Abstract

Inhaltsverzeichnis

1 Verlustbehaftete Kompression von volumetrischen Punkten

Wissenschaftliche Simulationen produzieren grosse Mengen an Daten. Es werden Schwingungen, Flugbahnen, Flüsse, Kraftfelder etc. als Linien im dreidimensionalen Raum visualisieren. Die Linien können als Folge von volumetrischen Punkten dargestellt werden. Die Datenmenge ist oft zu gross als dass sie in sinnvoller Zeit über eine Internetverbindung übertragen werden können. Deshalb soll im rahmen dieser Arbeit eine verlustbehaftete Kompression entwickelt werden, welche die Datenübertragung ermöglicht.

Im Rahmen dieses Projekts müssen Daten von Magnetfeldlinien der Sonne 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.
Es wurde eine Aufnahme der Sonnenoberfläche zusammen mit drei Arten von Feldlinien visualisiert: Linien,

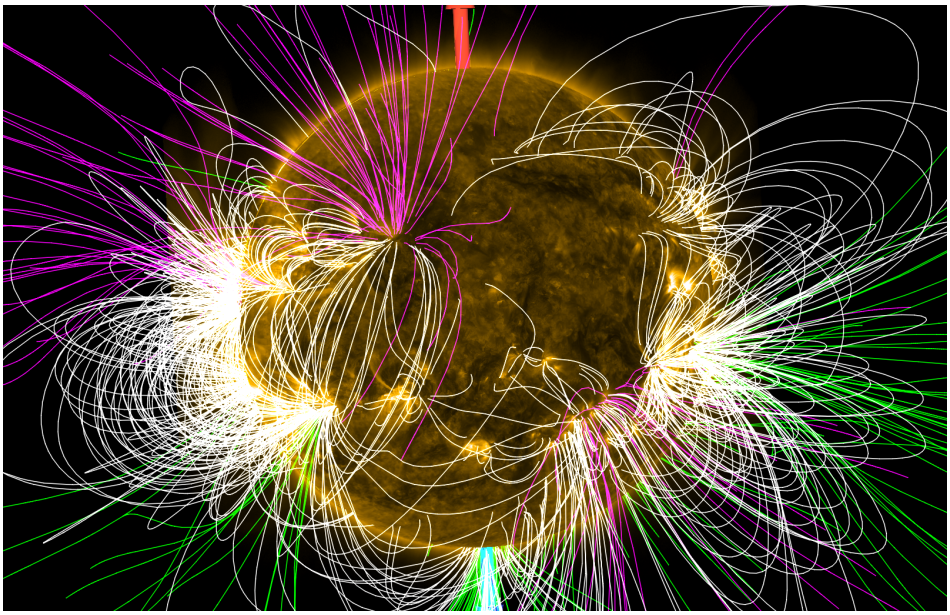

Abbildung 1: Visualisierung der Feldlinien im JHelioviewer

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 zu Weltall´´ und die Violetten ”Weltall zu Sonne´´.

Der JHelioviewer visualisiert mehrere Messungen oder Simulationen in Abfolge. Ziel ist es, möglichst schnell die Mess- und Simulationsdaten abzuspielen, sodass der Benutzer eine flüssige Animation erhält. Pro Simulation werden etwa 1.5 MiByte and Feldliniendaten generiert. Bei einer Visualisierung muss der JHelioviewer die Feldlinien- und andere Daten zur Laufzeit herunterladen. Internetverbindung wird stark beansprucht und Benutzer muss auf Frames warten.

Potential Field Source Surface (PFSS) Simulation, welche aus Messungen der Sonnenoberfläche die Magnetfeldlinien extrapoliert, produziert pro Aufnahme etwa 1.5 Mibyte an Daten. welche zur Laufzeit über eine Internetverbindung heruntergeladen wird. Mit einer verlustbehafteten Kompression soll die zu übertragende Datenmenge deutlich verkleinert werden.

Resultate

Wo was beschrieben wird.

2 State of the Art

Grob betrachtet beinhalten verlustbehaftete Kompressionen drei Teilschritte: Transformation, Quantisierung und Entropie Kodierung. Die Abbildung 2 zeigt eine vereinfachte Abfolge. Die Inputdaten werden zuerst duch ein oder mehrere Verfahren transformiert. So transformieren, dass in der Quantisierung unwichtige Informationen entfernt werden können. Oft gehen nur im Quantisierungsschritt Daten verlohren, während alle anderen Schritte verlustfrei umkehrbar sind. Danach werden die Daten Entropie Kodiert. Für jeden Teilschritt gibt es unterschiedliche Verfahren. So können auch mehrere Transformationen hintereinander durchgeführt werden, oder eine ganze Folge von Transformations- und Quantisierungsverfahren.

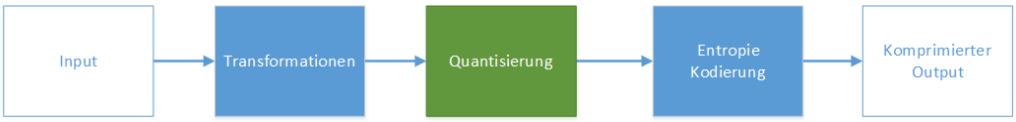

Abbildung 2: Typische Teilschritte einer Kompression

2.1 JPEG/JFIF Bildkompression

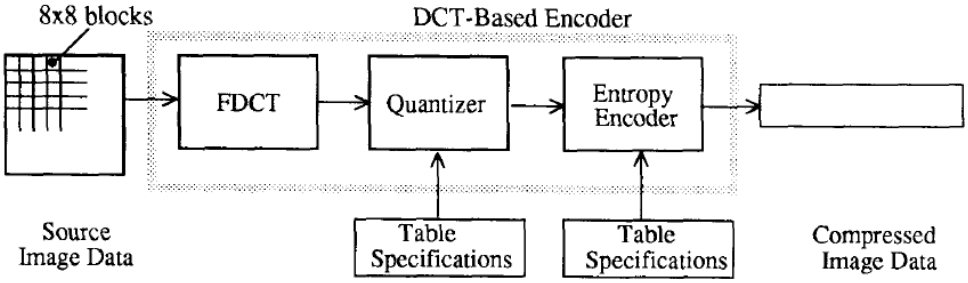

Abbildung 3: Aufbau der JPEG Kompression [**?**]

Das JPEG/JFIF Bildformat ist eines der meist-verwendeten Bildkompressionsverfahren für natürliche Bilder. Das Diagramm der Abbildung 3 zeigt den Aufbau der Kompressionspipeline. JPEG/JFIF unterteilt das Eingabebild in $8x8$ Blöcke und führt auf ihnen 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 mit 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 [**?**] [**?**] 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 zuerst eine Run-Length Kodierung durch und auf dem Rest dann eine Huffman-Kodierung. Auch hier bietet JPEG die Möglichkeit eine eigene Huffman-Tabelle abzuspeichern Hier bietet JPEG die Möglichkeit, eine eigene Huffman-Tabelle zu spezifizieren.

2.2 PointCloud Kompression

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

2.3 Curve Fitting

In der Signal Prozessierung Interpolation oder Rauschunterdrückung werden Signale mit Basisfunktionen. [**?**] Schnelle B-Spline Signal interpolation und Approximation möglich. Wird aber in der Signal Processing weniger verwendet. Da kontinuierlich, können Operationen wie Integration und Diffenezierung direkt auf der Spline-Funktion durchgeführt werden. Noise reduktion möglich, aber auch Approximation. Verschiedene Ansätze die Knotenpunkte zu wählen (de boor)

2.4 Signal Approximation

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

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

fourier, wavelet Compressed sensing

2.5 Entropie Kodierung

Allgemeine Verfahren, Huffman, Arithmetic, LZM, RLE. Fix fertige Pakete wie Gzip, Rar etc.

