UNIVERSITÄT LEIPZIG

Praktikum Visualisierungssysteme WS 2021/22

Hauptaufgaben

Institut für Informatik - Abteilung für Bild- und Signalverarbeitung
Prof. Dr. Scheuermann

Betreuerin: Vanessa Kretzschmar

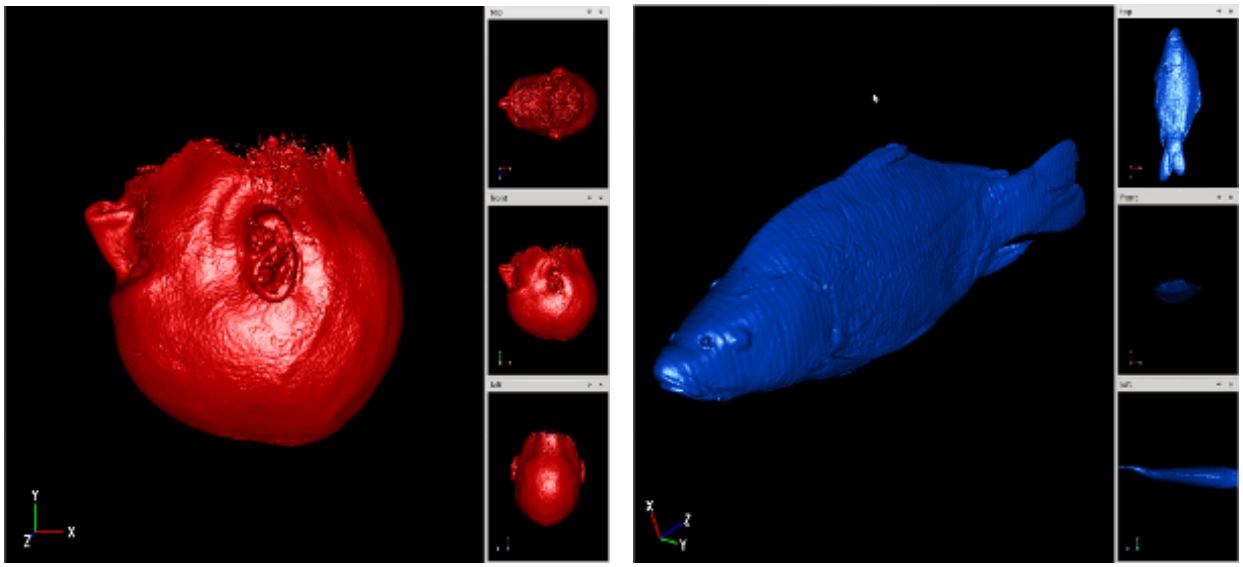
Von den hier dargestellten Aufgaben wählen Sie sich bitte **eine** aus. Eine Gruppenarbeit mit bis zu 3 Personen ist erlaubt. Wenn Sie Ihre Wahl getroffen haben, so teilen Sie diese bitte per Mail an kretzschmar@informatik.uni-leipzig.de mit.

Die Aufgaben sind im Folgenden in verschiedene Themen untergliedert, in denen Sie einzelne Algorithmen umsetzen sollen.

Hinweise:

- Wenn für Aufgaben (z.B. Topologie) die Berechnung von Ableitungen, Eigenwerten o.ä. benötigt wird, empfiehlt sich die *Eigen*-Bibliothek. Sie ist eine Header-only Bibliothek und lässt sich als solche einfach in ein Projekt einbinden.
- In den Aufgaben 1 und 2 sollen jeweils die gleichen Algorithmen auf verschiedene Art und Weise umgesetzt bzw. modifiziert oder verbessert werden. Es ist nur eine der Unteraufgaben umzusetzen, d.h. die Aufgaben stellen eigentlich mehrere Aufgaben dar.
- Wie immer gilt: Bitte kontaktieren Sie mich bei Fragen oder Problemen.

