Abstract

Inhaltsverzeichnis

1 Verlustbehaftete Kompression von volumetrischen Punkten

Computersimulationen produzieren grosse Datenmengen. In dieser Arbeit soll eineLinien, welche als Menge von volumetrischen Punkten abgebildet ist, verlustbehaftet komprimiert werden. sodass eine Übertragung über eine Internetverbindung in sinnvoller Zeit abgeschlossen ist. Die Artefakte der Kompression sollen sich im Rahmen halten. Aber dennoch schön aussehen.

teilchenphysik, sonnenForschung, feldlinien Konkreter Anwendungsfall der JHelioviewer. Visualisiert Satellitenmessungen und Simulationen. Mittels PFSS werden die Feldlinien simuliert. Datenmenge soll nun auf den jeweiligen JHelioviewer des Sonnenforschers übertragen werden. (Feldlinienbild). Resultate

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

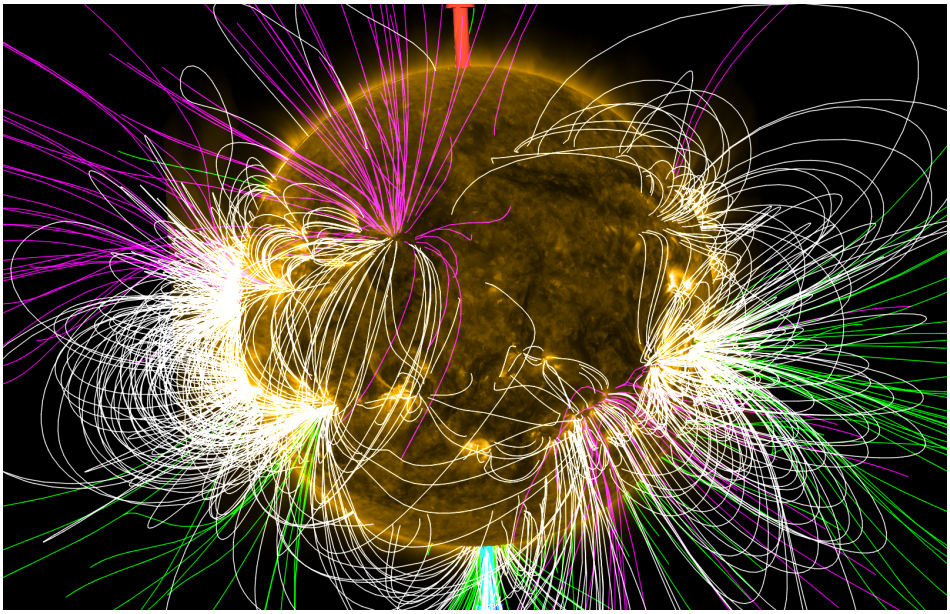

Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

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-

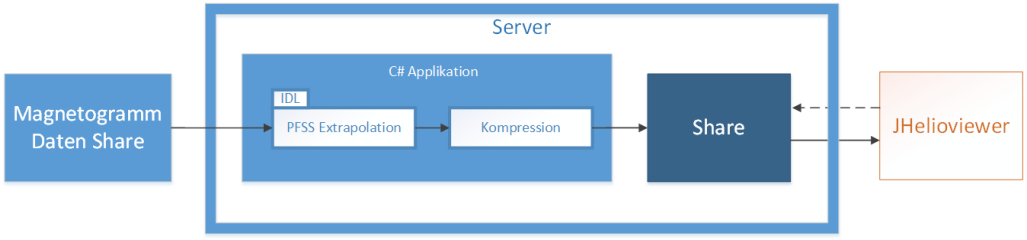

Abbildung 2: Aufbau und Datenfluss des Servers

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

Grob betrachtet können verlustbehaftete Kompressionen in drei Teilschritte aufgeteilt werden: Transformation, Quantisierung und Entropie Kodierung. Die Abbildung 3 zeigt eine vereinfachte Abfolge. Die Inputdaten werden zuerst duch ein oder mehrere Verfahren transformiert. So transformieren, dass in der Quantisierung unwichtige Informationen gelöscht werden können. Hier geschieht der Datenverlust. Danach werden die Daten Entropie Kodiert, was wieder verlustfrei ist. Unterschiedliche Verfahren für jeden Teilschritt. Meist auch mehrere Transformationen, oder Transformation Quantisierung Transformation Quantisierung.
Folgend werden Kompressionsverfahren oder typische Algorithmen für einen Teilschritt beschrieben.

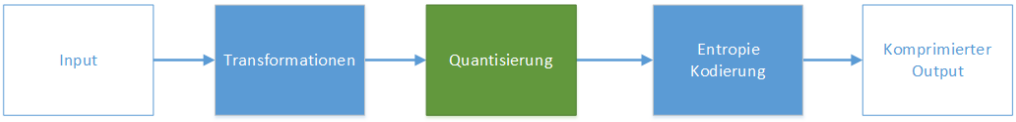

Abbildung 3: Typische Teilschritte einer Kompression

2.1 JPEG/JFIF Kompression

Weit verbreitete Bild YCbCr, DCT, Quantisierung, Entropie Kodierung.

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 [**?**] 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 Kompressionsverfahren der Feldlinien

konzept der unterschiedlichen kompressionsverfahren. zwei lösungsansätze. bereits eine simple Kompression implementiert.

3.1 Ist-Komprimierung

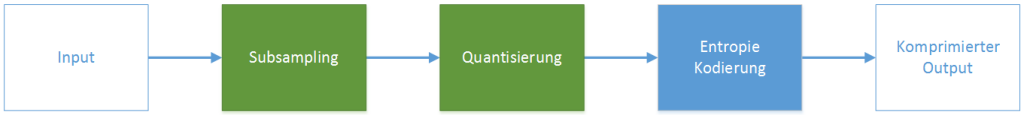

Abbildung 4: Aufbau der Ist-Kompression.

Quantisierung, Formatierung und Kodierung. Ist-Kompression. Jeder Kanal wird in 16-Bit Integer quantisiert. Abspeicherung: Jeder Kanal am Stück, sodass die Entropie Kodierung greifen kann. Entropiekodierung mittels gzip.