Spezialisierte Entropie Kodierun:Fast Loessless floating point compression [**?**]

3 Kompressionsverfahren der Feldlinien

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

3.1 Ist-Komprimierung

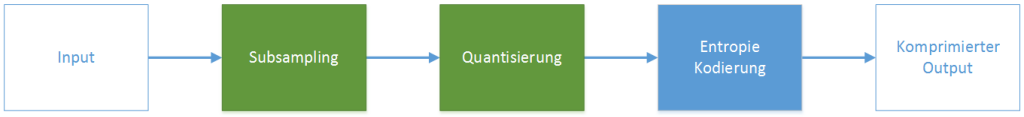

Abbildung 4: Aufbau der Ist-Kompression.

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. Alle X-Kanäle der Feldlinien werden hintereinander abgelegt, gefolgt von allen Y-Kanälen etc. 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 [**?**].
Da die Punktmenge für Low-End Grafikkarten zu gross ist, führt der JHelioviewer ein weiteres Subsampling durch, welches im Abschnitt 3.2.1 beschrieben ist.

3.2 Lösungsansatz: Adaptives Subsampling

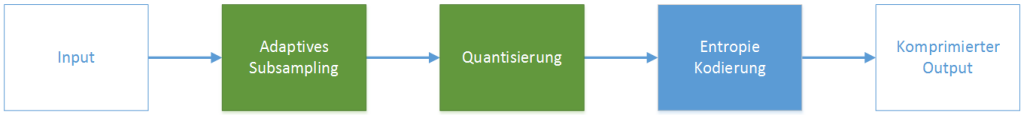

Abbildung 5: Aufbau des Lösungsansatzes: Adaptives Subsampling.

Dieser Lösungsansatz verwendet die selbe Pipeline wie die Ist-Kompression 3.1. Der Unterschied ist, dass ein anderes Subsampling Verfahren gewählt wurde und eine andere Entropie-Kodierung. Die Abbildung 5 zeigt den neuen Ablauf.

3.2.1 Adaptives Subsampling

Ziel des adaptiven Subsamplings ist es, die Daten durch eine Folge von Strecken zu approximieren. An stellen, welche die Feldlinie gekrümmt ist, braucht es mehr Strecken. An Stellen, welche die Feldlinie Linear verlauft können so viele Punkte gespart werden. Das Diagramm der Abbildung 6 stellt das Subsampling im zweidimensionalen Raum dar. Zu sehen sind die Punkte der Feldlinie. Das Adaptive Subsampling wählt nun Punkte P aus der Feldlinie aus, welche Start- und Endpunkte der Strecken darstellen.
P_1 wurde bereits ausgewählt. Es wird nun ein Punkt P_x gesucht, der als Endpunkt einer Strecke von P_1 zu P_x die Feldlinie approximiert. Dazu wird der Winkel der Strecke P_1 zu P_x mit der Strecke P_x zu $P_x + 1$ verglichen. Wenn der Winkel kleiner ist, als ein Faktor F, wird der nächste Punkt $P_x + 1$ überprüft. Wenn der Winkel grösser ist, wird P_x ausgewählt und von P_x aus weiter geprüft.

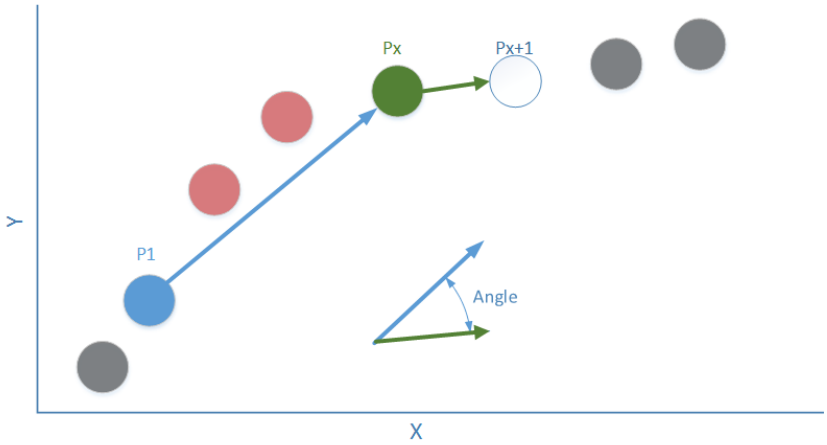

Abbildung 6: Darstellung des Adaptiven Subsapmlings im 2D Raum.

3.2.2 Entropie Kodierung mittels RAR

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

3.3 Lösung 1, Diskrete Kosinus Transformation

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

Es ist auch möglich die Punkte im sphärischen Koordinatensystem in den Frequenzraum zu überführen. Der ϕ-Kanal ist jedoch schwierig durch tiefe Kosinus Schwingungen darzustellen: Wie im Abschnitt 3.1 besprochen, beinhaltet der Kanal Sprünge bei der Passierung des Nullpunktes. Das führt zu sehr hochfrequenten Schwingungsanteile in der DCT. Nach einer Quantisierung sind dabei Artefakte nicht vermeidbar. Im kartesischen System hingegen sind alle Kanäle stetig und lassen sich einfacher durch Kosinus-Funktionen approximieren.

3.3.1 Subsampling

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 schneller.
Falls die Laufzeit der Dekompression weiter verbessert werden soll, kann die Fast-Cosine-Transformation umgesetzt werden. Diese hat eine Komplexität von $O(nlog(n))$. Der Nachteil ist, dass nur Daten der Länge 2^n transformiert werden können, was zusätzliche Programmlogik braucht. Falls die Fast-Cosine-Transformation nicht ausreicht, können die Linien in Blöcken mit einer bestimmten Anzahl von Punkten unterteilt werden. Dadurch wird die Komplexität auf $O(n)$ gesenkt. Jedoch ist es wahrscheinlich, dass durch die Unterteilung die Kompressionsrate leidet. Vermutlich braucht es für die Approximation der Blöcke insgesamt mehr Kosinus-Funktionen, als für die Approximation der gesamten Feldlinie.