1. Thema: Volumendatenvisualisierung



(a) (b)

Figure 1: Marching Cubes Rendering der Datensätze alex-anatomy-t2.vtk und Carp-Ascii.vtk.

Aufgabe 1: Efficient Marching Cubes

Den Marching Cubes - Algorithmus zur Isoflächenextraktion haben Sie im Detail bereits in der Vorlesung kennen gelernt. Das Original-Paper zu Marching Cubes wird Ihnen zur Verfügung gestellt. Unter

`http://paulbourke.net/geometry/polygonise/`

finden Sie darüberhinaus auch eine Implementierung und kurze Erklärung. Ziel der folgenden Aufgaben ist es, Marching Cubes effizient in FAnToM zu implementieren. Bitte beachten Sie, dass die Falltabellen in den Code-Samples bereits gegeben sind und nicht neu implementiert werden müssen.

Das zu schreibende Plugin soll dem Nutzer zusätzlich die Möglichkeit bieten, den Isowert interaktiv zu verändern (z.B. mittels eines Sliders), was dann natürlich eine sofortige Neuberechnung der Isofläche zur Folge hat.

Im naiven Ansatz muss der Algorithmus jede Zelle einzeln abarbeiten. Das Ziel dieser Aufgabe ist es, den Algorithmus effizient umzusetzen. Dazu werden Ihnen 4 Möglichkeiten der Beschleunigung vorgeschlagen von denen Sie eine umsetzen sollen:

- **1a Efficient Marching Cubes/Tetrahedra mit Span Space**
  Das Ziel dieser Aufgabe ist es, die Zellen des Datensatzes im Vorfeld auszusortieren, die von der Isofläche nicht berührt werden, um so die Berechnung der Isofläche zu beschleunigen. Dazu werden die Gitterzellen entsprechend ihren Isowerten in einer Suchstruktur organisiert. Alle benötigten Informationen dazu finden Sie im zur Verfügung gestellten Paper. Die Hauptaufgabe besteht also darin, die Beschleunigungsstruktur aufzubauen, um den Algorithmus nur auf den entsprechenden Gitterzellen ausführen zu müssen.

- **1b Flying Edges (advanced)**
  Bei der Parallelisierung (z.B. mit OpenCL) müssen entweder viele Punkte doppelt erstellt werden, oder es besteht ein Synchronisierungsbottle-Neck beim Verwalten der Schnittpunkte der Isoflächen mit den Zell-Kanten. Hier bringt Flying-Edges eine Lösung und erlaubt eine effizientere Parallelisierung von Marching Cubes. Das Paper ist als Proof-of-Concept auf der CPU, parallelisiert mit OpenMP umzusetzen.

Aufgabe 2: Improved Isosurfaces

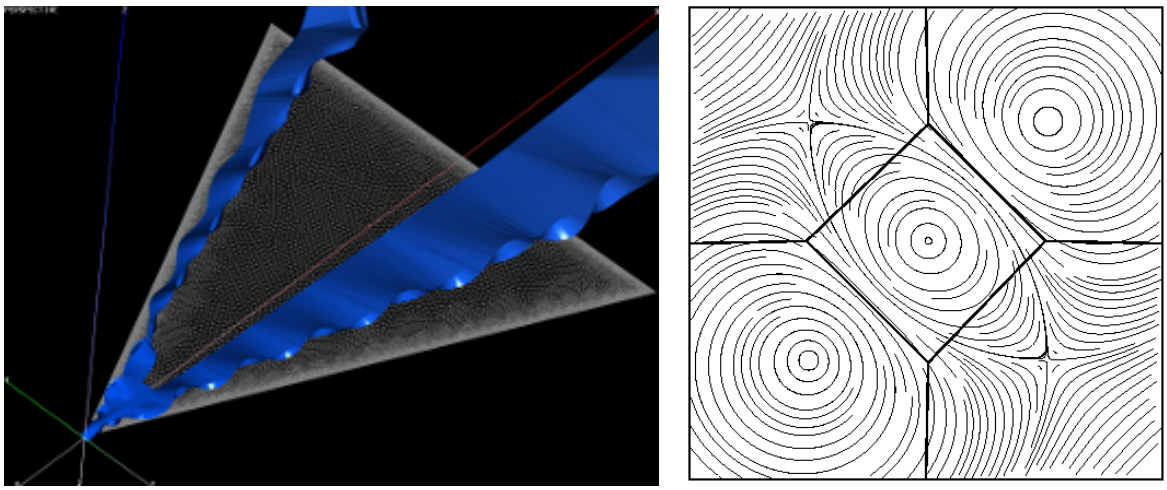
Neben der Beschleunigung gibt es bei Marching Cubes auch Möglichkeiten der visuellen Verbesserung:

