Abstract

Ziel dieser Arbeit ist es ein verlustbehaftetes Kompressionverfahren von wissenschaftlichen Daten zu entwickeln, welche die Übertragung und Zwischenspeicherung der Daten ermöglicht. In dieser Arbeit wurden drei Verfahren entwickelt: Kompression mit Adaptivem Subsampling, Kompression mit einer Diskreten Kosinus Transformation und Kompression mit Prediktoren. Die höchste Kompressionsrate wurde mit der Diskreteten Kosinus Transformation erreicht, während die Kompression mit Adaptiven Subsamplings keine sichtbaren Kompressionsartefakte aufweist. Die Kompression mit Prediktoren ist ein Kompromiss zwischen Kompressionsrate und Artefakte.

Mit den entwickelten Verfahren ist die Übertragung von wissenschaftlichen Daten in vernünfiger Zeit ?umsetzbar?

Ziele wurden erreicht?

Weitere Möglichkeiten ist ein Kompressionsverfahren mit Wavelet Transformation oder Compressive Sensing.

Inhaltsverzeichnis

1 Einleitung

Moderne Simulationen sind in der Lage grosse Mengen an Daten zu produzieren. Die rohe Datenmenge ist oft zu gross um sie zu archivieren oder in vernünftiger Zeit über eine Internetverbindung zu übertragen. In wissenschaftlichen Simulationen werden natürliche Phänomene wie Schwingungen, Flugbahnen, Kraftfelder etc. als Menge von Kurven im dreidimensionalen Raum abgebildet. Die Kurven sind dargestellt als Folge von Punkten. Ziel dieser Arbeit ist es eine verlustbehaftete Kompression für die Punktfolgen zu entwickeln, welche die Übertragung über eine Internetverbindung und die lokale Zwischenspeicherung ermöglicht.

Im Rahmen dieses Projekts werden Daten von Magnetfeldlinien der Sonne komprimiert, welche über eine Internetverbindung zum JHelioviewer übertragen werden. Der JHelioviewer ist eine Applikation zur visualisierung von Satellitenmessdaten und Simulationen der Sonne. Die Applikation wird von der ESA und der FHNW entwickelt. Die Abbildung 1 zeigt eine Visualisierung des JHelioviewers von der Sonnenoberfläche und Feldlinien.
Auf der Visualisierung sind Feldlinien in drei unterschiedlichen Farben zu erkennen, welche drei unterschiedle

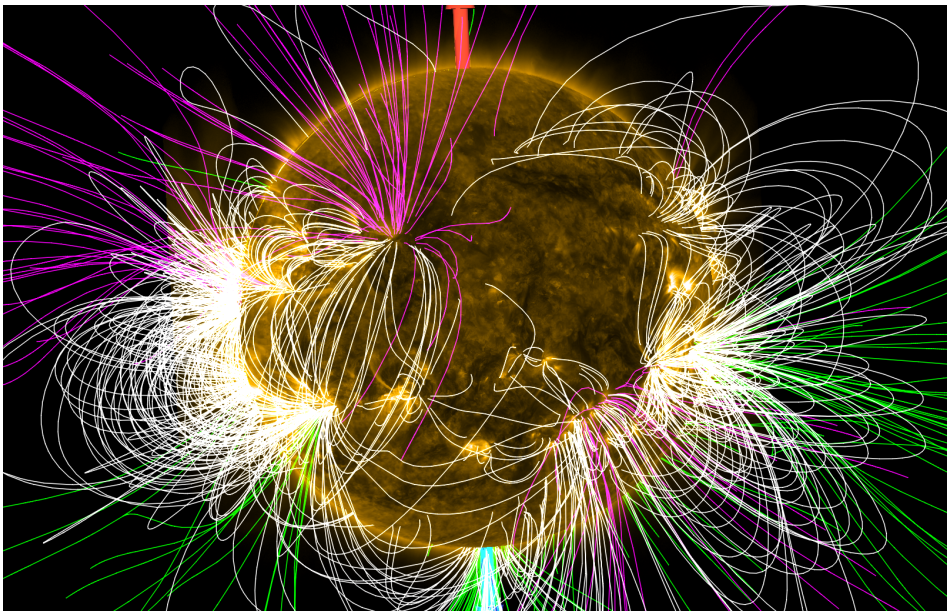

Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

Typen darstellen: Linien, die auf der Sonne starten und wieder 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 ins Weltall´´ und die Violetten ”Weltall zur Sonne´´.

Der JHelioviewer ist in der Lage Abfolge von Mess- und Simulationsdaten als Animation zu visualisieren. Die Daten für die Animation werden entweder on-the-fly heruntergeladen oder zuvor im Arbeitsspeicher gecached. Im Extremfall werden 1000 komprimierte Simulationen im Arbeitsspeicher zusammen mit anderen Mess- und Simulationsdaten abgelegt. Im Vorfeld wurde bereits eine verlustbehaftete Kompression entwickelt, welche die Datenmenge auf durchschnittlich 1 Megabyte pro Simulation reduziert. Das Caching von 1000 Feldlinien-Simulationen benötigt im Ist-Zustand durchschnittlich ein Gigabyte an Arbeitsspeicher. Das Ziel der Arbeit ist eine verlustbehaftete Kompression zu entwickeln, welche für das Caching 1000 Simulationen um den Faktor 8 bis 10 weniger Arbeitsspeicher benötigt.
Zusätzlich soll erforscht werden unter welchen Bedingungen und Kompromisse das on-the-fly Herunterladen von Simulationen möglich ist. Für die Animation der Feldlinien werden im Allgemeinen 1 bis maximal 10 Simulationen in der Sekunde benötigt. Ein Fernziel des JHelioviewers ist die kontinuierliche Animation der Feldlinien. Im Ist-Zustand ist jeder Wechsel der Simulation markant. Eine Möglichkeit das Ziel zu erreichen ist

die Kadenz zu erhöhen und 10-30 Simulationen in der Sekunde zu animieren.

Im laufe des Projekts wurden drei Kompressionen entwickelt. Die Kompressionsraten sind der Tabelle 1 ersichtlich.

Lösungsansatz	Kompressionsrate
Ist-Zustand	1
Adaptives Subsampling	11.6
Diskrete Kosinus Transformation	14.1
Prediktive Kodierung	13.4

Tabelle 1: Kompressionsraten der Lösungsansätze.

2 State of the Art

Die Teilschritte einer verlustbehaftete Kompressionen können in drei Verarbeitungsarten eingeteilt werden: Transformationen, Quantisierungen und Entropie Kodierungen. Die Abbildung 2 zeigt eine vereinfachte Abfolge. Die Inputdaten werden durch ein oder mehrere Verfahren transformiert. Die Transformationen haben das Ziel die Daten so aufzubereiten, dass die folgende Quantisierungen unwichtige Informationen löschen können. Die Transformationen sind meistens verlustfrei umkehrbar, sodass nur in den Quantisierungsschritten Informationen gelöscht werden. Die reduzierte Information wird Entropie Kodiert. Die Entropie Kodierung ist typischerweise für die Datenreduktion verantwortlich.

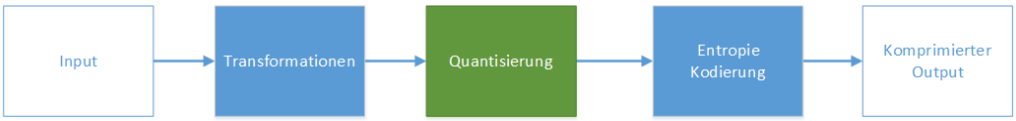

Abbildung 2: Vereinfachter Ablauf einer verlustbehafteten Kompression

2.1 JPEG/JFIF Bildkompression

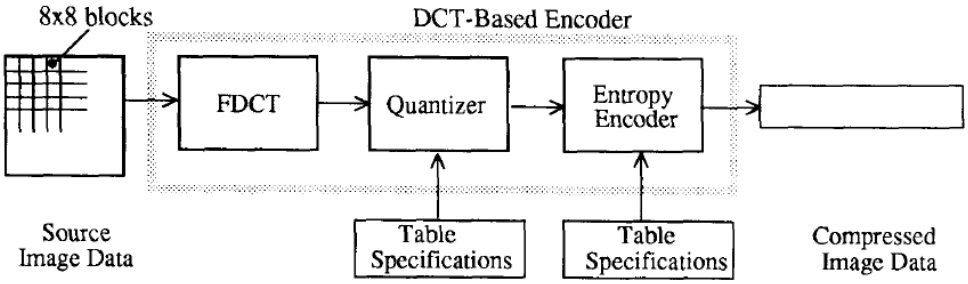

Abbildung 3: Aufbau der JPEG Kompression [1]

Der JPEG/JFIF Standard ist eines der meist verwendeten Bildkompressionsalgorithmen für natürliche Bilder. Das Diagramm der Abbildung 3 zeigt den Aufbau der Kompressionspipeline. JPEG/JFIF unterteilt das Eingabebild in $8 * 8$ Blöcke und führt eine Diskrete Kosinus Transformation (DCT) durch. Der Bildblock ist nun als Folge von Kosinus Funktionen dargestellt.
Die Quantisierung versucht nun Frequenzen, welche das menschliche Auge schlecht erkennen kann weniger Präzise darzustellen. Wenn die Quantisierung gut gewählt wurde, kann der Mensch das dekomprimierte Bild nicht vom Original unterscheiden, verbraucht aber weniger Speicherplatz. JPEG/JFIF bietet vorgefertigte Quantisierungstabellen an. Der Benutzer kann aber eigene Tabellen für spezifizieren. Wie die Quantisierungstabelle optimal gewählt wird, ist ein aktives Forschungsfeld [2] [3] und kann von Anwendungsfall zu Anwendungsfall unterschiedlich sein.
Nach der Quantisierung werden die quantisierten Blöcke im Zick-Zack-Muster angeordnet, sodass die Entropie Kodierung eine bessere Kompression durchführen kann. JPEG/JFIF führt eine Run-Length[**?**] und eine Huffman-Kodierung[**?**] durch. JPEG bietet auch hier an, eine benutzerspezifizierte Huffman-Tabelle zu verwenden.

Wenn für die Kompression von wissenschaftlichen Daten eine Diskrete Kosinus Transformation eingesetzt werden soll, kann ein ähnlicher Aufbau verwendet werden wie der JPEG/JFIF Strandard. Für eine optimale Kompression wird die Umsetzung der einzelnen Schritte vom Standard abweichen.

2.2 Point Cloud Kompression

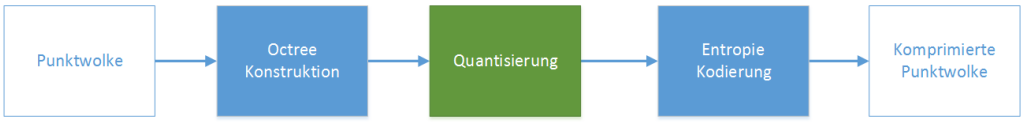

Abbildung 4: Aufbau einer Octree basierten Point Cloud Kompression.

3d Laser Sampling Geräte produzieren grosse Mengen an dreidimensionalen Punkten von alltäglichen Objekten. Die Kompression von solchen Punktwolken ist ein aktives Forschungsfeld. Eine vorgeschlagene Kompression von Schnabel und Klein [4] verwendet Octrees [5]. Das Diagramm der Abbildung 4 verdeutlicht den Ablauf.
Die dreidimensionalen Punkten werden in einem Octree mit einer begrenzten Anzahl an Levels abgelegt. Im Quantisierungsschritt werden die Punkte durch die Zellenmittelpunkte des Octrees ersetzt. Die Anzahl an Levels ist gleichzusetzen mit der Genauigkeit in Bits welche für jede Koordinatenachse zur Verfügung stehen. Wenn die Levels auf 8 begrenzt sind, steht für jede Achse 8 Bit Genauigkeit zur Verfügung.
In der Entropie Kodierung wird der Octree zuerst Binär abgebildet (0 für leere Knoten, 1 für befüllte Knoten) in Breadth-First Ordnung. Jedes Level im Octree stellt eine Approximation der Punktwolke dar. Durch die Breadth-First Ordnung werden die ungenaueren Approximationen zuerst abgelegt. Diese Eigenschaft wird mit einer Prediktiven Kodierung ausgenutzt: Aus den vorhergehenden Levels wird eine Vorhersage erstellt, wie die Punkte im nächsten Level zu liegen kommen. Die Kodierung kann die Information reduzieren, indem nur noch der Fehler der Kodierung abgespeichert wird. Je besser die Kodierung das Verhalten vorhersagen kann, desto besser ist die Kompression.

Für diesen Ansatz muss zu einem die Information, welcher Punkt zu welcher Linie gehört, gespeichert werden. Jedem Punkt kann eine Farbe zugewiesen werden. Die Farbe könnte man brauchen um die Punkte den Feldlinien zuzuordnen. Weiter müsste vermutlich die Prediktive Kodierung angepasst werden: Schnabel und Klein nehmen an, dass die Punktwolke eine Oberfläche darstellen. Die Feldlinien jedoch bilden im Allgemeinen keine Oberfläche und benötigen deshalb eine andere Prediktive Kodierung.

2.3 Curve Fitting

Curve Fitting ist ein Prozess, welcher ein Signal durch eine oder mehrere Funktionen abbildet. Es ist dabei zwischen einem exakten Curve Fitting und einer Approximation zu unterscheiden. Die exakte Repräsentation wird typischerweise für die Signalinterpolation verwendet, wäherend eine Approximation in der Rauschunterdrückung zum Einsatz kommt. Eine Datenkompression ist mit Curve Fitting möglich, wenn die Parameter der approximierenden Funktion weniger Speicherplatz benötigen, als das Signal.
Unser et al [6] zeigt ein Algorithmus, welcher ähnlich die Diskrete Kosinus Transformation ein diskretes Signal als Folge von B-Splines darstellt. Es wird aufgezeigt, wie eine verlustbehaftete Bildkompression mittels B-Splines möglich ist. Der Vorteil der B-Splines ist, dass die Artefakte der Dekompression als Rauschunterdrückung und Schärfung des Originalbildes ausdrücken.

Eine Datenkompression mit Curve Fitting ist im Vergleich mit der Kosinus Transformation weniger erforscht. Es sind ebenfalls keine Kompressionsstandards bekannt, welche ein Curve Fitting für die Datenkompression verwenden. Ein Kompressionsverfahren zu entwickeln zieht zusätzlichen Aufwand mit sich als Verfahren, welche in der Datenkompression etabliert sind.

2.4 Compressive Sensing

Das Compressive Sensing Verfahren stellt ein Inputsignal aus vordefinierten Funkionen dar. Das sogenannte Dictionary. Das Signal wird durch möglichst wenige Funktionen im Dictionary dargestellt (Sparse representation). Die Funktionen sind anders als bei der Wavelet Transformation nicht an Eigenschaften gebunden und können allgemein sein. Das Überführen in die Sparse representation ist ein $np - hard$ Problem [**?**]. Es existieren aber gute heuristiken wie Orhtogonal Matching Pursuit [**?**], welches in linearer Laufzeit eine genügend gute approximation erstellt.

Mit einem genügend allgemeinen Dictionary ist jedes Signal rekonstruierbar. Die Stärke des Compressive Sensing Ansatzes ist, dass das Dictionary auf die Signale optimiert werden kann. Ein möglicher Algorithmus ist der K-SVD [**?**] Ansatz.

2.5 Entropie Kodierung

Die Entropie Kodierung findet in allen Bereichen der Informatik Anwendungen. Ziel ist es die gleiche Information mit einem Minimum an Daten darzustellen. Verlustbehaftete Kompressionsverfahren reduzieren die Information und verwenden im letzten Schritt eine Entropie Kodierung um die Datenmenge zu reduzieren.

Es ist zwischen spezialisierten und allgemeinen Entropie Kodierer zu unterscheiden. Unter den spezialisierten Verfahren ist Beispielswiese die Arbeit von Ratanaworabhan et al.[7] einzuordnern, welche in der Lage ist Floating Point Daten performant zu kodieren und dekodieren. Unter den allgemeinen Verfahren sind Archivierer wie GZIP[8], 7-ZIP[9] und RAR[10] angesiedelt.
Archivierungsverfahren finden in allen Bereichen Anwendung und dienen in der Entwicklung von spezialisierten Kodierer als Messbasis. Im Vorfeld wurde die Kompressionsraten von GZIP, 7-ZIP und RAR von Feldliniendaten gemessen. Die Entropie Kodierung von RAR konnte stets eine höhere Kompressionsrate erreichen als andere Archivierungsverfahren. Die RAR Kompression ist urheberrechtlich Geschützt und die verwendeten Algorithmen sind nicht bekannt.

Es verwendet aber für unterschiedliche Datentypen unterschiedliche Verfahren

3 Umgesetzte Kompressionsverfahren

In diesem Abschnitt werden die einzelnen Verfahren zu den Lösungsansätzen vorgestellt. Im ersten Abschnitt 3.1 wird die Ist-Kompression analysiert. In den nächsten Abschnitten 3.2 bis 3.4 werden die Lösungsansätze und ihre Teilschritte vorgestellt.

3.1 Ist-Komprimierung

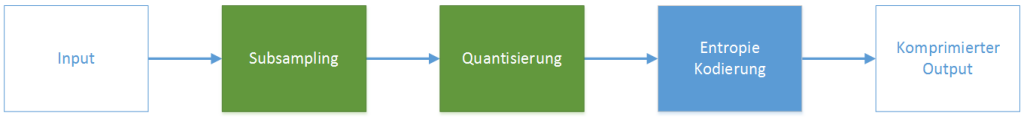

Abbildung 5: Aufbau der Ist-Kompression.

Das Ziel der Ist-Kompression ist es, die Datenmenge mit einfachen Mitteln drastisch zu reduzieren. Der Aufbau ist im Diagramm der Abbildung 5 dargestellt. Die Ist-Kompression führt zuerst ein Subsampling durch. Drei Viertel aller Punkte werden in diesem Schritt verworfen. Im nächsten Schritt werden die übrigen Punkte auf 16-Bit Integer diskretisiert. Das reduziert die Anzahl Bytes und verbessert die Kompression im Schritt Entropie Kodierung. Die Implementationen der Entropie Kodierer scheinen Integer-Werte einfacher komprimieren zu können. In der Entropie Kodierung werden die Daten geordnet wie in Tabelle 2 dargestellt.

Anzahl Punkte der Feldlinien	X Kanal aller Punkte	Y Kanal aller Punkte	Z Kanal aller Punkte

Tabelle 2: Anordnung der Simulationsdaten der Ist-Kompression

Als erstes werden die Längen aller Feldlinien abgelegt. Danach folgt der X, Y und der Z Kanal aller Punkte der Feldlinien. Diese Anordnung verbessert die Kompressionsrate der Entropie Kodierung. Je näher ähnliche Muster beieinander liegen, desto besser können sie Komprimiert werden. Für die eigentliche Entropie-Kodierung wird Gzip verwendet. Gzip basiert auf dem Deflate Algorithmus, welcher aus einer Kombination von LZ77 und Huffman Kodierung besteht [11].
Die Punktmenge ist für Low-End Grafikkarten zu gross. Um die Punktmenge für die Visualisierung zu verkleinern, führt der JHelioviewer ein weiteres Subsampling durch, welches im Abschnitt 3.2.1 beschrieben ist.