3.3.2 Randbehandlung

3.3.3 Principal Component Analysis

3.3.4 Cosinus-Transformation

Transformiert Daten in den Frequenzraum. Eine Menge von Punkten mit der Länge N kann in eine Menge von N Kosinusfunktionen überführt werden.
Es gibt verschiedene Möglichkeiten die Punkte zu transformieren. . Es wurde sich am JPEG/JFIF Standard orientiert , welcher, welche die DCT-II als Forwärts und die DCT-III als Rückwärtstransformation verwendet [**?**]. Der Unterschied liegt in der Randbehandlung

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

N ist anzahl Punkte. Transformiert in den Frequenzraum. x_n ist ein Punkt des Signales und X_k ist die Resultierende Kosinus Schwingung

Inverse Transform

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n cos[\frac{\pi}{N}n(k + \frac{1}{2})] \quad k = 0, 1, \ldots, N-1 \tag{3.2}$$

3.3.5 Quantisierung

3.3.6 Entropie Kodierung

Kodierung Zuerst einfaches Run-length

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

4 Qualitätsmessung der dekomprimierten Daten

Bei verlustbehafteten Kompressionen muss die Qualität des dekomprimierten Bildes sichergestellt werden. Im optimalen Fall ähneln die dekomprimierten Daten ihren Originalen, sodass für das menschliche Auge keine Artefakte sichtbar sind.
Im Verlauf der Arbeit wurden zwei Metriken verwendet: Die Standardabweichung und eine angepasste PSNR-HVS-M. Für die ersten Tests wurde nur die Standardabweichung gemessen. Es stellte sich heraus, dass die Standardabweichung nicht ausreicht: Sichtbare Artefakte, wie hochfrequente Schwingungen um die Originallinie fallen nicht ins Gewicht. Der absolute Fehler bleibt klein, für das menschliche Auge jedoch sind solche Artefakte störend (Ein Beispiel für die Artefakte ist in Abbildung 21 im Abschnitt 6.2.5 zu finden). Wie die Standardabweichung berechnet wird, ist im Abschnitt 4.2 beschrieben. Für weitere Tests wurde zusätzlich die PSNR-HVS-M Metrik berechnet. Die Metrik 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. Das Mass bewertet seltene, grosse Abstände stärker.
Die originale und dekomprimierte Feldlinie können unterschiedliche Abtastraten aufweisen. Die Standardabweichung muss deshalb unabhängig von der Abtastrate berechnet werden.

4.2.1 Allgemeiner Fall

Um die Abweichung unabhängig von der Abtastrate zu berechnen, wird zwischen den dekomprimierten Punkte eine Linie gezogen und den Abstand von dieser berechnet. Der Vorgang ist dargestellt im Diagramm der Abbildung 7. Für jeden Punkt $P1'$ aus den dekomprimierten Punkten D, nehme $P1'$ und den folgenden Punkt $P2'$. Ziehe eine Strecke s durch $P1'$ und $P2'$. Suche von $p1'$ den Originalpunkt $P1$ aus allen Originalpunkten

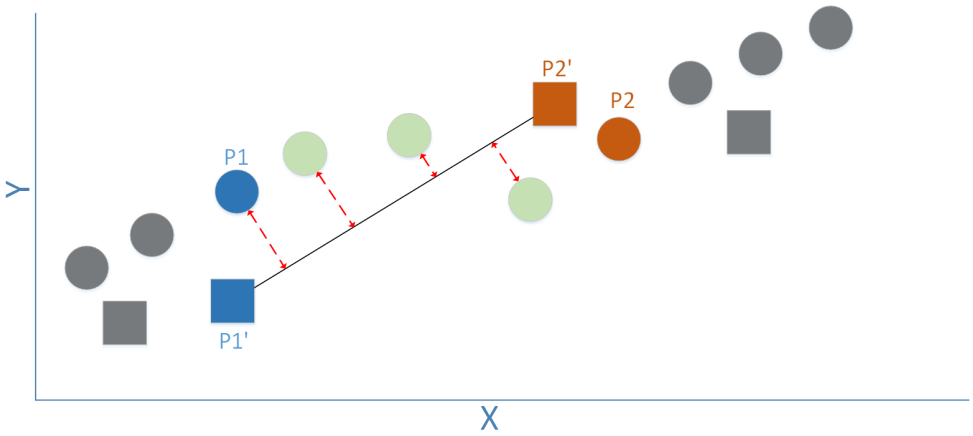

Abbildung 7: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

O und rechne den Abstand aus zur Strecke s. Führe das für alle folgenden Originalpunkte durch, bis $P2$ erreicht wurde. Der Abstand s zu $P2$ wird nicht mehr berechnet.

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 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.

4.2.2 Randbehandlung

Es ist möglich, dass die originalen Endpunkte der Feldlinien durch ein Subsampling verworfen wurden. Es ist möblich, dass am Anfang und am Ende Originalpunkte existieren, für die nie eine Distanz berechnet wurde. Das Diagramm der Abbildung 8 zeigt das Problem. Deshalb müssen die Abstände der Eckpunkte von der Komprimierten- zur Original-Linie berechnet werden.

4.2.3 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

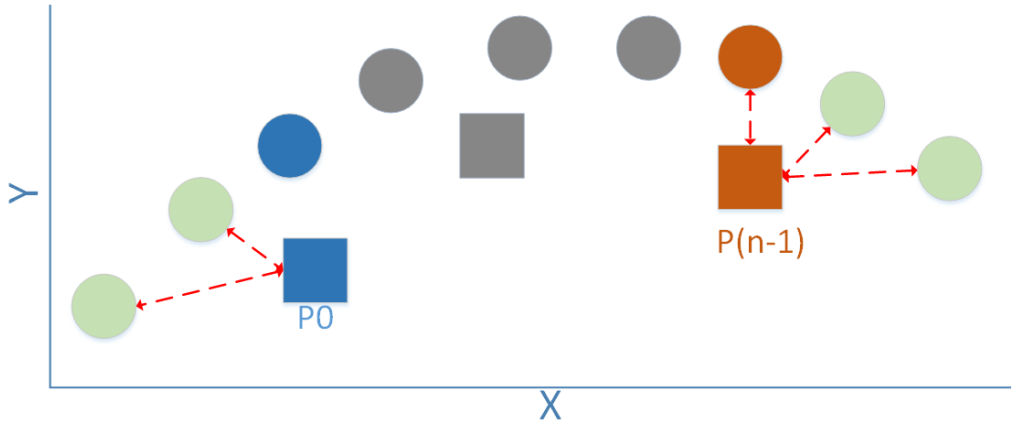

Abbildung 8: Darstellung der Fehlerberechnung. Die Punkte sind die Originaldaten, die Quadrate sind die Punkte nach der Kompression.

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 kann für die Messung des Fehlers zwischen dekomprimierten Bild und dem Original eingesetzt werden. Das Problem der Metrik ist aber, dass es nicht immer mit der menschlichen Wahrnemung korreliert. Ponomarenko et al. [**?**] 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.

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

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 $D()$. Der Unterschied zwischen der PSNR und der PSNR-HVS-M liegt in der Berechnung des durchschnittlichen quadratischen Fehlers. Das Diagramm der Abbidlung 9 zeigt den Ablauf der neuen Berechnung.
PSNR-HVS-M berechnet die Differenz zwischen dem Originalbild E und dem verrauschtem Bild D und 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:

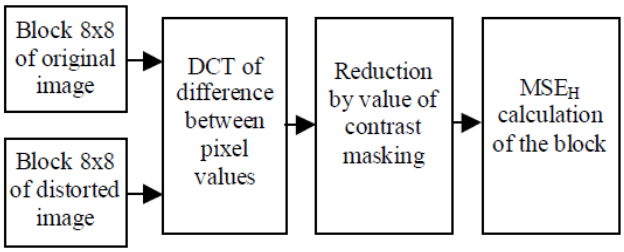

Abbildung 9: Flussdiagramm der PSNR-HVS-M Berechnung [**?**].

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

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

Wobei X die Kosinus-Koeffizienten eines Blockes sind und C die jeweiligen Gewichtungen zur Frequenz. 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).