- **2a Marching Diamonds**
  Beim ursprünglichen Marching Tetrahedra Algorithmus (für unstrukturierte Gitter), kommt es an den Zellgrenzen aufgrund der linearen Interpolation zu starken Unstetigkeiten. Andersen et al. stellen einen Ansatz vor, bei dem zwei Dreiecks (oder Tetraheder) - Zellen zu einer "Diamantzelle" zusammengefasst werden. Somit können glattere Isoflächen und genauere Isoflächen generiert werden. Es reicht, diese Aufgabe für Dreiecksgitter oder Tetrahedergitter zu bearbeiten. Optional kann diese Aufgabe (dann für beide Gittertypen) auch im Team gelöst werden.

Außerdem können Surfaces in bivariaten Feldern extrahiert werden.

- **2b Fiber Surfaces (advanced)**
  Normalerweise werden einzelne Skalarfelder untersucht. Manchmal bestehen Datensätze aber aus mehreren Feldern, z.B. Druck und Temperatur oder Dichte und Stärke der Dichteänderung. Um solche Felder zu analyiseren, werden normalerweise VolumeRenderings mit multivariaten Transferfunktionen verwendet. Letztes Jahr wurde aber ein Verfahren vorgestellt, aus diesen Daten auch verallgemeinerte Isoflächen zu extrahieren. Diese Extraktion soll im Rahmen dieser Aufgabe umgesetzt werden.

2. Thema: Strömungsvisualisierung 2D/3D



(a) (b)

Figure 2: a) Stromfläche im Delta Wing Datensatz. b) "schönes" Streamline Seeding (Quelle: http://slvg.soe.ucsc.edu/seed.html)

Aufgabe 3: FastLIC (advanced, shader programming)

In dieser Aufgabe soll ein Algorithmus geschrieben werden, der den Line Integral Convolution Algorithmus (LIC) implementiert. Ein zweidimensionales Vektorfeld wird mit Hilfe einer Textur dargestellt, die durch Grauwertmuster den Verlauf des Vektorfeledes visualisiert. Das ursprüngliche Verfahren ist sehr aufwändig und wurde deshalb verbessert (siehe LIC und FastLIC Papers). Sie sollen hier die FastLIC Methode und die beschriebenen Varianten implementieren und dabei unterschiedliche Filterlängen anwenden, um ihre Effekte auf die Textur zu untersuchen. Ein weiteres Skalarfeld — etwa die Norm (die Datei norm.vtk bezieht sich auf den Datensatz markCellSmall.vtk) — kann zusätzlich anhand einer Farbkodierung visualisiert werden. Die Berechnung der Integralkurven kann dabei durch den in den Grundaufgaben implementierten Algorithmus übernommen werden. Die verschiedenen Optionen und Parametrisierungen Ihres Algorithmus sollen durch den Nutzer eingestellt werden können. Zum Testen stehen unterschiedliche 2D Vektorfelder (z.B. markCell) zur Verfügung.
Bonusaufgabe: Histogram Equalization (o. ä.) zur Kontrasterhöhung Bonusaufgabe: Implementierung des Texture Controls von Matvienko

Aufgabe 4: Stromflächen

In dieser Aufgabe soll ein Algorithmus geschrieben werden, der Stromflächen aus einem Vektorfeld extrahiert und darstellt. Dabei soll der von Hultquist vorgestellte Algorithmus implementiert werden. Das entsprechende Paper wird zur Verfügung gestellt. Die Berechnung der Integrationslinien haben Sie in den Grundaufgaben bereits implementiert. Die verschiedenen Optionen und Parametrisierungen Ihres Algorithmus sollen durch den Nutzer eingestellt werden können.
Bonusaufgabe: Die Topologie-beachtenden Stromflächen: kritische Punkte markieren und Aufspaltung beenden.

