Abstract

Inhaltsverzeichnis

1 Verwendung und Umfeld des JHelioviewers

Satelliten, welche Messungen von der Sonne machen.

Der JHelioviewer ist eine Applikation, die zur Analyse von Sonnendaten verwendet wird. Es wird international zur Sonnenforschung eingesetzt und wird von der FHNW zusammen mit der ESA entwickelt. Momentan ist eine neue Version des JHelioviewer in Entwicklung, welche die Sonne im dreidimensionalen Raum darstellt. Ein Feature von JHelioviewer ist die Magnetfeldlinien darzustellen und zu animieren, die Abbildung 1 zeigt die Visualisierung. Es wird zwischen drei Feldlinien Unterschieden: Linien, die auf der Sonne starten und wieder

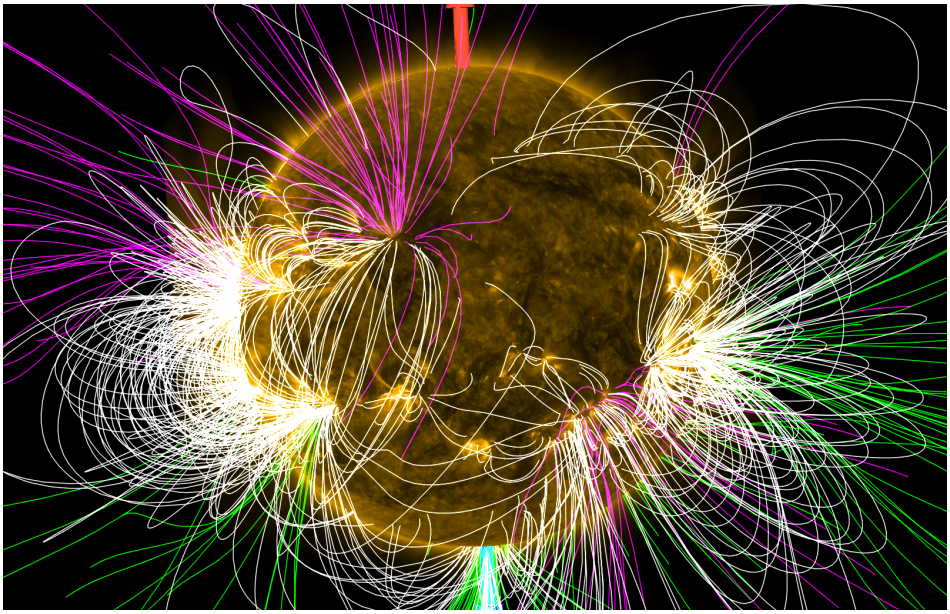

Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

auf der Sonne landen, auf der Sonne starten und ins Weltall führen oder vom Weltall auf der Sonne landen. Die weissen Feldlinien repräsentieren ”Sonne zu Sonne´´, die Grünen ”Sonne zu Weltall´´ und die Violetten ”Weltall zu Sonne´´. Die Feldlinien sind, allgemein Betrachtet, eine grosse Menge an Punkten, welche ein Server bereitstellt. Die Abbildung 2 visualisiert den Datenfluss. In regelmässigen Abständen sucht der Ser-

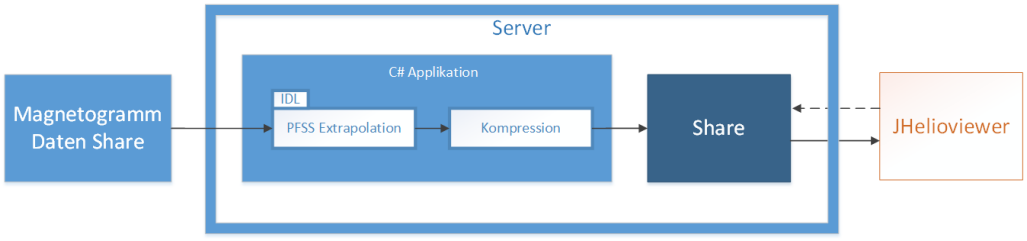

Abbildung 2: Aufbau und Datenfluss des Servers

ver nach neuen Oberflächen-Magnetogramm-Daten der Satelliten. Die momentane Lösung erhält alle sechs Stunden neue Messdaten der Satelliten. Daraus werden mittels Potential Field Source Surface (PFSS) Simulation die Feldlinien zu diesem Zeitpunkt errechnet. Danach führt der Server eine Kompression durch und stellt die Daten auf einem öffentlichen Share dem JHelioviewer zur Verfügung. Zusammen mit Aufnahmen von anderen Instrumenten animiert der JHelioviewer die Feldlinien über einen gewissen Zeitraum. Die Daten werden zur Laufzeit über eine Internetverbindung nachgeladen.