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

Wobei $E_{norm} = \sqrt{max[E_m(X_e), E_m(X_d)]/64}$. 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 und quadrierten Werte der Standard JPEG Quantisierungsmatrix [**?**]. 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 für diese Arbeit

Der grösste Unterschied zur Arbeit von Ponomarenko et. al. ist der Einbezug des DC-Koeffizienten ins Contrast-Masking. Das menschliche Auge kann leichte Verschiebungen der Feldlinien kaum unterscheiden. Des weiteren musste für die Feldlinie eine eigene Quantisierungsmatrik gefunden werden. Empirisch getestet, bis keine Artefakte mehr zu erkennen waren. Sehr konservativ gewählt. Maxvalue der PSNR wurde auf den 4Fachen Sonnenradius. Der maximale Wert der PFSS Simulation.
psnr-hvs-m ist nicht samplingunabhängig.

Letztes Problem, festlegung des Normalisierungsfaktors f_m. Ponomarenko et. al. hat wirklich nicht mehr dazu

	f_m 16	f_m 40	f_m 80
Qualität: gut	**125.1**	123.1	117.8
Qualität: ok	**94.4**	92.8	91.7
Qualität: schlecht	**90.7**	88.3	86.9

Tabelle 1: Tabelle mit unterschiedlichen Werten für f_m.

geschrieben, als dass der Wert empirisch festgelegt wurde. 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.

5 Implementation

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

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

Während der Benutzer sich eine Aufnahme der Feldlinie sieht, soll asynchron bereits die nächsten Aufnahmen heruntergeladen und dekomprimiert werden. So ist der Wechsel von Aufnahme zu Aufnahme so kurz wie möglich. Damit die Dekompression den Wechsel nicht verlangsamt, werden mehr als nur die aktuelle Aufnahme dekomprimiert. Die komprimierten Daten sollen ebenfalls im Vorfeld heruntergeladen werden. So sind die Daten bereits im Arbeitsspeicher, sobald die Dekompression gestartet wird.
Durch das Caching der unkomprimierten und der komprimierten Feldlinien soll der Wechsel beschleunigt werden, wenn der Benutzer nicht die nächste Aufnahme, sondern die vorhergehende nochmals anzeigen möchte. Je nach grösse des Arbeitsspeichers könnten die unkomprimierten Daten komplett zwischengespeichert werden, sodass nur noch die Dekompression durchgeführt werden muss.

5.1 Software Architektur

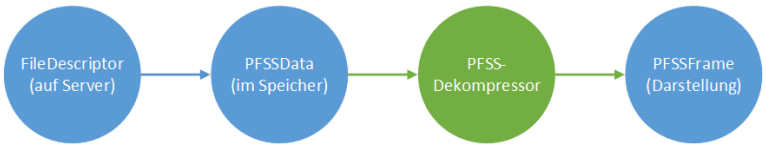

Abbildung 10: Zustandsdiagramm der Feldliniendaten

Die Daten der Feldlinien durchlaufen im JHelioviewer vier Zustände, welche durch vier Klassen abgebildet wurde. Die Klassen sowie die Zustandswechsel sind im Diagramm der Abbildung 10 dargestellt. Die Klasse ”FileDescriptor´´ repräsentiert eine Aufnahme von Feldlinien auf dem Server. In diesem Zustand sind die Daten bereit für das Herunterladen. Die folgende Klasse ”PFSSData´´ symbolisiert Feldlinien, welche in den lokalen Arbeitsspeicher geladen wurden. In diesem Zustand sind die Daten noch komprimiert und nicht bereit für eine Visualisierung. Für das Herunterladen ist ebenfalls die ”PFSSData´´ Klasse zuständig. ”PFSSDekompressor´´ ist ein Zwischenzustand und stellt den Wechsel von komprimierten zu unkomprimierten Daten dar. Da der Zustandswechsel aufwändig ist, wird es durch eine eigene Klasse abgebildet. Die letzte Klasse ”PFSSFrame´´ repräsentiert die dekomprimierten Feldlinien. In diesem Zustand sind die Daten bereit für die Darstellung. Die Darstellung wird ebenfalls von der ”PFSSFrame´´ Klasse übernommen.

5.1.1 Mehrstufiges Vorladen und Caching

Um den Flaschenhals herunterladen und Dekomprimieren zu umgehen, wird ein mehrstufiges Read-Ahead und Caching eingeführt. Es soll ein Vorladen und Caching für die unkomprimierten ”PFSSFrame´´ Feldllinien und eines für die ”PFSSData´´ Objekte implementiert werden. Die Implementation ist im Diagramm der

Abbildung 11 dargestellt. Der FileDescriptorManager sucht die Feldlinien auf dem Server und stellt die File-

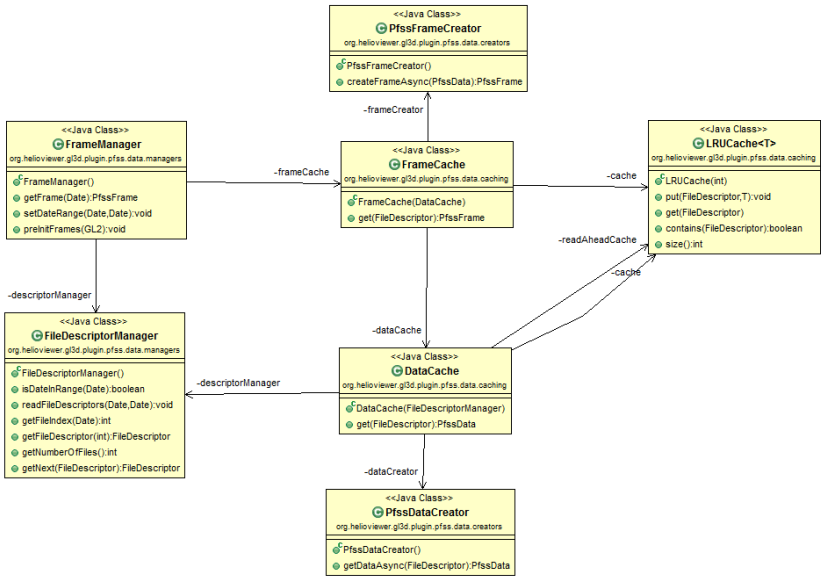

Abbildung 11: Diagramm der Implementation vom Vorladen und Caching

Descriptors zur Verfügung. Der Manager weiss ebenfalls, in welcher Reihenfolge die FileDescriptors zu liegen kommen.
Die Klasse ”Framemanager´´ ist die Facade der Vorladens- und Caching- Implementation. Sie abstrahiert das Zusammenspiel der verschiedenen Caches und den verschiedenen Zustände der Feldliniendaten, indem eine vereinfachte Schnittstelle angeboten wird. Für das Vorladen von PFSSFrame Objekte wurde wegen dem Prinzip von Low-Coupling ebenfalls im FrameManager umgesetzt. Die ”PFSSFrame´´ Objekte müssen jeweils Speicher auf der Grafikkarte alloziieren und abräumen. Die FrameManager Klasse muss genau wissen, wann ein PFSSFrame Objekt neu geladen oder nicht mehr gebraucht wird. Die Klasse FrameCache nur noch zuständig, PFSSFrame Objekte zwischenzuspeichern, welche keine Ressourcen der Grafikkarte alloziiert haben. Es ist auch eine Architektur denkbar, inder der FrameCache das Vorladen und das Abräumen des Grafikkartenspeichers übernimmt. Der Speicherplatz der Grafikkarte würde aber die Grösse des FrameCaches beschränken.
Das Vorladen und Caching der PFSSData Objekten wird von der Klasse DataCache übernommen. Die PFSSData Objekte alloziieren nur Arbeitsspeicher, die vom Garbage-Collector aufgeräumt werden können. Das Vorladen ist deshalb simpler und wird direkt im DataCache implementiert durch eine weitere Instanz des LRUCaches. Der eigentliche Cache beinhaltet alle PFSSData Objekte, welche gebraucht wurden und der readAheadCache alle, welche noch gebraucht werden können.