3.2 Lösungsansatz: Adaptives Subsampling

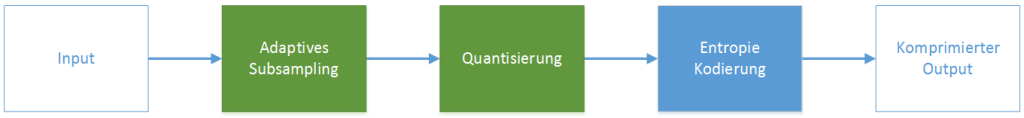

Abbildung 6: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Dieser Lösungsansatz verwendet einen ähnlichen Aufbau wie die Ist-Kompression 3.1. Der Unterschied ist, dass das Subsampling Verfahren gewählt wurde, welches der JHelioviewer auf dem Client durchführt. Die Abbildung 6 zeigt den neuen Ablauf.
Dieser Ansatz überträgt nur die Information, die der JHelioviewer zur Visualisierung benötigt. Der Vorteil ist dass die Information, die der JHelioviewer visualisiert ein Bruchteil darstellt und somit eine hohe Kompressionsrate ereicht werden kann. Es ist möglich dass die Visualisierung zu einem späteren Zeitpunkt mehr Punkte darstellen soll. In diesem Fall müssten entweder mehr Punkte übertragen werden, was eine schlechtere Kompressionsrate zur Folge hat, oder der JHelioviewer muss eine Interpolation durchführen.

3.2.1 Adaptives Subsampling

Konzeptionell approximiert das adaptive Subsambling eine Kurve diruch eine Folge von Strecken. Je stärker die Krümmung der Kurve, desto mehr Strecken werden für die Approximation benötigt.

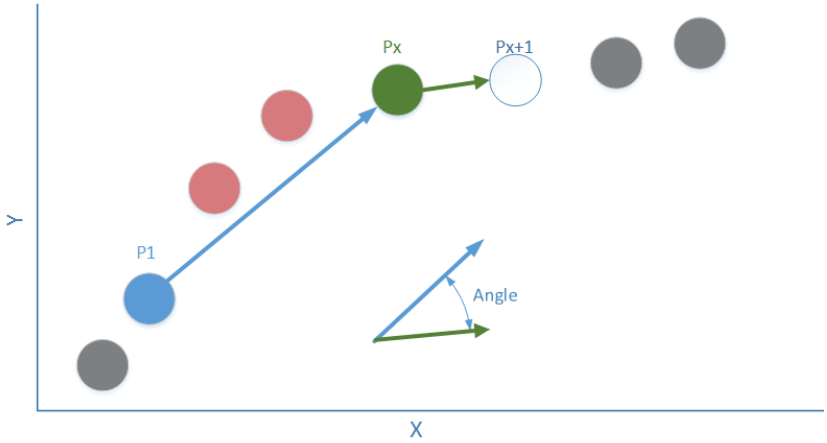

Abbildung 7: Darstellung des Adaptiven Subsapmlings im 2D Raum. Rot sind die Punkte, welche geprüft und gelöscht wurden. Grün ist der Punkt dargestellt, welcher geprüft wird.

Das Diagramm der Abbildung 7 stellt das Subsampling im zweidimensionalen Raum dar. 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 Winkel α, wird der nächste Punkt $P_x + 1$ überprüft. Wenn der Winkel grösser ist, wird P_x ausgewählt. Danach wird der wird eine nächste Strecke startend von P_x gesucht.

3.2.2 Entropie Kodierung mittels RAR

Die Anordnung der Daten wurde aus der Ist-Kompression übernommen, jedoch wird Rar anstatt GZip verwendet. GZip konnte bei den Ist-Komprimierten Daten eine Kompressionsrate von 1.2 erreichen, während Rar bei selben Daten eine Rate von 3.7 erreicht.

3.3 Lösungsantz: Diskrete Kosinus Transformation

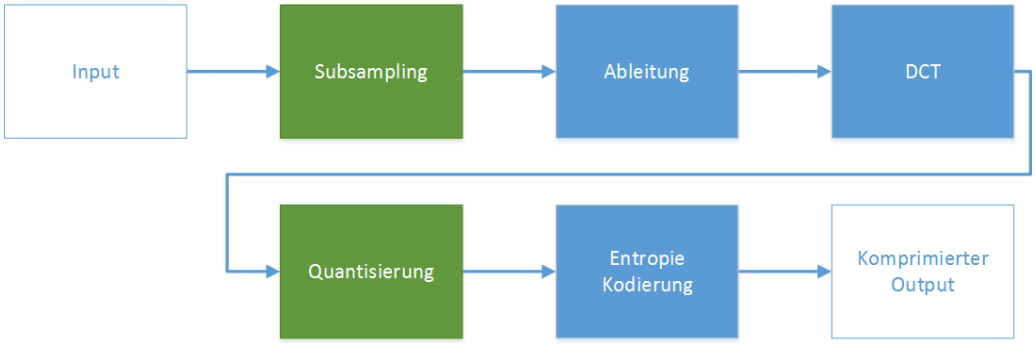

Abbildung 8: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Die Kompression dieses Lösungsansatzes ist Dargestellt im Diagramm der Abbildung 8. Konzeptionell ähnelt dieser Ansatz der JPEG/JFIF Kompression (dargestellt in der Abbildung 3), die einzelnen Teilschritte können aber andere Algorithmen verwenden. Im Vergleich zum JPEG/JFIF Standard ist der grösste Unterschied, dass das Eingangssignal vor der Kosinus Transformation abgeleitet wird.

Die Feldlinien ähneln oft harmonischen Halbwellen, welche sich durch wenige Kosinusfunktionen approximieren lassen können. Um eine optimale Kompression mit dieser Variante zu erreichen, müssen Ringing Artefakte [12] behandelt werden. Sie äussern sich als Oszillieren im dekomprimierten Signal, was das menschliche Auge als störend empfindet. Beispiele für die Ringing Artefakten von Feldlinien sind im Abschnitt 6.2.6 zu finden.
Es gibt Möglichkeiten, die Ringing Artefakte zu Dämpfen oder gar zu beheben: Die simpelste Variante ist es, das dekomprimierte Signal zu glätten. Die Glättung kann das Signal verfälschen und ist deshalb nicht die optimale Lösung. In der Bild- und Audioverarbeitung wird aktiv nach Post-Processing Filter geforscht, welche die Ringing Artefakte in der Dekompression vermindern [13] [14]. Ein angepasstes Postprocessing wurde im Rahmen dieser Arbeit nicht implementiert.

3.3.1 Subsampling

Das Subsampling wurde aus der Ist-Kompression 3.1 übernommen und dient, die DCT zu beschleunigen. Da die DCT eine Komplexität von $O(n^2)$ aufweist, wird durch das Subsampling die Transformation wesentlich beschlenuitgt.
Falls die Laufzeit der Dekompression weiter verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von $O(nlog(n))$. Falls das nicht ausreicht, können die Linien in Blöcke unterteilt werden und die DCT pro Block ausführen. Dadurch wird die Komplexität auf $O(n)$ gesenkt. Jedoch ist es wahrscheinlich, dass durch die Unterteilung die Kompressionsrate leidet. Die Approximation mehrer Blöcke des Eingabesignals benötigt schlussendlich mehr Kosinusfunktionen, als die Approximation des gesamten Signals.

3.3.2 Ableitung

Das Eingabesignal wird abgeleitet und alle folgenden Transformationen werden auf den Steigungen des Signals ausgeführt. Damit die Transformation umkehrbar ist, muss der Startpunkt zusätzlich abgespeichert werden.
Die Artefakte, welche das abgeleitete dekomprimierte Signal beinhaltet, sind bei der Feldliniensimulation weniger störend für das menschliche Auge. Die Feldlinie bleibt tendenziell glatt und Artefakte äussern sich meist

in veränderten Amplituden, welche erst erkennbar sind, wenn die Originalfeldlinie zum Vergleich bereit steht. Diese Eigenschaft wirkt sich ebenfalls auf die Ringing-Artefakte aus. Die Ableitung hat einen dämpfenden Effekt auf die Ringing Artefakte.

3.3.3 Diskrete Kosinus Transformation

Die Diskrete Kosinus Transformation stellt eine endliche Menge von N Datenpunkten als N Kosinusfunktionen zu verschiedenen Frequenzen dar. Die Werte DCT-Koeffizienten stellt dar, wie hoch der Anteil einer bestimmten Frequenz ist im Originalsignal. Im optimalen Fall kann ein Signal durch niederfrequente Funktionen approximiert werden. Die hochfrequenten Anteile stellen Details dar, welche meist nicht relevant sind.
Es gibt verschiedene Möglichkeiten die Punkte zu transformieren. Hier wurde sich am JPEG/JFIF Standard orientiert, welche die DCT-II (3.1) als Forwärts und die DCT-III (3.2) als Rückwärtstransformation verwendet [1].

$$X_k = \sum_{n=0}^{N-1} x_n * cos[\frac{\pi}{N} k(n + \frac{1}{2})] \quad k = 0, 1, \ldots, N-1 \tag{3.1}$$

$$x_n = \frac{1}{2} X_0 + \sum_{k=1}^{N-1} X_k * cos[\frac{\pi}{N} k(n + \frac{1}{2})] \quad n = 0, 1, \ldots, N-1 \tag{3.2}$$

N bezeichnet die Länge des Eingabesisignals, x_n bezeichnet einen Wert im diskreten Signal und X_k ist der Anteil der Frequens k. Ein Eingabesignal der Länge N resultiert in N Kosinus-Funktionen.

Die DCT transformiert ein periodisches, unendliches Signal. Um ein endliches Signal zu transformieren, wird es konzeptionell wiederholt. Die gewählten Verfahren wiederholen das Signal jeweils in umgekehrter Reihenfolge.
Die Wiederholung beeinflusst die Transformation nicht, wenn das Signal an den Rändern abflacht. Falls das Signal nicht abflacht, können nach der Quantisierung an den Rändern markante Artefakte auftreten. Ein Beispiel für solche Artefakte ist im Diagramm der Abbildung 19 im Abschnitt 6.2.1 zu sehen.

3.3.4 Quantisierung

In der Visualisierung werden die Feldlinien in drei Typen unterschieden: ”Sonne zu Sonne”, ”Sonne ins Weltall” und ”Weltall zur Sonne”. Die ”Sonne zu Sonne” Feldlinien können besser durch Kosinusfunktionen approximiert werden und sind weniger Anfällig auf die Ringing Artefakte. Dieser Typ von Feldlinien wird deshalb stärker Quantisiert. Für die ”Sonne zu Sonne” Feldlinien werden maximal 35 DCT Koeffizienten abgelegt, wobei die letzten zehn Koeffizienten kaum noch Einfluss besitzen. Sie werden mit einem hohen Faktor quantisiert, sodass sie im Normalfall ebenfalls Null sind.
Die anderen Feldlinien benötigen mehr Koeffizienten für die Dämpfung der Ringing Artefakte. Bei ihnen liegt das Maximum bei 50 Koeffizienten.

3.3.5 Entropie Kodierung

Um die Entropie Kodierung zu verbessern wurden zwei Byte-Kodierungen hinzugefügt. Die Längenkodierung reduziert 0-Einträge der Kanäle und die adaptive Genauigkeits-Kodierung reduziert die benötigten Bytes pro Wert eines Kanals.

Längenkodierung
Die Quantisierung erlaubt nur eine maximale Anzahl an Koeffizienten. Ab dem 50 Koeffizient, sind die Werte Null. Die Längenkodierung schneidet den Block von Null-Koeffizienten ab und fügt die Länge des Nicht-Null

Blockes hinzu. Alle Längen und alle Blöcke werden zusammen abgespeichert, die Tabelle 3 verdeutlicht das Konzept. n_i ist die Länge der Feldlinie i und $x_{i,j}$ ist der j DCT-Koeffizient der Feldlinie i.

X Kanäle							
Block Feldlinie 0					...		
n_0	$x_{0,0}$	$x_{0,1}$	...	$x_{0,n-1}$	n_1	$x_{1,0}$	...

Tabelle 3: Beispiel eines abgespeicherten Kanals mit der Längenkodierung.

Um einen Kanal zu Dekodieren, muss die Anzahl an DCT-Koeffizienten n_i gelesen werden und danach Nullen Anfügen, bis die ursprüngliche Länge des Kanals erreicht wurde. Die Anzahl Punkte jeder Feldlinie ist ebenfalls gespeichert.

Die Längenkodierung ist effizient, wenn die hochfrequenten DCT-Koeffizienten einen grossen, zusammenhängenden Null-Block bilden. Im schlechtesten Fall währe der letzte DCT-Koeffizient nicht Null. In diesem Fall würde die Längenkodierung nichts abschneiden.

Adaptive Genauigkeits-Kodierung
Die quantisierten Koeffizienten liegen meistens zwischen -50 und $+50$. Acht Bit Genauigkeit reichen im Allgemeinen aus um einen DCT-Koeffizienten abzuspeichern. Mit der adaptiven Genauigkeits-Kodierung sollen so wenige Bytes pro Koeffizient abgespeichert werden, wie benötigt werden. Dazu wird wie in Tabelle 4 eine neue Byte-Kodierung eingeführt. Das MSB wird als ”Continue Flag” verwendet. Wenn es gesetzt ist, gehört das folgende Byte ebenfalls zur Zahl.

Byte							
Continue Flag	X	X	X	X	X	X	X

Tabelle 4: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

Wenn ein Kanal hauptsächlich aus tiefen Zahlen besteht, können so Bytes gespart werden, ohne Genauigkeit zu verlieren. Wenn aber der Kanal aus hauptsächlich grossen Zahlen besteht, welche zwei oder mehr Bytes an Genauigkeit brauchen, wird weniger Speicherplatz gespart.

3.4 Lösungsansatz: Prediktive Kodierung

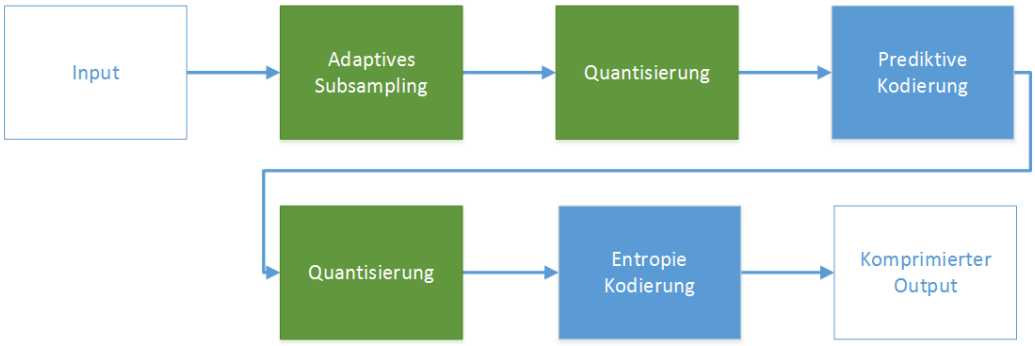

Abbildung 9: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Der Aufbau dieser Lösung ist im Diagramm der Abbildung 9 dargestellt. Im ersten Schritt wird ein Subsampling und eine Quantisierung der Daten durchgeführt. Darauf folgend werden die quantisierten Daten mit einer Prediktiven Kodierung versehen. Der zweite Quantisierungsschritt versucht unwichtige Informationen der Prediktiven Kodierung zu löschen.

Das Adaptive Subsampling reduziert die Anzahl Punkte und verändert die Eigenschaften der Kanäle der einzelnen Feldlinien. Stetige Steigungen werden zerstört, ein Beispiel ist im Diagramm der Abbildung 35 im Abschnitt **??**. Einfache Prediktoren wie zum Beispiel ein linearer Prediktor kann auf solche Daten keine zuverlässige Vorhersage treffen und verbessert die Kompressionsrate nicht. Es wurde deshalb der Rekursiver Linearer Prediktor entwickelt, welche auf der Idee der Point Cloud Kompression aus Abschnitt 2.2 basiert. Die Daten werden in unterschiedlich genauen Approximationen dargestellt. Der Prediktor benutzt die ungenaue Approximation um die nächst genauere vorherzusagen. Das Prinzip wurde für den Median Prediktor übernommen.

3.4.1 Angle Subsampling und Quantisierung

Das Angle-Subsampling vom Lösungsansatz aus Abschnitt 3.2 wurde mit angepassten Parametern übernommen. In diesem Ansatz werden 50% mehr Punkte übertragen. Die Zusätzliche Punkte werden übertragen, damit die Kompression nicht angepasst werden muss, falls der JHelioviewer in Zukunft mehr Punkte anzeigt.

Die Quantisierung überführt die Punkte in ein diskretes, sphärisches Koordinatensystem. In diesem Koordinatensystem können die Koordinatenachse mit 16 Bit Genauigkeit abgebildet werden, ohne hohe Abweichungen vom Originalpunkt in kauf zu nehmen.

3.4.2 Rekursiver Linearer Kodierung

Der Algorithmus der Rekursiver Linearer Kodierung versucht aus einer ungenauen Approximation die nächst genauere vorherzusagen. Im Diagramm der Abbildung 10 ist der erste Schritt des Algorithmus dargestellt. Es wird angenommen, dass der mittlere Wert des Kanals auf der Strecke zwischen Start- und Endwert liegt. Der nächste Schritt ist im Diagramm der Abbildung 11 dargestellt. Der Algorithmus wird rekursiv für die zwei Teilstücke des Kanals wiederholt, bis alle Werte des Kanals kodiert wurden, mit Ausnahme des Start- und Endwertes. Sie werden ohne Kodierung abgespeichert.

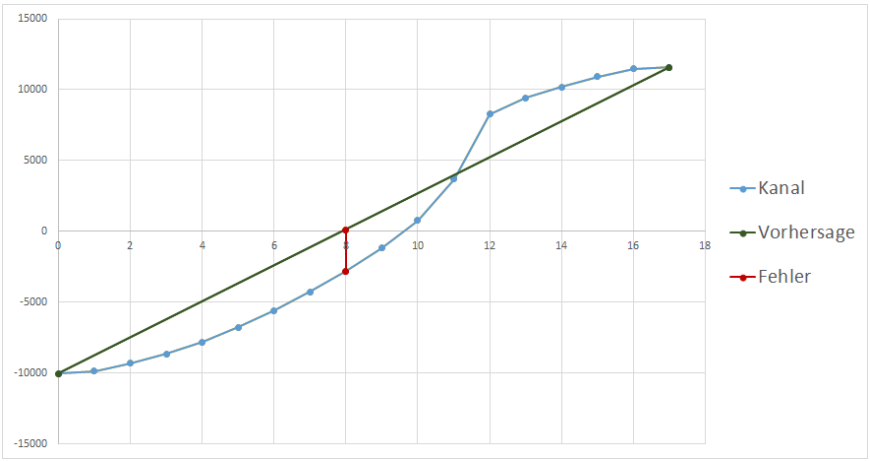

Abbildung 10: Erster Schritt der Rekursiver Linearer Kodierung. Zu sehen sind das zu kodierende Signal, die Vorhersage und den Fehler der Vorhersage.

