - Eigenwerte: λ_i mit $i \in \{1, 2\}$
  - Eigenvektoren: v_{λ_i} mit $i \in \{1, 2\}$

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:
  $T' = P \cdot T \cdot P^T$ mit $P = \begin{pmatrix} 1-n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1-n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1-n_z^2 \end{pmatrix}$.
- Eigenwertzerlegung nach Hasan et al. [HBPA01]
  - Eigenwerte: λ_i mit $i \in \{1,2\}$
  - Eigenvektoren: v_{λ_i} mit $i \in \{1,2\}$

1. nur 2 Eigenvektoren, da 3. Eigenvek = Normale

Tensorprojektion

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:

$$T' = P \cdot T \cdot P^T \text{ mit } P = \begin{pmatrix} 1 - n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1 - n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1 - n_z^2 \end{pmatrix}.$$

- Eigenwertzerlegung nach Hasan et al. [HBPA01]
  - Eigenwerte: λ_i mit $i \in \{1, 2\}$
  - Eigenvektoren: v_{λ_i} mit $i \in \{1, 2\}$
- Eigenvektoren noch immer im Objektraum der Geometrie

- Tensor durch GPU komponentenweise interpoliert
- Tensor auf die Ebene der Geometrie projizieren damit 3. Eigenvektor in Richtung der Normalen n zeigt:
  $T' = P \cdot T \cdot P^T$ mit $P = \begin{pmatrix} 1-n_x^2 & -n_y n_x & -n_z n_x \\ -n_x n_y & 1-n_y^2 & -n_z n_y \\ -n_x n_z & -n_y n_z & 1-n_z^2 \end{pmatrix}$.
- Eigenwertzerlegung nach Hasan et al. [HBPA01]
  - Eigenwerte: λ_i mit $i \in \{1,2\}$
  - Eigenvektoren: v_{λ_i} mit $i \in \{1,2\}$
- Eigenvektoren noch immer im Objektraum der Geometrie

1. nur 2 Eigenvektoren, da 3. Eigenvek = Normale

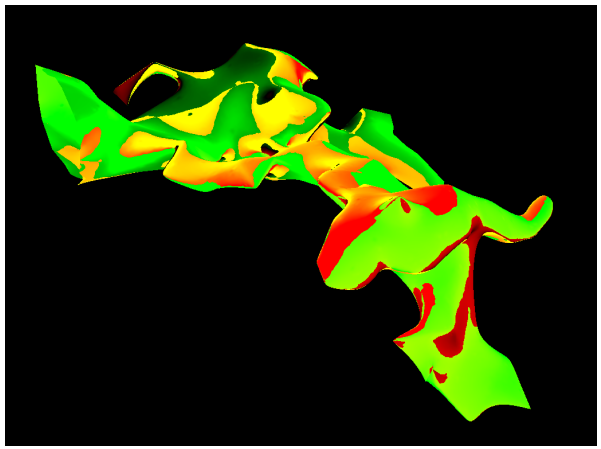
Tensorprojektion

- Projektion in Bildraum mittels OpenGL Modelviewmatrix M_M und Projectionmatrix M_P:
  $v'_{\lambda_i} = M_P \times M_M \times v_{\lambda_i}$, mit $(i \in 1, 2)$ und $v'_{\lambda_i} \in \mathbb{R}^2$

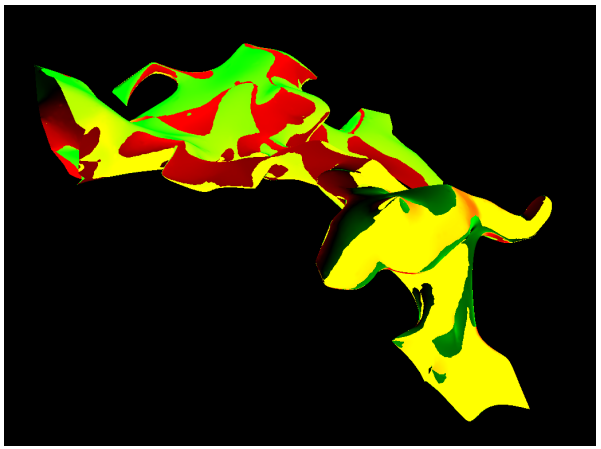
Tensorprojektion

- Projektion in Bildraum mittels OpenGL Modelviewmatrix M_M und Projectionmatrix M_P:
  $v'_{\lambda_i} = M_P \times M_M \times v_{\lambda_i}$, mit $(i \in 1, 2)$ und $v'_{\lambda_i} \in \mathbb{R}^2$
- Skalierung auf $[0, 1]$:
  $v''_{\lambda_i} = \frac{1}{2} + \frac{1}{2} * \frac{v'_{\lambda_i}}{\|v'_{\lambda_i}\|_\infty}$ mit $i \in \{1, 2\}$ und $\|v'_{\lambda_i}\|_\infty \neq 0$

Texturen (Tensorprojektion)

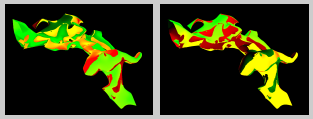


(a) v''_{λ_1}



(b) v''_{λ_2}

Abbildung: v''_{λ_1} und v''_{λ_2} separat in 2 Texturen zur Illustration.

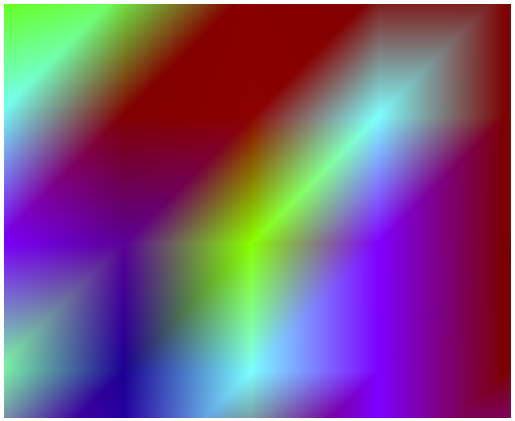


(a) v''_{λ_1} (b) v''_{λ_2}

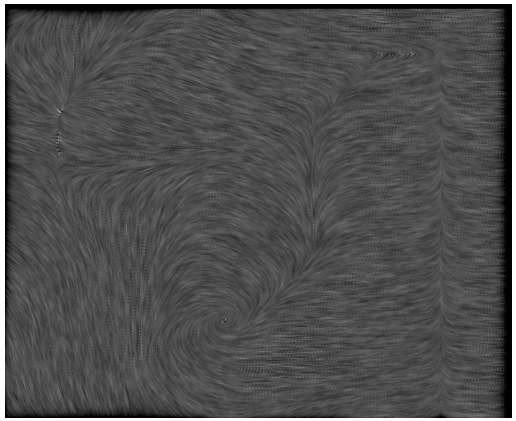
Abbildung: v''_{λ_1} und v''_{λ_2} separat in 2 Texturen zur Illustration.

1. Eigenvektoren dann nicht mehr orthogonal

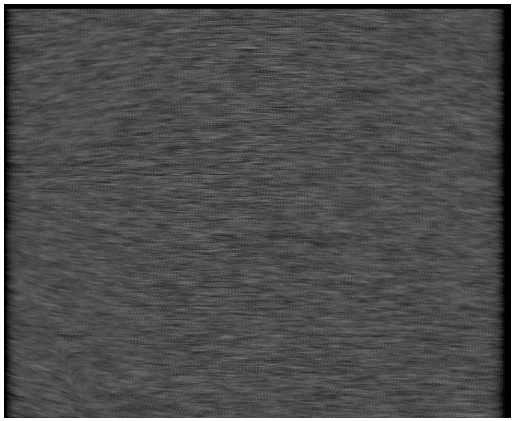
Texturen (Tensorinterpolation)



(a) Eigenvektoren



(b) LIC



(c) LIC (korrekt)

Abbildung: Bei vertexweiser Eigenwert/Eigenvektor- Zerlegung entstehen Probleme durch umschaltende Orientierungen von Eigenvektoren an benachbarten Vertex.