Aufgabe 5: Stream Ribbons und Stream Tubes

Um neben der Richtung im Vektorfeld auch andere Größen darzustellen, werden oft Stream Tubes und Stream Ribbons angezeigt. Für zeitunabhängige Vektorfelder entsprechen Streamtubes der Oberfläche, die alle Stromlinien bilden, die von einer geschlossenen Kurve gestartet wurden. Sie stellen die Deformation der Strömung dar. Stream Ribbons dienen zur Visualisierung von Rotationen im Vektorfeld. Sie werden konstruiert, indem man Nachbarpartikel der Stromlinie mitintegriert und diese über Polygone verbindet.

Aufgabe 6: Streamline Seeding

Besonders in turbulenten Strömungen ist es für die Visualisierung durch Stromlinien wichtig, diese an bestimmten Positionen zu starten, um ein übersichtliches und informatives Bild zu erhalten. In dieser Aufgabe sollen Sie die Methode für 2D-Datensätze von Verma et al. umsetzen, die sich an den kritischen Punkten im Feld orientiert. Das heisst die Herangehensweise ist sehr ähnlich zu der in Aufgabe 18: Finden und Typbestimmung der kritischen Punkte. Danach sollen die jeweils entsprechenden Seedingstragien angewandt werden.

Aufgabe 7: Streamsurface Seeding

Ein ähnliches Problem, wie das in der vorherigen Aufgabe, tritt auf, wenn Stromflächen zur Visualisierung eines vorliegenden Vektorfeldes verwendet werden sollen. Bei schlecht gewählten Startpositionen kann es dazu kommen, dass die dargestellten Stromflächen das Verhalten des Vektorfeldes nur unzureichend wiedergeben. Ebenso kann es zu starker Überdeckung kommen, welche die Informationsgewinnung erschwert. Das vorliegende Paper beschreibt einen Lösungsansatz für dieses Problem basierend auf einem Distanzmaß für Stromlinien. Mit dem fertigen Algorithmus, soll der Nutzer schlußendlich in der Lage sein, eine Region im Vektorfeld zu bestimmen, in dem diese "optimalen" Stromflächen generiert werden.