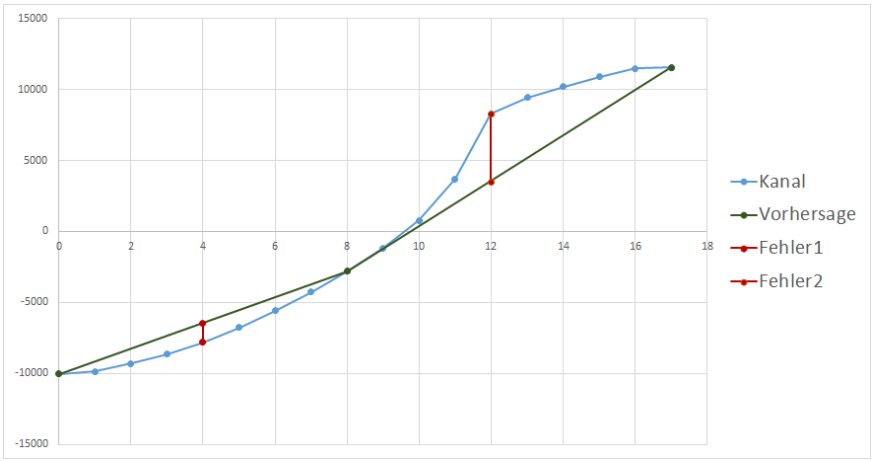

Abbildung 11: Zweiter Schritt der Rekursiver Linearer Kodierung. Die Vorhersage beinhaltet nun zwei Strecken.

3.4.3 Quantisierung

Die Vorhersagefehler Rekursiven Linearen Kodierung werden von Rekursionsstufe zu Rekursionsstufe kleiner. In der letzten Rekursion werden Detailinformationen kodiert, welche für meist nicht relevant sind. Angepasste Quantisierung, welche

Wenn die Durch die Quantisierung entstehen Abweichungen, welche im Extremfall Ringing ähnliche Artefakte verursachen können. Die Abbildung **??** im Abschnitt 6.3.4 zeigt die Artefakte der Kompression.

. Die Lösung ist eine Quantisierung ähnlich Wie die Abweichungen behandelt werden bestimmt die Genauigkeit und die

3.4.4 Entropie Kodierung

Die Vorhersagefehler werden in Breadth-First Ordnung abgespeichert. Der Fehler der Rekursionsstufe 0 wird zuerst abgelegt, gefolgt von den Fehlern der Stufe 1 etc. Die Tabelle 5 zeigt den Aufbau. Die Vorhersagefehler der ersten Levels sind grösser, als die der Letzten. Diese Ordnung hilft ähnliche Strukturen beieinander zu halten.

Vorhersagefehler							
Stufe 0	Stufe 1		Stufe 2				...
$Fehler_0$	$Fehler_1$	$Fehler_2$	$Fehler_3$	$Fehler_4$	$Fehler_5$	$Fehler_6$	...

Tabelle 5: Breadth First Ordnung der Vorhersagefehler.

Die Vorhersagefehler werden mit der Adaptiven Genauigkeitskodierung aus Abschnitt 3.3.5 kodiert. Nach der Quantisierung reichen im Allgemeinen 8 Bit aus um einen Fehler abzulegen. Die Längenkodierung wurde nicht verwendet. Es existiert im Allgemeinen kein zusammenhängender Block von 0-Fehler.

4 Qualitätsmessung der dekomprimierten Daten

Bei verlustbehafteten Kompressionen muss die Qualität der dekomprimierten Daaten sichergestellt werden. Im optimalen Fall ähneln die dekomprimierten Daten ihren Originalen, sodass bei einer Visualisierung das menschliche Auge keinen Unterschied erkennen kann.
Im Verlauf der Arbeit wurden zwei Metriken verwendet: Die Standardabweichung und eine angepasste PSNR-HVS-M. Die Standardabweichung bewertet sichtbare Artefakte wie Oszillationen im dekomprimierten Signal nicht genügend stark. Der absolute Fehler bleibt klein, für das menschliche Auge jedoch sind solche Artefakte störend. Ein Beispiel wie solche Artefakte in sich in der Visualisierung bemerkbar machen ist in der Abbildung 25 im Abschnitt 6.2.6 zu sehen. Wie die Standardabweichung berechnet wird, ist im Abschnitt 4.2 beschrieben. Die PSNR-HVS-M stammt aus der Bildverarbeitung. Das Ziel des Fehlermasses ist es, eine hohe Korrelation zwischen der Metrik und dem menschlichen Augenmass zu erreichen. Wie die PSNR-HVS-M Metrik angepasst und umgesetzt wurde, ist im Abschnitt 4.3 beschrieben.
Für die Messungen wurden spezielle Aufnahmen der Feldlinien gewählt. Wie die Aufnahmen ausgewählt wurden, ist im Abschnitt 4.1 beschrieben.

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, wird bereits eine einfache verlustbehaftete Kompression durchgeführt. Für die Testdaten wurde diese entfernt, was die rohe Datenmenge entsprechend anwachsen liess auf etwa 10 MiBytes pro Aufnahme.

4.2 Berechnung der Standardabweichung

Die dekomprimierte Linie ähnelt dem Original, wenn die Abweichung konstant und klein bleiben. Eine seltene, dafür grosse Abweichung kann das Aussehen massgebend verändern. Für diesen Fall wurde Die Standardabweichung ausgewählt. Die originale und dekomprimierte Feldlinie können unterschiedliche Abtastraten aufweisen. Die Standardabweichung muss deshalb unabhängig von der Abtastrate berechnet werden.
Um die Abweichung zu berechnen wird konzeptionell zwischen den dekomprimierten Punkte eine Linie gezogen und den Abstand zwischen dieser Linie und den Originalpunkte berechnet. Der Vorgang ist dargestellt im Diagramm der Abbildung 12. Für jeden Punkt P'_i aus den dekomprimierten Punkten D, nehme P'_i, P'_{i-1},P'_{i+1} und P'_{i+2}. Ziehe drei Strecken, von $P'_{i-1}P'_i$, $P'_iP'_{i+1}$ und $P'_{i+1}P'_{i+2}$. Suche von P'_i den Originalpunkt P_i aus allen Originalpunkten O, berechne den minimalen Abstand zu den drei Strecken. Führe das für alle folgenden Originalpunkte durch, bis P_{i+1} erreicht wurde.

Die Abstandsberechnung von Strecke s zu einem Punkt P erfolgt in zwei Schritten: Zuerst wird mit der Formel (4.1) überprüft , ob eine Senkrechte durch P auf der Strecke s zu liegen kommt. Falls das der Fall ist, wird der Abstand von P zu s berechnet (4.2). Falls nicht, wird die kürzeste Distanz der Eckpunkte der Strecke zu

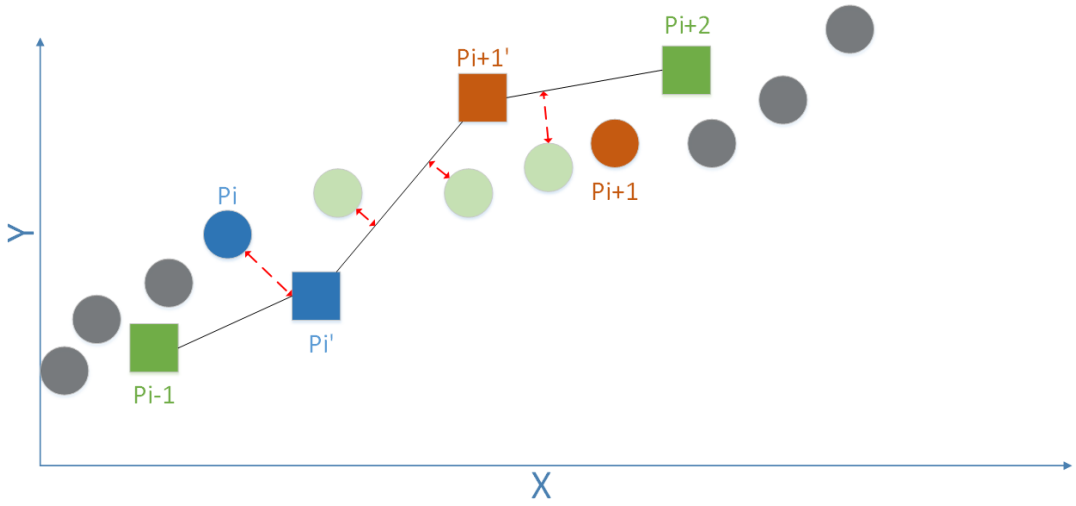

Abbildung 12: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

P berechnet.

$$t = \frac{\vec{AB} * \vec{AP}}{|\vec{AB}|^2} \quad 0 \leq t \leq 1 \tag{4.1}$$

$$distance = \frac{|\vec{BA} \times \vec{BP}|}{|\vec{BP}|} \tag{4.2}$$

A und B sind die Eckpunkte der Strecke. Falls $0 \leq t \leq 1$, existiert eine Senkrechte durch P mit Fusspunkt auf der Strecke s. Die Distanz von P zu s wird mit der Formel (4.2) berechnet.
Wenn das nicht möglich ist, wird der kürzere Distanz von P zu einem der Eckpunkte genommen.

4.2.1 Berechnung der Standardabweichung

$$\begin{aligned} \sigma(X) &= \sqrt{(variance(X))} \\ variance(X) &= \sum (x_i - E(x_i))^2 \end{aligned} \tag{4.3}$$

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 einen Wert für die Standardabweichung.

4.3 Berechnung der angepassten PSNR-HVS-M

Die Peak-Signal-Noise-Ratio (PSNR) Metrik ist ein weitverbreitetes Fehlermass in der Bildverarbeitung. Es wird für die Messung des Fehlers zwischen dekomprimierten Bild und dem Original eingesetzt werden.

$$PSNR = 20 * log_{10}(MAX_I) - 10 * log_{10}(MSE)$$
$$MSE = \frac{1}{n}\sum_{i=0}^{N-1}[E(i) - D(i)]^2 \tag{4.4}$$

Die PSNR (4.4) setzt sich zusammen aus dem maximal möglichen Wert MAX_I und dem "Mean Squared Error´´ (MSE), der durchschnittliche Quadratische Fehler zwischen den Originaldaten $E()$ den dekomprimierten Daten $D()$. Das Problem der Metrik ist, dass sie nicht immer mit der menschlichen Wahrnehmung übereinstimmt. Ponomarenko et al. [15] haben eine modifizierte PSNR entwickelt; die PSNR Human Visual System Masking (HVS-M). In ihren Messungen erreichten sie eine hohe korrelation zwischen menschlicher Wahrnehmung von verrauschten Bildern und dem neuen Fehlermass.

Der Unterschied zwischen der PSNR und der PSNR-HVS-M liegt in der Berechnung des durchschnittlichen quadratischen Fehlers. Das Diagramm der Abbidlung 13 zeigt den Ablauf der neuen Berechnung.
PSNR-HVS-M berechnet die Differenz zwischen dem Originalbild E und dem verrauschtem Bild D und

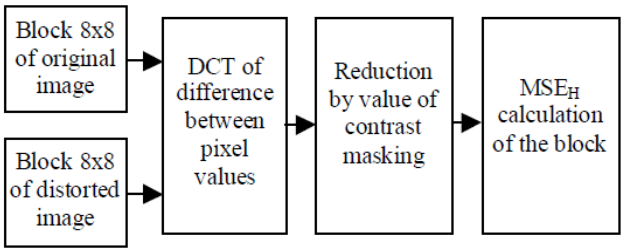

Abbildung 13: Flussdiagramm der PSNR-HVS-M Berechnung [15].

führt die Daten mittels einer DCT in den Frequenzraum. Der nächste Schritt Contrast Maskin reduziert die Differenz, wenn das menschliche Auge den Frequenzunterschied nicht erkennen kann. Die Berechnung des MSE_H Wertes erfolgt wieder gleich wie bei der PSNR.

4.3.1 Contrast Masking

Um das Contrast Masking zu berechnen, führt Ponomarenko et al. die gewichtete Energie der DCT Koeffizienten E_w (4.5)und den masking effect E_m (4.6) ein:

$$E_w(X) = \sum_{i=0}^{7}\sum_{j=0}^{7}[X_{ij}]^2 C_{ij} \tag{4.5}$$

$$E_m(X) = \frac{1}{f_m}E_w(X) \tag{4.6}$$

Wobei X die Kosinus-Koeffizienten eines Bildblocks sind und C die Gewichtungen der Frequenzen. Der Normalisierungsfaktor f_m wurde experimentell ermittelt und auf 16 festgelegt. Ponomarenko et al. argumentiert, dass der Unterschied zwischen einem Block X_e und einem verrauschten Block X_d unsichtbar sind, wenn die Formel (4.7) erfüllt ist.

$$E_w(X_e - X_d) < max[E_m(X_e), E_m(X_d)] \tag{4.7}$$

Das Contrast Masking fliesst mit folgender Formel in die Distanzberechnung mit ein (4.8).

$$X_{\Delta ij} = \begin{cases} X_{eij} - X_{dij}, & i = 0, j = 0 \\ 0, & \left|X_{eij} - X_{dij}\right| \leq E_{norm} / C_{ij} \\ X_{eij} - X_{dij} - E_{norm} / C_{ij}, & X_{eij} - X_{dij} > E_{norm} / C_{ij} \\ X_{eij} - X_{dij} + E_{norm} / C_{ij}, & \text{otherwise} \end{cases} \tag{4.8}$$

Wobei $E_{norm} = \sqrt{max[E_m(X_e), E_m(X_d)]/64}$. X_{eij} ist der DCT Koeffizient des Originalblockes und X_{dij} der Koeffizient des verrauschten Blockes. Dabei ist es irrelevant ob die DCT Koeffizienten subtrahiert werden oder wie in der Abbildung 13 angedeutet, zuerst die Differenz der Pixelwerte berechnet und diese DC transformiert. Aus den Distanzen $X_{\Delta ij}$ wird die PSNR (4.4) berechnet.

Wie gut die PSNR-HVS-M Metrik mit dem menschlichen Auge übereinstimmt, hängt vom Normalisierungsfaktor f_m und von der Wahl der Gewichtungen C ab. Ponomarenko et al. verwendeten die normalisierten ($\frac{10}{x}$) und quadrierten Werte der JPEG/JFIF Quantisierungsmatrix [1]. Es ist zu beachten, dass der DC-Koeffizient nicht im Contrast Masking berücksichtigt wird, für den Wert wird die normale PSNR berechnet. Der DC-Koeffizient stellt die durchschnittliche Helligkeit in einem Block dar. Das menschliche Auge kann auch kleine Unterschiede in dieser Frequenz erkennen.

4.3.2 Umsetzung und Anpassung der PSNR-HVS-M für diese Arbeit

Der grösste Unterschied zur Arbeit von Ponomarenko et. al. ist der Einbezug des DC-Koeffizienten ins Contrast-Masking. Der DC-Koeffizient repräsentiert bei den Feldlinien eine Verschiebung. Das menschliche Auge kann leichte Verschiebungen der Feldlinien kaum unterscheiden. Des weiteren musste für die Feldlinie eine eigene Quantisierungsmatrix gefunden werden, ausdenen die Gewichtungen berechnet werden. Es wurde eine DCT-Kompression entwickelt, welche keine sichtbaren Artefakte aufweist. Die Quantisierungsmatrix wurde übernommen und auf die selbe Art normalisiert.
Für die PSNR wird der maximale Wert MAX_I benötigt. Hier wurde der maximal mögliche Wert der PFSS Simulation verwendet, den vierfachen Sonnenradius.

Der letzte Wert, den es zu setzen gibt, ist der Normalisierungsfaktor f_m. Ponomarenko et. al. schrieb keine zusätzliche Information oder Begründung zu diesem Wert. Dieser Wert stellt ein, wie stark das Contrast-Masking einfliessen soll. Je höher der Wert ist, desto ähnlicher ist die PSNR-HVS-M Metrik der normalen PSNR.
Die Tabelle 6 erforscht den Einfluss des f_m Faktors. Dazu wurde die PSNR-HVS-M zu drei dekomprimierten Simulationen berechnet, welche eine unterschiedliche Qualität aufweisen. Für diesen Anwendungsfall scheint

Qualität	f_m 8	f_m 16	f_m 32
Keine sichtbare Artefakte	96.8 dB	**95.6** dB	94.5 dB
Kaum sichtbare Artefakte	95.8 dB	**94.4** dB	93.3 dB
Sichtbare Artefakte	90.0 dB	**88.3** dB	87,0 dB

Tabelle 6: Einfluss des f_m Faktors auf die PSNR-HVS-M.

der Faktor f_m sich stabil zu verhalten: der Faktor kann verdoppelt oder halbiert werden und die resultierenden Distanzen bleiben in der selben Grössenordnung. Es wurde der Standardwert von 16 übernommen. Eine Anpassung von f_m scheint nicht die Metrik massgebend zu verändern und hat vermutlich weniger Einfluss als die Gewichtung der DCT Koeffizienten.

Die PSNR-HVS-M ist nicht unabhängig von der Abtastrate. Es erwartet, dass die zu vergleichenden Datenmengen gleich viele Punkte enthalten. Für diese Arbeit werden die Originaldaten auf die selbe Punktmenge reduziert. Wenn pro Feldlinie genau ein Punkt exakt abgespeichert wird, würde die PSNR-HVS-M trotzdem eine hohe Ähnlichkeit zwischen Original und dekomprimierten Feldlinien ergeben. Dies ist Vertretbar, da die Standardabweichung diesen Fall abdeckt und eine hohe Distanz ausgeben wird. Die PSNR-HVS-M soll hauptsächlich Artefakte aufdecken, welche in der Standardabweichung nicht ins Gewicht fallen wie die Ringing Artefakte vom Abschnitt 6.2.6.

5 Implementation der Dekompression

Der JHelioviewer lädt eine Folge von komprimierten Simulationen über eine Internetverbindung und muss diese vor der Visualisierung dekomprimieren. Das Herunterladen und die Dekomprimierung sind zeitaufwändige Operationen und kann zu Wartezeiten beim Benutzer führen. Um die Operationen zu beschleunigen werden im Ist-Zustand die Simulationen im Voraus asynchron heruntergeladen. Somit sind die Daten bereits im Arbeitsspeicher, bevor die Daten dekomprimiert und visualisert werden.
Die Dekompression wird direkt vor der Visualisierung durchgeführt. Bei einer Animation der Feldliniendaten führt das zu einer bemerkbaren Verzögerung bei jedem Wechsel. Um die Qualität der Animation zu verbessern und die Wartezeit weiter zu verkürzen wurden folgende Massnahmen umgesetzt:

1. Asynchrone Dekompression.
2. Vorladen der Dekomprimierten Feldlinien.
3. Caching von Komprimierten und Dekomprimierten Feldlinien.