Subsampling auf JHelioviewer 3.2.1

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 Lösungsansatz Adaptives Subsampling

Subsampling wird nun auf dem Server ausgeführt. in den folgenden Abschnitten sind die Teilschritte erklärt.

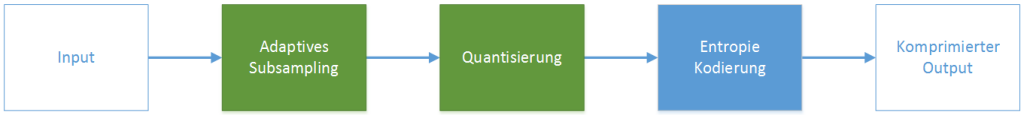

Abbildung 5: Aufbau des Lösungsansatzes: Adaptives Subsampling.

3.2.1 Adaptives Subsampling

Ziel des adaptiven Subsamplings ist es, die Daten durch eine Folge von Strecken zu approximieren. An stellen, welche die Feldlinie gekrümmt ist, braucht es mehr Strecken. An Stellen, welche die Feldlinie Linear verlauft können so viele Punkte gespart werden. Das Diagramm der Abbildung 6 stellt das Subsampling im

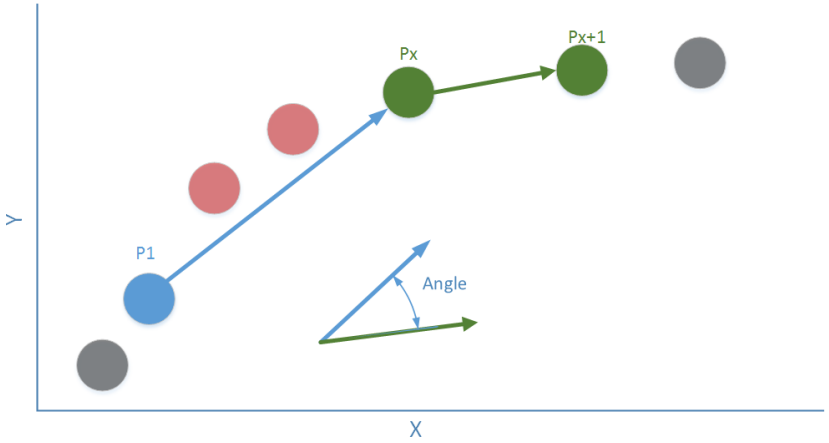

Abbildung 6: Darstellung des Adaptiven Subsapmlings im 2D Raum.

zweidimensionalen Raum dar. Zu sehen sind die Punkte der Feldlinie. Das Adaptive Subsampling wählt nun Punkte P aus der Feldlinie aus, welche Start- und Endpunkte der Strecken darstellen.
P_1 wurde bereits ausgewählt. Es wird nun ein Punkt P_x gesucht, der als Endpunkt einer Strecke von P_1 zu P_x die Feldlinie approximiert. Dazu wird der Winkel der Strecke P_1 zu P_x mit der Strecke P_x zu $P_x + 1$ verglichen. Wenn der Winkel kleiner ist, als ein Faktor F, wird der nächste Punkt $P_x + 1$ überprüft. Wenn der Winkel grösser ist, wird P_x ausgewählt und von P_x aus weiter geprüft.

3.2.2 Entropie Kodierung

Abspeicherung, Alle X Kanäle hintereinander, alle Y Kanäle etc. So kann die Entropide-Kodierung besser greifen. Rar hat sich bewährt bei der purer Feldlinien kompression im Vergleich zu anderen Verfahren wie LZ77/gZip. Bringt etwa 4 mal bessere Kompression hin, hat aber mehr mühe mit Floating-Point nummern als mit Integers

3.3 Lösung 1, Diskrete Kosinus Transformation

Bild DCT, da alles nahe an harmonischen Halbwellen kartesische Koordinaten –¿ kein wrap around,

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 lassen sich einfacher durch Kosinus-Funktionen approximieren.

3.3.1 Subsampling

Wie im Abschnitt 4.1 beschrieben, wurde aus den Testdaten die Quantisierung und das Subsampling entfernt. Die Feldlinien der Testdaten haben das vier Mal mehr Puntke, als beim Ist-Zustand übertragen werden. Die DCT-Implementierung weist eine Komplexität von $O(n^2)$ auf. Vor der Kosinus-Transformation wird deshalb das selbe Subsampling durchgeführt, wie im Ist-Zustand. So kann der Rechenaufwand in Grenzen gehalten werden.
Falls die Laufzeit der Dekompression verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von $O(nlog(n))$. Der Nachteil ist, dass nur Daten der Länge 2^n transformiert werden können, was zusätzliche Programmlogik braucht. Falls die Fast-Cosine-Transformation nicht ausreicht, können die Linien in Blöcken mit einer bestimmten Anzahl von Punkten unterteilt werden. Dadurch wird die Komplexität auf $O(n)$ gesenkt. Jedoch ist es wahrscheinlich, dass durch die Unterteilung die Kompressionsrate leidet. Vermutlich braucht es für die Approximation der Blöcke insgesamt mehr Kosinus-Funktionen, als für die Approximation der gesamten Feldlinie.

3.3.2 Randbehandlung

3.3.3 Cosinus-Transformation

Transformiert Daten in den Frequenzraum. Eine Menge von Punkten mit der Länge N kann in eine Menge von N Kosinusfunktionen überführt werden.
Es gibt verschiedene Möglichkeiten die Punkte zu transformieren. Der Unterschied liegt in der RAndbehandlung. Es wurde sich am JPEG/JFIF Standard orientiert(cite JPEG paper 1997), welcher, welche die DCT-II als Forwärts und die DCT-III als Rückwärtstransformation verwendet. Der Rand $X_k = \sum_{n=0}^{N-1} x_n cos[\frac{\pi}{N}(n + \frac{1}{2}k)]k = 0, 1, \ldots, N-1$ N ist anzahl Punkte. Transformiert in den Frequenzraum. x_n ist ein Punkt des Signales und X_k ist die Resultierende Kosinus Schwingung

Inverse Transform $X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n cos[\frac{\pi}{N}n(k + \frac{1}{2}]k = 0, 1, \ldots, N-1$

3.3.4 Quantisierung

3.3.5 Ablegung ins Fits Format

3.3.6 Entropie Kodierung

Kodierung Zuerst einfaches Run-length

Weitere Kodierung mittels Rar. Rar hat sich bewährt bei der purer Feldlinien kompression im Vergleich zu anderen Verfahren wie LZ77/gZip

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. Es ist wichtig, dass die Form der Kurve. Grosse, seltene Abweichungen können das Aussehen beeinflussen.