1.1 Rohdatenmenge der Feldlinien

Die PFSS Extrapolation produziert für eine Aufnahme etwa 15 MiBytes an Daten. Verlustfrei/verlustbehaftet. Vorfeld schon mit Verlustlosen und Verlustbehafteten Kompressionsverfahren auf etwa 1.5 MiByte verkleinert. Ziel ist es, so wenige Daten wie Möglich zu brauchen. Zukunftsvision

2 State of the Art

2.1 JPEG/JFIF Kompression

2.2 3d Mesh Kompression

In der Computergrafik müssen alle Gegenstände und Akteure in der virtuellen Welt modelliert werden. So wird ein Mensch eine Menge von Dreiecken, das Mesh, modelliert. Jeder Punkt eines Dreieck kann zusätzliche Informationen gespeichert haben wie Texturkoordinaten, Farbe oder Oberflächen-Normalen. Je nach Anwendungsfall kann ein Modell aus ein paar hundert oder mehreren hunderttausend Dreiecken bestehen. Um diese Datenmenge zu verkleinern versuchen Formate wie OpenCTM [2] die Datenmenge verlustfrei oder verlustbehaftet zu komprimieren.

wie funktioniert es

wie es verwendet werden könnte

2.3 PointCloud Kompression

Industrie Lasersampling Bild pointcloud Grosse Punktmenge, welche im 3d Raum komprimiert werden soll. verlustfrei/ Verlustbehaftet Je nach Implementation können zu jedem Punkt zusatzinfos gespeichert werden wie Farbe/Normalen etc, darüber könnte die Information, zu welcher Linie ein Punkt gehört, gespeichert werden.

2.4 Signal Approximation

Messtechnik von Medizin bis Fotografie, überall wo man ein Signal über Zeit hat Versucht ein Signal durch eine Folge von Funktionen zu approximieren.

Gute Approximation kommt mit wenigen Funktionen aus. Verlustbehaftete Kompression, indem man mit einer begrenzten Anzahl

fourier, wavelet Compressed sensing, Spline Approximation

2.5 Entropie Kodierung

Verlustfreie Komprimierung basierend auf Shannons coding theorem

Arithmeic, Huffman Dictionary Type: LZ77, Byte paoir encoding, RLE

3 Eigenschaften und Kompression der Feldlinien

Jede Simulation beinhaltet 1200 Feldlinien und insgesamt etwa 60'000 Punkte. Fits format [**?**]

3.1 Ist-Komprimierung

Der JHelioviewer bietet an, die Feldlinien zu einem gegebenen Zeitpunkt darzustellen. Damit der Benutzer eine vernünftige Zeit auf die Feldlinien wartet, wurde bereits im Vorfeld eine Kompression implementiert. Zuerst werden die Daten im sphärischen Koordinatensystem(Radius, Längengrad ϕ und Breitengrad θ) auf dem Server quantisiert und mit GZip verlustfrei komprimiert. Der JHelioviewer dekomprimiert die Daten und transformiert sie in das kartesische Koordinatensystem um. Die Punktmenge wäre für schwächere Grafikkarten zu gross, weshalb der JHelioviewer eine weitere Quantisierung durchführt.

Quantisierung und Dateiformat auf dem Server
Zuerst werden die Kanäle R,ϕ und θ Kanäle zu shorts diskretisiert:

1. R: 4 = 2^{15}.
2. ϕ: 2π = 2^{15}
3. θ: 2π = 2^{15}

θ Wertebereich geht aber nur von 0 bis π, die letzten Bits werden gar nicht verwendet. Die Kanäle R und ϕ haben das Problem, dass der Wert 2^{15} einen Signed Integer Overflow verursacht und auf -2^{15} zu liegen kommt. R scheint den maximalen Wert nie zu erreichen. Wenn aber eine Feldlinie den Nullpunkt passiert, springt der Kanal von $2^{15}-1$ auf -2^{15} und dann auf 0.

Subsampling, jeder vierte Punkt 0 Löschen.

Format: zuerst Konstanten, alle Radien

Quantisierung des JHelioviewers
Clientseitig wieder ein subsampling und umrechnung ins kartesische System xyz, dann Anglesubsampling

3.2 Lösung 0, Angle-Subsampling

Subsampling 5Grad auf dem Server Minimale Lösung, ziel ist es diese Lösungen zu schlagen.

3.3 Lösung 1, Diskrete Kosinus Transformation

DCT, da alles nahe an harmonischen Halbwellen

subsampling koordinatentransformation –¿ kein wrap around, Ableitung Cosinus-Transformation

DCT 2 idct ist dct3

Quantisierung

speicherung für Entropie encoding, alles was ähnlich ist zusammen.

encoding-¿ rar

4 Qualitätsmessung der Kompression

Um die Datenmenge der Feldlinien zu verringern werden verlustbehaftete Kompressionsverfahren angewendet. Trotz des Dateverlustes sollen die dekomprimierten Linien möglichst ihren Originalen ähneln. Kleine Abweichungen werden in der Sonnenforschung toleriert. Es ist wichtig, dass die Form der Kurve erhalten bleibt. Grosse, seltene Abweichungen sollten vermieden werden, da sie das Aussehen der Feldlinie verändern können.