In dieser Arbeit wurde ein Least-Recently-Used (LRU) Cache Algorithmus verwendet. Wenn der Cache voll ist, löscht der Algorithmus das längste nicht verwendete Objekt. Da der JHelioviewer im allgemeinen Fall sequenzell nach den Datenobjekten abfragt, kann der LRU Cache mit einer First-in-First-out Queue implementiert werden. Das Objet, welches am längsten nicht mehr gebraucht 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 Aufnahme der Feldlinien angekommen ist, wird an Wrap-around durchgeführt und wieder die erste Aufnahme verlangt. Wenn der Cache $n - 1$ von n Aufnahmen abspeichern kann, so löscht der LRU-Algorithmus immer die Aufnahme, welches als übernächstes abgefragt wird. Das führt dazu, dass gleich viele Cache-Misses geschehen, als wenn der Cache nur halb so gross wäre.

5.1.2 Asynchrone Aufrufe mittels Executor Services

Im Diagramm der Abbildung 11 zu sehen ist, wird das Erstellen von PFSSData und PFSSFrame Objekten jeweils von zwei Creators übernommen. Sie sind zuständig für das asynchrone Herunterladen und Dekomprimieren der Feldliniendaten und sind mit einem Standard Executor Service umgesetzt.
Beim asynchronen Aufruf wird jeweilds das angeforderte PFSSData oder PFSSFrame Objekt bereits zurückgegeben, während der Creator die Daten herunterläd oder dekomprimiert. Die Klassen PFSSData und PFSSFrame müssen Threadsafe implementiert werden. Da die Dekompression erst starten kann, wenn alle Daten heruntergeladen wurden, muss die PFSSData Klasse eine Wait-Logik anbieten. PFSSFrame objekt ist die Frage, ob blocking oder non-blocking. PFSSData muss eine Wait-Logik implementieren. Der PFSSDecompressor kann erst anfangen, wenn alle Daten heruntergeladen wurden.

Lösungsansatz	Kompressionsraten
Adaptives Subsampling	11.6

Tabelle 2: Tabelle der

6 Resultate

Es gibt x Lösungenansätze mit x varianten. hier sind die Kompressionsraten der jeweils besten Varianten der Ansätze.
Bis auf den Lösungsansatz unter 6.1 Unterschiedliche Qualitätsstufen Im Verlauf wurde eine neue Metrik entwickelt, deshalb gibt es manchmal zwei Plots. Einmal mit der Standardabweichung, und einmal mit PSNR-HVS-M. Standardabweichung ist der beste Ort unten Links, bei PSNR-HVS-M Plots Oben Links. Bis aus den Lösungsansatz 6.1 sind alle Ansätze mit unterschiedlichen Qualitätsstufen getestet.

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-

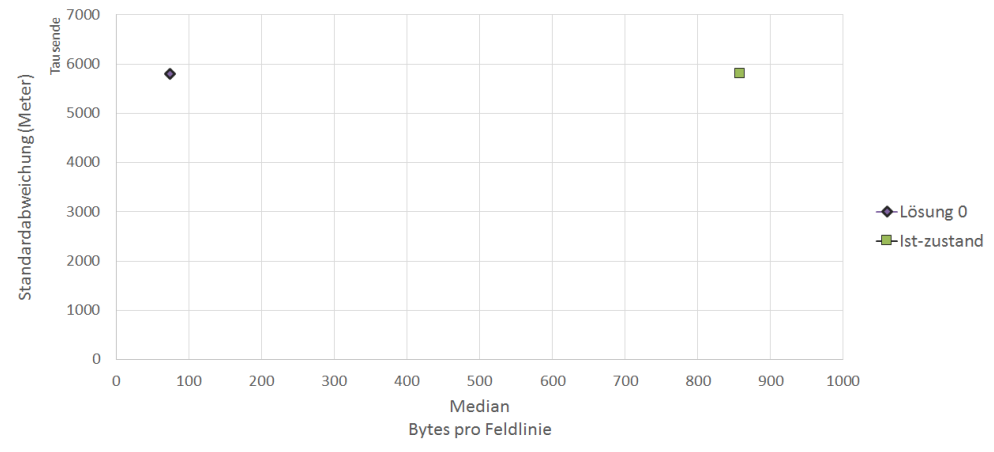

Abbildung 12: Vergleich des Lösungsansatzes: Adaptives Subsampling zur Ist-Kompression.

gramm 12 erkennbar ist, braucht dieser Lösungsansatz deutlich weniger Speicher als die Ist-Kompression. Das Adaptive Subsampling reduziert deutlich die Anzahl Punkte, während die Rar eine bessere Kompression erbringt. Dieser Ansatz kann die Daten um Faktor 11.6 besser komprimieren.
Die Komplexität dieses Ansatzes bleibt ist $O(n)$ (n ist die Anzahl Punkte) und bleibt somit in der selben Komplexitätsklasse wie die Ist-Kompression. Dieser Lösungsansatz ist aber bei der Dekompression schneller, da n etwa vier mal Kleiner ist. und ist som Da bei der Dekompression n etwa vier Mal weniger Punkte bearbeiten muss, ist die Dekompression sogar schneller als die Ist-Lösung.
Die Abbildung 13 zeigt die Artefakte, die bei der Komprimierung der Lösung 0 entstehen. Es ist anzumerken, dass der Ist-Zustand die selben Artefakte aufweist.

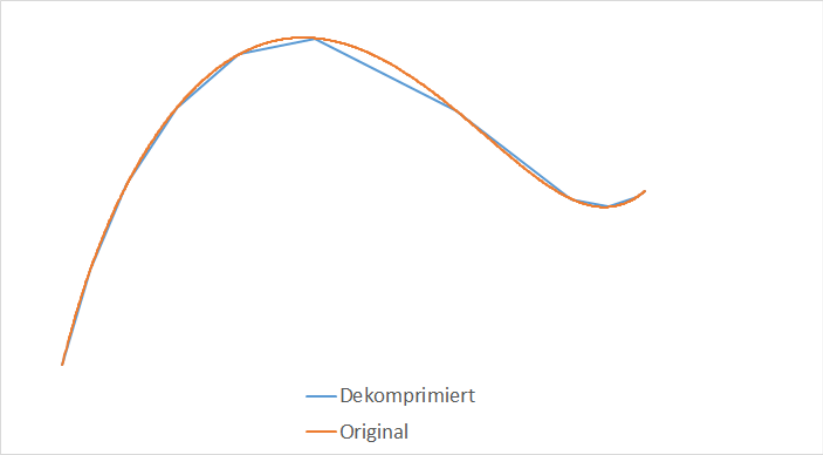

Abbildung 13: Artefakte der Lösung 0

6.2 Lösungansatz: Diskrete Kosinus Transformation

In den folgenden Abschnitten werden die Resultate verschiedener Varianten vorgestellt. Alle Varianten bestehen grob aus fünf Teilschritten: Einem Subsampling[1], einer Folge von verschiedenen Transformationen, bei der eine die Diskrete Kosinus Transformation ist, Abspeicherung ins Fits Format[2] undeiner Quantisierung und einer Entropie Kodierung mit Rar

ringing artefakte

x Varianten mit den verschiedenen vorhergehenden Transformationen die DCT zu verbessern. Die Quantisierung wurde linear für alle durchgeführt. d r[3].