Im Verlauf der Arbeit wurden zwei unterschiedliche Metriken verwendet: Die Standardabweichung und eine angepasste PSNR-HVS-M. Für die ersten Messungen wurde die Standardabweichung gemessen. Es stellte sich heraus, dass die Metrik für den Anwendungsfall nicht ausreicht: Sichtbare Artefakte, wie hochfrequente Schwingungen um die Originallinie fallen nicht ins Gewicht. Wie die Standardabweichung berechnet wird, ist im Abschnitt 4.2 beschrieben. Für weitere Messungen wurde eine angepasste PSNR-HVS-M Metrik verwendet. Es berücksichtigt nur Artefakte, die vom menschlichen Auge sichtbar sind. 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.
Subsample

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 und ein Subsampling durch. Für die Testdaten wurde das Subsampling und die Quantisierung entfernt. Für jede Dimension eines Punktes wird anstatt 16 Bit 32 Bit Genauigkeit verwendet. Die rohe Datenmenge ist dementsprechend Angewachsen auf etwa 10 MiByte pro Aufnahme.

4.2 Berechnung der Standardabweichung

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

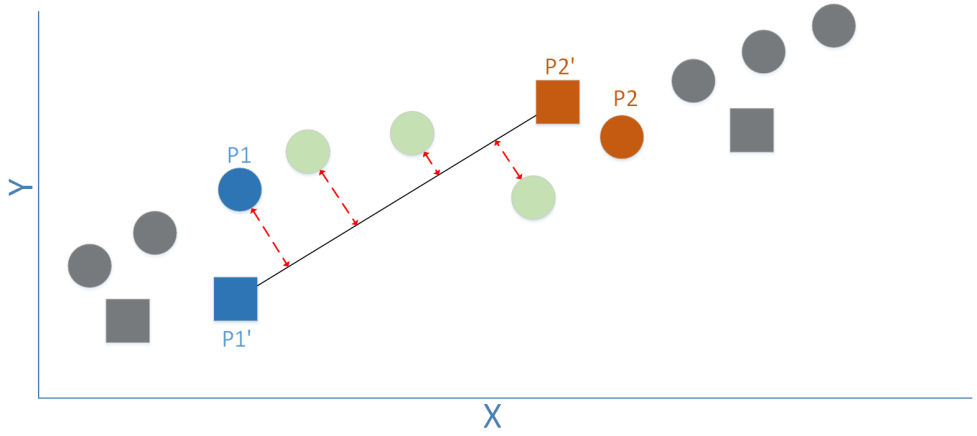

Abbildung 7: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

Die Berechnung ist Dargestellt im Diagramm 7. 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 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 8 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.

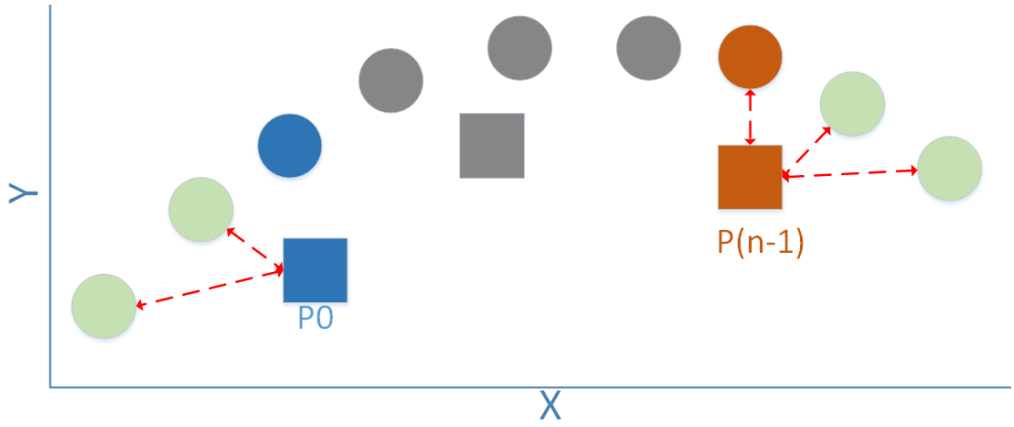

Abbildung 8: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

4.2.3 Berechnung der Standardabweichung

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

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.

4.3 Berechnung der angepassten PSNR-HVS-M

grund dafür, ref auf resultate Warum ist die Standardabweichung ungenügend: Mathematisch sinnvoll, aber die Daten werden schlussendlich von einem Menschen angeschaut. Gewisse Artefakte verschwinden, gewisse sind gut sichtbar. Ein sinnvolles mass für menschlichen Sehapparat zu finden ist ein aktives Vorschungsgebiet der Bildverarbeitung.

PSNR:

Verbesserung der PSNR-HVS

Awbwandlung

4.3.1 Umsetzung

Faktoren, Empirisch getestet mit algos. eher konservativ maxvalue der psnr

4.3.2 Wertebereich und Grenzen der PSNR

Wann ist gut, wann knapp. Jetzt eher knapp

5 Implementation

Um die Feldlinien darzustellen müssen diese heruntergeladen und dekomprimiert werden. Die zwei Operationen sind die Hauptverantwortlichen für die Wartezeit. Um die Wartezeit zu verkürzen, werden Feldlinien bereits im Voraus asynchron heruntergeladen. Somit sind die Daten bereits im Arbeitsspeicher, bevor die Visualisierung sie benötigt.
Die Wartezeit kann mit zusätzlichen Massnahmen weiter verkürzt werden Folgende Massnahmen wurden implementiert:

1. Asynchrone Dekompression.
2. Vorladen der Dekomprimierten Feldlinien.
3. Mehrstufiges Caching.

Die dekomprimierten Feldlinien sollen bereit stehen, bevor sie gebraucht werden.
Damit die Visualisierung durch das Read-Ahead dekomprimieren nicht unterbrochen wird, wird die Dekompression neu Asynchron ausgeführt. Zusätzlich führt die Asynchrone Implementation zu Performanceverbesserung, da die Dekompression paralellisiert werden kann.
Durch das CachingDie unkomprimierten sowie die komprimierten Feldlinien werden im Arbeitsspeicher zwischengespeichert. Wenn der Benutzer nun ”Zurückspuhlt´´ in der Visualisierung, kann im besten Fall auf unkomprimierte FEldlinien zurückgegriffen werden. Im zweitbesten muss nur noch die Dekompression erfolgen. Im schlechtesten müssen wieder Feldlinien hinzugeladen werden.Sonderfall, bei sehr langsamen verbindungen möchte man alle Daten cachen: Möglichkeit alle komprimierten Feldlinien im Speicher zu cachen.

was nicht gemacht wurde, mehrere Dateien zusammengefasst.