Abbildung der Noise Textur

- Viele Ansätze verfügbar
  - Texture Atlases ([PCK04, IOK00])
  - Reaction Diffusion direkt auf der Geometrie ([Tur91])
  - 3D Textur in der Domain des Datensatzes ([WE04])
  - aber meist zu Rechenintensiv oder zu Geometrieabhängig

Abbildung der Noise Textur

- Viele Ansätze verfügbar
  - Texture Atlases ([PCK04, IOK00])
  - Reaction Diffusion direkt auf der Geometrie ([Tur91])
  - 3D Textur in der Domain des Datensatzes ([WE04])
  - aber meist zu Rechenintensiv oder zu Geometrieabhängig
- Eigene Heuristik entwickelt
  - Nicht C^1 stetig
  - Kann also zu irrelevanten Verzerrungen führen
  - Aber durch Verwischung im Schritt 3 nicht von Bedeutung
  - Außerdem erlaubt sie nahtlose Skalierung (Anpassung an LOD)

Abbildung der Noise Textur

- Zunächst Transformation des Vertex in *voxelisierten* Raum: $v_{voxel} = v_g \cdot \begin{pmatrix} l & 0 & 0 & -b_{min_x} \\ 0 & l & 0 & -b_{min_y} \\ 0 & 0 & l & -b_{min_z} \\ 0 & 0 & 0 & 0 \end{pmatrix}$
- l ist dabei Größe eines Voxels und b_{min} der Ursprung in Weltkoordinaten

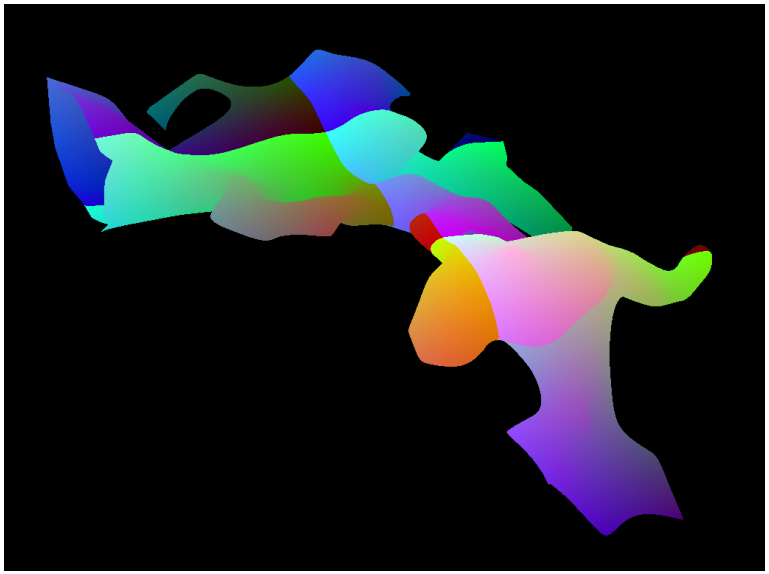
Abbildung der Noise Textur

- Zunächst Transformation des Vertex in *voxelisierten* Raum: $v_{voxel} = v_g \cdot \begin{pmatrix} l & 0 & 0 & -b_{min_x} \\ 0 & l & 0 & -b_{min_y} \\ 0 & 0 & l & -b_{min_z} \\ 0 & 0 & 0 & 0 \end{pmatrix}$
- l ist dabei Größe eines Voxels und b_{min} der Ursprung in Weltkoordinaten
- Abbildung auf die Ebene des Voxels mit $v_{hit} = v_{voxel} - \lfloor v_{voxel} \rfloor$

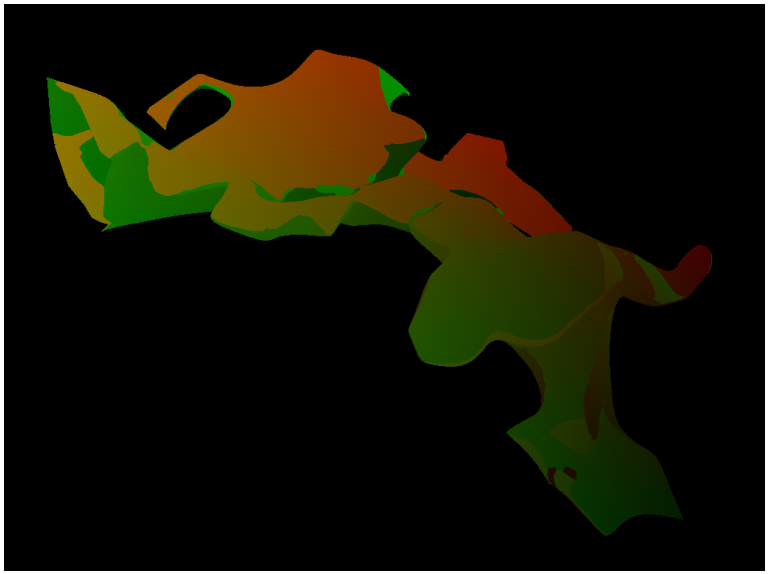
Abbildung der Noise Textur

- Zunächst Transformation des Vertex in *voxelisierten* Raum: $v_{voxel} = v_g \cdot \begin{pmatrix} l & 0 & 0 & -b_{min_x} \\ 0 & l & 0 & -b_{min_y} \\ 0 & 0 & l & -b_{min_z} \\ 0 & 0 & 0 & 0 \end{pmatrix}$
- l ist dabei Größe eines Voxels und b_{min} der Ursprung in Weltkoordinaten
- Abbildung auf die Ebene des Voxels mit $v_{hit} = v_{voxel} - \lfloor v_{voxel} \rfloor$
- Texturkoordinate t dann definiert als: $t = (v_{hit_i}, v_{hit_j})$, mit $i \neq j \neq k \wedge (n_k = max\{n_i, n_j, n_k\})$
  - Nutzt also die Komponenten von v_{hit} bei denen die entsprechenden Komponenten der Normale n nicht maximal ist

Texturen (Noise Mapping)



(a) v_{hit}



(b) t

Abbildung: Zwischenergebnisse zur Illustration in separaten Texturen.

Texturen (Noise Mapping)

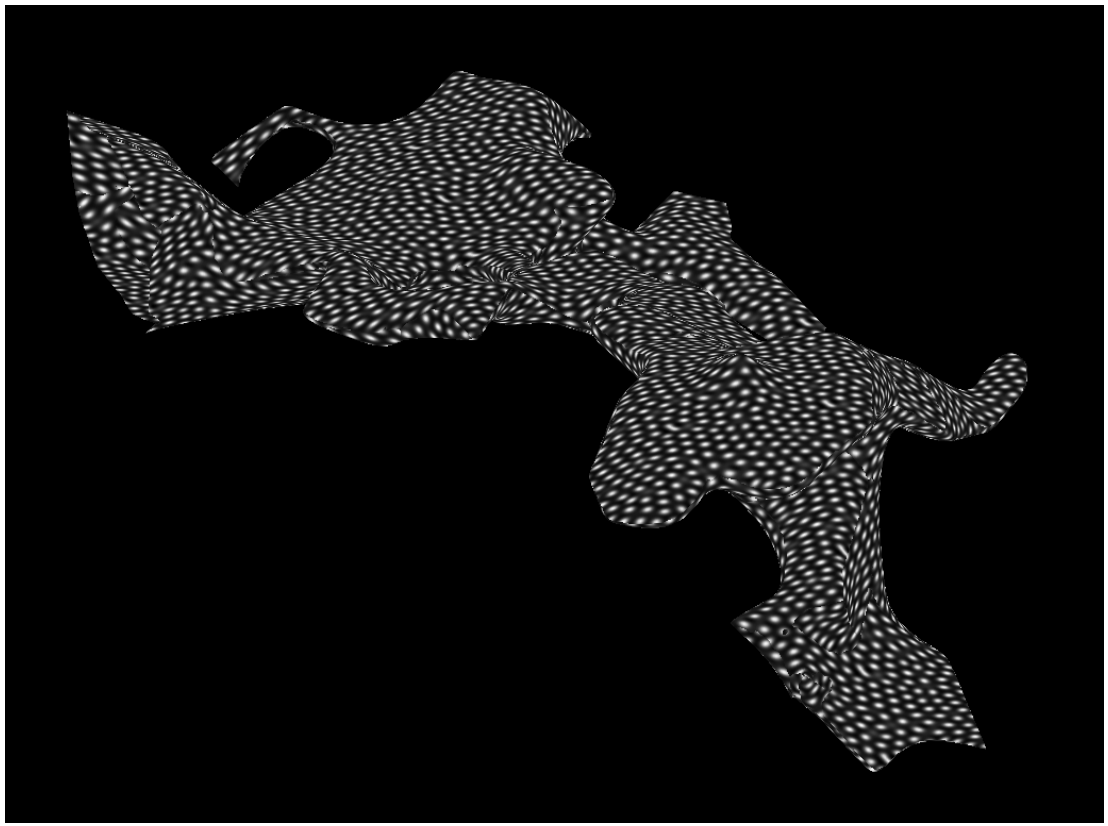


Abbildung: Die auf die Oberfläche abgebildete Rauschtextur $\beta_{i,j}$.

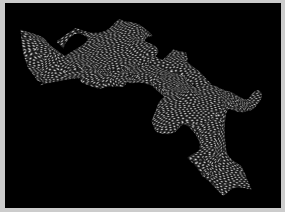


Abbildung: Die auf die Oberfläche abgebildete Rauschtextur $\beta_{i,j}$.