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

6.2.1 Variante: DCT

Diese Variante verwendet die Diskrete Kosinus Transformation und die lineare Quantisierung. Die Abbildung 14 zeigt den Vergleich der DCT Kompression mit dem Lösungsansatz des Adaptiven Subsamplings (siehe 6.1). Es ist deutlich zu erkennen, dass die Standardabweichung schnell steigt bei leicht sinkender Grösse. Der Grund dafür kann im Diagramm der Abbildung 16 entnommen werden. Un den meisten Fällen kann die DCT die Feldlinie gut approximieren. Bei dieser Feldlinie wird der Anfang der Kurve nicht richtig dargestellt. Das ist ein typisches Problem der DCT. Die Diskrete Kosinus Transformation nimmt an, dass das Signal sich periodisch wiederholt. Die implememententierte Transformation (siehe Abschnitt 3.3.4) nimmt an, dass am Anfang und am Ende das Signal in umgekehrter Reihenfolge wiederholt. Bei den Feldlinien kann das zu einer Diskontinuität im Signal führen, welches hochfrequente Kosinus Anteile. Wenn die Quantisierung Hochfrequente Schwingungen verschluckt, entstehen die Artefakte der Abbildung 16.
Eine Möglichkeit ist die Feldlinie um Punkte zu erweitern. Wenn die Feldlinie am Anfang und am Ende abflacht,

[1] siehe Abschnitt 3.3.1
[2] siehe Abschnitt **??**
[3] siehe Abschnitt 3.2.2

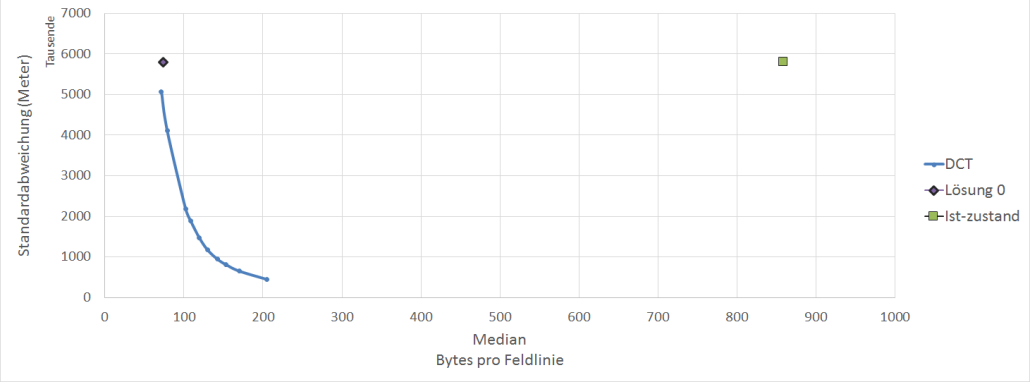

Abbildung 14: Vergleich der DCT Kompression mit der Lösung0

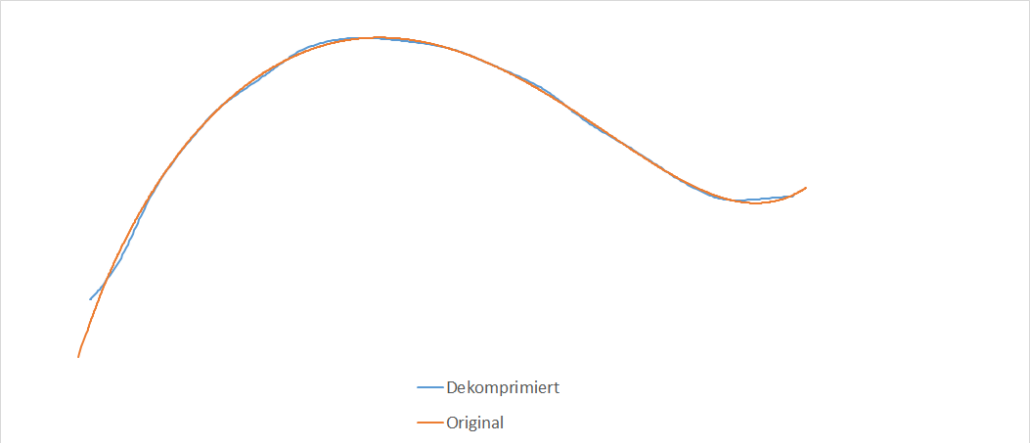

Abbildung 15: Artefakte der DCT Dekompression anhand Beispieldaten

sollte die resultierende Transformation weniger hochfrequente Schwingungen enthalten. Diese Variante wird im Abschnitt 6.2.5 behandelt und führt zu einer deutlich besseren Approximation. Durch eine andere Darstellung der Daten kann das Problem ebenfalls gelöst werden.

6.2.2 Variante: Ableitung+DCT

Vor der DCT werden die Feldlinien abgeleitet. Mit der Ableitung soll das Randproblem dargestellt in 16 gelöst werden. Die Steigungen sind kleinere Zahlen, was an den Rändern keine grosse Spitze verursachen sollte. Der Nachteil ist, dass Ungenauigkeiten sich durch die Kurve durchziehen und summieren.
Die abgeleiteten Feldlinien können besser komprimiert werden und erreichen dadurch eine Kompressionsrate

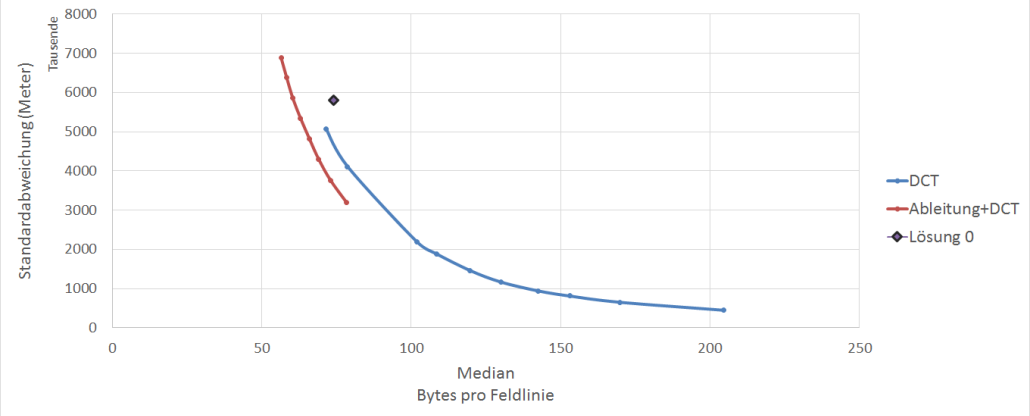

Abbildung 16: Vergleich der DCT Kompression der Ableitung mit der DCT Kompression

von 14.3 mit der vergleichbaren Genauigkeit. Das Randproblem wurde ebenfalls verbessert. Eine Darstellung der Artefakte ist im Diagramm der Abbildung19 zu finden.