5.1 Software Architektur

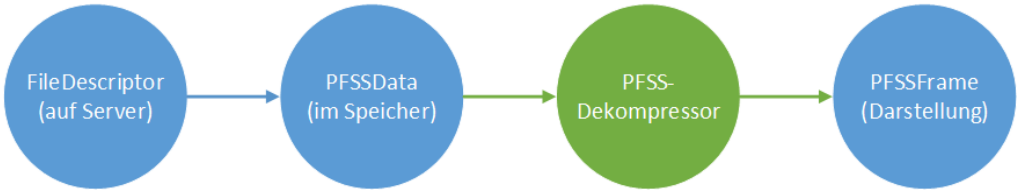

Abbildung 9: Zustandsdiagramm der Feldliniendaten

Die Daten der Feldlinien durchlaufen im JHelioviewer vier Zustände, welche durch vier Klassen abgebildet wurde. Die Klassen sowie die Zustandswechsel sind in Abbildung 9 dargestellt. Die Klasse ”FileDescriptor´´ repräsentiert eine Aufnahme von Feldlinien auf dem Server. In diesem Zustand sind die Daten bereit für das Herunterladen. Die folgende Klasse ”PFSSData´´ symbolisiert Feldlinien, welche in den lokalen Arbeitsspeicher geladen wurden. In diesem Zustand sind die Daten noch komprimiert und nicht bereit für eine Visualisierung. Für das Herunterladen ist ebenfalls die ”PFSSData´´ Klasse zuständig. ”PFSSDekompressor´´ ist ein Zwischenzustand und stellt den Wechsel von komprimierten zu unkomprimierten Daten dar. Da der Zustandswechsel aufwändig ist, wird es durch eine eigene Klasse abgebildet. Die letzte Klasse ”PFSSFrame´´ repräsentiert die dekomprimierten Feldlinien. In diesem Zustand sind die Daten bereit für die Darstellung. Die Darstellung wird ebenfalls von der ”PFSSFrame´´ Klasse übernommen.

5.1.1 Mehrstufiges Read Ahead und Caching

Um den Flaschenhals herunterladen und Dekomprimieren zu umgehen, wird ein mehrstufiges Read-Ahead und Caching eingeführt. Es soll ein Read Ahead und Caching für die unkomprimierten ”PFSSFrame´´ Feldllinien und eines für die ”PFSSData´´ Objekte implementiert werden. Für das wurde folgende Klassen implementiert: (Klassendiagramm) Die Klasse ”Framemanager´´ stellt die erste Stufe dar des Read-Aheads dar. Sie ist dafür zuständig, das aktuelle ”PFSSFrame´´ Objekt im speicher zu behalten und falls welche Fehlen, sie vom FrameCache anzufordern. ”PFSSFrame´´ Objekte müssen jeweils Speicher auf der Grafikkarte alloziieren und selbst abräumen. Der Manager gibt den Objekten die Chance sich vor dem Darstellen zu inizialisieren und stellt sicher, dass die Ressourcen der Grafikkarte abgeräumt wurden. Dür das Caching der Frame-Objekte ist die Klasse FrameCache zuständig.
Das Preloading und Caching der PFSSData Objekte wird von der Klasse DataCache übernommen. Das Preloading ist als zweite Cache-Instanz umgesetzt. Hier braucht es keine Kontrolle darüber, wann ein Objekt aus einem Cache entladen wird. Der Garbage-Collector verwaltet alle Ressourcen des Data-Objekts.

Der Cache arbeitet mit einer FiFo queue für die entscheidung, wann ein Objekt aus dem cache entladen wird. Der JHelioviewer fragt im allgemeinen Fall sequenzell nach den Datenobjekten. Das älteste Objekt ist meistens das, welches am längsten nicht mehr gebraucht wird.
Der LRU-Cache funktioniert gut, wenn die anzahl Objekte deutlich grösser ist, als der Cache zu halten mag. Der LRU-Cache versagt aber beim Wrap-Around. Wenn es aber n Objekte gibt und der Cache $n-1$ Objekte speichern kann, so löscht er immer das Nächste Objekt, welches gebraucht wird.

5.1.2 Asynchrone Aufrufe mittels Executor Services

Um vom Preload und Caching sinnvoll gebrauch zu machen, muss die Dekompression und das Herunterladen der Feldlinien asynchron implementiert sein. Dazu müssen alle Klassen aus Abbildung 9 threadsafe sein. Die Klasse FileDescriptor ist einfach, diese ist Immutable und somit Threadsafe. PFSSData und Frame threadsafe, data + compressor implementieren runnable Nun Creators verwenden den Executor Service für bla

6 Resultate

Es gibt x Lösungenansätze mit x varianten. hier sind die Kompressionsraten der jeweils besten Varianten der Ansätze.
Bis auf den Lösungsansatz unter 6.1 Unterschiedliche Qualitätsstufen

6.1 Lösungsansatz: Adaptives Subsampling

Bild Der Ist-Zustand wird das Adaptives-Subsampling[1] im JHelioviewer durchgeführt. Diese Variante führt das Subsampling vor der Dateiübertragung durch. Die resultierende FITS-Datei wird mittels Rar kodiert[2]; Wie