1. ist Transformationsunempfindlich
2. Schon jetzt quasi keine Artefakte zu sehen

Weitere Berechnungen

- Beleuchtungsintensität $\mathcal{L}$ mittels Phong Modell

Weitere Berechnungen

- Beleuchtungsintensität $\mathcal{L}$ mittels Phong Modell
- Mean Diffusivity

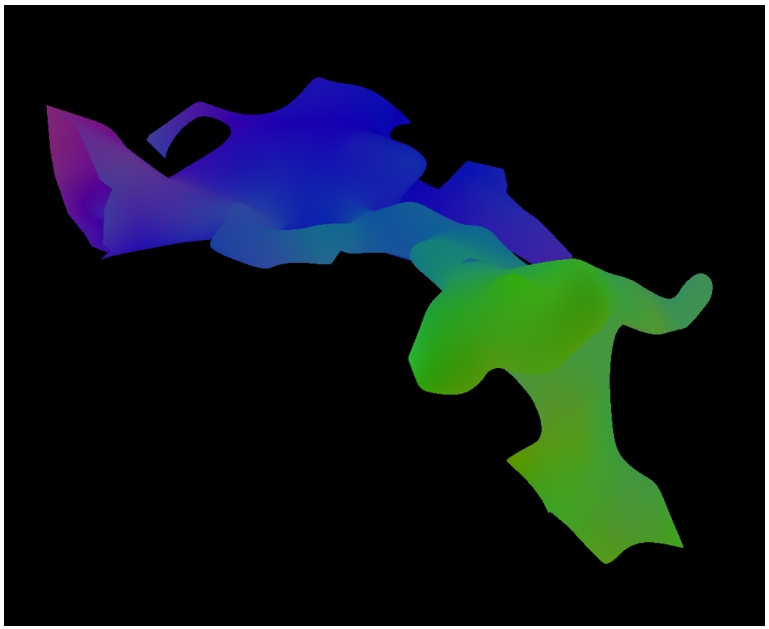
Weitere Berechnungen

- Beleuchtungsintensität $\mathcal{L}$ mittels Phong Modell
- Mean Diffusivity
- Fractional Anisotropy

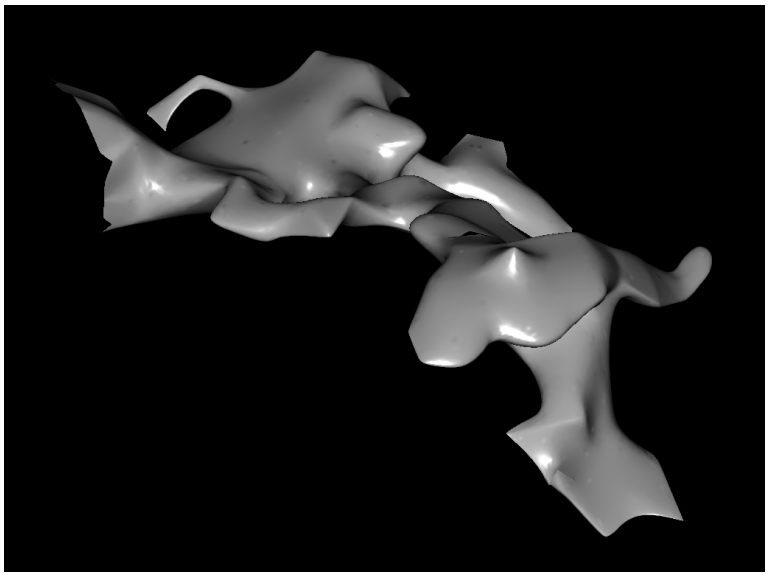
Weitere Berechnungen

- Beleuchtungsintensität $\mathcal{L}$ mittels Phong Modell
- Mean Diffusivity
- Fractional Anisotropy
- Colormapping: $c^{FA \cdot v_{\lambda_i}}(T) = \frac{|v_{\lambda_i}|}{\|v_{\lambda_i}\|} * FA(T)$

Texturen (weitere Berechnungen)



(a) $c^{FA \cdot v_{\lambda_i}}$



(b) $\mathcal{L}$

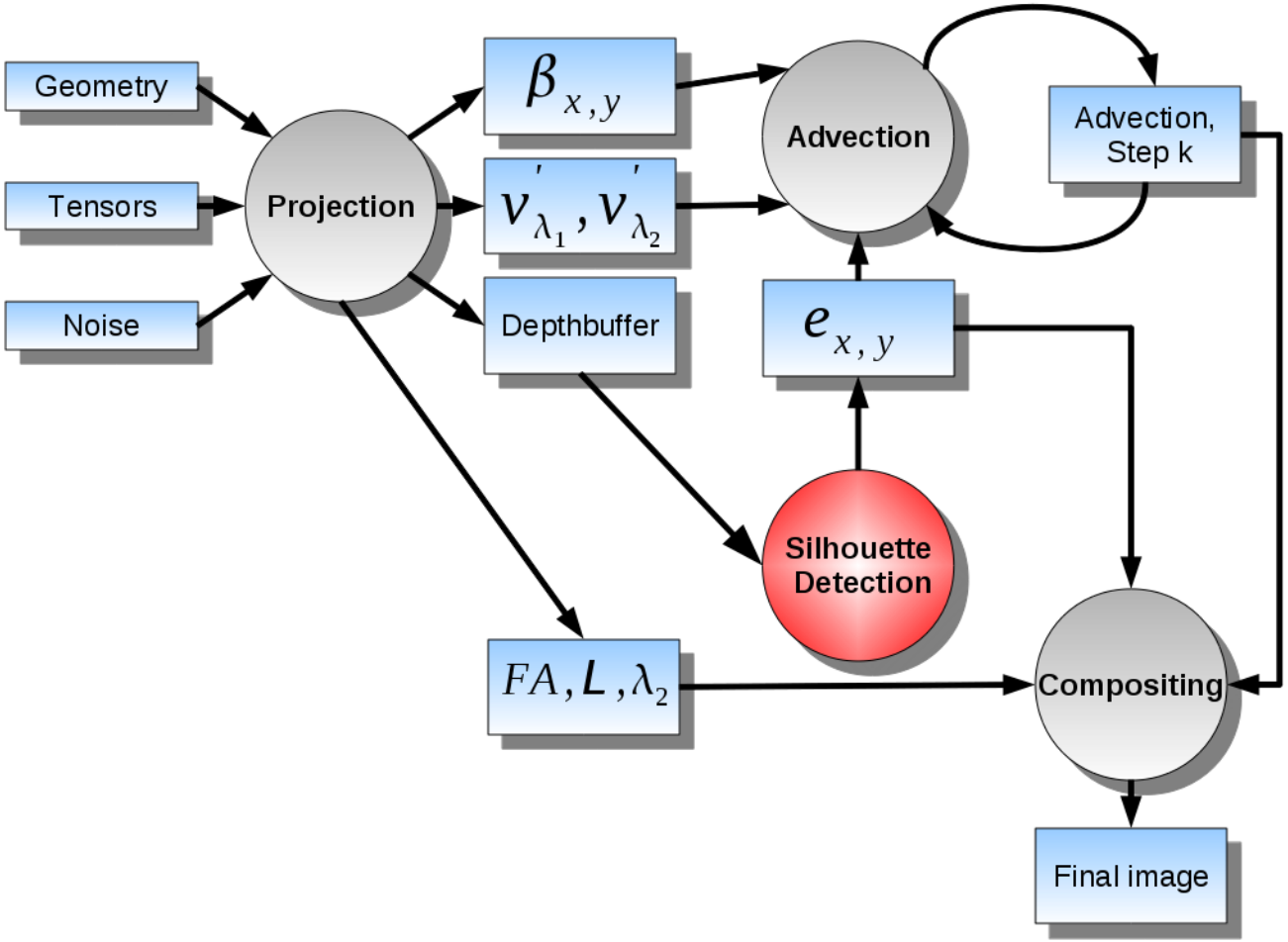
Abbildung: Beispielhaft die erzeugte Colormap und die Beleuchtung.

Ausgaben (Übersicht)

Wert	Textur	Kanal
v''_{λ_1} v''_{λ_2}	1	R,G B,A
FA $\mathcal{L}$ λ_2 β_{t_x,t_y}	2	R G B A
$c^{FA \cdot v_{\lambda_1}}$ MD	3	RGB A
depth buffer	4	L

Tabelle: Anordnung der Daten in verschiedenen Texturen.