6.2.3 Variante: PCA+Ableitung+DCT

Die Feldlinien liegen meist auf einer Ebene im dreidimensionalen Raum. Wenn die X,Y und Z Kanäle Kosinus-Transformiert werden, ist die Information etwa gleichmässig auf den Kanälen verteilt. Eine Linie könnte sich durch weniger Kosinus-Funktionen approximieren lassen, wenn die Linie zuerst in ein lokales Koordinatensystem transformiert wird.
Die Principal Component Analysis (PCA)[**?**] ist ein Verfahren aus der Statistik, welches Daten in ein neues koordinatensystem Transformiert. Dabei werden die Achsen so gelegt, dass die Daten entlang der ersten Achse die grösste Varianz aufweisen. Entlang der zweiten Achse, welche orthogonal zur ersten liegt, die zweithöchste Varianz etc. Wenn das Vefahren auf die Feldlinien angewandt wird, werden die Feldlinien in ein lokales System transformiert indem der Z-Kanal 0 ist, wenn die Feldlinie in einer Ebene liegt. Der Nachteil ist, dass für die Rücktransformation pro Feldlinie die Koordinatenachsen und die Koordinatenverschiebung abgespeichert werden.
Vor der DCT wird nun eine PCA durchgeführt. Die sechs Faktoren der neuen Koordinatenachsen werdenmit 16 Bit Genauigkeit in die Fits-Datei abgelegt und die drei Verschiebung-Faktoren mit 32 Bit. Der Vergleich 17 zeigt deutlich, dass sich der Mehraufwand nicht lohnt, obwohl die PCA vielversprechend scheint. Eine Feldlinie lässt sich mit 5 bis maximal 20 Kosinus-Funktionen pro Kanal approximieren. Durch die PCA-Transformation lässt sich das noch minim verkleinern, aber die zusätzlichen Informationen wie die Werte für neuen Koordinatenachsen und für die Verschiebung verbrauchen mehr speicher, als durch die Transformation gewonnen werden kann.

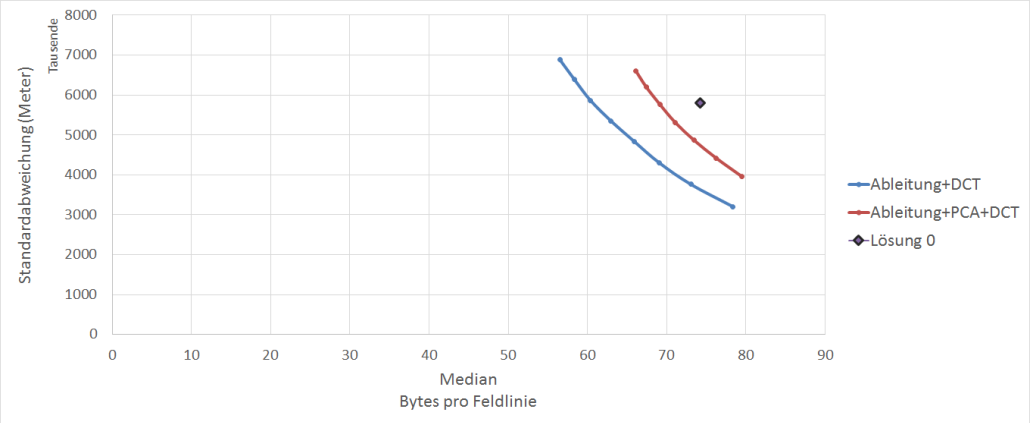

Abbildung 17: Vergleich der PCA DCT Kompression der Ableitung mit der DCT Kompression der Ableitung

Die PCA-Variante könnte noch verkleinert werden. Die Verschiebung kann Quantisiert werden, oder man kann weniger Genauigkeit für die Koordinatenachsen verwenden. Die PCA-Variante ist aber nicht genauer wie die Ableitung+DCT Variante. Wie auch in 17 ersichtlich, ist die PCA-Variante bei vergleichbarer Kompressionsgrad ungenauer. Unter dem Strich hat die PCA keine Verbesserung erbracht.

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

6.2.4 Variante: Ableitung+DCT+Byte Kodierung

Es wird versucht, mit einer Byte Kodierung die DCT-Koeffizienten der Variante 6.2.2 besser zu komprimieren. Die Koeffizenten werden mit zwei Verfahren kodiert: Mit einer simplen Run-Length Kodierung und einer adaptiven Byte Kodierung. Die adaptive Byte Kodierung versucht jeden koeffizienten mit einem Byte darzustellen. Wenn die Genauigkeit nicht ausreicht, wird ein weiteres Byte hinzugenommen. Die Kodierung ist im Abschnitt 3.3.6 beschrieben. 18 zeigt eine deutliche Verbesserung der Kompressionsrate, wenn die Koeffizienten mit der

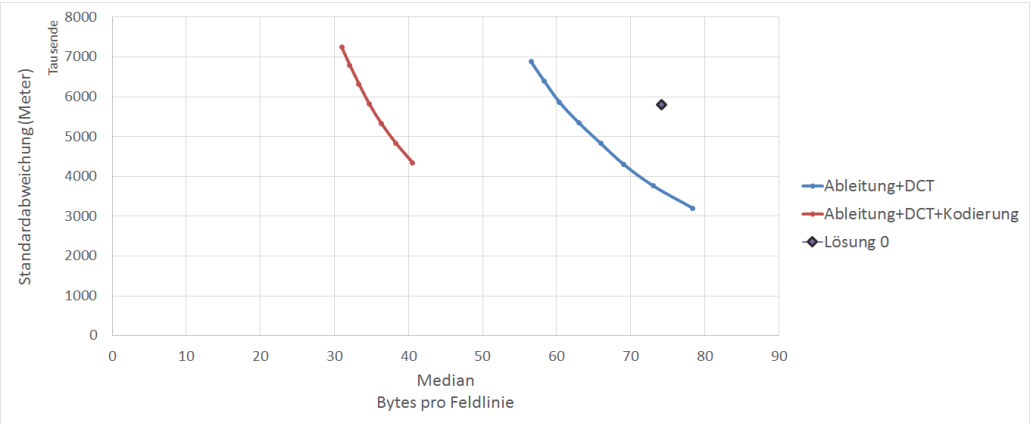

Abbildung 18: Vergleich der Kompression mit und ohne Byte-Kodierung

unter 3.3.6 beschriebenen Verfahren kodiert werden. Bei ähnlicher Genauigkeit wie der Ist-Zustand braucht

diese Variante durchschnittlich 35 Bytes pro Feldlinie. Bei 1200 Feldlinien eine ergibt das eine Dateigrösse von 42 KiByte pro Aufnahme. Im Vergleich zum Ist-Zustand sind die Dateien um das 24 Fache kleiner.

Bei der Variante 6.2.1 war das Problem, dass die Ränder schlecht darzustellen waren. Es stellt sich die Frage, was für Artefakte diese Kompression aufweist. Im Diagramm der Abbildung19 ist deutlich zu sehen, dass die