5.1 Software Architektur

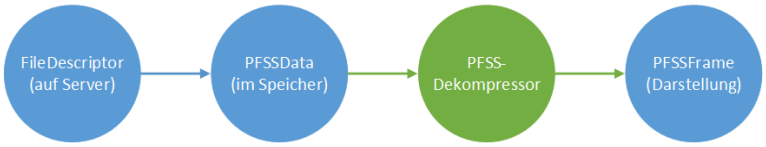

Abbildung 14: Zustandsdiagramm der Feldliniendaten

Die Daten der Feldlinien durchlaufen im JHelioviewer vier Zustände, welche durch vier Klassen abgebildet wurden. Die Klassen sowie die Zustandswechsel sind im Diagramm der Abbildung 14 dargestellt. Die Klasse ”FileDescriptor´´ repräsentiert eine Simulation 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. Die Klasse ”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 Vorladen und Caching von Komprimierten und Dekomprimierten Feldlinien

Um die Animation der Feldlinien möglichst Unterbrechungsfrei zu gestalten, werden die komprimierten und dekomprimierten Feldliniendaten vorgeladen und gecached. Mit dem Vorladen wird erreicht, dass der Wechsel von der Visualisierung einer Simulation zur anderen möglichst ohne Unterbrechung durchgeführt werden kann. Das Caching hilft, wenn der Benutzer einen anderen Zeitpunkt der Animation nochmals darstellen möchte. Aus dem Zustandsdiagramm der Abbildung 14 kann entnommen werden, dass die Komprimierten Feldlinien von den ”PFSSData” Objekten und die dekomprimierten Daten von den ”PFSSFrame” Objekten repräsentiert werden. Die Implementation ist im Diagramm der Abbildung 15 dargestellt. Die Klasse FrameManager ist zuständig für das Vorladen der dekomprimierten Daten, der PFSSFrame Objekte. Das Caching wird in der FrameCache Klasse umgesetzt. Die Klasse DataCache ist für das Vorladen und Caching der komprimierten Daten zuständig. Die FileDescriptor Objekte repräsentieren eine Feldliniensimulation auf dem Server, der FileDescriptorManager ist zuständig für das Auffinden der Simulationen.

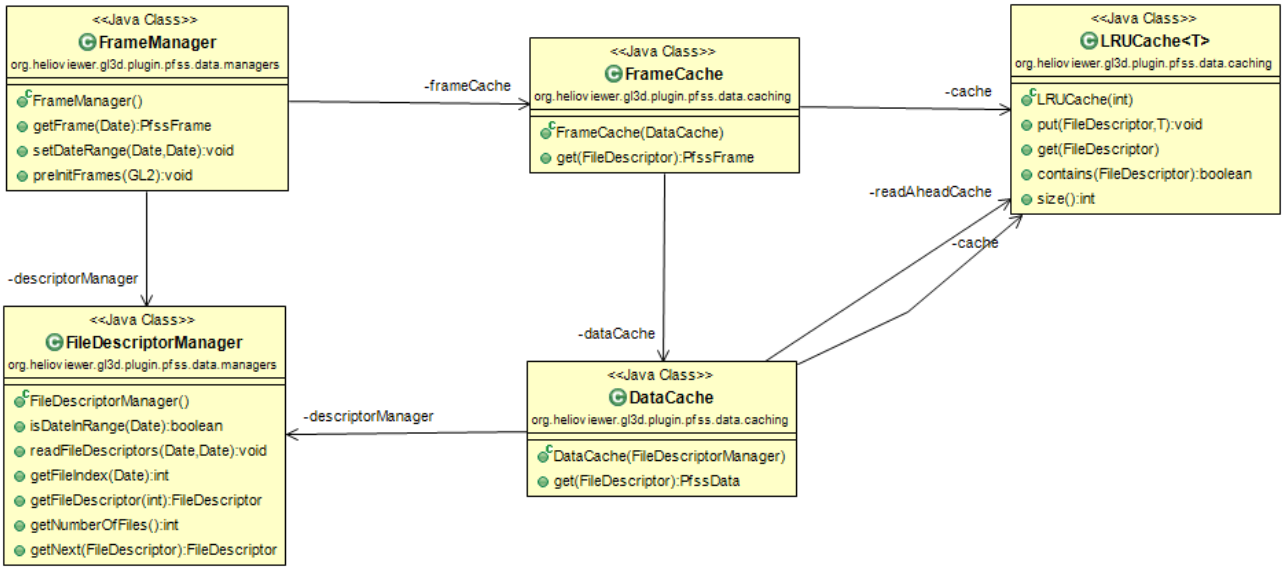

Abbildung 15: Diagramm der Implementation vom Vorladen und Caching

Die Klasse FrameManager repräsentiert die Facade der Vorladens- und Caching- Implementation. Sie abstrahiert das Zusammenspiel der verschiedenen Caches und den verschiedenen Zustände der Feldliniendaten und bietet eine vereinfachte Schnittstelle an.
Die Vorladen und Caching Implementation der PfssFrame Objekte wurde in zwei Klassen aufgeteilt: Die PfssFrame Objekte müssen vor der Visualisierung Resourcen der Grafikkarte allozieren. Nachdem ein Objekt visualisiert wurde, müssen die Resourcen freigegeben werden. Der FrameManager ist zuständig die Allozierung anzustossen und die Freigabe sicherzustellen.
Die PFSSData Objekte können vom Garbage-Collector verwaltet werden. Das Vorladen kann mit einer weiteren Instanz des LRU-Caches umgesetzt werden und wurde direkt in der DataCache Klasse implementiert.

In dieser Arbeit wurde ein Least-Recently-Used (LRU) Cache Algorithmus verwendet. Der LRU-Cache löscht das am längsten nicht verwendete Objekt, wenn der Cache gefüllt ist. Da der JHelioviewer im allgemeinen Fall sequenzell Objekte verlangt, kann der LRU Cache mit einer First-in-First-Out Queue implementiert werden. Das Objet, welches am längsten nicht verwendet wurde, ist das Letzte in der Queue. Ein LRU-Cache funktioniert in diesem Anwendungsfall optimal, wenn die Anzahl Objekte grösser ist, als der Cache. In einem Spezialfall ist der LRU-Algorithmus nicht optimal. Wenn der JHelioviewer zur letzten Simulation der Feldlinien angekommen ist, wird an Wrap-around durchgeführt und wieder die erste Simulation verlangt. Wenn der Cache $n-1$ von n Simulation abspeichern kann, so löscht der LRU-Algorithmus immer die Simulation, welches als übernächstes abgefragt wird. Das führt dazu, dass gleich viele Cache-Misses geschehen, als wenn der Cache wesentlich kleiner währe.

5.1.2 Asynchrone Aufrufe mittels Executor Services

Im Diagramm der Abbildung 15 zu sehen ist, wird das Erstellen von PFSSData und PFSSFrame Objekten jeweils von zwei Klassen übernommen werden, den Creators. Sie sind zuständig für das asynchrone Herunterladen und Dekomprimieren der Feldliniendaten. Die asynchrone Ausführung ist mit dem Java Executor Service umgesetzt. Der Executor Service verwaltet und begrenzt die Anzahl an Threads welche die Aufrufe bearbeiten, sodass auch bei hoher Auslastung ein möglichst hoher Durchsatz erreicht wird. Im Ist-Zustand wurden alle asynchrone Aufrufe jeweils in einem eigenen Thread ausgeführt. Bei hoher Auslastung steigt der Verwaltungsaufwand der Threads und bremst das System.

6 Resultate

In diesem Abschnitt werden die Ergebnisse der Tests vorgestellt. Aufgrund dieser Resultate wurden die Lösungsansätze vom Abschnitt 3 entwickelt. Für die PSNR-HVS-M Metrik existiert nicht immer ein Diagramm. Die Metrik wurde im Laufe der Arbeit entwickelt und wird ab Abschnitt 6.2.7 ebenfalls gemessen.

6.1 Lösungsansatz: Adaptives Subsampling

Im Ist-Zustand führt der JHelioviewer nach der Dekompression ein adaptives Subsampling durch. Dieser Lösungsansatz führt das adaptive Subsampling vor der Datenübertragung durch und Kodiert die Daten mit Rar anstatt mit Gzip. Eine genauere Beschreibung des Ansatzes ist im Abschnitt 3.2 zu finden. Wie im Dia-

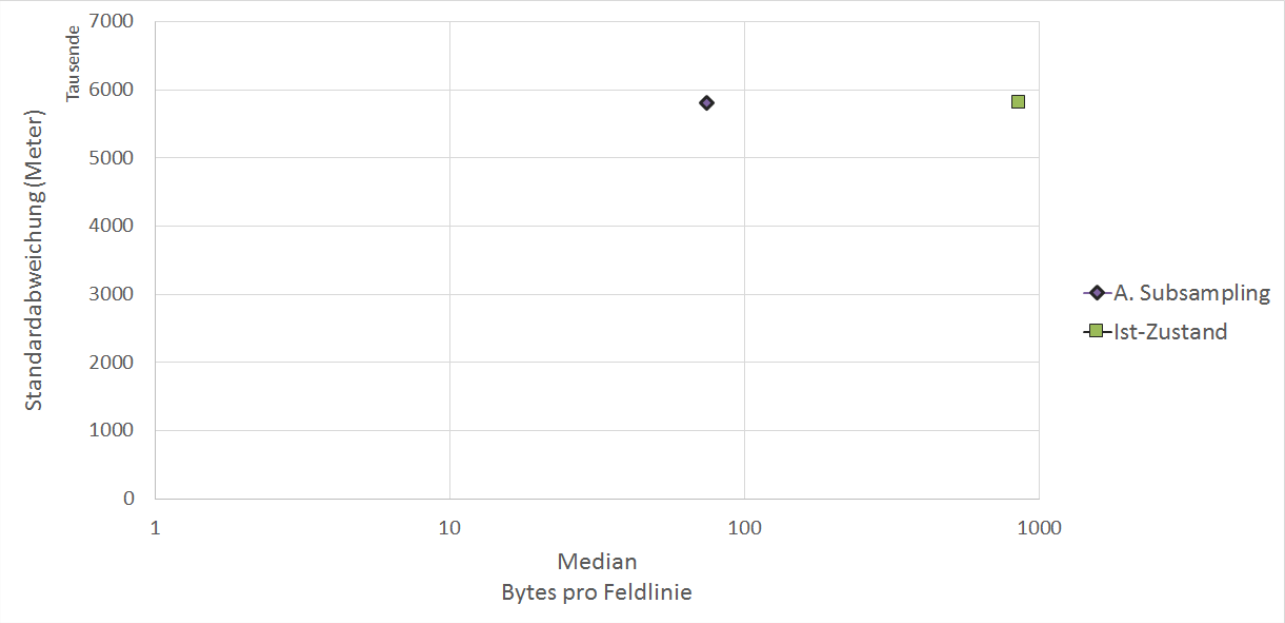

Abbildung 16: Vergleich des Lösungsansatzes: Adaptives Subsampling zur Ist-Kompression.

gramm 16 erkennbar ist, braucht dieser Lösungsansatz deutlich weniger Speicher als die Ist-Kompression zur selben Genauigkeit. Der Lösungsansatz des Adaptiven Subsamplings erreicht eine Kompressionsrate von 11.6 gegenüber dem Ist-Zustand, indem es nur die Punkte überträgt, welche der JHelioviewers darstellt.
Das Diagramm Abbildung 17 zeigt die Artefakte der Kompression. Sie sind identisch mit den Artefakten der Visualisierung im Ist-Zustand und sind erst bei höheren Zoom-Stufen erkennbar.

Der JHelioviewer muss in der Lage sein 1000 komprimierte Simulationen im Arbeitsspeicher abzulegen. Mit dieser Kompression werden durchschnittlich 85 Megabyte an Arbeitsspeicher benötigt. Wenn von einer 10 Megabit Internetverbindung ausgegangen wird, werden für das Herunterladen von 1000 Simulationen 70 Sekunden benötigt. Das ist eine deutliche Verbesserung gegenüber zum Ist-Zustand, welcher unter den selben Bedingungen 790 Sekunden $(13$ Minuten) benötigt. Mit dieser Kompression können etwa 14 komprimierte Simulationen pro Sekunde übertragen werden. Der Benutzer kann die Simulation on-the-fly herunterladen, wenn weniger 14 als Simulationen pro Sekunde benötigt werden.

Ein Vorteil dieses Lösungsansatzes ist die Laufzeit Dekompression: Da keine rechenaufwändige Transformationen verwendet werden braucht dieser Lösungsansatz 19 Millisekunden für eine Dekompression (Siehe Abschnitt 9). Mit einem Thread ist die Testmaschine in der Lage 50 Simulationen pro Sekunde zu dekomprimieren. Anders als der DCT Lösungsansatz wird für die Dekompression keine Parameter zwischengespei-

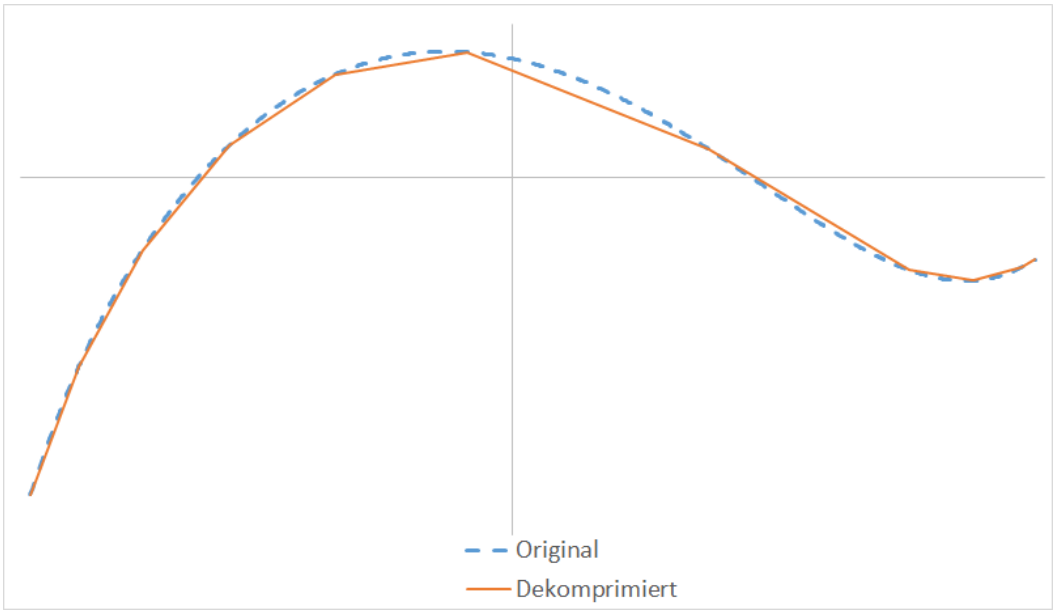

Abbildung 17: Artefakte des Lösungsansatzes Adaptives Subsampling.

chert. Dieser Lösungsansatz verbraucht für die Dekompression am wenigsten Ressourcen, vorausgesetzt dass keine Interpolation benötigt wird.

In Zukunft ist es möglich, dass im JHelioviewer deutlich mehr Punkte dargestellt werden sollen. In diesem Fall müssen entweder mehr Punkte übertragen werden oder der JHelioviewer mit einer Interpolation erweitert werden.

6.2 Lösungansatz: Diskrete Kosinus Transformation

In diesem Abschnitt wird der Lösungsansatz mittels Diskreter Kosinus Transformation behandelt. Es wurden verschiedene Transformationen getestet, welche die Approximation mittels Kosinus Funktionen verbessern. Die Ableitung der Feldlinien dämpft die Kompressionsartefakte und ist massgebend für die Kompressionsrate verantwortlich.
Um die Feldlinien optimal mit der DCT zu approximieren, müssen Ringing Artefakte, behandelt werden. Das Auftreten der Artefakte wird im Abschnitt 6.2.6 und die Behandlung im Abschnitt 6.2.7 besprochen .

6.2.1 Variante: DCT

Diese Variante verwendet die Diskrete Kosinus Transformation. Es wird erforscht welche Kompressionsrate ohne zusätzliche Transformationen möglich ist. Die Abbildung 18 zeigt den Vergleich der DCT Kompression

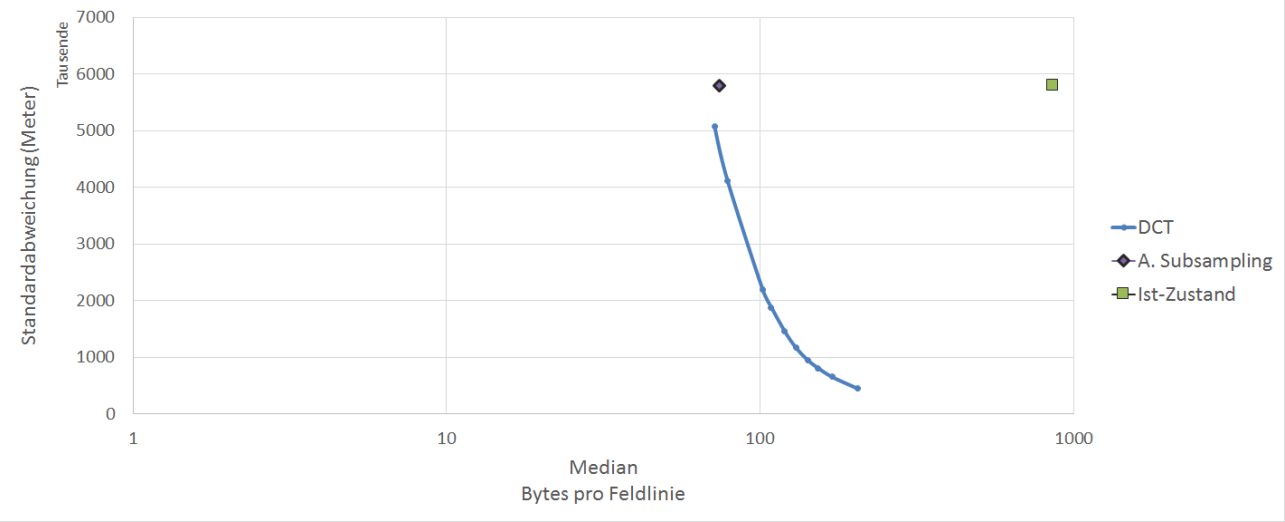

Abbildung 18: Vergleich der DCT Kompression mit dem Lösungsansatz des Adaptiven Subsamplings

mit dem Lösungsansatz des Adaptiven Subsamplings (siehe 6.1). Die Standardabweichung steigt unerwartet schnell an. Der Grund kann im Diagramm der Abbildung 19 entnommen werden. Der Start und Endpunkt der Feldlinie kann nicht richtig dargestellt werden. Dies ist ein typisches Problem der DCT: Ein Inputsignal wird in der DCT konzeptionell wiederholt. Bei der implemementierten Kosinus Transformation (siehe Abschnitt 3.3.3) wird das Signal jeweils in umgekehrter Reihenfolge wiederholt. Bei der Beispielfeldlinie aus Abbildung 19 führt die Wiederholung zu einer Diskontinuität im Signal. Die Diskontinuität wird in der DCT mit hochfrequenten Anteilen abgebildet, welche von der Quantisierung gelöscht werden. Das Ergebnis ist eine Verschiebung an den Stellen, wo die Diskontinuität auftritt. In diesem Fall ist es jeweils am Anfang und am Ende der Feldlinie.
Das Problem kann entweder durch eine andere Darstellung der Feldlinie oder durch zusätzliche Punkte am Anfang und am Ende der Linie gelöst werden. Die Variante, welche die Feldlinie mit Punkten erweitert, wird im Abschnitt refresultate:loesung1:dct:randbeh+byte behandelt. Eine andere Darstellung der Feldlinie wird in den folgenden Abschnitten behandelt.