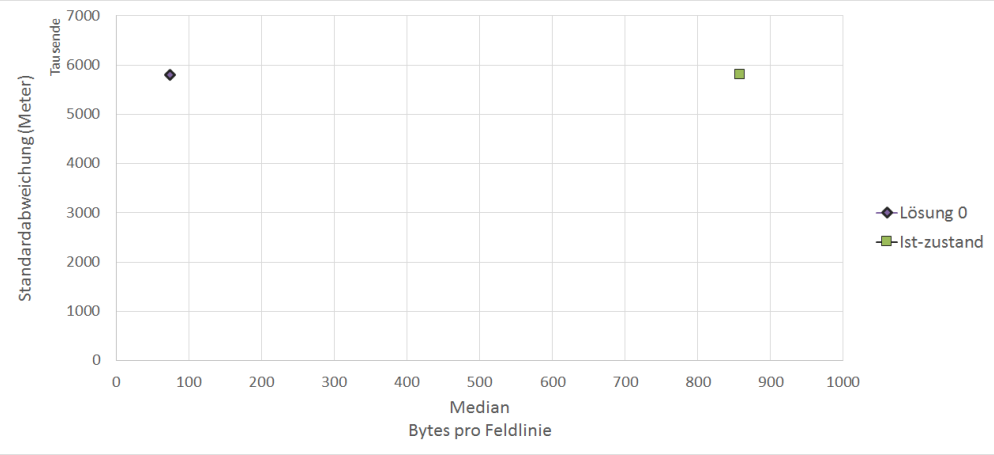

Abbildung 10: Vergleich der Lösung 0 zum Ist-Zustand.

im Diagramm 10 erkennbar ist, verbraucht die Lösung 0 deutlich weniger Speicher. Das Angle-Subsampling reduziert deutlich die Anzahl Punkte, während die Rar eine bessere Kompression erbringt. 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 11 zeigt die Artefakte, die bei der Komprimierung der Lösung 0 entstehen. Es ist anzumerken, dass der Ist-Zustand die selben Artefakte aufweist.

6.2 Lösungansatz: Diskrete Kosinus Transformation

Lösungsvarianten d In den folgenden Abschnitten werden die Resultate verschiedener Varianten vorgestellt. Alle Varianten bestehen grob aus fünf Teilschritten: Einem Subsampling[3], einer Folge von verschiedenen Transformationen, bei der eine die Diskrete Kosinus Transformation ist, Abspeicherung ins Fits Format[4] und-einer Quantisierung und einer Entropie Kodierung mit Rar[5].

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,

[1] siehe Abschnitt 3.2.1
[2] siehe Abschnitt3.2.2
[3] siehe Abschnitt 3.3.1
[4] siehe Abschnitt **??**
[5] siehe Abschnitt 3.2.2

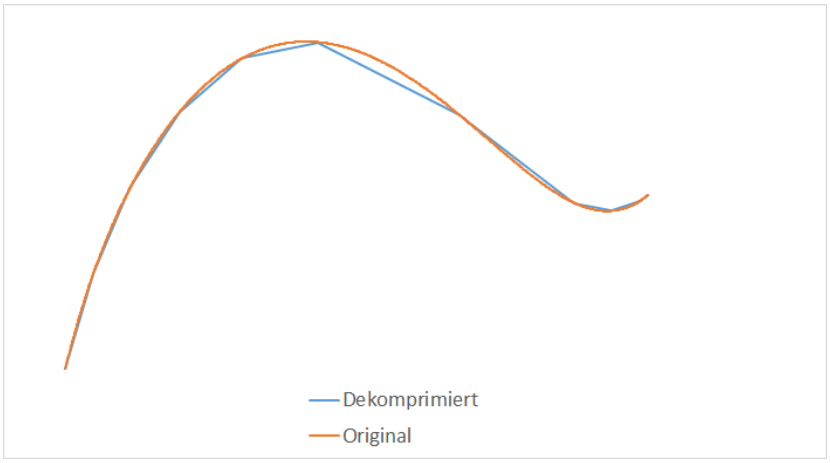

Abbildung 11: Artefakte der Lösung 0

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 Variante: DCT

Bild verwendet als Transformation nur DCT. Eizige Veränderung, jeder Kanal wird mit 32 Bit Integer anstatt mit 16 Bit abgespeichert. DCT-Koeffizienten sind sonst zu gross. Die Abbildung 12 zeigt den Vergleich der DCT

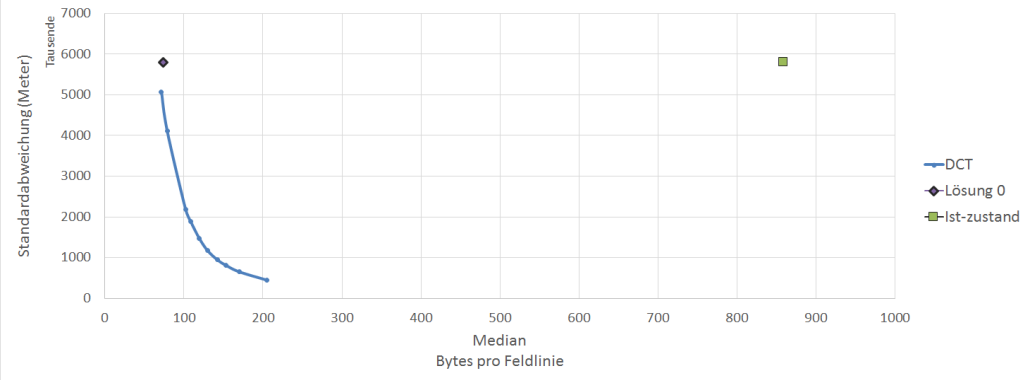

Abbildung 12: Vergleich der DCT Kompression mit der Lösung0

Kompression mit der Lösung 0. Es ist deutlich zu erkennen, dass die Standardabweichung schnell steigt bei leicht sinkender Grösse. Der Maximale Fehler ist ist mehr als vier Mal grösser, als die der Lösungsvariante 6.1 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 14 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. In diesem Fall wird für die Transformation angenommen, dass sich das Signal am Anfang und am Ende in umgekehrter Reihenfolge wiederholt [**?**]. Das führt am Anfang der Kurve zu einer grossen Spitze, welche sich nur durch sehr hochfrequente Schwingungen darstellen lässt. Wenn diese Anteile durch die Quantisierung verschwinden, entstehen solche Artefatkte.