Schritt 2: Kantendetektion



Schritt 2: Kantendetektion

- Depthbuffer als Eingabetextur benutzt

Schritt 2: Kantendetektion

- Depthbuffer als Eingabetextur benutzt
- Pixelweise Anwendung des Laplace Filter Kernels:

$$D^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Schritt 2: Kantendetektion

- Depthbuffer als Eingabetextur benutzt
- Pixelweise Anwendung des Laplace Filter Kernels:

$$D^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Erzeugt Kantentextur: $e : (x, y) \rightarrow s$, mit $x, y, s \in [0, 1]$

Schritt 2: Kantendetektion

- Depthbuffer als Eingabetextur benutzt
- Pixelweise Anwendung des Laplace Filter Kernels:

$$D^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Erzeugt Kantentextur: $e : (x, y) \rightarrow s$, mit $x, y, s \in [0, 1]$
- In Praxis werden hier einige der Daten des Schritt 1 zu einer Textur zusammengefügt
  - Da Anzahl der Eingabetexturen durch Hardware limitiert
  - Spart im letzten Schritt eine Textur
  - Depthbuffer wird in Kantentextur gespeichert

Texturen (Kantenerkennung)

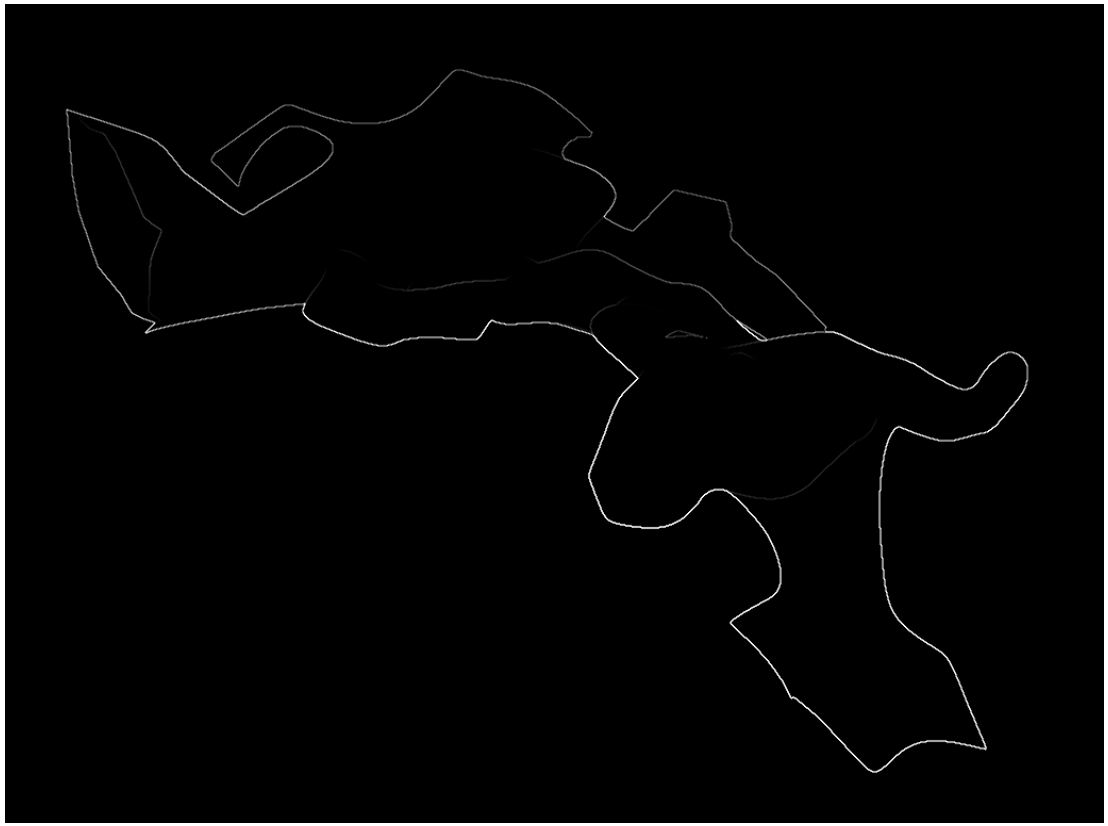
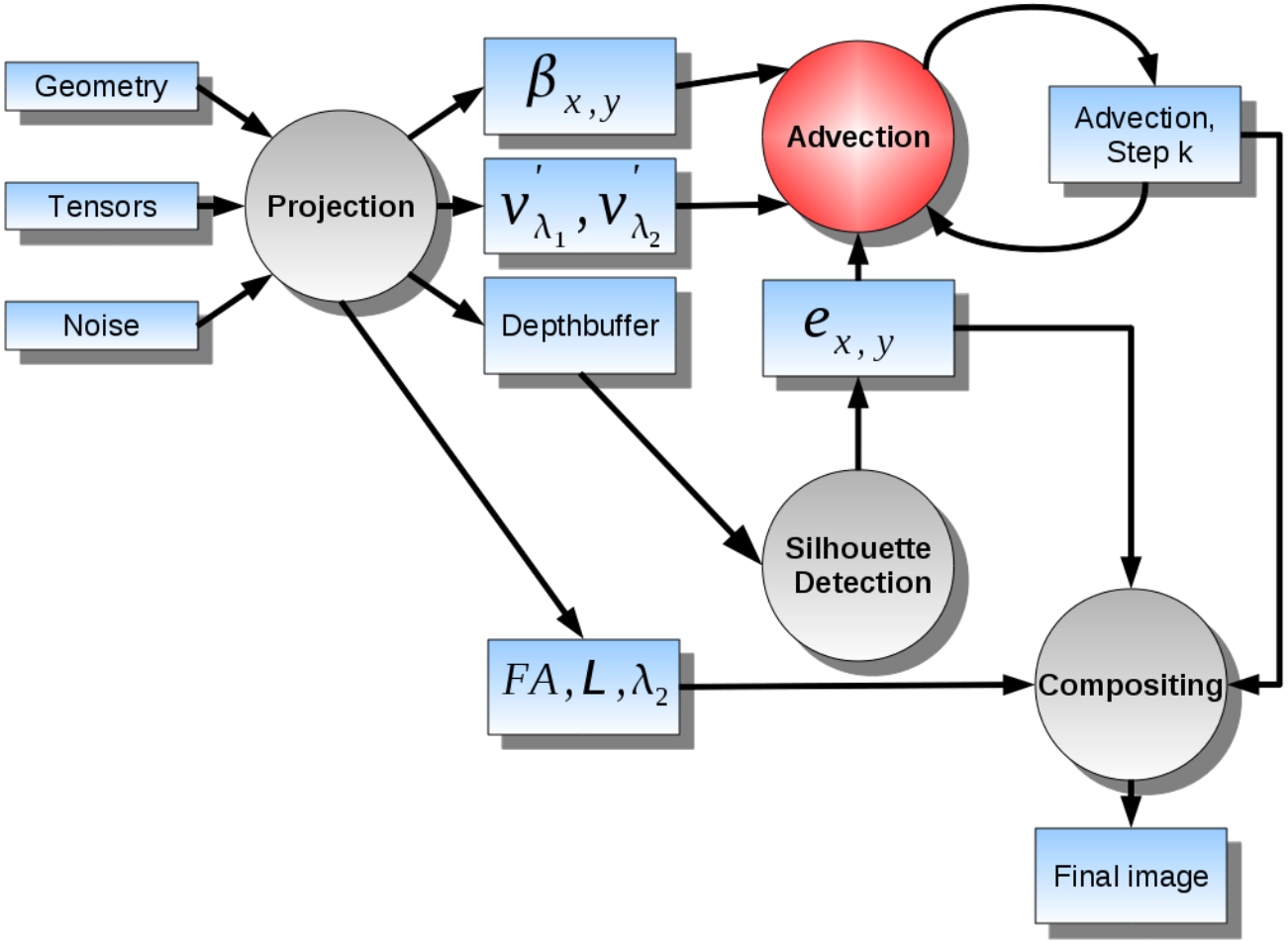


Abbildung: Silhouette der Geometrie, errechnet aus Tiefenbuffer.

Schritt 3: Advektion



Advektion

- Zugriff auf die diskreten LIC- Texturen P^{λ_1} und P^{λ_2} über Funktion: $f_P : (x, y) \rightarrow p$, mit $x, y, p \in [0, 1]$
  - Interpolation

Advektion

- Zugriff auf die diskreten LIC- Texturen P^{λ_1} und P^{λ_2} über Funktion: $f_P : (x, y) \longrightarrow p$, mit $x, y, p \in [0, 1]$
  - Interpolation
- Iteration auf beiden Feldern für jedes Pixel:

$$\forall x, y \in [0,1] : \forall \lambda \in \{\lambda_1, \lambda_2\} :$$

$$p_0^{\lambda} = \beta_{x,y},$$

$$p_{i+1}^{\lambda} = k \cdot \beta_{x,y} + (1-k) \cdot \frac{f_{p_i^{\lambda}}(x + v'_{\lambda_x}, y + v'_{\lambda_y}) + f_{p_i^{\lambda}}(x - v'_{\lambda_x}, y - v'_{\lambda_y})}{2}.$$

Advektion

- Zugriff auf die diskreten LIC- Texturen P^{λ_1} und P^{λ_2} über Funktion: $f_P : (x, y) \longrightarrow p$, mit $x, y, p \in [0, 1]$
  - Interpolation
- Iteration auf beiden Feldern für jedes Pixel:

$$\forall x, y \in [0, 1] : \forall \lambda \in \{\lambda_1, \lambda_2\} :$$

$$p_0^{\lambda} = \beta_{x,y},$$

$$p_{i+1}^{\lambda} = k \cdot \beta_{x,y} + (1 - k) \cdot \frac{f_{p_i^{\lambda}}(x + v'_{\lambda_x}, y + v'_{\lambda_y}) + f_{p_i^{\lambda}}(x - v'_{\lambda_x}, y - v'_{\lambda_y})}{2}.$$

  - k beschreibt die "Rauhheit"(je kleiner k je weicher und verwaschener wirkt das zusammengesetzte Bild)