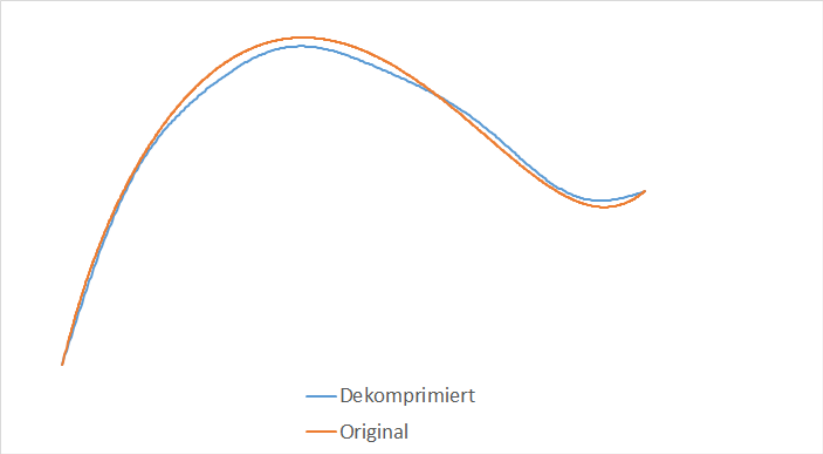

Abbildung 19: Artefakte der DCT Kompression der Ableitung

Kurve durch die Quantisierung gedämpft wird. Die Maximum der Kurve ist tiefer, sowie das lokale Minima der letzten Halbwelle höher. Der Vorteil dieser Variante ist, dass die resultierende Feldline sehr glatt verläuft. Ohne die Originalkurve währen die Artefakte nicht zu identifizieren.
Wenn die Artefakte 19 und 13 vergleicht, fällt auf, dass die Variante 6.2.1 die Feldlinie genauer approximiert. Wenn die Ränder besser dargestellt werden, ist es Denkbar, dass die Variante 6.2.1 wenigere Kosinus-Funktionen braucht für eine ähnlich genaue Approximation.

6.2.5 Variante: Randbehandlung+DCT+Byte Kodierung

Wieder nur die Diskrete Kosinus Transformation, aber noch mit künstlich erzeugten Punkten3.3.2 und der Byte Kodierung3.3.6. Das Diagramm der Abbildung 20 zeigt den Vergleich der Variante mit Randbehandlung

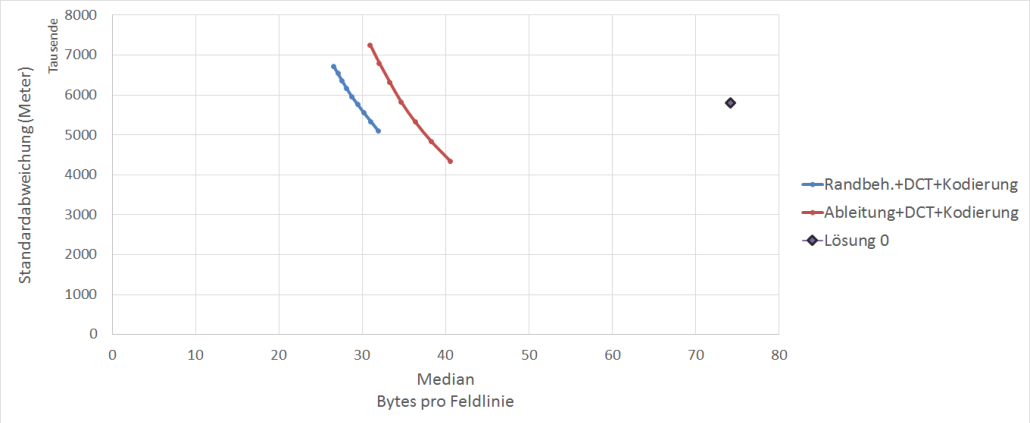

Abbildung 20: Vergleich der Kompression mit und ohne Byte-Kodierung

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

Trotz einer vergleichbaren Genauigkeit wie die Ist-Kompression, sind auf der JHelioviewer visualisierung deutliche Artefakte zu sehen. Die Abbildung 21 vergleicht die originalen mit dekomprimierten Feldlinien. Die Dekompression Lässt die Feldlinien um das originale Signal schwingen. In diesem Fall scheint die Standardab-

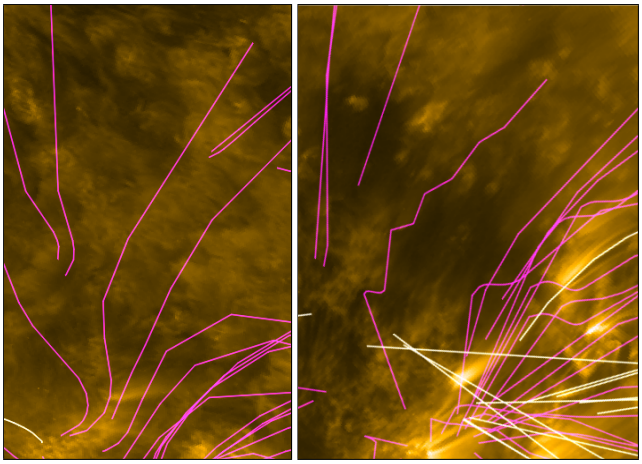

Abbildung 21: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten.

weichung als Fehlermass zu versagen: Da die Schwingungen nahe am originalen Signal liegen, bleiben die Abstände klein. Jedoch sind die Artefakte für das menschliche Auge inakzeptabel.
Interessant ist, dass die die Variante der Ableitung (Abschnitt 6.2.4) ähnliche Artefakte aufweist. Im Diagramm der Abbildung 19, welches die Artefakte anhand einer Beispiellinie zeigt, sind keine Schwinungen zu entdecken. In der JHelioviewer Visualisierung jedoch, sind auch bei dieser Variante deutliche Schwingungen zu erkennen. Die Abbildung 22 zeigt die Artefakte. Es ist anzumerken, dass die Artefakte weniger ausgeprägt sind, aber dennoch störend für das menschliche Auge. Ebenfalls interessant ist, dass hauptsächlich Feldlinien betroffen sind, welche vom Weltall zur Sonne oder von der Sonne zum Weltall führen.
Es wird vermutet, dass es sich um Ringing Artifacts [**?**] handelt. Sie geschehen, wenn ein Signal harte

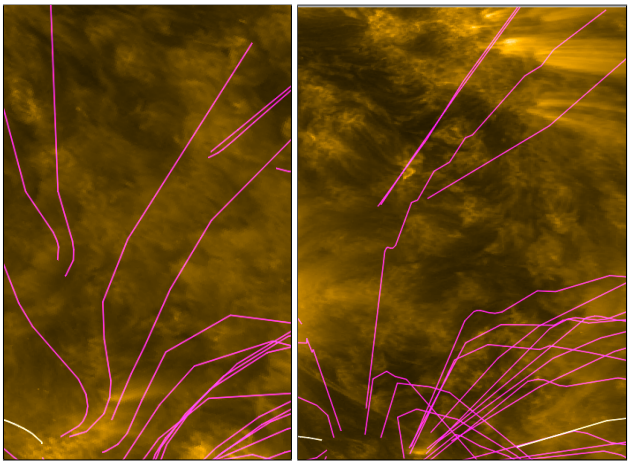

Abbildung 22: Artefakte der Kompression, links sind die originalen Feldlinien, rechts die Dekomprimierten der Variante 6.2.4.

übergänge beinhaltet. Ein weiteres Indiz dafür ist, dass die Feldlinien von der Sonne zur Sonne nicht betroffen sind. Sie verlaufen sehr kontinuierlich und ändern richtungen langsam. Die Feldlinien vom Weltall zur Sonne oder von der Sonne zum Weltall dagegegen können abrupte Richtungswechsel vornehmen.

ist ein Vergleich der Original- mit den dekomprimierten Feldlinien dieser Variante zu sehen. Die Artefakte sind deutlich schwächer ausgeprägt. Metrik ist unbrauchbar, Deshalb wurde eine neue Metrik entwickelt, welche

unter4.3 beschrieben ist.

7 Diskussion

8 Fazit

Abbildungsverzeichnis

Tabellenverzeichnis

9 Anhang

subsectionInstallationsanleitung

10 Ehrlichkeitserklärung