Zusammen mit Sonnenforschern wurden zwei Fehlermasse bestimmt: Der absolute maximale Fehler und die Standardabweichung von der komprimierten Linie zum Original. Die Standardabweichung ist für diesen Fall geeignet: Grosse, seltene Abweichungen werden stärker gewichtet, als kleine dafür häufige Abweichungen.
Der absolute maximale Fehler wird noch als Absicherung gemessen. In den meisten Fällen wird die Kompression mit der tieferen Standardabweichung auch den kleineren maximalen Fehler haben. Da aber die Messung über ein paar hunderttausend Punkte durchgeführt wird, ist das Gegenteil denkbar.

Eine Grenze für die Genauigkeit ist nicht festzulegen. Auch wenn eine Grenze gefunden wird, kann diese sich in der Zeit verändern. Im Fall der Feldlinien ist die Internetverbindung der Flaschenhals. Es kann sein, dass in Zukunft mehr Präzision bei mehr Platzbedarf verlangt wird. Deshalb werden die Verfahren, wenn möglich, mit unterschiedlichen Qualitätsstufen getestet und verglichen.

4.1 Auswahl und Erhebung der Testdaten

Die Testdaten sollen zu einem alle Randfälle abdecken, als auch durchschnittliche Fälle enthalten. Aus diesem Grund wurden insgesamt zehn Datensätze ausgewählt: Vier Datensätze mit hoher Sonnenaktivität, zwei mit wenig und vier zufällig. Für die vier Datensätzen mit hoher Aktivität wurde in den Jahren 2014 und 2013 nach den grössten Solare Flares gesucht. Für die Datensätze mit wenig Aktivität wurde das Gegenteil gemacht, nach Zeiträumen mit möglichst kleinen Solar Flares gesucht.
Die feldlinien werden aber nur alle sechs Stunden berechnet und Solar Flares sind sehr spontane Ereignisse. Auch eine grosse Flare kann während den sechs Stunden angefangen und wieder aufgehört haben. Für die grossen Solar Flares wurde deshalb beachtet, dass die Datensätze vor dem Ereignis verwendet wurden. Grosse Solar Flares entladen das Feld, vor dem Ereignis ist das Magnetfeld komplexer.

Wie im Abschnitt 3.1 beschrieben, führt der IDL-Code schon eine Quantisierung durch. Diese wurde entfern und deshalb wurde der IDL Code angepasst. Die Quantisierung wurde für die Testdaten entfernt. Die Punkte werden nun mit 32 Bit Floating-Point Genauigkeit gespeichert.

4.2 Aublauf des Tests und Messung des Fehlers

Float daten werden geladen. Daten kopiert und Kompression/Dekompression durchgeführt für alle Testdaten. Die kopierten Daten wissen aber noch, welches ihr Originalpunkt ist. Zwei Mengen, Originalpunkte O, dekomprimierte Punkte D. Es gibt immer gleich viele oder mehr Originalpunkte wie dekomprimierte Punkte. Die Fehlerberechnung muss der allgemeine Fall und die Randbehandlung unterschieden werden.

4.2.1 Allgemeiner Fall

Die Berechnung ist Dargestellt im Diagramm 3. Für jeden Punkt $p1'$ aus D, nehme $p1'$ und den folgenden Punkt und $p2'$. Ziehe eine Strecke s durch $p1'$ und $p2'$. Suche von $p1'$ den Originalpunkt $p1$ aus O und rechne

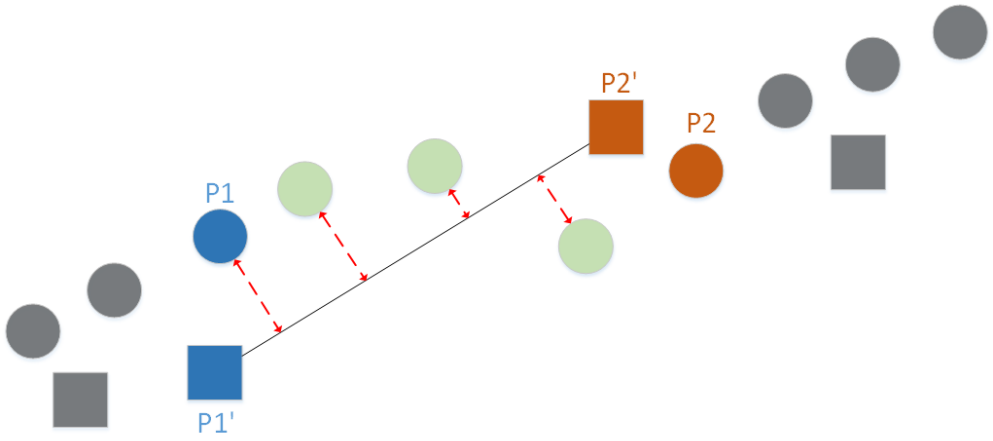

Abbildung 3: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

den Abstand aus zur Strecke s. Führe das für alle folgenden Originalpunkte durch, bis $p2$ erreicht wurde. Der Abstand s zu $p2$ wird nicht mehr berechnet.