Advektion

- Zugriff auf die diskreten LIC- Texturen P^{λ_1} und P^{λ_2} über Funktion: $f_P : (x, y) \longrightarrow p$, mit $x, y, p \in [0, 1]$
  - Interpolation
- Iteration auf beiden Feldern für jedes Pixel:

$$\forall x, y \in [0, 1] : \forall \lambda \in \{\lambda_1, \lambda_2\} :$$

$$p_0^{\lambda} = \beta_{x,y},$$

$$p_{i+1}^{\lambda} = k \cdot \beta_{x,y} + (1 - k) \cdot \frac{f_{p_i^{\lambda}}(x + v'_{\lambda_x}, y + v'_{\lambda_y}) + f_{p_i^{\lambda}}(x - v'_{\lambda_x}, y - v'_{\lambda_y})}{2}.$$

  - k beschreibt die "Rauhheit"(je kleiner k je weicher und verwaschener wirkt das zusammengesetzte Bild)
  - Advektion muss in beide Richtungen erfolgen, da Eigenvektoren keine Orientierung haben

Advektion

- Zugriff auf die diskreten LIC- Texturen P^{λ_1} und P^{λ_2} über Funktion: $f_P : (x, y) \longrightarrow p$, mit $x, y, p \in [0, 1]$
  - Interpolation
- Iteration auf beiden Feldern für jedes Pixel:

$$\forall x, y \in [0, 1] : \forall \lambda \in \{\lambda_1, \lambda_2\} :$$

$$p_0^{\lambda} = \beta_{x,y},$$

$$p_{i+1}^{\lambda} = k \cdot \beta_{x,y} + (1 - k) \cdot \frac{f_{p_i^{\lambda}}(x + v'_{\lambda_x}, y + v'_{\lambda_y}) + f_{p_i^{\lambda}}(x - v'_{\lambda_x}, y - v'_{\lambda_y})}{2}.$$

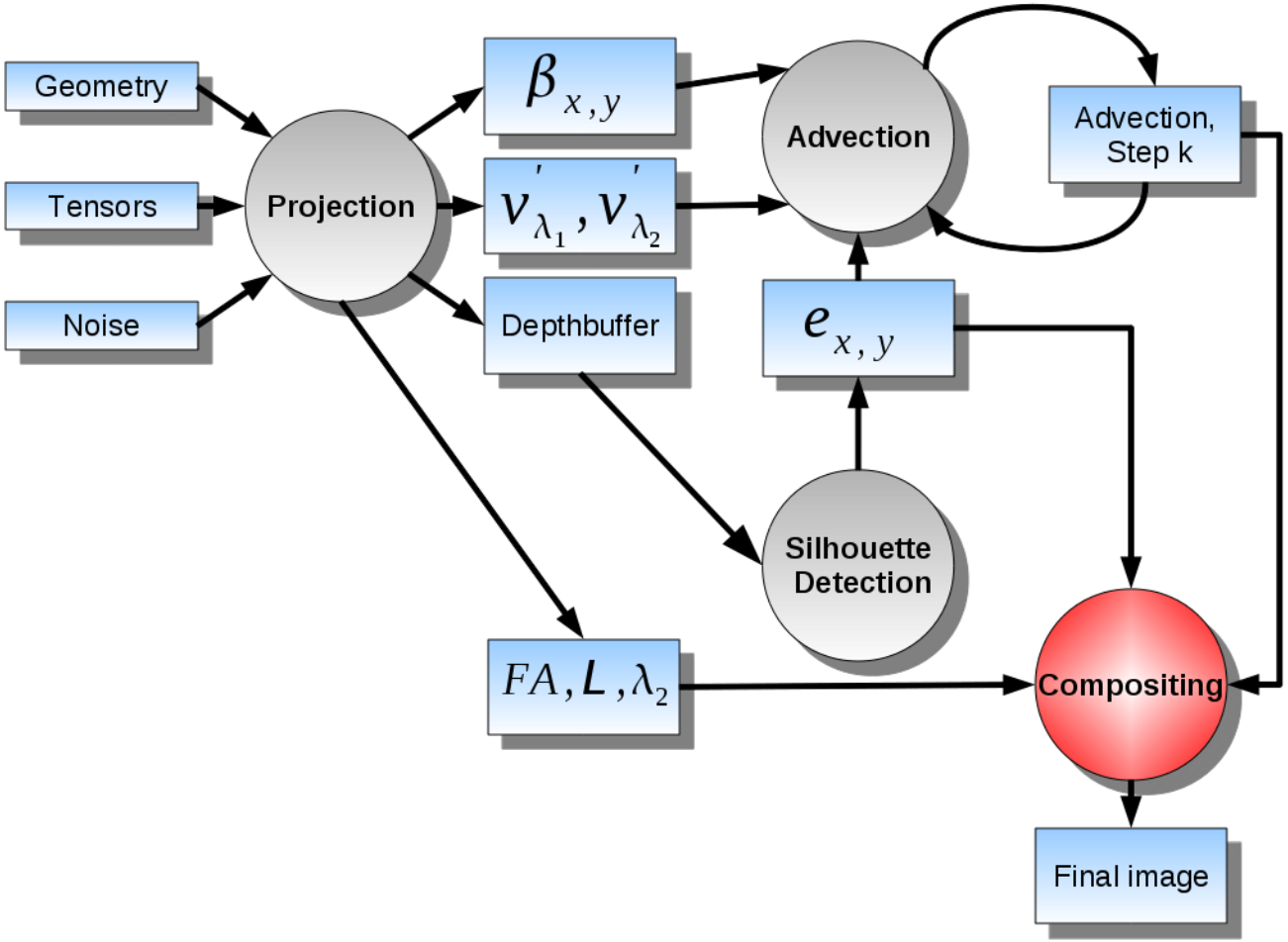
  - k beschreibt die "Rauhheit"(je kleiner k je weicher und verwaschener wirkt das zusammengesetzte Bild)
  - Advektion muss in beide Richtungen erfolgen, da Eigenvektoren keine Orientierung haben
  - Iteration abbrechen wenn keine Änderung mehr auftritt (Schwellwert)

Texturen (Iteration)

(a) $P_1^{\lambda_1}$ (b) $P_{10}^{\lambda_1}$ (c) $p_{100}^{\lambda_1}$ mit $k = 0.01$

Abbildung: Advektion der Eingabetextur entlang der Eigenvektoren v'_{λ_1}. Nach wenigen Schritten konvergiert die Iteration bereits. Bei kleinerem k dauert signifikant die Iteration länger bis Konvergenz eintritt.

Schritt 4: Bildkomposition



Schritt 4: Bildkomposition

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden

Nicht bei Projektion clippen, da zu viele kleine Areale das Gesamtbild beeinflussen

Schritt 4: Bildkomposition

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich

Nicht bei Projektion clippen, da zu viele kleine Areale das Gesamtbild beeinflussen

Schritt 4: Bildkomposition

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen

Nicht bei Projektion clippen, da zu viele kleine Areale das Gesamtbild beeinflussen

Schritt 4: Bildkomposition

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen
- Depth-enhancing ([CCG+08]) für bessere Plastizität

- Finaler Schritt nach i Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen
- Depth-enhancing ([CCG+08]) für bessere Plastizität

Nicht bei Projektion clippen, da zu viele kleine Areale das Gesamtbild beeinflussen

Schritt 4: Bildkomposition

- Finaler Schritt nach *i* Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen
- Depth-enhancing ([CCG+08]) für bessere Plastizität
- Hohe Flexibilität, viele Möglichkeiten
  - z.Bsp. einblenden einer Colormap

- Finaler Schritt nach i Iterationen
  - kann auch auf „unfertigen“ Advektionstexturen ausgeführt werden
- Clipping auf Basis von MD oder FA möglich
- Depthbuffer Informationen setzen
- Depth-enhancing ([CCG+08]) für bessere Plastizität
- Hohe Flexibilität, viele Möglichkeiten
  - z.Bsp. einblenden einer Colormap

Nicht bei Projektion clippen, da zu viele kleine Areale das Gesamtbild beeinflussen

Schritt 4: Bildkomposition

- Farbwert für jedes Pixel mit den Advektionsfeldern $P_i^{\lambda_1}$ und $P_i^{\lambda_2}$:

$$R = \frac{r \cdot f_{p_k^{\lambda_2}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_1}}(x,y)} + e_{x,y} + light(\mathcal{L}_{x,y}),$$

$$G = \frac{(1-r) \cdot f_{p_k^{\lambda_1}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_2}}(x,y)} + e_{x,y} + light(\mathcal{L}_{x,y}), \text{ und}$$

$$B = e_{x,y} + light(\mathcal{L}_{x,y}).$$

  - Kantentextur: e und Licht: $\mathcal{L}$
  - r bestimmt Verhältnis beider Felder im Gesamtbild