Eine Möglichkeit ist die Feldlinie um Punkte zu erweitern. Wenn die Feldlinie am Anfang und am Ende abflacht, sollte die resultierende Transformation weniger hochfrequente Schwingungen enthalten. Diese Variante wird

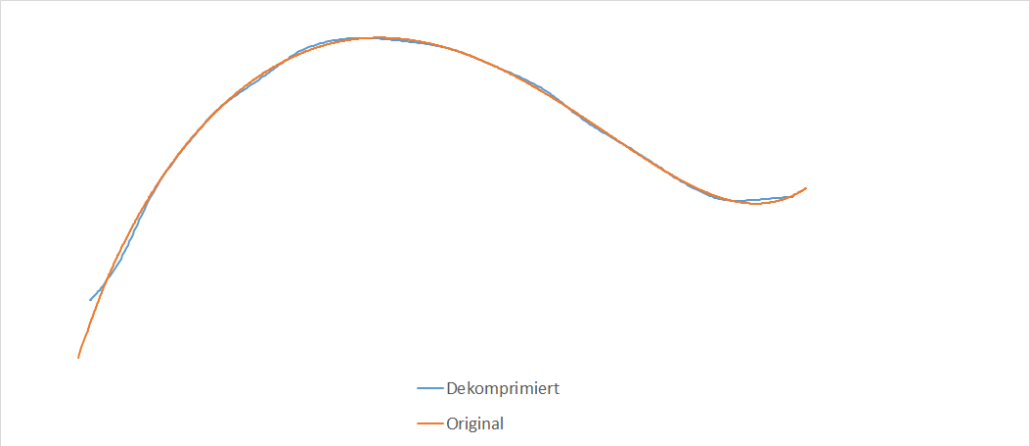

Abbildung 13: Artefakte der DCT Dekompression anhand Beispieldaten

im Abschnitt 6.2.5 behandelt und führt zu einer deutlich besseren Approximation. Durch eine andere Darstellung der Daten kann das Problem ebenfalls gelöst werden. Diese Variante wird in den folgenden Abschnitten besprochen.

6.2.2 Variante: Ableitung+DCT

Bild Bevor die Feldlinie Kosinus-Transformiert und Quantisiert wird, soll sie abgeleitet werden. Dämpfung der koeffizienten, die diskretisierten Koeffizienten sollten weniger speicherplatz verbrauchen. Mit der Ableitung soll das Randproblem dargestellt in 14 gelöst werden. Die Steigungen sind kleinere Zahlen, was an den Rändern keine grosse Spitze verursachen sollte. Der Nachteil ist, dass Ungenauigkeiten sich durch die Kurve durchziehen und summieren.
Damit die Ableitung umkehrbar ist, wird neu jeder Startpunkt der Feldlinie mit in die Fits Datei abgelegt. Die DCT-Koeffizienten werden mit 16 Bit Genauigkeit abgelegt.

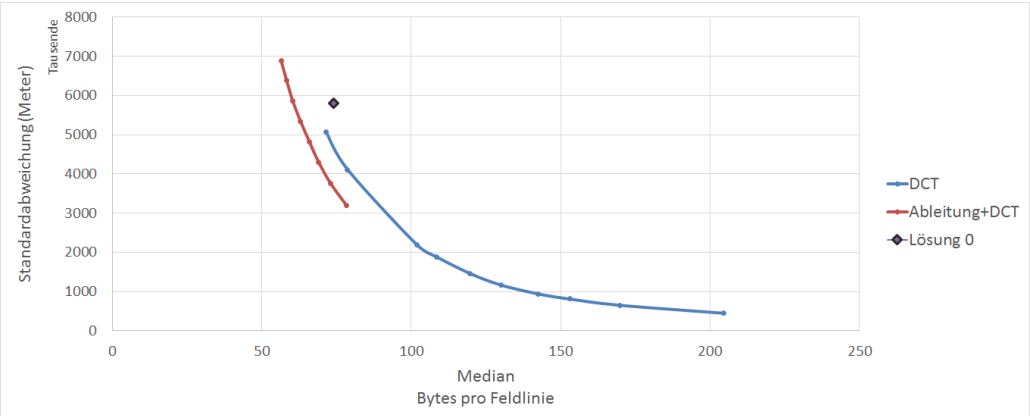

Abbildung 14: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

Die abgeleiteten Feldlinien können sehr gut quantisiert werden. Im Vergleich zur Lösung 0 braucht diese Variante etwa 15 Byte weniger um eine Feldlinie darzustellen. Durchschnittlich ist jetzt eine PFSS Simulation auf 72 KiByte komprimiert. Die Ränder können dargestellt werden, eine Darstellung der Artefakte ist auf der Abbildung 17 zu finden.
Die Feldlinien liegen meist auf einer Ebene im dreidimensionalen Raum. Wenn die X,Y und Z Kanäle Kosinus-

Transformiert werden, ist die Information etwa gleichmässig auf den Kanälen verteilt. Eine Linie könnte sich durch weniger Kosinus-Funktionen approximieren lassen, wenn die Linie zuerst in ein lokales Koordinatensystem transformiert wird. Die Koordinatenachsen können für jede Linie so gelegt werden, dass der X und Y Kanal den grossteil der Informationen beinhalten. Es wird vermutet, dass für Approximation des X und Y Kanals mit etwa gleich viele Kosinus-Funktionen gebraucht werden, aber für den Z Kanal bedeutend weniger.

6.2.3 Variante: Ableitung+PCA+DCT

Bild Die Principal Component Analysis (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.
Vor der DCT wird nun eine PCA durchgeführt. Die sechs Faktoren der neuen Koordinatenachsen werdenmit 16 Bit Genauigkeit in die Fits-Datei abgelegt und die drei Verschiebung-Faktoren mit 32 Bit. Der Vergleich 15