6.2.2 Variante: Ableitung+DCT

Vor der Diskreten Kosinus Transformation werden die Feldlinien abgeleitet. Durch diese Darstellung werden die Artefakte aus der Abbildung 19 behoben.
Das Diagramm der Abbildung 20 zeigt, dass die abgeleiteten Feldlinien besser approximiert werden können

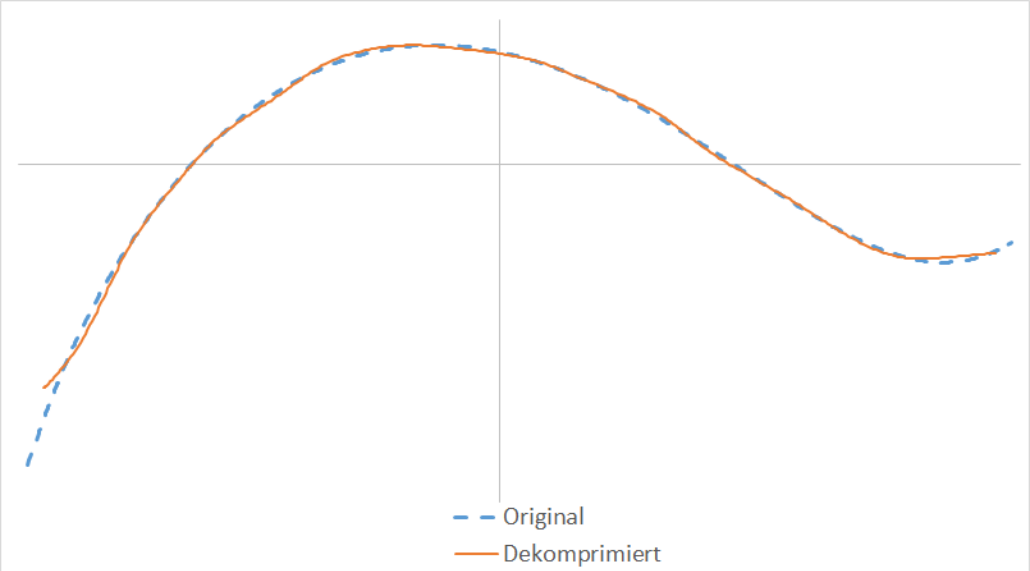

Abbildung 19: Artefakte der DCT Dekompression anhand Beispieldaten

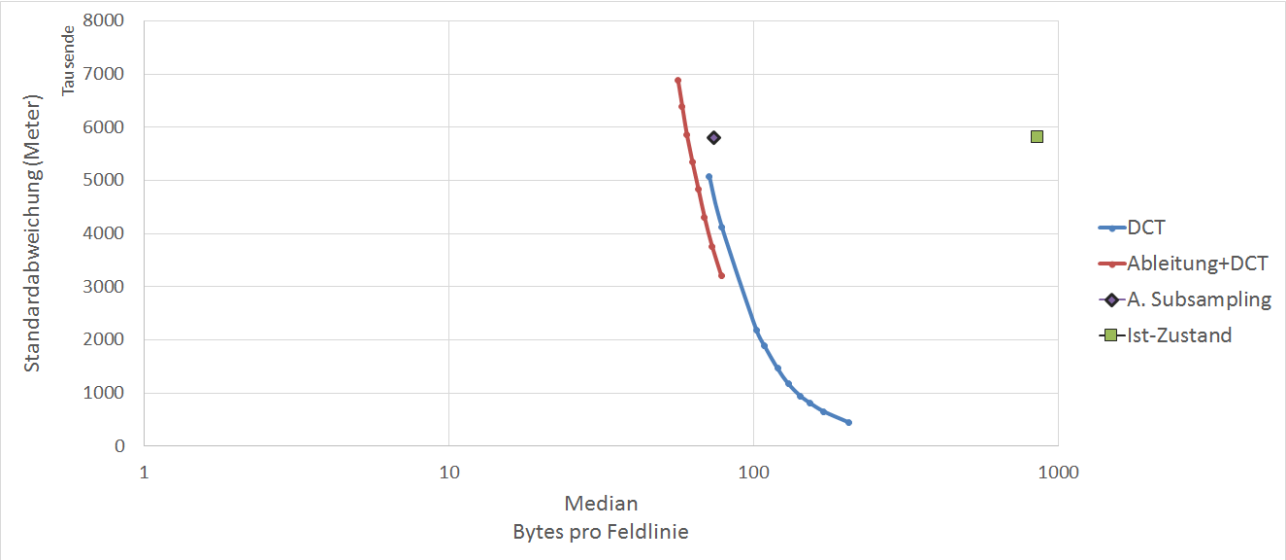

Abbildung 20: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

als die vorhergehende Variante. Bei einer vergleichbaren Genauigkeit wie der Ist-Zustand erreicht diese Variante eine Kompressionsrate von 14.3. Eine Darstellung der Artefakte ist im Diagramm der Abbildung21 zu finden. Die Artefakte der Kompression äussern sich als Dämpfungen. Die Artefakte sind jedoch für das menschliche Auge nicht sichtbar. Die Dämpfung ist erst zu erkennen, wenn das Original zur Verfügung steht.

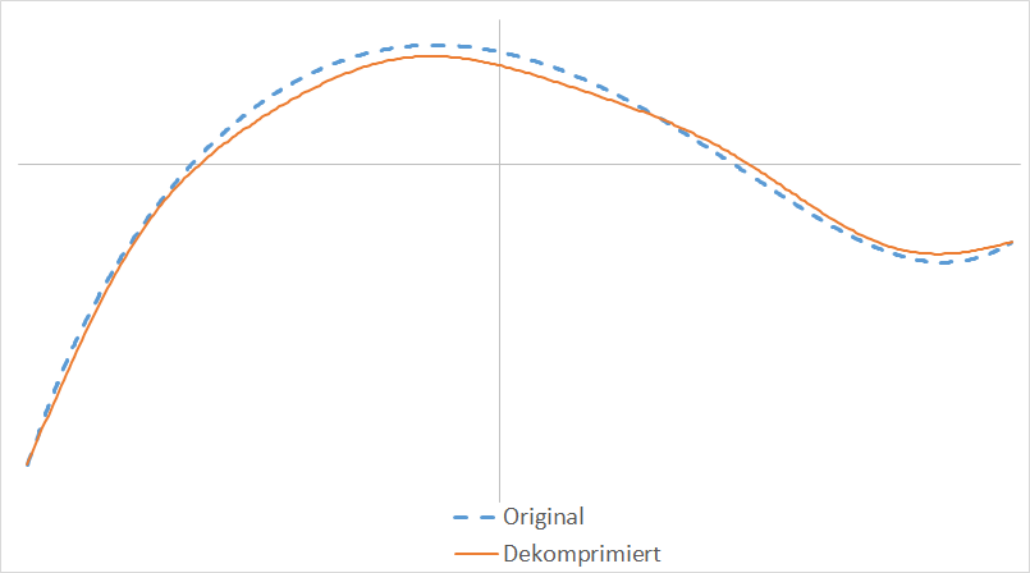

Abbildung 21: Artefakte der DCT Kompression der Ableitung

6.2.3 Variante: PCA+Ableitung+DCT

Die Feldlinien liegen im Allgemeinen in einer Ebene im dreidimensionalen Raum. Durch eine Principal Component Analysis (PCA)[16] können die Feldlinien in ein lokales Koordinatensystem transformiert werden, in welchem der Z Kanal in den meisten Fällen nicht gebraucht wird.
Für die Rücktransformation ins Sonnen-Koordinatensystem werden pro Feldlinie zusätzlich sech Parameter für die neuen Koordinatenachsen und drei Parameter für die Verschiebung abgespeichert. Im Diagramm der

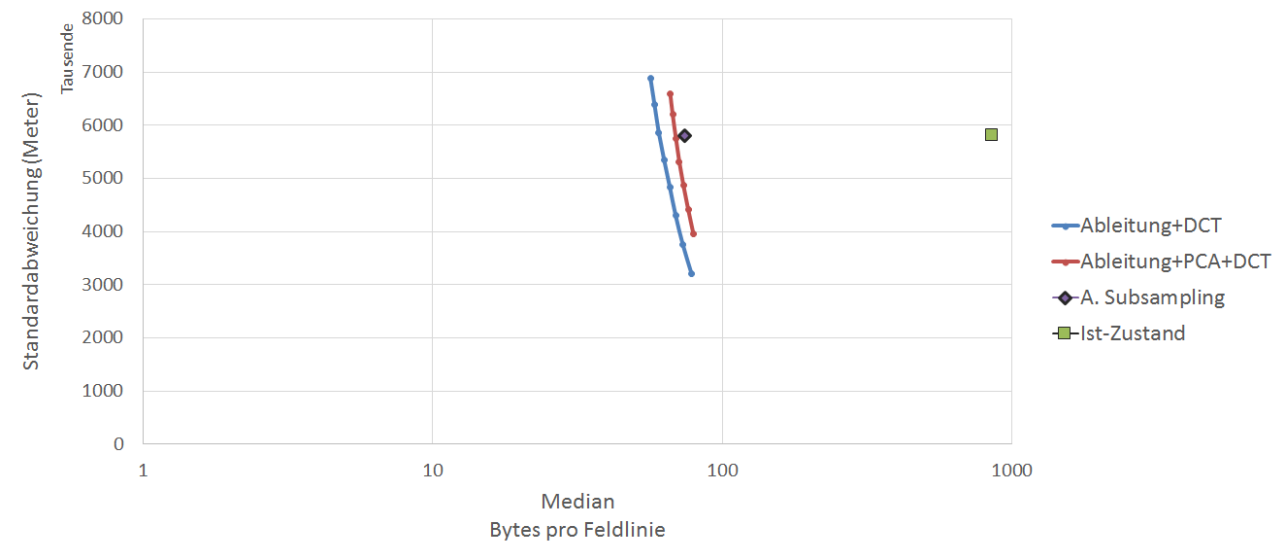

Abbildung 22: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

Abbildung 22 sind die Resultate der Messung dargestellt. Die PCA konnte keine Verbesserung erbringen. Die zusätzlichen Parameter verbrauchen mehr Speicherplatz als durch die PCA gewonnen werden kann.

6.2.4 Variante: Ableitung+DCT+Byte Kodierung

Die quantisierten Koeffizienten können im Allgemeinen mit 8 Bit Genauigkeit dargestellt werden. Nur wenige Ausnahmen benötigen die 16 Big Genauigkeit, mit der sie abgespeichert werden. Der Grossteil der DCT-Koeffizienten wird ebenfalls auf 0 Quantisiert. Mit einer Byte-Kodierung werden diese Eigenschaften ausgenutzt. Die Byte Kodierung ist im Abschnitt 3.3.5 beschrieben.
Das Diagramm der Abbildung 23 zeigt den Einfluss der Byte-Kodierung auf die Kompressionsrate. Es bewirkt

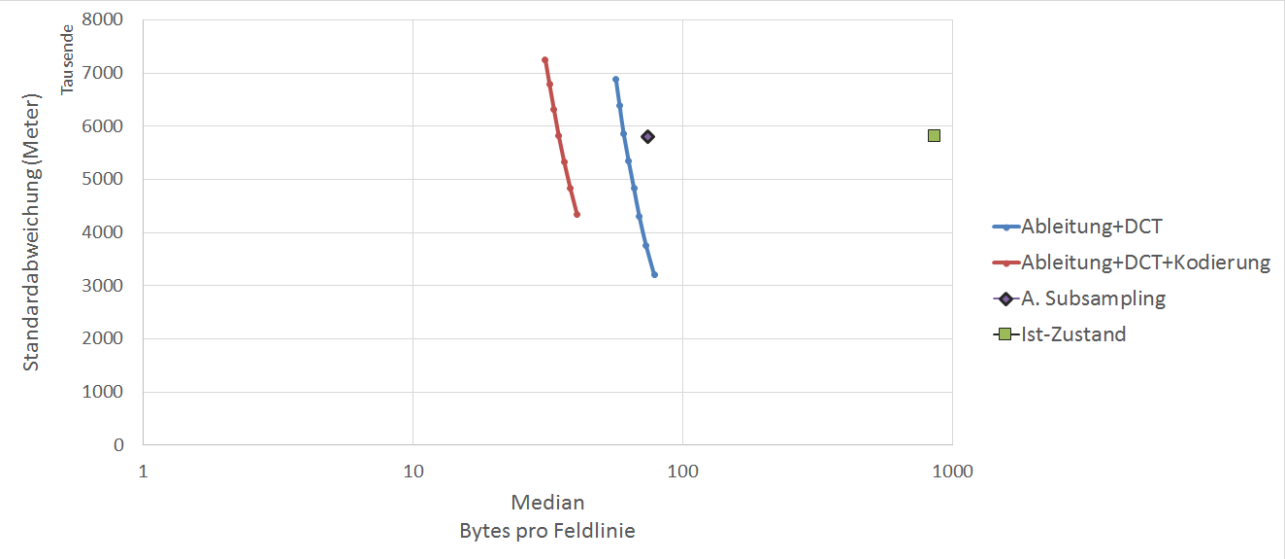

Abbildung 23: Vergleich der Kompression mit und ohne Byte-Kodierung

eine deutliche Verbesserung gegenüber der vorhergehenden Variante, ohne die Qualität der Kompression negativ zu beeinflussen. Bei einer Vergleichbaren Genauigkeit wie die Ist-Lösung weist diese Variante eine Kompressionsrate von 24.5 auf.

6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung

Bei dieser Variante wird mit zusätzlichen Punkten am Anfang und am Ende der Feldlinie die Artefakte aus Abschnitt 6.2.1 behoben. Die Zusätzlichen Punkte lassen die Feldlinie abflachen. Es wird erforscht, ob diese Darstellung der Feldlinien eine bessere Kompressionsrate zur ähnlicher Abweichung erlaubt. Das Diagramm der Abbildung 24 zeigt den Vergleich der Variante mit Randbehandlung und der vorhehenden Variante, welche die Feldlinie ableitet. Die zusätzlichen Punkte erlauben eine höhere Kompressionsrate zu einer ähnlichen Abweichung.
Diese Variante führt Artefakte ein, welche in der Standardabweichung nicht ins Gewicht fallen. Im folgenden Abschnitt 6.2.6 werden diese Artefakte besprochen.

6.2.6 Ringing Artefakte

Obwhol die Variante aus Abschnitt 6.2.5 eine vergleichbare Genauigkeit aufweist, wie die Ist-Lösung, sind auf der JHelioviewer Visualisierung deutliche Artefakte zu sehen. Die Abbildung 25 vergleicht die originalen mit

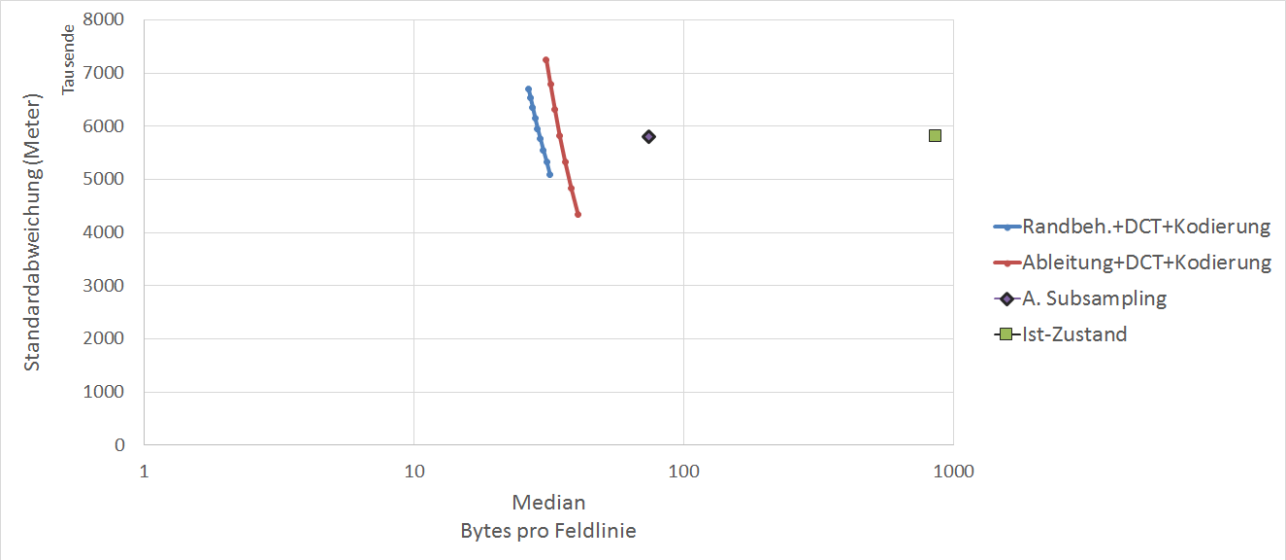

Abbildung 24: Vergleich des Einflusses der Randbehandlung

dekomprimierten Feldlinien. Die Artefakte äussern sich als Oszillationen in den dekomprimierten Feldlinien. In

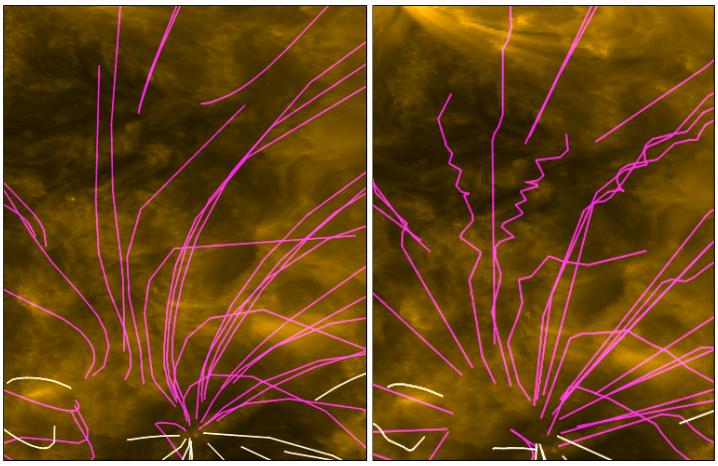

Abbildung 25: Artefakte der Kompression. Links sind die originalen Feldlinien, rechts die Dekomprimierten.

diesem Fall scheint die Standardabweichung als Fehlermass zu versagen: Da die Oszillationen nahe an der originalen Feldlinie liegen, bleiben die Abstände klein. Jedoch sind die Artefakte für das menschliche Auge inakzeptabel.
Interessant ist, dass die abgeleiteten Feldlinien aus Abschnitt 6.2.4 ähnliche Artefakte aufweisen, jedoch sind sie weniger ausgeprägt. Die Ableitung scheint die Artefakte zu dämpfen. Die Abbildung 26 zeigt die Artefakte der abgeleiteten Feldlinien.
Die oszillierenden Artefakte sind typisch für eine Datenkompression mit einer Diskreten Kosinus Transformation als Ringing Artefakte [12] bekannt. Sie treten ebenfalls bei JPEG/JFIF oder MP3 Kompressionen auf: Abrupte Steigungen im Inputsignal werden in der DCT durch hochfrequente Anteile dargestellt. Durch die Quantisierung der hochfrequenten Anteile werden oszillierende Artefakte eingefügt.
Die Feldlinien, welche am stärksten von den Artefakten betroffen sind, sind die ”Weltall zur Sonne” oder ”Sonne ins Weltall” Feldlinien. Sie verhalten sich nicht wie harmonische Halbwellen sondern steigen oft monoton, mit teils abrupten Richtungswechseln in der Nähe der Sonnenoberfläche. Die Abbildung 27 zeigt ein Beispiel solcher Feldlinien. Die abrupten Wechsel führen zu abrupten Steigungen in den einzelnen Kanälen.