- Farbwert für jedes Pixel mit den Advektionsfeldern $P_l^{\lambda_1}$ und $P_l^{\lambda_2}$:

$$R = \frac{r \cdot f_{p_k^{\lambda_2}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_1}}(x,y)} + e_{x,y} + light(\mathcal{L}_{x,y}),$$

$$G = \frac{(1-r) \cdot f_{p_k^{\lambda_1}}(x,y)}{8 \cdot f^2_{p_k^{\lambda_2}}(x,y)} + e_{x,y} + light(\mathcal{L}_{x,y}), \text{ und}$$

$$B = e_{x,y} + light(\mathcal{L}_{x,y}).$$

  - Kantentextur: e und Licht: $\mathcal{L}$
  - r bestimmt Verhältnis beider Felder im Gesamtbild

1. *light* wird benutzt um den Einfluss des Lichts zu steuern um Netz- Farbgebung nicht zu beeinflussen

Texturen (Komposition)

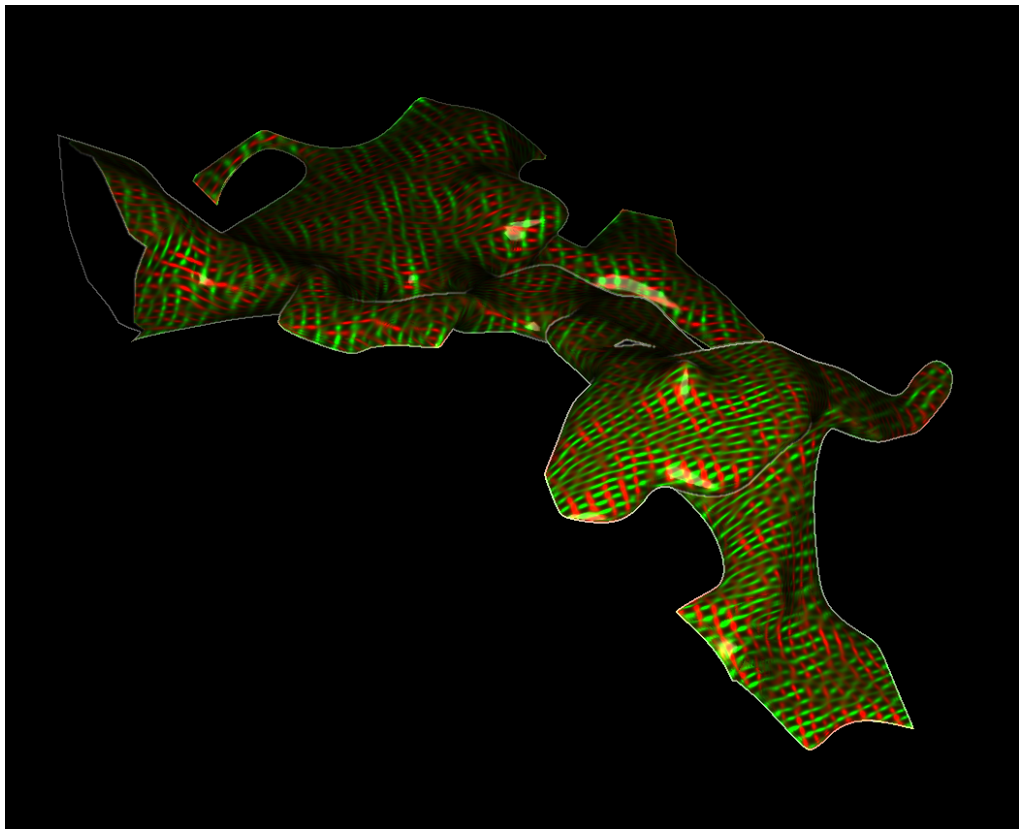


Abbildung: Die Diffusionsrichtungen deutlich durch Maschenmuster ergennbar.

Texturen (Komposition)

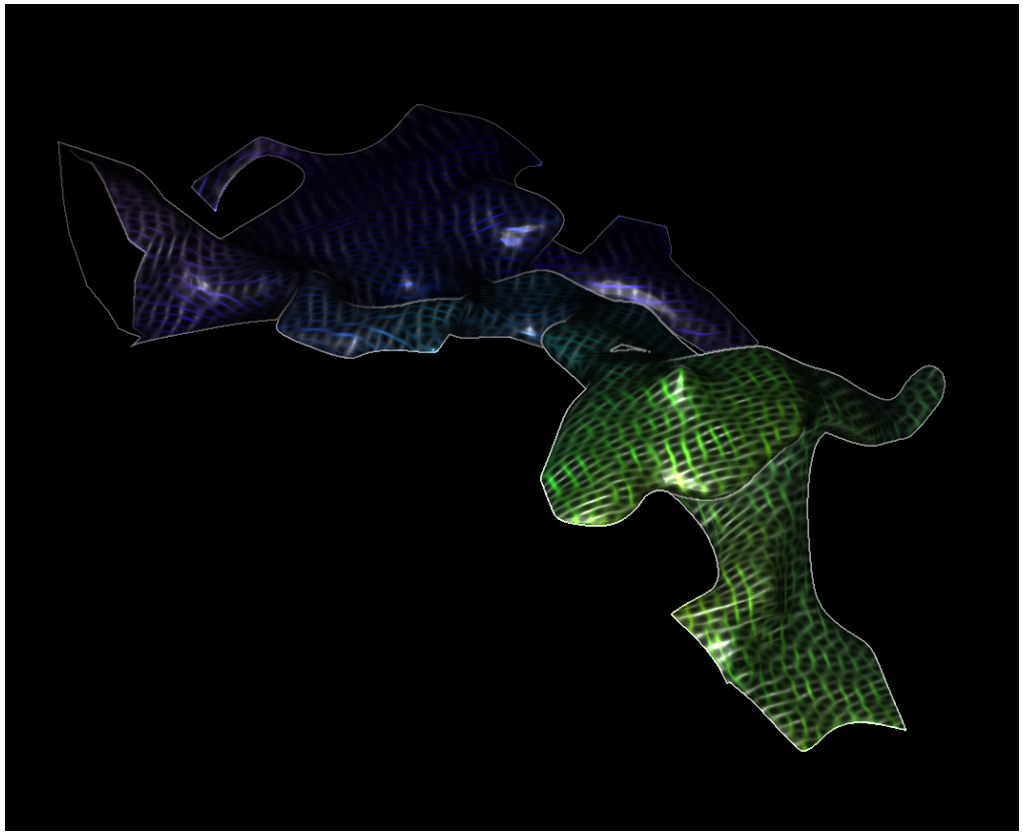


Abbildung: Wie oben, aber mit $c^{FA \cdot v_{\lambda_1}}$ Colormap.

Gliederung

Ergebnisse

- Dynamisch anpassbare Detailstufe

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich
- Strukturen unter Transformation konsistent

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich
- Strukturen unter Transformation konsistent

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich
- Strukturen unter Transformation konsistent
- Strukturen deutlich erkennbar durch „Maschenmuster”

Ergebnisse

- Dynamisch anpassbare Detailstufe
- Performance unabhängig von Größe der Eingabedaten
  - Engpass stellt Fähigkeit der Grafikkarte dar die Geometrie einmal zu rendern.
- Keine Einschränkungen an die Geometrie
  - Durch beliebiges Verfahren mit beliebiger Metrik erzeugbar (vgl. PBM)
  - Auch implizite Oberflächen nutzbar (Raytracing auf der GPU)
- Nicht auf DTI begrenzt
  - z. Bsp. Point Load Datensätze o.a. Second Order Datensätze möglich
- Strukturen unter Transformation konsistent
- Strukturen deutlich erkennbar durch „Maschenmuster”

1. Dadurch keine Einschränkungen auf Ausschnitte der Daten und kein Informationsüberfluss bei großen Ausschnitten

Performance (Rahmenbedingungen)

- Messung auf:
  - AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (512K L2 Cache)
  - NVIDIA G80 GPU (GeForce 8800 GTS) und 640MB Grafikspeicher
  - bei 1024×768 Pixel

- Messung auf:
  - AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (512K L2 Cache)
  - NVIDIA G80 GPU (GeForce 8800 GTS) und 640MB Grafikspeicher
  - bei 1024 × 768 Pixel

1. Da man schritte abkürzen kann wenn keine Transformation->einheitliche Definition eines Frames

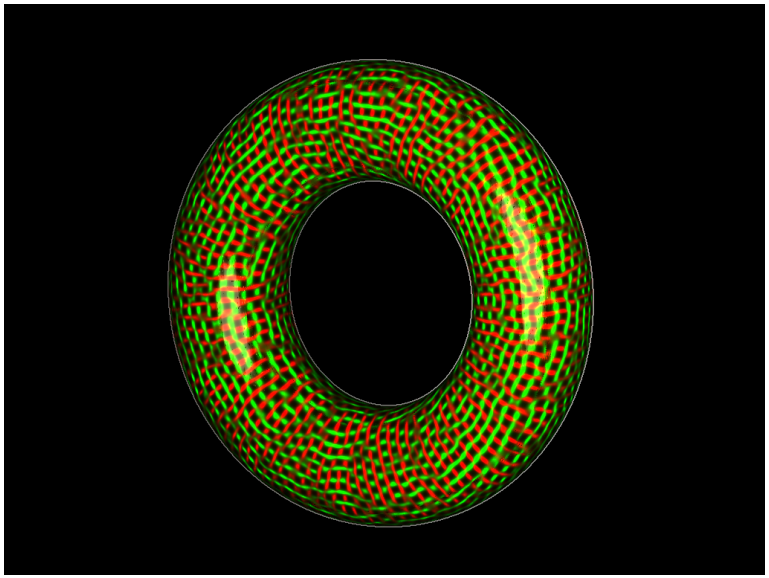
Performance (Rahmenbedingungen)