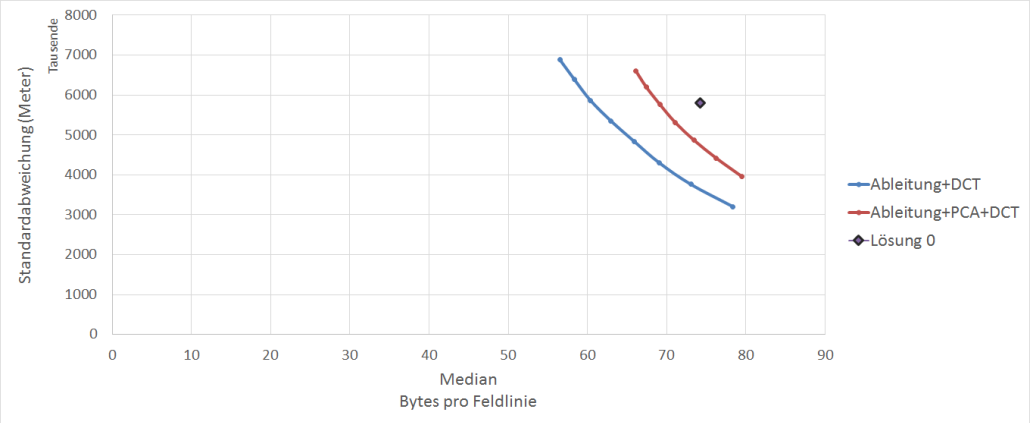

Abbildung 15: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

zeigt deutlich, dass sich der Mehraufwand nicht lohnt, obwohl die PCA vielversprechend scheint. Eine Feldlinie lässt sich mit 5 bis maximal 20 Kosinus-Funktionen pro Kanal approximieren. Durch die PCA-Transformation lässt sich das noch minim verkleinern, aber die zusätzlichen Informationen wie die Werte für neuen Koordinatenachsen und für die Verschiebung verbrauchen mehr speicher, als durch die Transformation gewonnen werden kann.
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. Wie auch in 15 ersichtlich, ist die PCA-Variante bei vergleichbarer Kompressionsgrad ungenauer. Unter dem Strich hat die PCA keine Verbesserung erbracht.

Es gibt weitere Transformationen, welche die Feldlinien so darstellen, dass weniger Kosinus-Funktionen für die selbe Approximation gebraucht werden. Die Transformationen brauchen aber für die Rückwärds-Operation meist zusätzliche Informationen. Zusätzlich bringen führen weitere Transformationen Ungenauigkeiten wie Rundungsfehler mit sich. Bei 5 bis 20 Kosinus-Funktionen pro Kanal ist es schwierig eine Transformation zu entwickeln, welche mindestens so genau ist und dabei weniger Speicherplatz verbraucht. Die ganze Kodierung wird momentan Rar überlassen. Dort gibt es noch Optimierungspotential.

6.2.4 Variante: Ableitung+DCT+Byte Kodierung

Es wird versucht, mit einer Byte Kodierung die DCT-Koeffizienten der Variante 6.2.2 besser zu komprimieren. Die Koeffizenten werden mit zwei Verfahren kodiert: Mit einer simplen Run-Length Kodierung und einer adaptiven Byte Kodierung. Die adaptive Byte Kodierung versucht jeden koeffizienten mit einem Byte darzustellen. Wenn die Genauigkeit nicht ausreicht, wird ein weiteres Byte hinzugenommen. Die Kodierung ist im Abschnitt 3.3.6 beschrieben. 16 zeigt eine deutliche Verbesserung der Kompressionsrate, wenn die Koeffizienten mit der

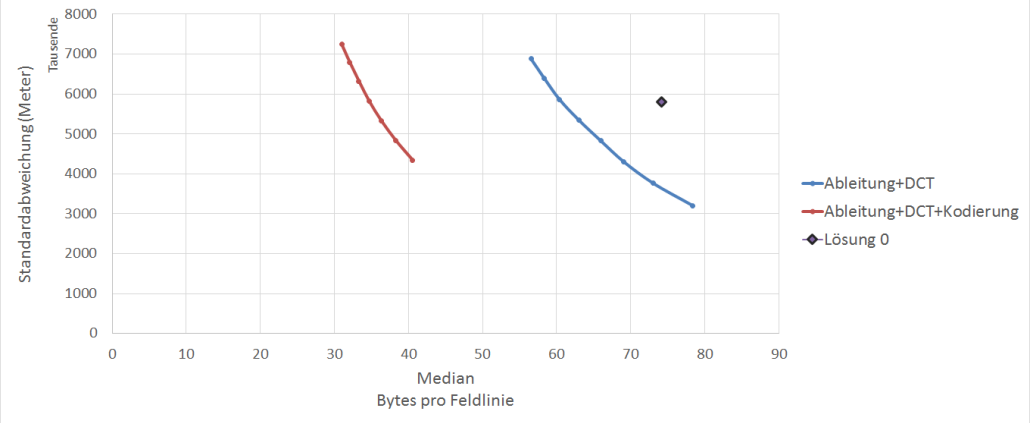

Abbildung 16: Vergleich der Kompression mit und ohne Byte-Kodierung

unter 3.3.6 beschriebenen Verfahren kodiert werden. Bei ähnlicher Genauigkeit wie der Ist-Zustand braucht diese Variante durchschnittlich 35 Bytes pro Feldlinie. Bei 1200 Feldlinien eine ergibt das eine Dateigrösse von 42 KiByte pro Aufnahme. Im Vergleich zum Ist-Zustand sind die Dateien um das 24 Fache kleiner.

Bei der Variante 6.2.1 war das Problem, dass die Ränder schlecht darzustellen waren. Es stellt sich die Frage, was für Artefakte diese Kompression aufweist. Im Diagramm der Abbildung17 ist deutlich zu sehen, dass die

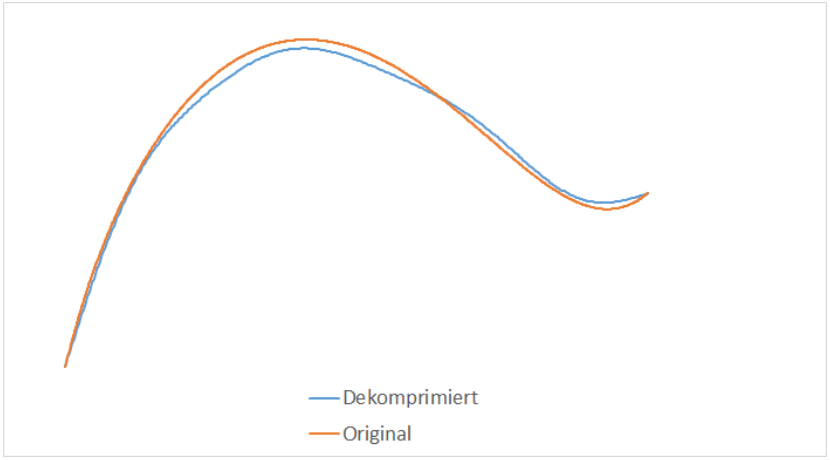

Abbildung 17: Artefakte der DCT Kompression der Ableitung

Kurve durch die Quantisierung gedämpft wird. Die Maximum der Kurve ist tiefer, sowie das lokale Minima der letzten Halbwelle höher. Der Vorteil dieser Variante ist, dass die resultierende Feldline sehr glatt verläuft. Ohne die Originalkurve währen die Artefakte nicht zu identifizieren.
Wenn die Artefakte 17 und 11 vergleicht, fällt auf, dass die Variante 6.2.1 die Feldlinie genauer approximiert. Wenn die Ränder besser dargestellt werden, ist es Denkbar, dass die Variante 6.2.1 wenigere Kosinus-Funktionen braucht für eine ähnlich genaue Approximation.