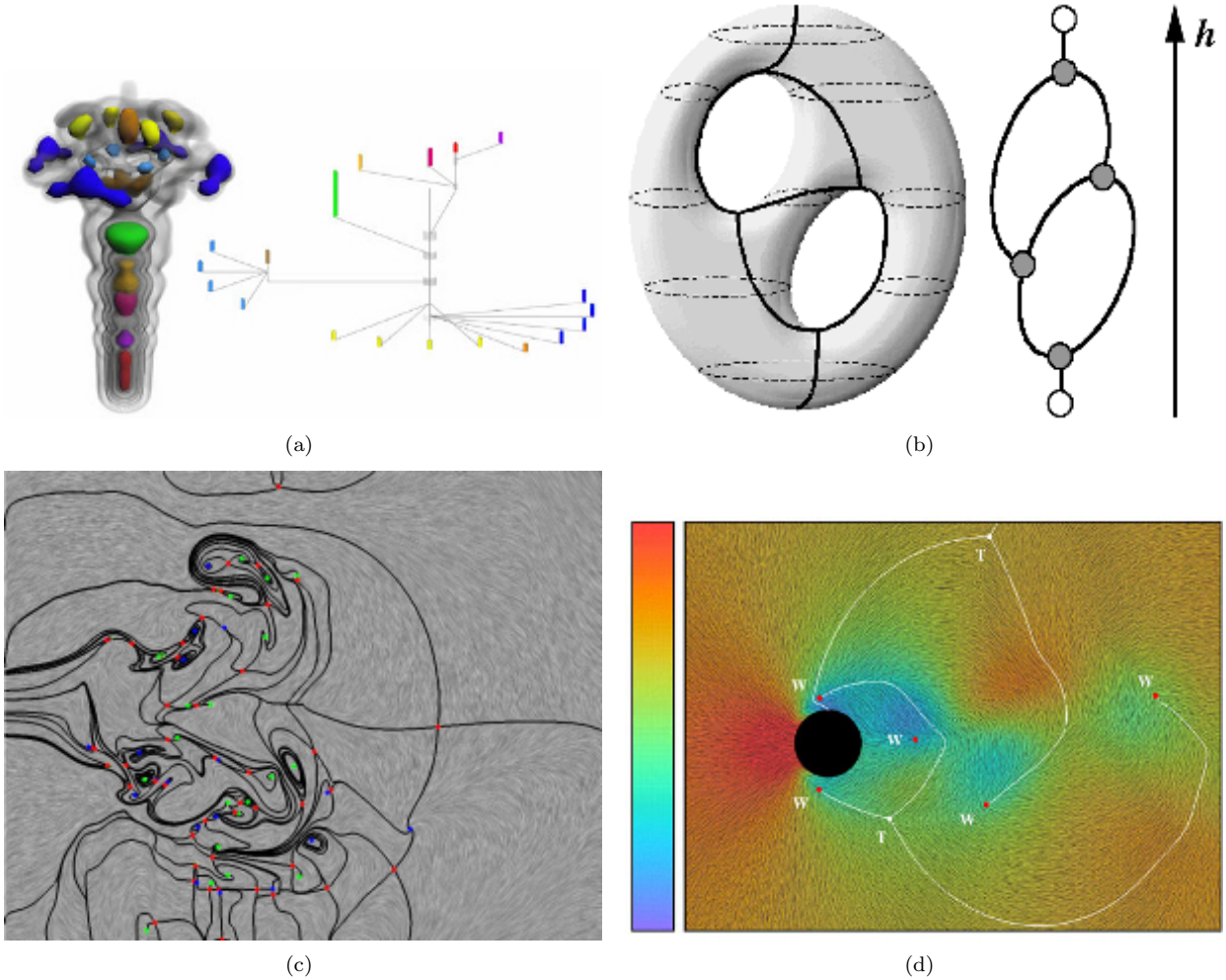
Aufgabe 8: Localized FTLE (advanced)

Der Finite-time Lyapunov Exponent ist ein Maß für die Separation von Partikeln, die sich durch das Vektorfeld bewegen. Der ursprüngliche Algorithmus zur Berechnung (siehe Garth et al.) wurde erweitert (und damit effizienter), indem in jedem neuen Schritt Berechnungsergebnisse vorheriger Schritte benutzt werden (ähnliche Idee wie beim FastLIC). Das Ziel dieser Aufgabe ist die Implementierung der Berechnung und der Visualisierung des localized FTLE nach der Methode von Kasten et al.

Aufgabe 9: Uncertainty in Flow Ensembles (advanced)

Z.B. bei der Simulation von Wetter wird häufig mit randomisierten Berechnungen gearbeitet. Diese werden mehrfach wiederholt, sodass man eine Anzahl von Vektorfeldern mit möglichen Ergebnissen erhält. Konkret: z.B. 50 verschiedene Vorhersagen für die Windgeschwindigkeiten in Europa in 96 Stunden. Bei der Analyse dieser Feld-Ensembles spielt die Visualisierung der Unsicherheit eine große Rolle um z.B. gesicherte Stromlinienverläufe von sehr unwahrscheinlichen abgrenzen zu können. Das Ziel dieser Aufgabe ist die Implementierung des Verfahrens von Ferstl et al. um die Trends der Stromlinienverläufe von einem gegebenen Punkt zu extrahieren. Dabei sollen sowohl die Daten selbstständig erhoben werden, die Trend-Components extrahiert werden und mitsamt der repräsentativen Stromlinien in 2D visualisiert werden.