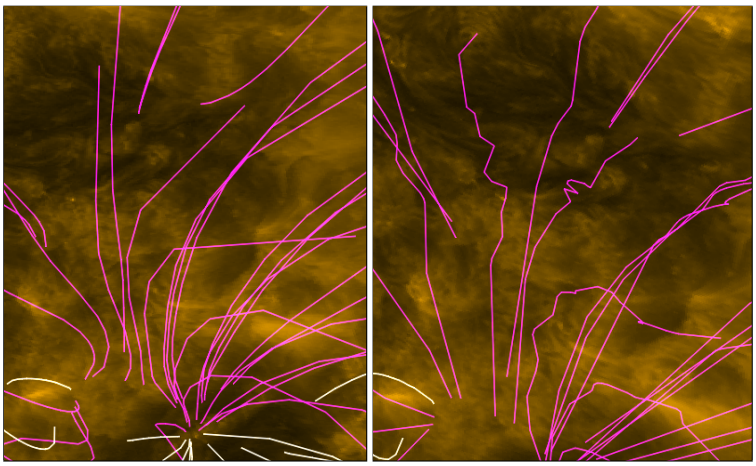

Abbildung 26: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4.

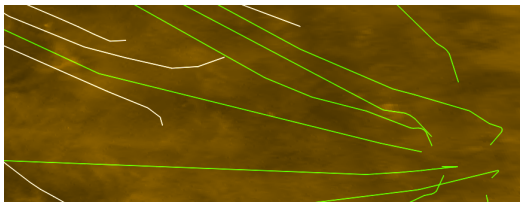

Abbildung 27: Abrupte Steigungen bei Feldlinien, welche von der Sonne ins Weltall führen.

6.2.7 Behandlung der Ringing Artefakte

Um eine optimale Kompression der Feldlinien zu erreichen, müssen die Ringing Artefakte behandelt werden. Abschnitt 6.2.6 wurde erwähnt, dass nicht alle Varianten gleich starke Artefakte verursachen. Es wurde ebenfalls besprochen, dass die Feldlinien, welche von der Sonnenoberfläche zur Oberfläche führen, weniger ausgeprägte Artefakte mit sich bringen. In diesem Abschnitt wird erforscht, welche Variante die ”Sonne zu Sonne” und welche die ”Weltall zur Sonne” oder ”Sonne ins Weltall” Feldlinien optimal approximieren kann, ohne zusätzliche Ringing Artefakte hinzuzufügen. Für die Messung der Artefakte wird die PSNR-HVS-M Metrik aus Abschnitt 4.3 verwendet.
Für den Test werden insgesamt vier Varianten verglichen. Die Variante der abgeleiteten Feldlinien aus Abschnitt 6.2.4 und die Variante der Randbehandlung aus Abschnitt 6.2.5. Die Varianten werden jeweil mit und ohne einer PCA gemessen. Die PCA wird hier nochmals überprüft, da sie die Ringing Artefakte auf einen Kanal beschränken kann.
Das Diagramm der Abbildung 28 zeigt die Resultate der unterschiedlichen Varianten bei den Feldlinien ”Sonnenoberfläche zur Sonnenoberfläche”. Bei einer PSNR-HVS-M von 95 dB sind die Artefakte so schwach ausgeprägt, dass das menschliche Auge sie nicht mehr entdecken kann. Interessant ist, dass drei Varianten ähnlich viel Speicherplatz benötigen, für eine artefaktfreie Approximation. Bei stärkerer Quantisierung treten bei den abgeleiteten Feldlinien deutlich weniger Artefakte auf. Dies deckt sich mit den Beobachtung aus dem Abschnitt 6.2.6.
Ebenfalls interessant ist die Variante der PCA Transformation zusammen mit der Ableitung: Diese benötigt für eine beinahe artefaktfreie Approximation am wenigsten Speicherplatz, führt aber bei stärkerer Quantisierung viele Artefakte ein.

Das Diagramm der Abbildung 29 zeigt, wie gut die Varianten die Feldlinien ”Sonne ins Weltall” und ”Weltall zur Sonne” approximieren können. Bei diesen Typen von Feldlinien ist ebenfalls die Ableitung der Feldlinie Artefaktfreier als die anderen Varianten. Jedoch weniger Deutlich als bei den ”Sonne zur Sonne” Feldlinien.

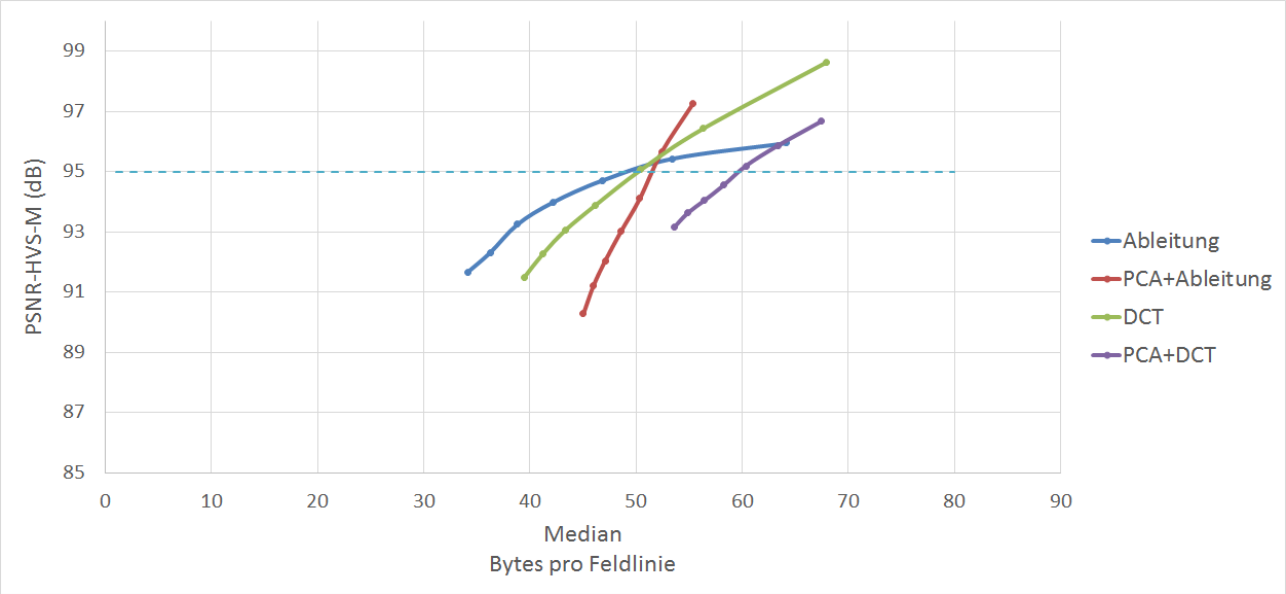

Abbildung 28: Approximation der Feldlinien ”Sonnenoberfläche zu Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation.

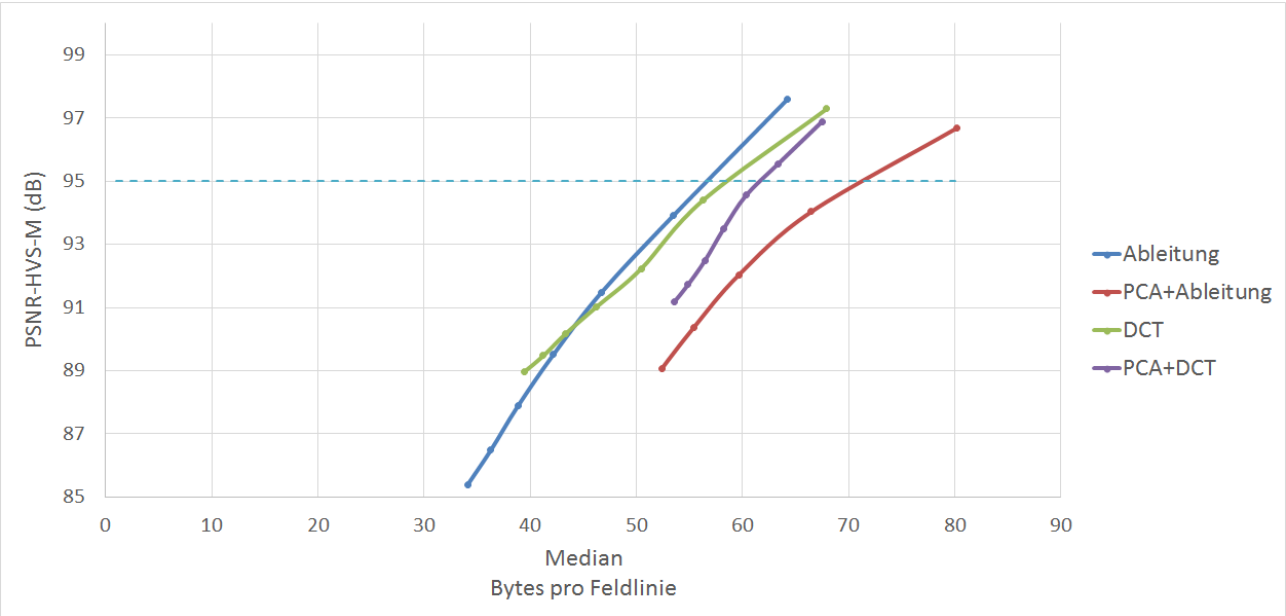

Abbildung 29: Approximation der der Feldlinien ”Sonnenoberfläche ins Weltall” oder ”Weltall zur Sonnenoberfläche”. Je höher die PSNR-HVS-M, desto besser ist die Approximation.

Die PCA konnte auch bei diesen Typen von Feldlinien keinen messbaren Vorteil erbringen. Die zusätzlichen Parameter der PCA verbrauchen mehr Speicherplatz als durch die Transformation gewonnen werden.
Die DCT Variante aus Abschnitt 6.2.5 fällt ab einer PSNR-HVS-M von $90dB$ weniger schnell ab als die abgeleiteten Feldlinien. Bei diesem PSNR-HVS-M Wert sind aber die Artefakte bereits zu deutlich und nicht mehr akzeptabel. Aufgrund dieser Resultate wurde die Variante der abgeleiteten Feldlinien von Abschnitt 6.2.4 ausgewählt.

6.2.8 Abschliessende Variante

Für den abschliessenden Tests wurde die Variante der abgeleiteten Feldlinien aus Abschnitt 6.2.4 ausgewählt. Durch die Ableitung können die Ringing Artefakte gedämpft werden. Die Quantisierung wurde angepasst, sodass die Feldlinien vom Typ ”Sonne zu Sonne” stärker quantisiert werden, als die anderen Feldlinien. Durch diese Massnahme wird eine hohe Kompressionsrate erreicht ohne starke Ringing Artefakte mit sich zu ziehen. Das Diagramm der Abbildung 30 zeigt die Standardabweichung der abschliessenden DCT Variante. Sie

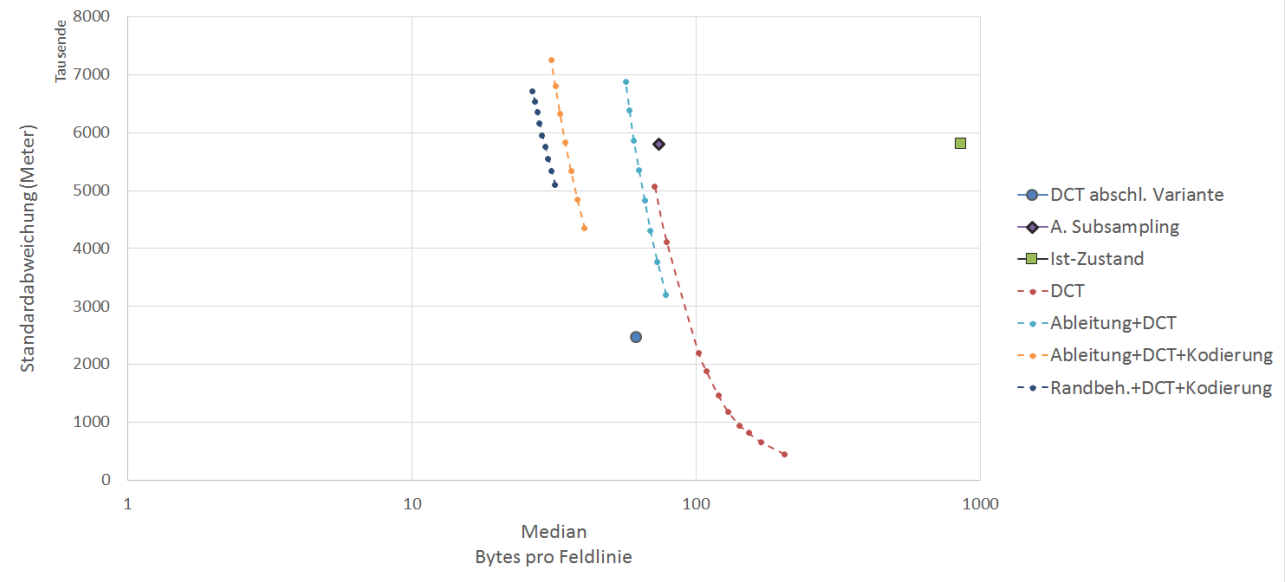

Abbildung 30: Standardabweichung der abschliessenden DCT Variante.

erreicht eine bessere Kompression als der Lösungsansatz des adaptiven Subsamplings und kann die Punkte deutlich genauer approximieren. Es wurde eine durchschnittliche Kompressionsrate 14.1 erreicht, obwohl eine höhere Anzahl an Punkte übertragen wird als beim Lösungsansatz des adaptiven Subsamplings. Die Problematik dieses Ansatzes liegt darin, dass die DCT Kompression Ringing Artefakte hinzufügt.
Durch die unterschiedliche Quantisierungen konnten die Ringing Artefakte in Grenzen gehalten werden und erreichen einen PSNR-HVS-M Wert von 94.0. Die ”Sonne zu Sonne” Feldlinien können deutlich besser approximiert werden als die anderen Feldlinien. Würde die Simulation nur aus diesen Feldlinien bestehen, könnte eine Kompressionsrate von $15 - 18$ erreicht werden zu einem ähnlichen PSNR-HVS-M Wert. Die Feldlinien ”Sonne ins Weltall und ”Weltall zur Sonne” sind schwieriger mit einer DCT Artefaktfrei zu approximieren. Durch das Zoom Feature vom JHelioviewer ist der Benutzer in der Lage die Ringing Artefakte zu finden, sobald sie existieren. Das Linke Bild der Abbildung 31 zeigt die Artefakte der Kompression. Die Artefakte sind aus der Simulation, welche die ausgeprägtesten Ringing Artefakte aufweist. Durch eine Kurvenglättung kann der JHelioviewer die Artefakte verschleiern. Die komprimierten Daten enthalten mehr Punkte, als die JHelioviewer Visualisierung darstellen kann. Die zusätzlichen Punkte können für die Glättung der Feldlinie verwendet werden. Den Effekt der Glättung ist in der Abbildung 31 abgebildet.
Im Abschnitt 3.3 wurde erwähnt, dass reduktion von Ringing Artefakte ein aktives Forschungsfeld der Bildver-

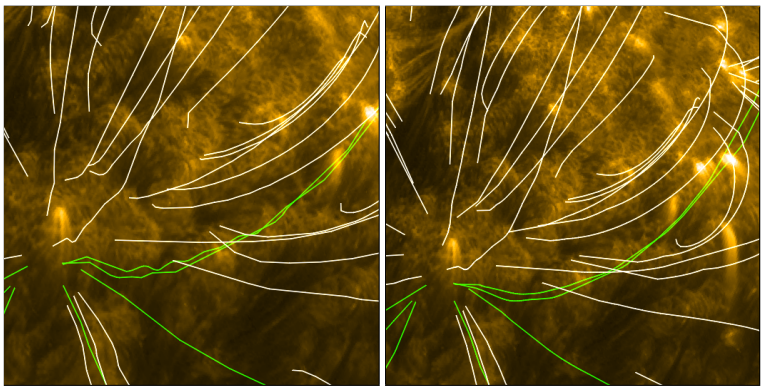

Abbildung 31: Die am stärksten ausgeprägten Artefakte der abschliessenden Variante. Links ohne Glättung, rechts mit Glättung

arbeitung ist. Es existieren Post-Processing Filter, welche Ringing Artefakte vermindern. Eine Möglichkeit die Kompression zu verbessern ist es, einen Post-Processing Filter für wissenschaftliche Daten zu entwickeln. Eine weitere Möglichkeit ist die Diskrete Kosinus Transformation durch eine Wavelet Transformation zu ersetzen, diese ist weniger anfällig auf Ringing Artefakte und hat das Potential eine ähnlich gute Kompression zu erreichen.

Beim Caching von 1000 Simulationen benötigt dieser Ansatz 70 Megabyte Arbeitsspeicher, etwa 15 Megabyte weniger als der Lösungsansatz des Adaptiven Subsamplings. Wenn von der selben 10 Megabit Internenetverbindung ausgegangen wird, werden 56 Sekunden benötigt um 1000 Simulationen herunterzuladen. Pro Sekunde werden 17 anstatt 14 Simulationen übertragen. Die Simulationen können mit dieser Kompression zur gegebenen Internetverbindung on-the-fly heruntergeladen werden.