Abstandsberechnung eines Punktes zu einer Strecke
Gegeben: Stecke s mit Eckpunkten A und B und Punkt P.
Gesucht: Kürzeste Distanz zwischen s und P

Zuerst wird überprüft, ob eine Senkrechte durch P überhaupt auf der Strecke s zu liegen kommt. Das ist der Fall, wenn die Strecke AP auf die Strecke s projizierbar ist:

$$t = \frac{\vec{AB} * \vec{AP}}{|\vec{AB}|^2}$$
$$0 \leq t \leq 1$$

Wenn das nicht möglich ist, wird der kürzere Distanz von P zu einem der Eckpunkte genommen. Falls aber eine Senkrechte auf s zu liegen kommt, muss jetzt die Länge der Senkrechte berechnet werden. Aus der vorhgehenden Berechnung könnte man den Fusspunkt auf s berechnen und dadurch die Distanz, oder man kann die Distanz direkt über das Kreuzprodukt berechnen.

$$distance = \frac{|\vec{BA} \times \vec{BP}|}{|\vec{BP}|}$$

4.2.2 Randbehandlung

Es ist möglich, dass die originalen Endpunkte durch eine Quantisierung verworfen wurden. Das bedeutet, wenn man den Fehler für den allgemeinen Fall berechnet, am Anfang und am Ende Originalpunkte existieren, für die nie eine Distanz berechnet wurde. Die Abbildung 4 zeigt das Problem. Deshalb müssen die Abstände der Ränder von der Komprimierten- zur Original-Linie noch berechnet werden. Der Abstand vom ersten komprimierten Punkt (in der Abbildung P0) zu seinem Original wird schon im allgemeinen Fall berechnet.

4.2.3 Berechnung der Standardabweichung

$$\sigma(X) = \sqrt{(variance(X))}$$
$$variance(X) = \sum (x_i - E(x_i))^2$$

Abbildung 4: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

Die Standardabweichung σ einer Beobachtungsreihe X $(x_1, x_2, x_3, \ldots, x_n - 1)$ ergibt sich aus der Wurzel der Varianz von X. Die Varianz von X kann errechnet werden, wenn man den Distanz jeder Beobachtung x_i mit dem Erwartungswert $E(x_i)$ berechnet und quadriert. Die Beobachtung ist im diesen Fall ein Punkt der dekomprimierten Linie, während der Erwartungswert der Originalpunkt ist. Die Distanz wird mit dem besprochnen Verfahren 4.2 berechnet. Die Summe der quadratischen Abstände ergibt die Varianz. Die Varianz wird über alle Testdaten berechnet, somit erhält man für einen Test genau eine Standardabweichung.

5 Implementation

5.1 Komprimierung auf dem Server

C# architektur des servers, nicht viel. neuer Ordner auf Share

5.2 Asynchrone Dekomprimierung im JHelioviewer

alter zustand?

neuer Zustand DAtenfluss, FileDescriptor Ziel, Asynchroner Ablauf. Daten werden Asynchron vom Server heruntergeladen, dekomprimiert und auf die Grafikkarte geladen.

Klassendiagramm

Read ahead Frames Cache

PfssData Read ahead + cache

6 Resultate

Wie im Kapitel 3 wird erwähnt, dass der Ist-Zustand die Feldlinien im FITS-Dateiformat abspeichert. Es wurde geprüft, wie viel Speicher das Format nach der Entropie-Kodierung beansprucht. Für den Test wurde die Komprimierung 6.1 werwendet. Zum Vergleich wurden die Daten im FITS-Dateiformat und als Binärdatei abgelegt. Die Resultate 1 zeigen keinen signifikanten Unterschied des Speicherverbrauchs. Pro Feldlinie liegt

pro Feldlinie (FITS)	Bytes pro Feldlinie (Binär)
74.2 Bytes	73.6 Bytes

Tabelle 1: Einfluss des FITS Formates auf den Speicherverbrauch

der Unterschied bei 0.6 Bytes. Hochgerechnet auf 1200 Feldlinien sind es 0.7 KiBytes, welches das Fits-Format für sich beansprucht. Fits erlaubt es, den Datenreihen einen Namen so wie eine kurze Beschreibung anzuhänken. Diese Strings sind Optional und können auch weggelassen werden.
Der wesentliche Vorteil von FITS ist, dass es bereits Plattformunabhängig ist. Probleme wie Endianness [**?**] werden vom Format gelöst. Aus diesen Gründen wurde entschieden das Format beizubehalten.