- Messung auf:
  - AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (512K L2 Cache)
  - NVIDIA G80 GPU (GeForce 8800 GTS) und 640MB Grafikspeicher
  - bei 1024×768 Pixel
- Schritte pro Frame (simuliert also permanente Benutzerinteraktion)
  - Schritt 1; inklusive initialem rendern der Geometrie
  - Schritt 2
  - 3 Iterationen pro Frame in Schritt 3
  - Schritt 4 zur finalen Ausgabe

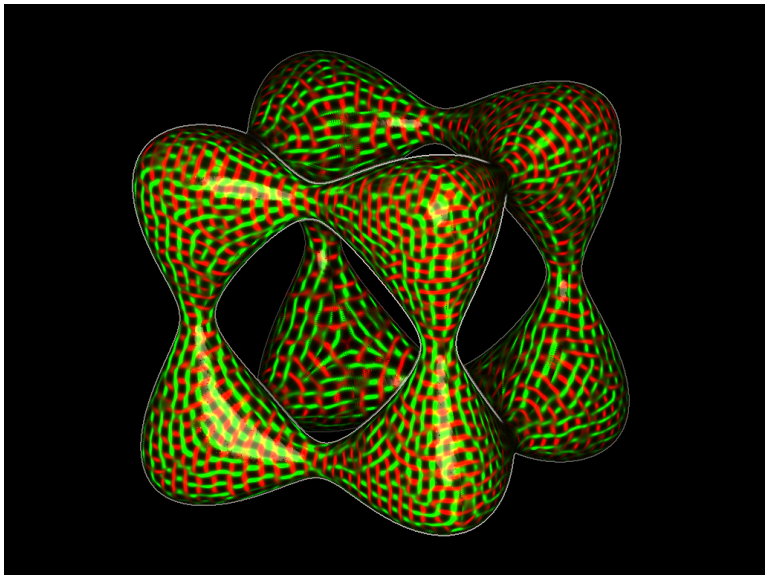
- Messung auf:
  - AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (512K L2 Cache)
  - NVIDIA G80 GPU (GeForce 8800 GTS) und 640MB Grafikspeicher
  - bei 1024 × 768 Pixel
- Schritte pro Frame (simuliert also permanente Benutzerinteraktion)
  - Schritt 1; inklusive initialem rendern der Geometrie
  - Schritt 2
  - 3 Iterationen pro Frame in Schritt 3
  - Schritt 4 zur finalen Ausgabe

1. Da man schritte abkürzen kann wenn keine Transformation->einheitliche Definition eines Frames

Bilder (Berechnete Datensätze)

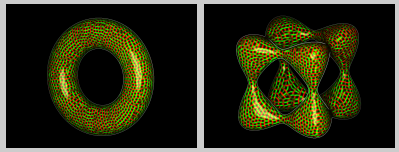


(a) Torus



(b) Tangle

Abbildung: Implizite, C^1 stetige Oberflächen ([KHH+07])



(a) Torus (b) Tangle

Abbildung: Implizite, C^1 stetige Oberflächen ([KHH+07])

1. Tensorfeld ist 2 Fach angewandter Gradientenfilter

Bilder (DTI)

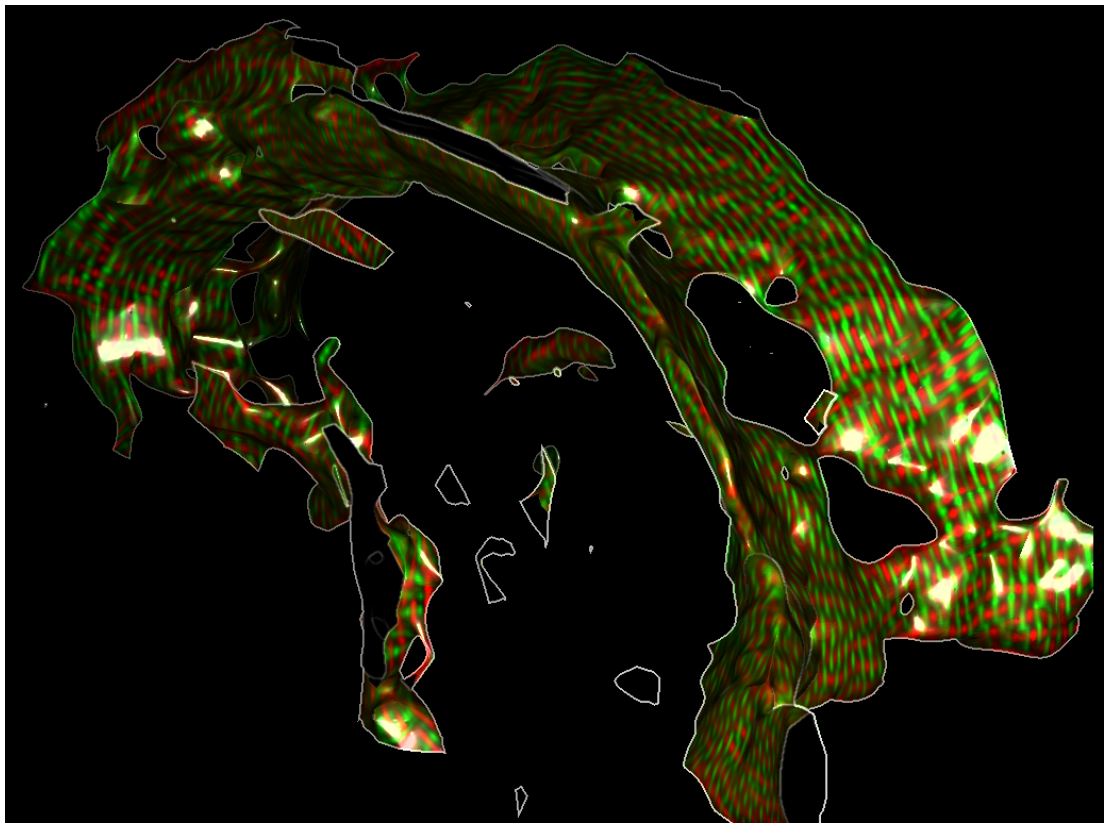


Abbildung: Corpus Callosum aus einem DTI Datensatz. (571776 Dreiecke, 14 FPS (Geometrie: 90%))

Bilder (DTI)

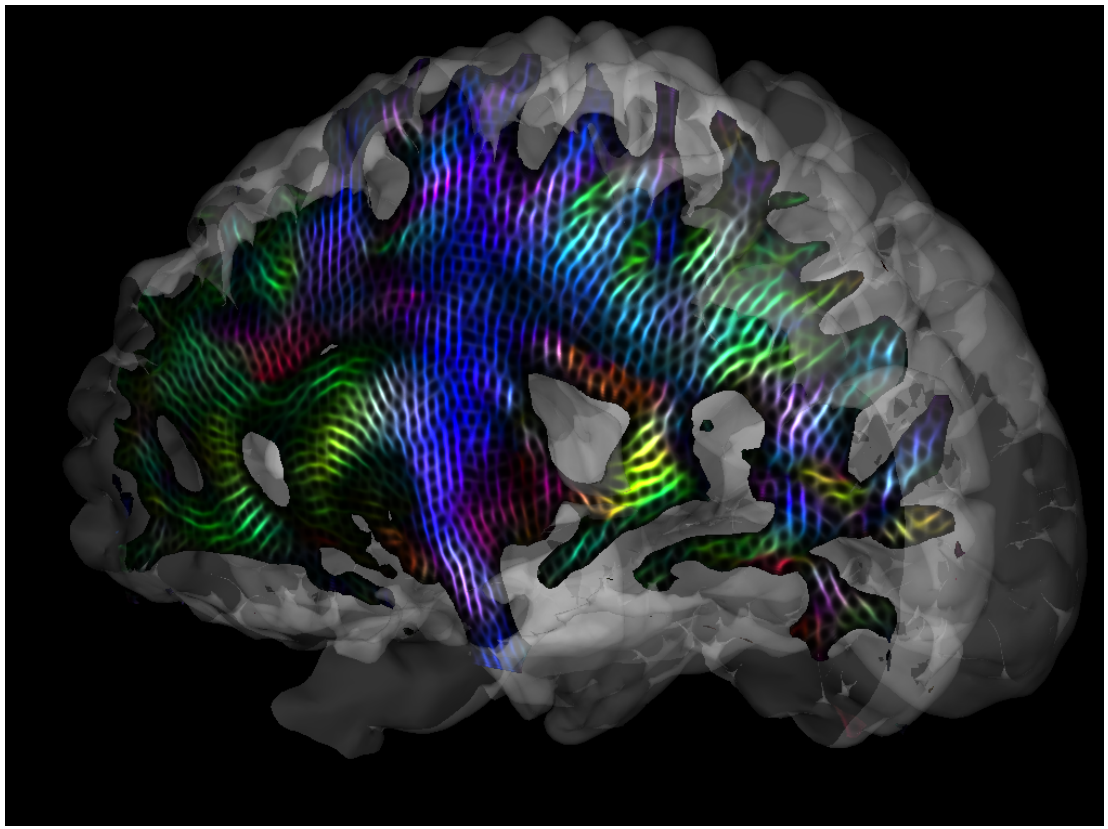


Abbildung: Diffusion entlang Neural Fibers (Anwander et al. [ASH+09]). (41472 Dreiecke, 32 FPS (Geometrie: 72%))