Die bessere Kompressionsrate kommt auf Kosten der Komplexität der Dekompression. Die Laufzeit der Dekompression hängt im Wesentlichen von der Implementation der inversen DCT ab. Eine naive Implementation braucht für eine Dekompression etwa drei Sekunden. Die grösste Zeit wird in der Berechnung der Kosinus-Werte verbraucht. Bei der Implementation in dieser Arbeit werden die Kosinus-Werte über SoftReferences gecached. Der Cache passt sich somit an den Arbeitsspeicher an. Wie schnell die Dekompression ist, hängt im Wesentlichen vom verfügbaren Arbeitsspeicher ab. Im besten Fall ergibt das eine Laufzeit von 65 Millisekunden und im schlechtesten Fall 350, was zu einem Durchsatz zwischen 3 und 15 Simulationen pro Sekunde pro Thread führt.

6.3 Lösungsansatz: Prediktive Kodierung

Prediktoren sind typische Ansätze für verlustfreie Kompressionsverfahren. Die Frage ist, wie Informationen gelöscht werden. In erster Linie werden zwei Subsampling verfahren getestet, ein einfaches Subsampling und das Adaptive Subsampling. Das Adaptive Subsampling ist gut in unwichtige Informationen zu löschen, ist aber schwieriger zu kodieren. Das

6.3.1 Variante: einfaches Subsampling

Für die erste Variante wird das Subsampling des DCT-Lösungsansatzes verwendet. Es reduziert die Punktmenge auf die Anzahl des Ist-Zustands. Die Feldlinien werden mit einer PCA in ein lokales Koordinatensystem transformiert. Im lokalen System können die Koordinaten mit 16 Bit Genauigkeit dargestellt werden. 16 Bit im Koordinatensystem der Sonne reichen nicht aus und führen zu einem grösseren Fehler als der Ist-Zustand.
In dieser Variante wird der Einfluss von vier Prediktoren getestet (x sind die Bekannten Punkte und y die Vorhersage):

- Konstanter Prediktor: Nimmt an, dass der nächste Wert im Kanal gleich dem letzten Wert ist ($x = y$).
- Linearer Prediktor: Nimmt an, dass die Steigung die Steigung zum nächsten Wert konstant bleibt ($x_1 + (-x_2 + x_1) = y$).
- Linearer Prediktor mit Moving Average: Nimmt die durchschnittliche Steigung der letzten Werte.
- Adaptiver Linearer Prediktor mit Moving Average: Berücksichtigt den Fehler der letzten Vorhersage.

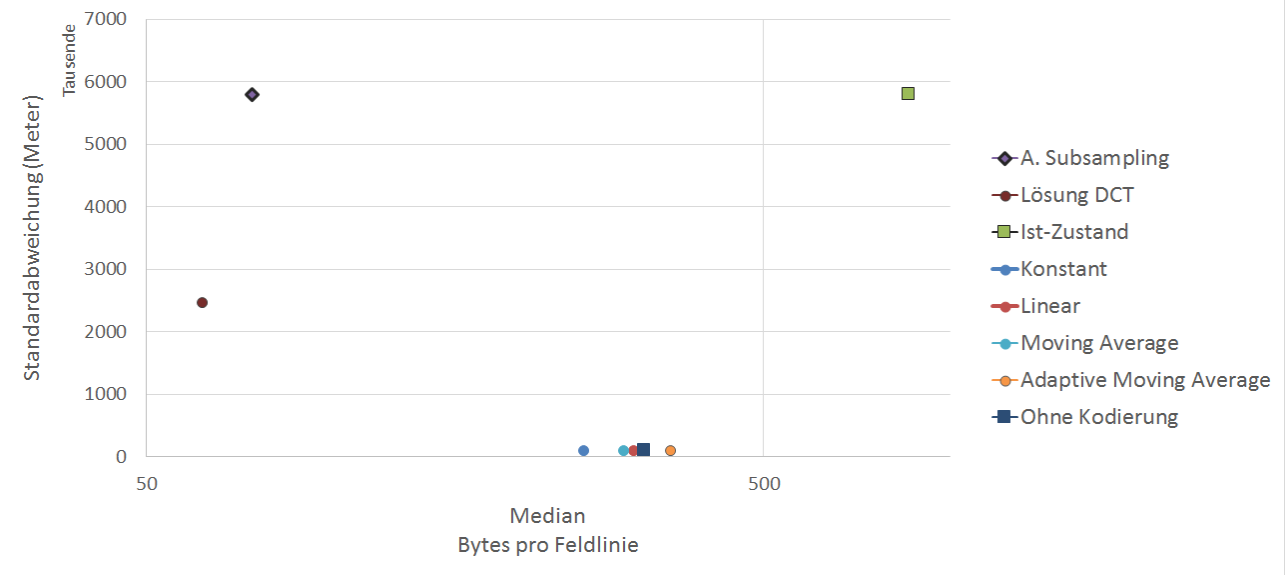

Abbildung 32: Kompressionsraten der vier Prediktoren im Vergleich zum Ist-Zustand.

Im Diagramm der Abbildung 32 sind die Kompressionsraten der jeweiligen Prediktoren dargestellt. Ein Diagramm mit der PSNR-HVS-M wurde nicht erstellt. Sie ist für alle Prediktoren gleich und liegt bei 140.7 dB. Unerwartet ist, dass der Konstante Prediktor mit 255 Bytes pro Feldlinie die beste Kompression erreichte, obwohl die Daten nicht zuverlässig vorhersagen kann. Im Vergleich mit dem Moving Average Prediktor sind die Fehler der Vorhersagen bis zu 5 Mal grösser, verbrauchen aber 40 Bytes weniger um eine Feldlinie abzuspeichern. Der Fehler bleibt jedoch Konstant. Eine Möglichkeit ist, dass die Rar Kodierung sich wiederholende Muster findet.
Eine mögliche Optimierung ist die Adaptive Byte Kodierung der DCT-Variante, beschrieben im Abschnitt 3.3.5.

Das Diagramm der Abbildung 33 zeigt die Resultate mit der Byte Kodierung. Der Konstante Prediktor verbraucht mit der Adaptiven Kodierung mehr Speicherplatz. Die Kompressionsrate der anderen Prediktoren wird durch die Adaptive Kodierung deutlich verbessert. Der Lineare Prediktor erreicht mit 214 Bytes pro Feldlinie die beste Kompression. Das bedeutet, dass die Fehler des Konstanten Prediktors grösser sind, als die der anderen Prediktoren. Es bestätigt die Vermutung, dass die anderen Prediktoren die Daten besser vorhersagen können. Die Kompressionsrate des Konstanten Prediktors ist auf die Rar Kodierung zurückzuführen, welche in den Prediktor-Fehler Muster erkennen und effizient kodieren kann.

Der Linare Prediktor kann eine Feldinie mit etwa 214 Bytes darstellen, was eine Kompressionsrate von 4

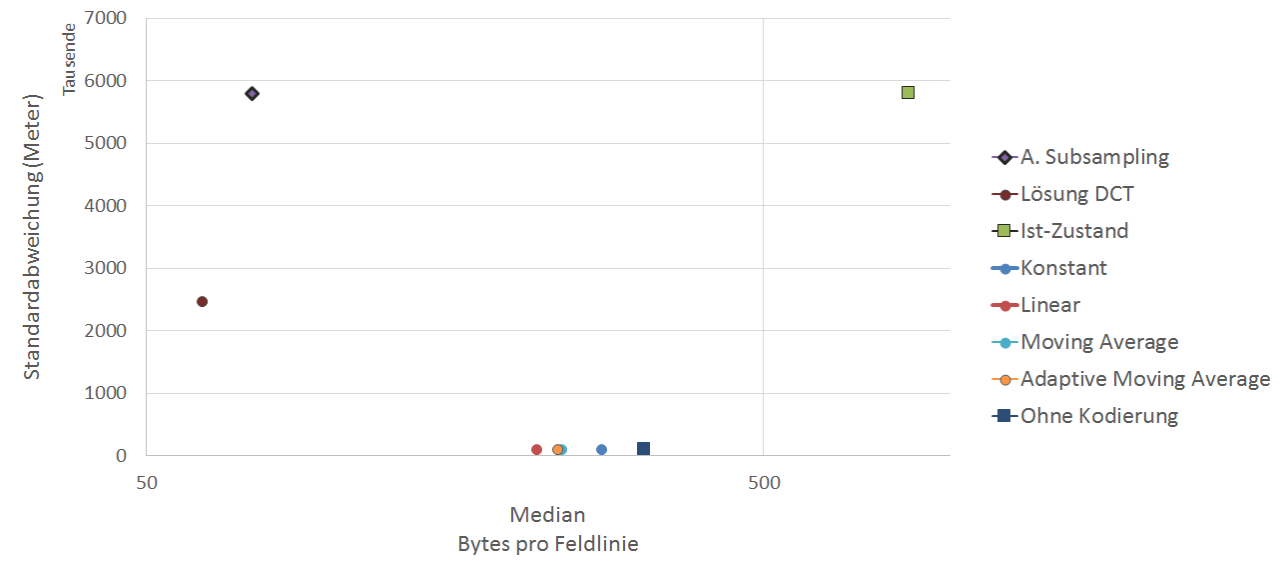

Abbildung 33: Kompressionsraten der Prediktoren mit Adaptiver Byte Kodierung.

ergibt.

6.3.2 Variante: Adaptives Subsampling

Um mit diesem Ansatz in die selbe Grössenordnung zu gelangen wie die Lösungsansätze Adaptives Subsampling und DCT müssen mehr Informationen gelöscht werden. Es werden andere Parameter verwendet und im Schnitt 50% mehr Punkte übertragen als im Lösungsansatz des Adaptiven Subsampling.

Das Diagramm der Abbildung 34 zeigt die Kompressionsraten. Die Resultate liegen dicht beieinander und der Abstand zwischen Kodierung und der Kompression, welche ohne Kodierung erreicht wird wurde drastisch vermindert. Die Kodierungen erreichen eine PSNR-HVS-M von $139,2$. Das Angle Subsampling verändert die Eigenschaften der Daten. Die Diagramme der Abbildung 35 visualisiert die Veränderung. Monotone Steigungen sind nach dem Subsampling nicht mehr vorhanden. Die einfachen Prediktoren können die Daten nicht zuverlässig vorhersagen.

6.3.3 Rekursive Lineare Kodierung

Die Rekursive Lineare Kodierung ist in der Lage nicht-stetigen Kanälen eine sinnvolle Vorhersage zu berechnen. Das Verfahren ist im Abschnitt 3.4 genauer erleutert. Die Punkte werden ins sphärische Koordinatensystem überführt. In diesem Koordinatensystem sind 16 Bit Genauigkeit pro Koordinatenachse ausreichend und die PCA muss nicht berechnet werden. Im Schnitt können dadurch 30 Bytes pro Feldlinie eingespart werden. Zusätzlich werden die Vorhersagefehler quantisiert. Bei den einfachen Prediktoren wurde keine Quantisierung

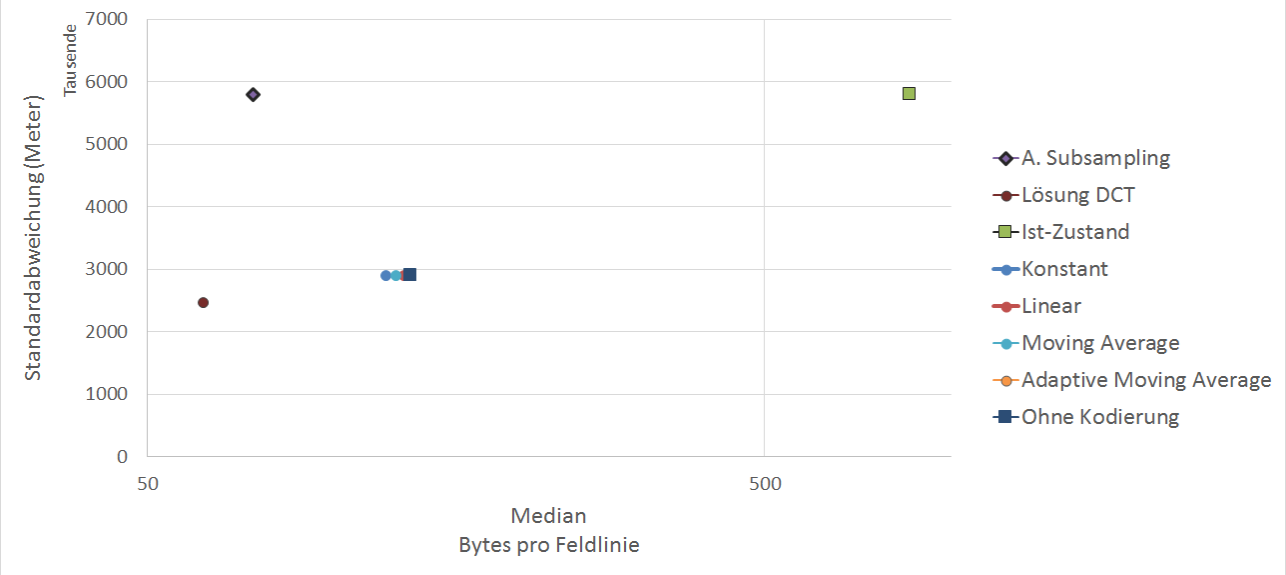

Abbildung 34: Kompressionsraten der Prediktiven Kodierungen mit dem adaptiven Subsampling.

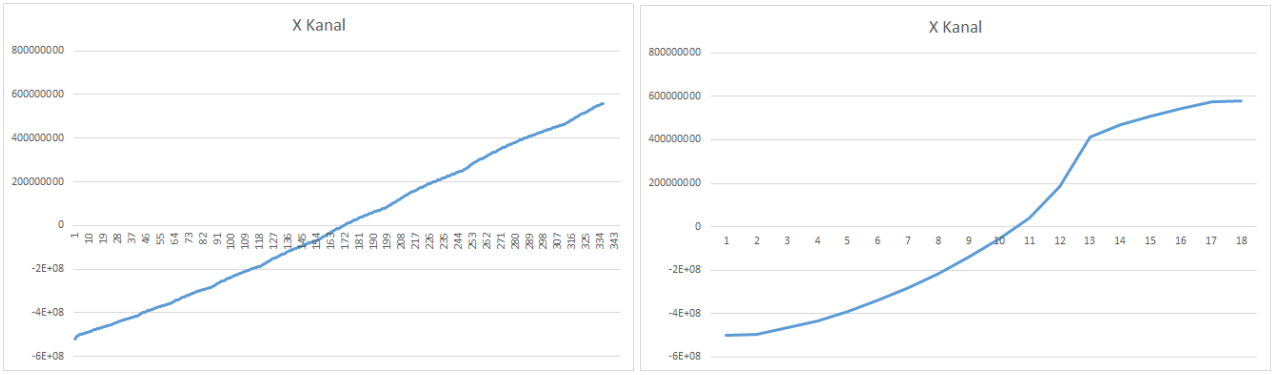

Abbildung 35: Änderung der Eigenschaften der Daten durch das Angle Subsampling. Links der Kanal einer Feldlinie vor, rechts der Kanal nach dem Angle Subsampling.

des Fehlers vorgenommen. Jede Quantisierung führte zu einem markanten Anstieg der Abweichung.
Das Diagramm der Abbildung 36 zeigt die Standardabweichung der Rekursiven Linearen Kodierung zu unter-

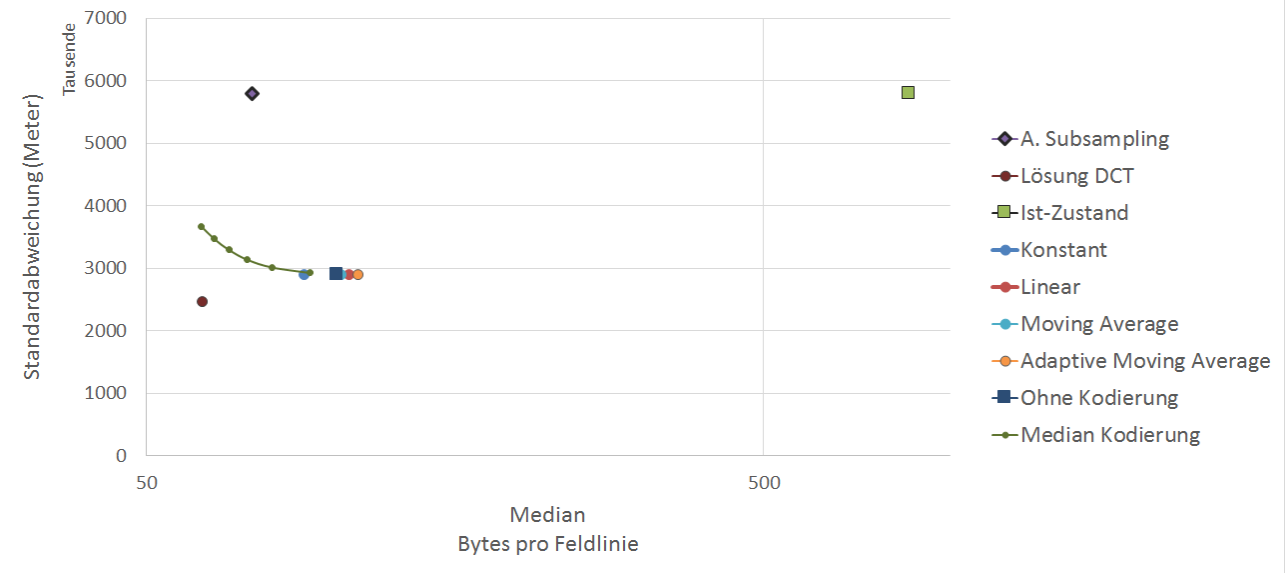

Abbildung 36: Kompressionsraten der Rekursive Lineare Kodierung mit dem adaptiven Subsampling.