3. Thema: Topologie



(a) (b) (c) (d)

Figure 3: a) Kontourbaum des Datensatzes Fuel-Ascii.vtk (Dillard '05). b) Reeb Graph eines Bitorus und einigen Kreuzungssektionen (Biasotti '01). c) 2D Vektorfeldtopologie mit Rand, d: Tensorfeldtopologie (Hesselink '94).

Aufgabe 10: Konturbaum

Der Konturbaum ist eine essentielle Struktur bei der Untersuchung der Topologie von Skalarfeldern. Der Algorithmus sollte Ihnen aus der Vorlesung bekannt sein. Details entnehmen Sie bitte dem zur Verfügung gestelletem Paper. Überlegen Sie sich ausserdem ein (naives) Layout, um den entstehenden Konturbaum übersichtlich darzustellen.
Bonusaufgabe: Option der Persistenzvereinfachung, um die Übersichtlichkeit zu steigern

Aufgabe 11: ReebGraph (advanced)

Der Reeb-Graph ist die Verallgemeinerung des Konturbaums auf Mannigfaltigkeiten, die "Löcher" enthalten können, auf denen sich also nicht alle Pfade zwischen zwei Punkten stetig ineinander überführen lassen. Der Reeb-Graph hat seine Bedeutung oft in der Analyse von Oberflächen, denen ein künstliches Skalarfeld entlang einer Koordinatenachse zugewiesen wird um zum Beispiel Schleifen in der Graphenstruktur zu finden, die auf torusartige Strukturen der Oberfläche hinweisen.
In der Bildverarbeitung und dem Computersehen ist er ein beliebtes Mittel zur Formerkennung, wohingegen er in der Visualisierung als Verallgemeinerung des Konturbaums auf Oberflächen mit komplexerer Topologie zum Einsatz kommen kann.

Aufgabe ist die Implementierung des Reeb-Graphen nach der Methode von Pascucci et al. (2007) für simpliziale 2- und 3-Mannigfaltigkeiten, das Speichern des Graphen zur Weiterverarbeitung z.B. im "dot"-Format (www.graphviz.org), sowie die Anzeige des Graphen in 3D auf dem Datensatz.

Herausforderungen:

- Effiziente Datenhaltung
- Graphentraversal

Aufgabe 12: Morse-Smale-Complex (advanced)

Für Topologie-basierte Visualisierung spielt der Morse-Smale-Komplex eine wichtige Rolle. Dabei handelt es sich um eine Struktur, die den Gradientenfluss beschreibt und damit eine aussagekräftige Unterteilung der Domain liefert. Die Anzeige soll direkt im Datensatz erfolgen. Das mitgelieferte Paper liefert einen Algorithmus zur Berechnung in 2D.

Aufgabe 13: Contour Spektrum

Bjaj et al. haben gezeigt, dass es möglich ist, die Fläche von Isoflächen sowie das eingeschlossene Volumen in Simplexen analytisch zu bestimmen. Dies ermöglicht die schnelle Suche nach interessanten Teilen eines Datensatzes, da neben den Werten oft auch das eingeschlossene Volumen eine Rolle für die Wichtigkeit der Werte spielt.
Zu implementieren ist die Berechnung der Länge der Isolinien und der eingeschlossenen Fläche in 2D sowie die Fläche und das Volumen innerhalb der Konturen in 3D. Die Implementierung soll dabei basierend auf den Parametern der Zellen Splines erstellen, die aufsummiert als Funktionsgraphen angezeigt werden.
Zeitabhängige Daten sollen als Farbspektrum (Isowert auf einer Achse, Zeit auf der zweiten Achse und Farbe als Fläche der Isofläche — siehe Bajaj, Figure 7) dargestellt werden können.