Bilder (Mechanik)

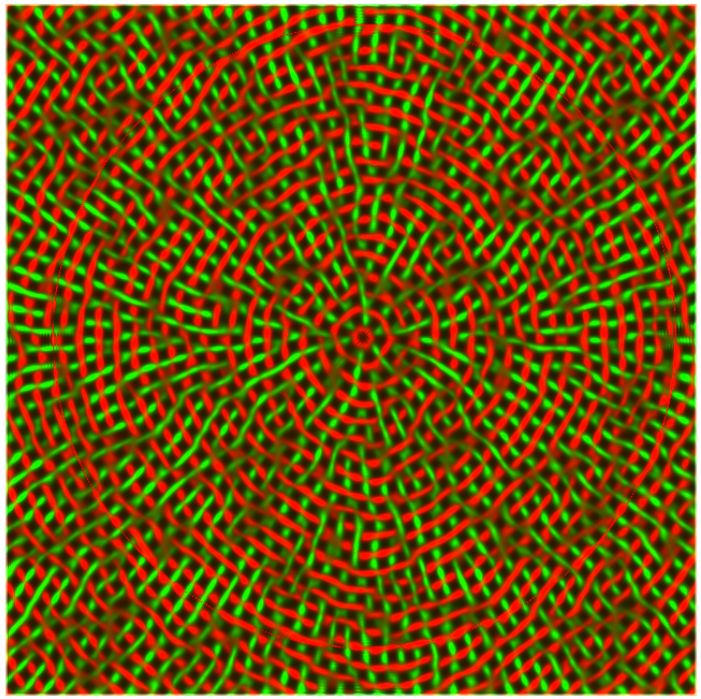


Abbildung: Single Point Load Datensatz.

Video

Gliederung

Probleme

- Keine Variation der Rauschtextur möglich

Probleme

- Keine Variation der Rauschtextur möglich
- Abbildung der Rauschtextur auf Geometrie sehr stark von Normalen abhängig
  - Funktioniert nur gut, wenn Normalen auf der Oberfläche sehr „smooth” sind.

Probleme

- Keine Variation der Rauschtextur möglich
- Abbildung der Rauschtextur auf Geometrie sehr stark von Normalen abhängig
  - Funktioniert nur gut, wenn Normalen auf der Oberfläche sehr „smooth” sind.
- Blurring- Effekte im zusammengesetzten Bild an manchen Stellen
  - Geringe numerische Präzision

Probleme

- Keine Variation der Rauschtextur möglich
- Abbildung der Rauschtextur auf Geometrie sehr stark von Normalen abhängig
  - Funktioniert nur gut, wenn Normalen auf der Oberfläche sehr „smooth” sind.
- Blurring- Effekte im zusammengesetzten Bild an manchen Stellen
  - Geringe numerische Präzision
- Beleuchtung oft nicht ausreichend für räumliches Aussehen der Geometrie
  - Licht wirkt sich auf Kontrast zwischen beiden Eigenvektorfeldern aus

Potenziale

- Reduktion der benötigten Geometrie
  - Level of Detail
  - BSP, kdTree oder Octree zum Space Partitioning

Potenziale

- Reduktion der benötigten Geometrie
  - Level of Detail
  - BSP, kdTree oder Octree zum Space Partitioning
- Erweiterung auf Tensordaten höherer Ordnung
  - Anzahl möglicher Fiber Richtungen begrenzt

Potenziale

- Reduktion der benötigten Geometrie
  - Level of Detail
  - BSP, kdTree oder Octree zum Space Partitioning
- Erweiterung auf Tensordaten höherer Ordnung
  - Anzahl möglicher Fiber Richtungen begrenzt
- Variation der Punktgrößen in der Rauschtextur oder deren Dichte
  - Entsprechend Eigenwerten oder anderer Metrik

Vielen Dank

Vielen Dank für Ihre Aufmerksamkeit.

Bibliographie I

Alfred Anwander, Ralph Schurade, Mario Hlawitschka, Gerik Scheuermann, and Thomas R. Knösche.
White matter imaging with virtual Klingler dissection.
Human Brain Mapping 2009, 2009.

James F. Blinn.
Light reflection functions for simulation of clouds and dusty surfaces.
In *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 21–29, New York, NY, USA, 1982. ACM.

Bibliographie II

Peter J. Basser and Carlo Pierpaoli.
Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor mri.
Journal of Magnetic Resonance, Series B, 111(3):209 – 219, 1996.

Alan Chu, Wing-Yin Chan, Jixiang Guo, Wai-Man Pang, and Pheng-Ann Heng.
Perception-aware depth cueing for illustrative vascular visualization.
In *BMEI '08: Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics*, pages 341–346, Washington, DC, USA, 2008. IEEE Computer Society.

Bibliographie III

Thierry Delmarcelle and Lambertus Hesselink.
Visualization of second order tensor fields and matrix data.
In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 316–323, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

Khader M. Hasan, Peter J. Basser, Dennis L. Parker, and Andrew L. Alexander.
Analytical computation of the eigenvalues and eigenvectors in DT-MRI.
Journal of Magnetic Resonance, 152(1):41 – 47, 2001.

Bibliographie IV

Ingrid Hotz, Louis Feng, Hans Hagen, Bernd Hamann, and Kenneth I. Joy.
Visualization and Processing of Tensor Fields, chapter Tensor Field Visualization Using a Metric Interpretation, pages 269–281.
Springer–Verlag Berlin Heidelberg, 2006.

Ingrid Hotz, Z. X. Feng, Bernd Hamann, and Kenneth I. Joy.
Tensor field visualization using a fabric-like texture on arbitrary two-dimensional surfaces.
In Torsten Möller, Bernd Hamann, and R. D. Russel, editors, *Mathematical Foundations of Scientific*

Bibliographie V

Visualization, Computer Graphics, and Massive Data Exploration. Springer-Verlag Heidelberg, Germany, 2009.


Charles D. Hansen and Chris R. Jonson.
The Visualization Handbook.
Elsevier Butterworth-Heinemann, 2005.


Hans Karl Iben.
Tensorrechnung.
Mathematik für Ingenieure und Naturwissenschaftler.
Teubner, 1995.

Bibliographie VI


Yuya Iwakiri, Yuuichi Omori, and Toyohisa Kanko.
Practical texture mapping on free-form surfaces.
In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 97, Washington, DC, USA, 2000. IEEE Computer Society.


Aaron Knoll, Younis Hijazi, Charles Hansen, Ingo Wald, and Hans Hagen.
Interactive ray tracing of arbitrary implicits with simd interval arithmetic.
In *RT '07: Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, pages 11–18, Washington, DC, USA, 2007. IEEE Computer Society.

Bibliographie VII

G Kindlmann.
Superquadric tensor glyphs.
In *Proceedings of IEEE TVCG/EG Symposium on Visualization 2004*, pages 147–154, May 2004.

James T. Kajiya and Brian P Von Herzen.
Ray tracing volume densities.
In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1984. ACM.

Bibliographie VIII

Budirijanto Purnomo, Jonathan D. Cohen, and Subodh Kumar.
Seamless texture atlases.
In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 65–74, New York, NY, USA, 2004. ACM.

Alan Turing.
The chemical basis of morphogenesis.
Philosophical Transactions of the Royal Society of London, 237(641):37 – 72, 1952.

Bibliographie IX

Greg Turk.
Generating textures on arbitrary surfaces using reaction-diffusion.
In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1991. ACM.

Bibliographie X

Daniel Weiskopf and Thomas Ertl.
A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces.
In *GI '04: Proceedings of Graphics Interface 2004*, pages 263–270, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

Bibliographie XI


David Weinstein, Gordon Kindlmann, and Eric Lundberg.
Tensorlines: advection-diffusion based propagation through diffusion tensor fields.
In *VIS '99: Proceedings of the conference on Visualization '99*, pages 249–253, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.


Carl-Fredrik Westin, Sariel Peled, Hakon Gudbjartsson, Ron Kikinis, and Ferenc A. Jolesz.
Geometrical diffusion measures for MRI from tensor basis analysis.
In *ISMRM '97*, page 1742, Vancouver Canada, April 1997.

Bibliographie XII


Xiaoqiang Zheng and Alex Pang.
Hyperlic.
In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 33, Washington, DC, USA, 2003. IEEE Computer Society.