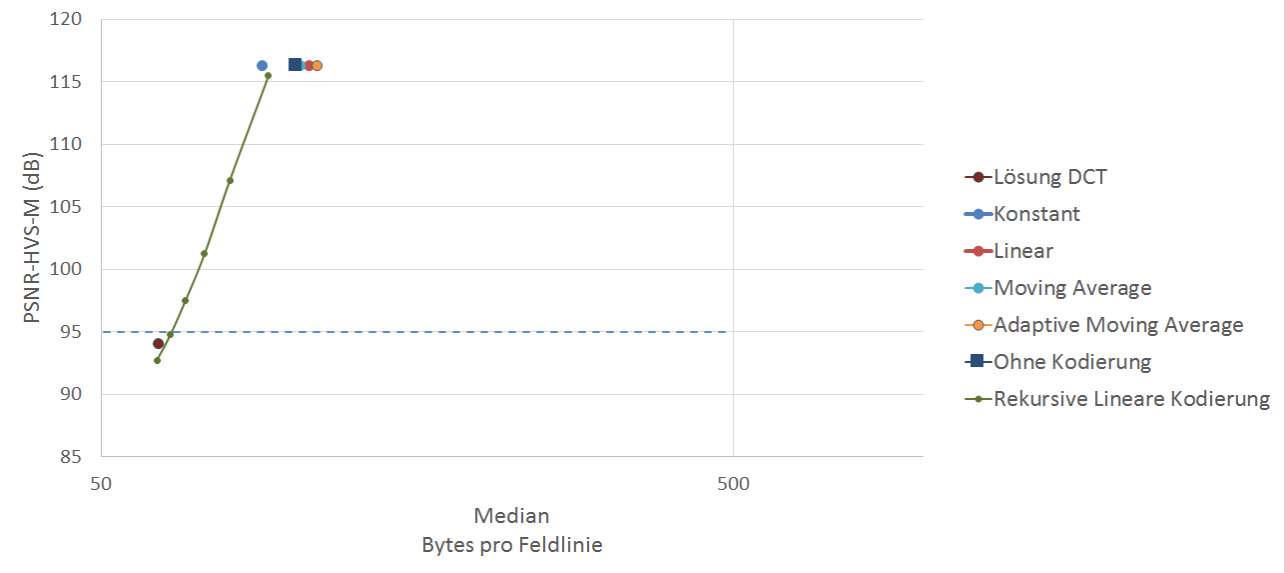

Abbildung 37: Kompressionsraten der Rekursive Lineare Kodierung mit dem adaptiven Subsampling.

schiedlichen Quantisierungen. Das Diagramm der Abbildung **??** zeigt die PSNR-HVS-M Werte der Variante. Die Resultate der einfachen Prediktoren wurden ebenfalls ohne PCA im sphärischen Koordinatensystem gemessen. Die Rekursive Lineare Kodierung kann eine bessere Kompressionsrate erreichen als die Lösung des Adaptiven Subsamplings. Bei Datenmenge von 64 Bytes pro Feldlinie erreicht diese Variante eine tiefere Standardabweichung als das Adaptive Subsampling zu einem PSNR-HVS-M Wert von 94.7. Dieser Ansatz besitzt weniger ausgeprägte Artefakte als die DCT Lösung, weist aber eine höhere Standardabweichung auf. Die Variante erreicht eine Kompressionsrate von 13.4 und fällt somit zwischen den Lösungsansätzen DCT und Adaptives Subsampling.

Die Artefakte der Dekompression äussern sich meist als Verschiebungen einzelner Punkte. Im Extremfall können Ringing ähnlichen Artefakte entstehen, welche in der Abbildung **??** dargestellt sind. Die Artefakte sind weniger stark ausgeprägt als die des DCT Lösungsansatzes. Der Grund für die Artefaktbildung liegt in der Rekursion: Wenn der Vorhersagefehler des ersten Wertes (siehe Diagramm der Abbildung 10) quantisiert wird, beeinflusst der Quantisierungsfehler den gesamten Kanal. Die Werte, welches als letztes Kodiert werden, erfahren die grösste Verschiebung bei der Dekompression. Diese Variante verwendet einen konstanten Faktor für die Quantisierung. Die Lösung ist eine angepasste Quantisierung, welche die ersten Vorhersagefehler mit höherer Genauigkeit abspeichert.

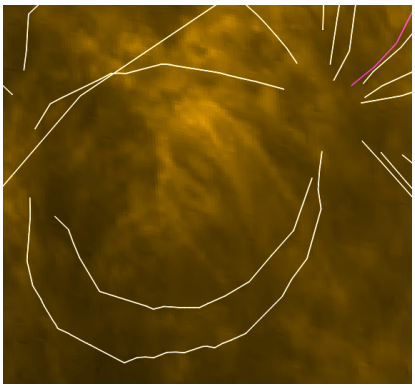

Abbildung 38: Artefakte der Rekursiven Linearen Kodierung.

6.3.4 Rekursive Lineare Kodierung mit angepasster Quantisierung

Um die Artefakte der Dekompression zu dämpfen werden die Vorhersagefehler der Rekursionsstufe angepasst quantisiert. Die ersten Rekursionsstufen werden mit höherer Genauigkeit abgelegt. Figure Durch die angepasste Quantisierung konnte weiter Genauigkeit gewonnen werden und liegt nun fast gleichauf mit der DCT Lösung. Die Artefakte

Artefakte wurden ebenfalls verbessert. Eine leichte Glättung ist aber dennoch willkommen, da sonst alles ein wenig wirr ist. Durch die veränderte Quantisierung konnte weiter an Genauigkeit gewonnen werden. Das Diagramm der Abbildung

zu finden. Je nach Verfahren kann die Abweichung von der Originalfeldlinie oder die Ausprägung der Artefakte minimiert werden. In dieser Variante werden die Artefakte minimiert. Mit einer zusätzlichen Glättung können die letzten Anzeichen von Artefakten versteckt werden und sind vom menschlichen Auge nicht mehr als Artefakte erkennbar.

Das Caching von 1000 Simulationen und die Übertragungszeit fallen zwischen die anderen Lösungsansätzen mit 73 Megabyte an Arbeitsspeicher beziehungsweise 59 Sekunden Übertragungszeit. Zum Vergleich: Die DCT Lösung verbraucht unter den selben Bedingungen 70 und das Adaptive Subsampling 85 Megabyte an Arbeitsspeicher. Bei einer Internetverbindung von 10 Megabit ist bei diesem Lösungsansatz mit einer Übertragungsrate von 16 Simulationen in der Sekunde zu rechnen. Die Laufzeit der Dekompression liegt mit 30.6 Millisekunden ebenfalls zwischen den Lösungsansätzen Adaptives Subsampling und DCT. Mit dieser Laufzeit kann ein Thread durchschnittlich 32 Simulationen pro Sekunde dekomprimieren.

Dieser Lösungsansatz ist ein Kompromiss zwischen Kompressionsrate und Qualität. Die Daten enthalten Kompressionsartefakte. Sie sind aber schwach ausgeprägt sodass auf eine Glättung gegebenenfalls verzichtet werden kann.

7 Diskussion

7.1 Lösungsansatz Adaptives Subsampling

Allgemeinheit der Lösung? Simples Verfahren, kann einfach auf andere Daten benutzt werden Es wird angenommen, dass ein Kanal 16 Bit Genauigkeit verwendet. Solange die Kurven polinomähnlich und niederfrequent sind, wird auch eine gute Kompression erreicht.

7.2 Lösungsansatz Diskrete Kosinus Transformation

Die DCT Kompression kann eine Kompressionsrate von 14.1 erreicht werden. Im Vergleich mit dem Lösungsansatz des adaptiven Subsampling wird eine höhere Kompressionsrate erreicht, obwohl eine höhere Anzahl an Punkte übertragen werden. Die Problematik dieses Ansatzes liegt darin, dass die DCT Kompression Ringing Artefakte hinzufügt (siehe Abschnitt 6.2.6). Die Artefakte können bei allen Kompressionsverfahren mit einer Kosinus Transformation auftreten, wie bei JPEG/JFIF Bilder und MP3 Audiodateien.
Der JHelioviewer bietet die Möglichkeit, an die Feldlinien heranzuzoomen.Sobald Ringing Artefakte existieren ist der Benutzer in der Lage sie zu finden. Bei der Entwicklung der Kompression wurde nach Möglichkeiten gesucht, die Ringing Artefakte zu dämpfen. Die Daten sind Artefaktfrei bei etwa der Hälfte des Zoombereichs. Mit einer Glättung der Feldlinien konnten die Ringing Artefakte für alle Zoomstufen behoben werden. In der Bildverarbeitung wird nach weiteren Post-Processing Methoden geforscht, welche Ringing Artefakte vermindern. Die Qualität oder die Kompressionsrate könnte durch ein angepasstes Post-Processing weiter verbessert werden. Eine andere Möglichkeit ist die Diskrete Kosinus Transformation durch eine Wavelet Transformation zu ersetzen. Die Wavelets sind im Allgemeinen weniger Anfällig auf Ringing Artefakte und haben das Potential eine ähnliche Kompressionrate zu erreichen ohne Ringing Artefakte einzuführen.

Beim Caching von 1000 Simulationen benötigt dieser Ansatz 70 Megabyte Arbeitsspeicher, etwa 15 Megabyte weniger als der Lösungsansatz des adaptiven Subsamplings. Wenn von der selben 10 Megabit Internenetverbindung ausgegangen wird, werden 56 Sekunden benötigt um 1000 Simulationen herunterzuladen. Pro Sekunde werden 17 anstatt 14 Simulationen übertragen. Die Simulationen können mit dieser Kompression zur gegebenen Internetverbindung on-the-fly heruntergeladen werden.

Die bessere Kompressionsrate kommt auf Kosten der Komplexität der Dekompression. Die Laufzeit der Dekompression hängt im Wesentlichen von der Implementation der inversen DCT ab. Eine naive Implementation braucht für eine Dekompression etwa drei Sekunden. Die grösste Zeit wird in der Berechnung der Kosinus-Werte verbraucht. Bei der Implementation in dieser Arbeit werden die Kosinus-Werte über SoftReferences gecached. Der Cache passt sich somit an den Arbeitsspeicher an. Wie schnell die Dekompression ist, hängt im Wesentlichen vom verfügbaren Arbeitsspeicher ab. Im besten Fall ergibt das eine Laufzeit von 65 Millisekunden und im schlechtesten Fall 350, was zu einem Durchsatz zwischen 3 und 15 Simulationen pro Sekunde pro Thread führt.

Dieser Lösungsansatz ist unter mehr Bedingungen auf andere Daten anwendbar, als der Lösungsansatz des Adaptiven Subsamplings. Eigentlich möglich, man muss aber klar die Quantisierung anpassen. Je nach Genauigkeit welche verlangt ist, sind die Byte-Kodierungen nicht Sinnvoll und ersparen keinen Speicherplatz mehr.

7.3 Lösungsansatz Prediktive Kodierung

Mit dem Lösungsansatz der Prediktiven Kodierung wurde eine Kompressionsrate von 12.6 gegenüber dem Ist-Zustand erreicht und liegt somit zwischen dem Lösungsansatz der DCT und des Adaptiven Subsamplings. Die Artefakte der Dekompression äussern sich im Allgemeinen als leichte Verschiebungen einzelner Punkte

und sind erst bei maximalem Zoom als Artefakte erkennbar. In Ausnahmefällen können Ringing-Ähnlichen Artefakte auftreten. Sie sind schwächer ausgeprägt als bei der DCT Kompression und ebenfalls erst bei höheren Zoomstufen erkennbar. Mit einer Glättung können die letzten Anzeichen von Artefakten versteckt werden.

Das Caching von 1000 Simulationen und die Übertragungszeit fallen ebenfalls zwischen die anderen Lösungsansätzen mit 78 Megabyte an Arbeitsspeicher beziehungsweise 62 Sekunden Übertragungszeit. Bei einer Internetverbindung von 10 Megabit ist eine Übertragungsrate von 16 Simulationen in der Sekunde zu rechnen. Die Laufzeit der Dekompression liegt mit 30.6 Millisekunden auch zwischen den Lösungsansätzen Adaptives Subsampling und DCt. Mit dieser Laufzeit kann ein Thread durchschnittlich 32 Simulationen pro Sekunde dekomprimieren.

Dieser Lösungsansatz ist ein Kompromiss zwischen Kompressionsrate und Qualität. Die Daten enthalten Kompressionsartefakte. Sie sind aber schwach ausgeprägt sodass auf eine Glättung gegebenenfalls verzichtet werden kann. Die Kompressionsrate ist höher als die des Adaptiven Subsamplings, obwhol etwa 50% mehr Punkte übertragen werden. Falls die JHelioviewer Visualisierung mehr Punkte darstellen möchte ist es ohne Interpolation oder zusätzliche Punkte möglich. Einfache Implementation.

Mit wenigen Problemen auch auf andere Daten anwendbar. Die Byte Kodierung muss aber hier ebenfalls hinterfragt werden.

8 Fazit

Auswahl des Lösungsansatzes. Kandidatur DCT und Prediktive Kodierung. Es wurde der Lösungsansatz der Prediktiven Kodierung ausgewählt.

Was bedeutet es, wenn Movies mit höherer Kadenz abgespielt werden sollen. –¿ Verzicht auf Artefaktfreien zoom. DCT währe hier optimal. Vorausgesetzt der Computer hat eine genügend hohe Rechenleistung für die Dekompression. Zoom Feature verzicht.

Weitere forschungen

Literatur

[1] Gregory K Wallace. The jpeg still picture compression standard. Consumer Electronics, IEEE Transactions on, 38(1):xviii–xxxiv, 1992.

[2] Siu-Wai Wu and Allen Gersho. Rate-constrained picture-adaptive quantization for jpeg baseline coders. In Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on, volume 5, pages 389–392. IEEE, 1993.

[3] Ching-Yang Wang, Shiuh-Ming Lee, and Long-Wen Chang. Designing jpeg quantization tables based on human visual system. Signal Processing: Image Communication, 16(5):501–506, 2001.

[4] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In SPBG, pages 111–120, 2006.

[5] Wikipedia. Octree — wikipedia, the free encyclopedia, 2014. [Online; accessed 4-January-2015].

[6] Michael Unser, Akram Aldroubi, and Murray Eden. B-spline signal processing. i. theory. Signal Processing, IEEE Transactions on, 41(2):821–833, 1993.

[7] Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. Fast lossless compression of scientific floating-point data. In Data Compression Conference, 2006. DCC 2006. Proceedings, pages 133–142. IEEE, 2006.

[8] Jean-loup Gailly. The gzip home page, November 2014.

[9] Igor Pavlov. 7-zip, November 2014.

[10] Alexander Roshal. rarlab, November 2014.

[11] Wikipedia. Gzip — wikipedia, the free encyclopedia, 2014. [Online; accessed 17-December-2014].

[12] Wikipedia. Ringing artifacts — wikipedia, the free encyclopedia, 2014. [Online; accessed 23-December-2014].

[13] André Kaup. Reduction of ringing noise in transform image coding using simple adaptive filter. Electronics Letters, 34(22):2110–2112, 1998.

[14] HyunWook Park and Yung Lyul Lee. A postprocessing method for reducing quantization effects in low bit-rate moving picture coding. Circuits and Systems for Video Technology, IEEE Transactions on, 9(1):161–171, 1999.

[15] Nikolay Ponomarenko, Flavia Silvestri, Karen Egiazarian, Marco Carli, Jaakko Astola, and Vladimir Lukin. On between-coefficient contrast masking of dct basis functions. In Proceedings of the Third International Workshop on Video Processing and Quality Metrics, volume 4, 2007.

[16] Hervé Abdi and Lynne J Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459, 2010.

[17] NASA. The fits support office, November 2014.

Abbildungsverzeichnis

Tabellenverzeichnis

9 Anhang

Installationsanleitung

Installation des JHelioviewers

Installation der Kompression

Dateiformat und Dekompression

This section describes the encoding and decompression algorithm used for compressing fieldline data. The data is written in the FITS Format [17] and compressed with Rar. The content of the FITS file is described in Table 7.
After decompressing the FITS file with Rar apply the decodings described in section 9. The next step is to

Name	Datatype	Encoding	Content Description
B0	Double	None	Latitude to Earth
L0	Double	None	Longitude to Earth
TYPE	Byte Array	None	Type of quantization used for each Fieldline
LINE_LENGTH	Byte Array	Adaptive Precision Unsigned	Length of each Fieldline
StartPointR	Byte Array	Length + Adaptive Precision	Radius of the Startpoint for each Fieldline
StartPointPhi	Byte Array	Length + Adaptive Precision	Latitude of the Startpoint for each Fieldline
StartPointTheta	Byte Array	Length + Adaptive Precision	Longitude of the Startpoint for each Fieldline
X	Byte Array	Length + Adaptive Precision	X Channel DCT Coefficients of each Fieldline
Y	Byte Array	Length + Adaptive Precision	Y Channel DCT Coefficients of each Fieldline
Z	Byte Array	Length + Adaptive Precision	Z Channel DCT Coefficients of each Fieldline

Tabelle 7: Content of the FITS File

split up the data for each fieldline. Each fieldline contains exactly one Type, one Length and one Startpoint. The length describes the number of DCT Coefficients of the fieldline. The first batch of coefficients belong to the first fieldline, and the second batch to the second line etc.
After splitting the data there are four steps left to do:

1. Transform the Startpoint to Euler coordinates
2. Multiply the DCT Coefficients according to the Type
3. Apply the Inverse Cosine Transform (Formula (3.2) in section 3.3.3)
4. Integrate the channels

How to Transform the Point to Euler coordinates is described in section 9. The multiplication factors can be gathered from the JHelioviewer implementation. Take note that the decompression time will depend on your implementation of the Inverse Cosine Transform, and a naive implementation can take three seconds or more for one FITS file.
The last step ”Integrate the channels” needs further clarification: Each channel is derived before the Discrete Cosine Transform is applied. After the Inverse Transform, each channel contains the rises and not the actual points.

Encodings

Length Encoding

Encoded Channel							
Block of Fieldline 0					...		
n_0	$x_{0,0}$	$x_{0,1}$	...	$x_{0,n-1}$	n_1	$x_{1,0}$	...

Tabelle 8: Beispiel eines abgespeicherten Kanals mit der Längenkodierung.

Unsigned Adaptive Precision Encoding

Byte							
Continue Flag	X	X	X	X	X	X	X

Tabelle 9: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

Adaptive Precision Encoding

Byte							
Continue Flag	Sign Flag	X	X	X	X	X	X

Tabelle 10: Aufteilung eines Bytes der adaptiven Genauigkeitskodierung. X sind Nutz-Bits.

Transformation to cartesian Coordinates

The spherical coordinates can be transformed into the sun's cartesian coordinate system with a few simple calculations. Step 1: Add the displacement of the Compression.

$$\begin{aligned} R_{raw} &= R_{raw} + 8192 \\ Phi_{raw} &= Phi_{raw} + 16384 \\ Theta_{raw} &= Theta_{raw} + 8192 \end{aligned} \tag{9.1}$$

Step 2: Convert to Floating point numbers.

$$\begin{aligned} R &= R_{raw}/8192 * Sunradius(Meter) \\ Phi &= Phi_{raw}/32768.0 * 2 * \pi \\ Theta &= Theta_{raw}/32768.0 * 2 * \pi \end{aligned} \tag{9.2}$$

Step 3: Rotate the coordinates so they are centered around the viewpoint of Earth.

$$\begin{aligned} Phi &= Phi - l0/180 * \pi \\ Theta &= Theta + b0/180 * \pi \end{aligned} \tag{9.3}$$

Step 4: Transform from spherical to cartesian.

$$\begin{aligned} X &= R * sinus(Theta) * sinus(Phi) \\ Y &= R * cosinus(Theta) \\ Z &= R * sinus(Theta) * cosinus(Phi) \end{aligned} \tag{9.4}$$

Performance Tests

Lösunsansatz	Durchschnittliche Dekompressionszeit (ms)
Adaptives Subsampling	19 ms
DCT - Naive Inverse DCT	3100 ms
DCT - Mit Kosinus Caching	$65 - 350$ ms
Prediktive Kodierung	31 ms

Testmaschine

Prozessor	Intel i7-4600
Arbeitsspeicher	16GB
OS	Windows 8.1 Enterprise 64bit

10 Ehrlichkeitserklärung