Herausforderungen:

- Speicherung, Aufsummierung, Integration und eventuell Vereinfachung vieler Splinefunktionen
- Naive Parallelisierung möglich

Aufgabe 14: Vektorfeldtopologie mit Rand in 2D

Ziel dieser Aufgabe ist die Extrahierung des topologischen Skeletts eines 2D Vektorfeldes. Die Aufgabe besteht darin die kritischen Punkte im Feld zu extrahieren, deren Typus zu bestimmen und sie durch Separatrizen (welche das Strömungsverhalten separieren) zu verbinden. Zusätzlich dazu sollen alle Punkte am Rand extrahiert werden, die tangential zum Rand laufen. Auch hier sollen entprechende Stromlinien gestartet werden, welche dann in einen kritischen Punkt laufen.

Aufgabe 15: Ridges and Valleys

Ein besonders in der Bildverarbeitung beliebtes Verfahren zur Bestimmung von Bereichsgrenzen ist die Extraktion sogenannter verallgemeinerter Extrema. Bei verallgemeinerten Extrema müssen die Bedingungen, die an ein Extremum geknüpft sind, nur in eine Richtung gelten. Stellt man sich das Bild (den 2D-Datensatz) als Höhenfeld vor, entspricht das der Extraktion von Bergkämmen und Tälern. Umzusetzen sind in dieser Aufgabe die Extraktion eben dieser Merkmale und deren Visualisierung.

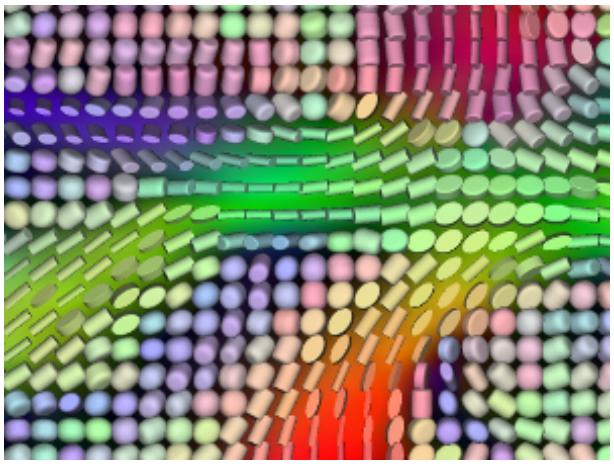
Aufgabe 16: Attachement and Separation Lines

Wenn beispielsweise Flugzeug- oder auch Autochassis entworfen werden, ist es von größtem Interesse, wann und wo sich der Luftstrom entweder an die Flugzeugoberfläche anschmiegt oder auch ablöst. Solche Ablösungen können beispielsweise den Auftrieb des Flugzeugs reduzieren, was bei niedrigen Geschwindigkeiten zu erheblichen Steuerungsproblemen führen kann. Das Finden und Darstellen solcher Ablösungs- und Anschmiegelinien ist das Ziel dieser Aufgabe.

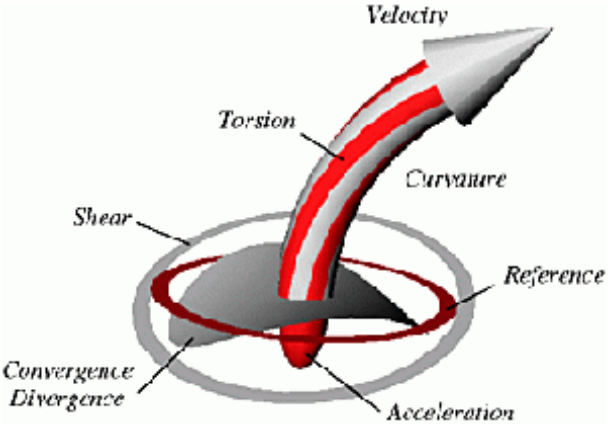
Aufgabe 17: Tensorfeldtopologie

Bei der Topologie von Tensorfeldern ist das Tracking von Tensorlines ein essentieller Bestandteil. Die Aufgabe besteht darin, die kritischen Punkte im Tensorfeld zu finden und die Major, Middle und Minor Tensorlines zu extrahieren und (verschiedenfarbig) darzustellen. Die Aufgabe ist beschränkt auf Dreiecksgitter (2D) und Tetraedergitter (3D). Die zu untersuchenden Tensoren sind symmetrisch und vom Rang 2.