6.1 Lösung 0, Angle-Subsampling

Das Angle-Subsampling führt der JHelioviewer selbst durch. Für die Lösung 0 soll nun das Subsampling vor der Dateiübertragung vorgenommen werden. Dazu werden die Punkte in das kartesische Koordinatensystem umgerechnet, das Angle-Subsampling durchgeführt und anschliessend wird in das sphärische System rücktransformiert. Die resultierende FITS-Datei wird mittels Rar kodiert. Wie im Diagramm 5 erkennbar ist, ver-

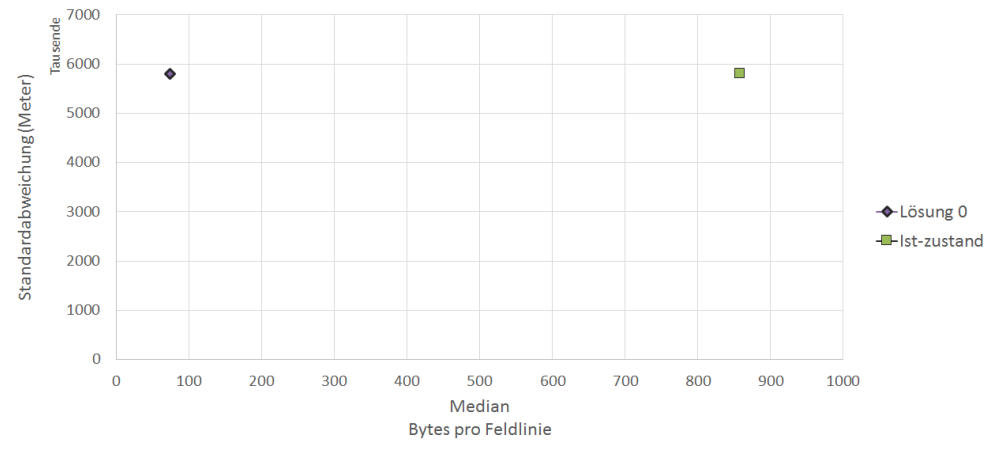

Abbildung 5: Vergleich der Lösung 0 zum Ist-Zustand.

braucht die Lösung 0 um grössenordnungen weniger Speicher. Zu einem wird durch das Angle-Subsampling weniger Daten gespeichert,etwa nur ein Viertel der ursprünglichen Punkte. Zum anderen erbringt Rar die bessere Kompression. Die Komplexität der Kompression und Dekompression bleibt in der Grössenordnung $O(n)$ (n ist die Anzahl Punkte). Da bei der Dekompression n etwa vier Mal weniger Punkte bearbeiten muss, ist die Dekompression sogar schneller als die Ist-Lösung. Die Abbildung 6 zeigt die Artefakte, die bei der Komprimierung der Lösung 0 entstehen. Aus dem JHelioviewer sind diese Artefakte kaum sichtbar.

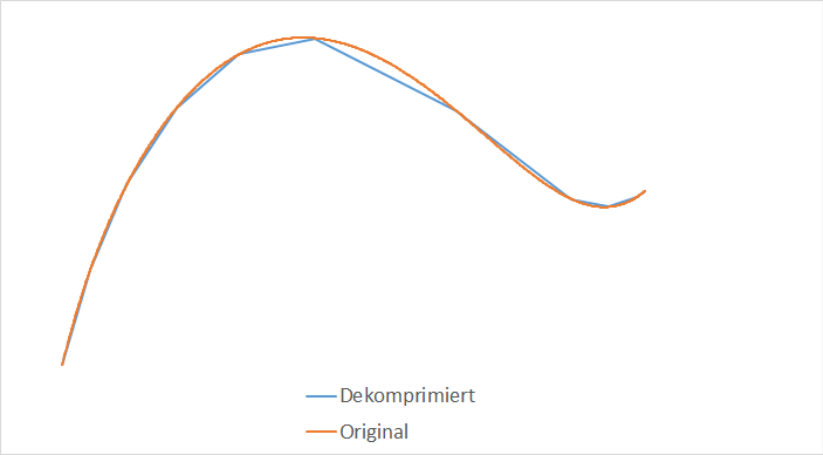

Abbildung 6: Artefakte der Lösung 0

6.2 Lösung 1, Diskrete Kosinus Transformation

Das Ziel ist eine bessere Kompression zu erreichen, indem die Feldlinien mit Kosinusfunktionen approximiert werden. Die DCT-Implementierung für die Tests weist eine Komplexität von $O(n^2)$. Bei etwa 600 Punke pro Linie ist das zu erheblichen Rechenaufwand. Deshalb wird vor der DCT deshalb ein Subsampling durch. Falls die Laufzeit der Dekompression verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von $O(nlogn)$. Falls das nicht ausreicht, können die Linien in Blöcken mit einer bestimmten Anzahl von Punkten unterteilt werden. Das senkt die Komplexität auf $O(n)$. Die Unterteilung könnte negative Effekte auf die Kompression haben.