6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung

Wieder nur die Diskrete Kosinus Transformation, aber noch mit künstlich erzeugten Punkten3.3.2 und der Byte Kodierung3.3.6. Das Diagramm der Abbildung 18 zeigt den Vergleich der Variante mit Randbehandlung

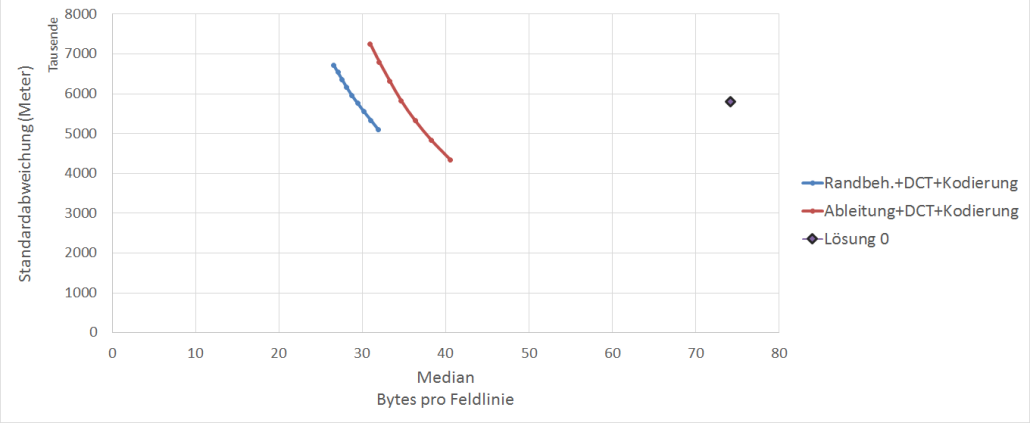

Abbildung 18: Vergleich der Kompression mit und ohne Byte-Kodierung

und der Variante der abgeleiteten Feldlinie (beschrieben im Abschnitt 6.2.4. Es ist zu erkennen, dass dank der Randbehandlung die Feldlinien mit weniger Bytes ähnlich genau approximiert werden können.

Trotz einer vergleichbaren Genauigkeit wie die des Lösungsansatzes unter 6.1, sind auf der JHelioviewer visualisierung deutliche Artefakte zu sehen. Die Abbildung 19 vergleicht die originalen mit dekomprimierten Feldlinien. Die Dekompression Lässt die Feldlinien um das originale Signal schwingen. In diesem Fall scheint

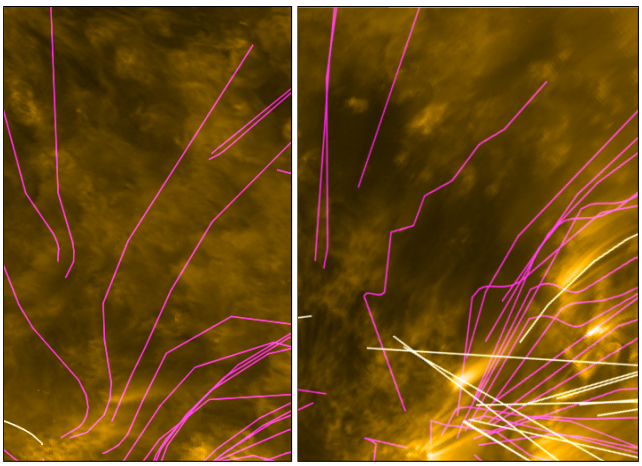

Abbildung 19: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten.

die Genauigkeitsmetrik zu versagen: Da die Schwingungen nahe am originalen Signal liegen, bleiben die Abstände klein. Die Approximation ist mathematisch betrachtet genau, aber für das menschliche Auge inakzeptabel.
Interessant ist, dass die die Variante der Ableitung beschrieben(6.2.4) ähnliche Artefakte aufweist. In der Abbildung 20 ist ein Vergleich der Original- mit den dekomprimierten Feldlinien dieser Variante zu sehen. Die Artefakte sind deutlich schwächer ausgeprägt. Metrik ist unbrauchbar, Deshalb wurde eine neue Metrik entwickelt, welche unter4.3 beschrieben ist.

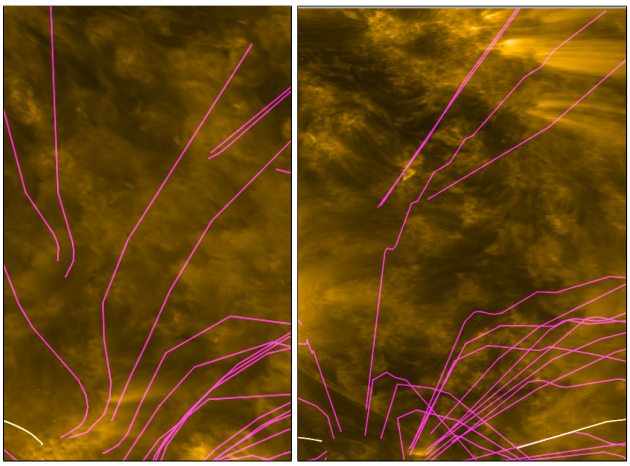

Abbildung 20: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4.

7 Diskussion

8 Fazit

Abbildungsverzeichnis

Tabellenverzeichnis

9 Anhang

subsectionInstallationsanleitung

10 Ehrlichkeitserklärung