4. Thema: Glyphen



(a)



(b)

Figure 4: a) Superquadrics (Kindlmann '04). b) Komponenten einer Localized Flow Probe.

Aufgabe 18: Superquadrics, Localized Flow Probe

In der Vektor- und Tensorfeldvisualisierung werden verschiedene Glyphen zur Darstellung bestimmter Eigenschaften verwendet. Die Localized Flow Probe ist ein "erweiterter" Hedgehog, der zusätzlich Informationen wie Divergenz, Rotation und Krümmung anzeigen kann.

Superquadrics sind eine Art Erweiterung von Streamlets auf Tensorfelder. Durch die Tiefe, Höhe und Breite dieser Glyphen werden die Richtungen der Tensorlinien visualisiert. Bei einer MRT Aufnahme beispielsweise kann man (falls keine Normalisierung der Glyphen vorliegt) auch die Stärke der Diffusion in jede Richtung ablesen. Ziel der Aufgabe ist die Implementierung der Localized Flow Probe und der Boxen/Superquadrics als separate Visualisierungsalgorithmen.

Aufgabe 19: Glyph Packing

Bei der Verwendung von Glyphen ergeben sich manchmal aufgrund der Platzierung der Glyphen in den Daten visuelle Muster, die nur aufgrund des regelmäßigen Gitters entstehen. Dadurch werden große, in den Daten vorhandene Muster überdeckt. Glyph Packing platziert die Glyphen auf eine Weise im Gitter, die dieses Problem löst. Ziel der Aufgabe ist die Implementierung des Glyph-Packing-Ansatzes von Kindlmann unter Verwendung von einfachen elliptischen Glyphen.

5. Thema: Traktographie

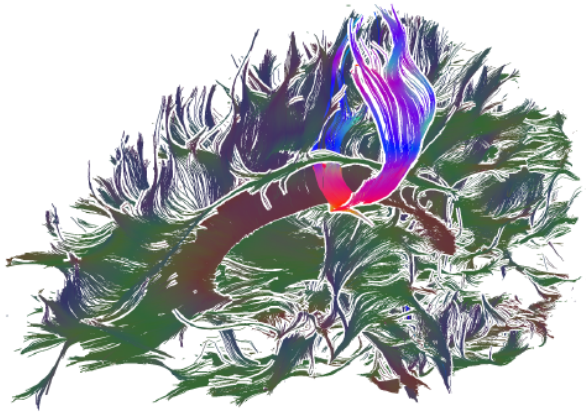


Figure 5: Contracted Fibers of a human brain (Everts 2015).

Aufgabe 20: Fiber Contraction

Bei der Analyse von Konnektivität im Gehirn werden z.B. fibers verwendet, d.h. Linien, die den Verlauf von Axonen in der weißen Gehirnmasse repräsentieren. Aus Visualisierer-Sicht haben wir es also mit einer großen Menge an Linien zu tun. Verschiedene Ansätze der Computergrafik (z.B. Shading) können bei der Visualisierung helfen, aber eine Abstraktion der Linien, die große Strukturen im Gehirn sichtbar machen, sind erstrebenswert. Ein Vorschlag dazu ist ein Paper von Everts aus dem Jahr 2015. Im Rahmen dieser Aufgabe soll das iterative Verfahren zur Linien-Kontraktion umgesetzt werden. Interaktion und Rendering steht dabei nicht im Vordergrund.