In den Tests wurde eine lineare Quantisierung verwendet. Jeder DCT Koeffizient wird durch einen Faktor geteilt, der sich stetig ehöht. Zum Beispiel wird der erste Koeffizient durch zwei geteilt, der zweite durch Vier, der Dritte durch Sechse etc. Die Kompressionsrate kann durch einen höheren oder tieferen Faktor gesteuert werden. Diese Quantifizierung ist nicht das Optimum. Eine bessere Quantifizierung wird für die beste Lösung ausgearbeitet.

6.2.1 DCT

Nach dem Subsampling wird auf den Punkten (im kartesischen Koordinatensystem) die Diskrete Kosinus Transformation ausgeführt. Es ist auch möglich die Punkte im sphärischen Koordinatensystem in den Frequenzraum zu überführen. Der ϕ-Kanal ist jedoch schwierig durch tiefe Kosinus schwingungen darzustellen: Wie im Abschnitt 3.1 besprochen, beinhaltet der Kanal Sprünge bei der Passierung des Nullpunktes. Das führt zu sehr hochfrequenten Schwingungsanteile in der DCT. Nach einer Quantisierung sind dabei Artefakte nicht vermeidbar. Im kartesischen System hingegen sind alle Kanäle stetig und haben somit wenige hochfrequente Anteile.
Die Anzahl Punkte pro Linie werden zuerst als 16 Bit Integer Array abgelegt, gefolgt von allen DCT Koeffizienten des X Kanals, danach des Y und zum Schluss des Z Kanals mit 32 Bit Genauigkeit.
Die Abbildung 7 zeigt den Vergleich der DCT Kompression mit der Lösung 0. Es ist deutlich zu erkennen, dass die Standardabweichung schnell steigt bei leicht sinkender Grösse. Der Maximale Fehler steigt ebenfalls schnell und erreicht beim letzten Test eine höhe von $140'686'000$ Meter. Zum Vergleich: Der maximale Fehler der Lösung 0 ist mehr als vier Mal kleiner und liegt bei $30'014'000$ Meter.
Die Darstellung der Artefakte 9 zeigen das Problem: in den meisten Fällen kann die DCT die Feldlinie gut approximieren. Bei dieser Feldlinie wird der Anfang der Kurve nicht richtig dargestellt. Das ist ein typisches Problem der DCT: Die Transformation nimmt an, dass das Signal sich am Anfang und am Ende in umge-

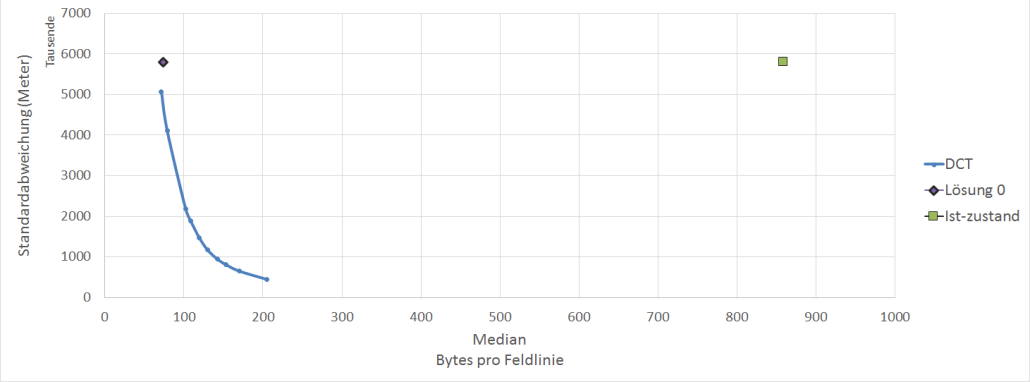

Abbildung 7: Vergleich der DCT Kompression mit der Lösung0

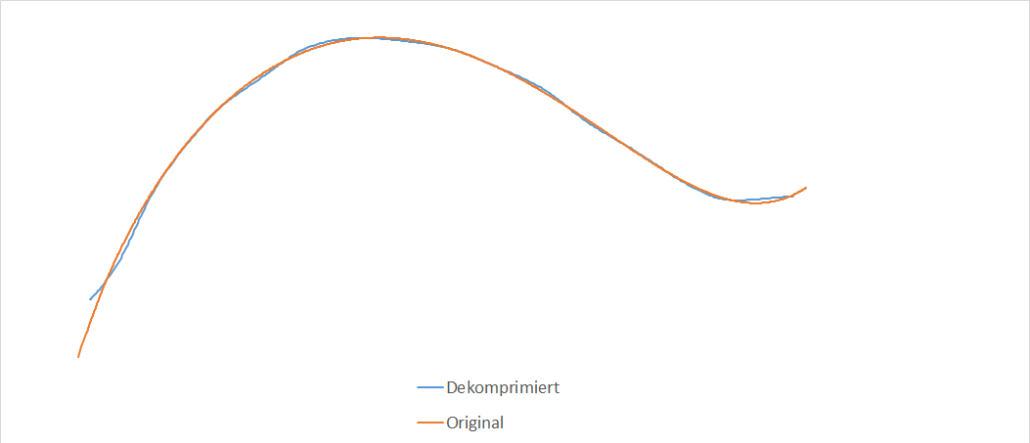

Abbildung 8: Artefakte der DCT Dekompression

kehrter Reihenfolge wiederholt [**?**]. Gerade beim Anfang bedeutet das ein Unterbruch, welche sich nur durch hochfrequente Anteile darstellen lässt.
Eine Möglichkeit ist die Feldlinie um Punkte zu erweitern, welche die Transformation vereinfachen. Wenn die Punkte geschickt gewählt werden, sollte eine Feldlinie nach der Transformation und Quantisierung ähnlich viele Bytes benötigen. Durch eine andere Darstellung könnten diese Probleme ebenfalls gelöst werden, weshalb zuerst weitere Transformationen analysiert werden.

6.2.2 Feldlinien Ableiten mit DCT

Bevor die Feldlinie Kosinus-Transformiert und Quantisiert wird, soll sie abgeleitet werden. Die Kosinustransformation wird auf den Steigungen ausgeführt. Damit die Operation umkehrbar ist, wird der Startpunkt der Feldlinie im kartesischen Koordinatensystem gespeichert. Mit der Ableitung sollen zu einem die Koeffizienten gedämpft werden, sodass sie sich mit weniger Genauigkeit quantisieren lassen. Zum Anderen soll das Randproblem dargestellt in 9 gelöst werden. Die Steigungen sind kleinere Zahlen, was nicht so einen starken Wechsel verursachen sollte. Der Nachteil ist, dass Ungenauigkeiten sich durch die Kurve durchziehen und summieren. Anfangs stimmen komprimierte und Originalkurve sehr genau überein. Aber gegen Ende können sie immer mehr abweichen.
Die DCT Koeffizienten und die X,Y und Z Kanäle der Startpunkte weren mit 16 Bit quantisierten getrennt abgelegt. Die DCT, die Quantisierung sowie die Entropie Kodierung ist die Selbe wie bei 6.2.1. Kann gut

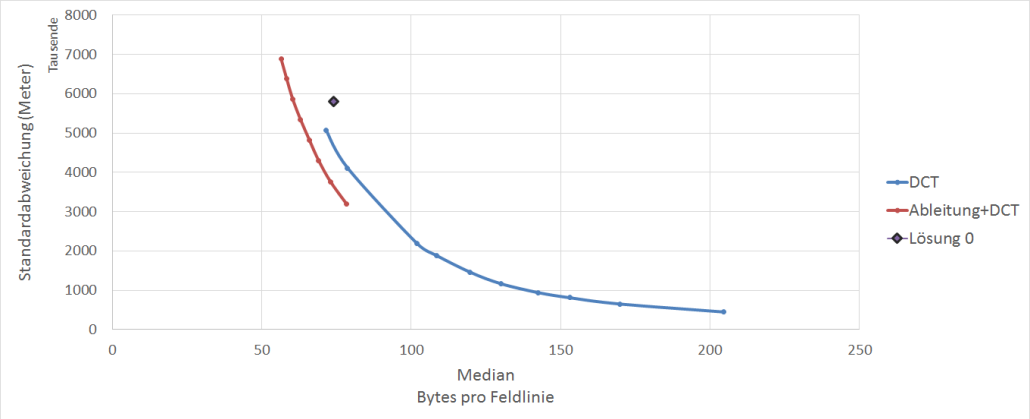

Abbildung 9: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

Quantisiert werden, deutliche Verbesserung zur Lösung 0. Ebenfalls deutlich besser zur lösung 1. Aber noch mehr möglich. Momentan ist die Information gleichmässig auf alle Kanäle verteilt. Die Linien liegen aber oft auf einer Ebene im 3d Raum. Die Komprimierung könnte verbessert werden, wenn die Feldlinien nicht im Sonnenkoordinatensystem, sondern im Lokalen Komprimiert werden. Wenn eine Linie in einer Ebene liegt, ist eine Komponente 0.

6.2.3 Feldlinien Ableiten mit PCA und DCT

Mit einer vorhergehenden Principal Component Analysis (PCA) [**?**] werden die Feldlinien vom Sonnen-Koordinatensystem in ihr lokales System transformiert. PCA ist ein Verfahren aus der Statistik, welches Daten in ein neues koordinatensystem Transformiert. Dabei werden die Achsen so gelegt, dass die Daten entlang der ersten Achse die grösste Varianz aufweisen. Entlang der zweiten Achse, welche orthogonal zur ersten liegt, die zweithöchste Varianz etc. Wenn das Vefahren auf die Feldlinien angewandt wird, werden die Feldlinien in ein lokales System transformiert indem der Z-Kanal 0 ist, wenn die Feldlinie in einer Ebene liegt.

Der Nachteil ist, dass für die Rücktransformation pro Feldlinie die Koordinatenachsen und die Koordinatenverschiebung abgespeichert werden. Für Achsen werden 16 Bit Genauigkeit und für die Verschiebung 32 Bit verwendet. Der Vergleich 10 zeigt deutlich, dass sich der Mehraufwand nicht lohnt, obwohl die PCA vielver-

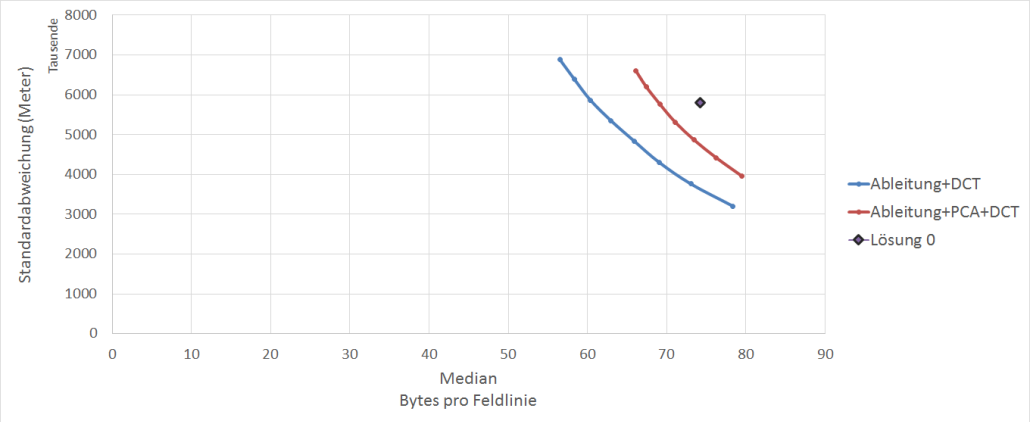

Abbildung 10: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

sprechend scheint. Wenn man quantisierte DCT-Koeffizienten einzelner Feldlinien analysiert, so fällt auf dass von etwa 200 Koeffizienten zwischen 180 und 195 Nullen sind. Die selben Linien mit der PCA-Transformation weisen tendenziell tiefere, aber ähnlich viele Koeffizienten. Die PCA-Transformierten Feldlinien brauchen aber noch zusätzlich Werte der neuen Koordinatenachsen und der Verschiebung.
Die PCA-Variante könnte noch verkleinert werden. Die Verschiebung kann Quantisiert werden, oder man kann weniger Genauigkeit für die Koordinatenachsen verwenden. Die PCA-Variante ist aber nicht genauer wie die Ableitung+DCT Variante. Unter dem Strich hat die PCA keine Verbesserung erbracht.

Weitere Verfahren lohnen sich beinahe nicht. Aufwand grösser als Nutzen Nur unterschiedliche Kurven Speichern: Transformation beinahe teurer als Kurve selbst. Verschiebung + Rotation wenn bei ähnlichen Kurven nur eine abespeichert werden soll. Median DCT Kurve Ableitung der DCT?

6.2.4 Feldlinien Ableiten mit DCT und Byte-Codierung

wieder 6.2.2, nur mit einer Byte-Codierung. Meistens sehr tiefe Koeffizienten, aber ein paar wenige Ausreisser. Adaptive Genauigkeit. Für die meisten Fällen wird ein Byte benutzt, wenn mehr genauigkeit gebraucht wird kommt ein Byte hinzu. Speicherung bis zum letzten nicht-null koeffizienten. figure vergleich. gute resultate. Zeigt, wie tief die eigentlichen komponenten sind. Byte RLE, rar sollte das für mich übernehmen, oder? Artefakte Problem gelöst? (Vergleich der Artefakte). Nicht schlecht, artefakte sind bei Lösung 0 schon nicht von auge her sichtbar. Die Ränder sind auch gut, jedoch scheint die pure DCT die Kurve besser approximieren zu können, falls die Ränder nicht wären.

6.2.5 DCT mit Byte-Codierung

Zum Vergleich wird das jetzt noch gemacht.

7 Diskussion

8 Fazit

Literatur

[1] Nobody Jr. My article, 2006.

[2] Marcus Geelnard. Openctm the open compressed triangle mesh file format, November 2014.

Abbildungsverzeichnis

Tabellenverzeichnis

9 Anhang

subsectionInstallationsanleitung

10 Ehrlichkeitserklärung