

BACHELORARBEIT

Rauschsignalsynthese aus Filmkorn

vorgelegt am 27. Juli 2026
Malte Bungard

Erstprüfer: Prof. Dr. Jan Mietzner
Zweitprüfer: Prof. Dr. Robert Mores

**HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG**

Fakultät Elektro-, Medien- und Informationstechnik (EMI)
Finkenau 35
22081 Hamburg

Inhaltsverzeichnis

Abbildungsverzeichnis	III
1 Einleitung	1
2 Algorithmische Erzeugung von Rauschen	2
2.1 Generieren von Zufallszahlen	2
2.2 Leistungsspektren unterschiedlicher Rauscharten	3
2.3 Psychoakustische Wahrnehmung von Rauschen	3
3 Grundlagen analoger Fotografie	6
3.1 Aufbau fotografischen Films	6
3.1.1 Schichtaufbau	6
3.1.2 Fotoemulsion	7
3.2 Entstehung des latenten Bildes	7
3.3 Entwicklung des Films	8
3.4 Scannen und Digitalisieren	8
3.5 ISO-Empfindlichkeit und Korngröße	9
3.6 Unterschied zu digitalen Fotos	13
4 Erzeugung von Rauschen aus Filmkorn	14
4.1 Extraktion des Filmkorns	14
4.1.1 Kantendetektion	14
4.1.2 Hochpassfilter	16
4.1.3 Differenz der Gauß-Filter	17
4.1.4 FFT-basierte Extraktion	18
4.2 Erzeugen des akustischen Rauschens	20
4.2.1 Lineares Interpolieren von Frequenzen	20
4.2.2 Verwenden der Kornverteilung als Spektrogramm	24
4.2.3 Erzeugen und Mischen der Frequenzen in Matlab	25
4.3 Verrechnen des Rauschens mit Musik	30

5	Fazit und Auswertung	32
5.1	Vergleich zu konventionell erzeugtem Rauschen	32
5.1.1	Frequenzanalyse	32
5.1.2	Subjektive Wahrnehmung	46
5.2	Auswertung der Verrechnung von Rauschen und Musik	47
5.3	Vergleich zu natürlichem analogen Rauschen	53
5.4	Auswirkung von verschiedenen Parametern	57
5.4.1	Blockgröße	57
5.4.2	Perzentil	58
5.4.3	Anzahl Frequenzbänder	58
5.4.4	Framedauer	59
5.5	Weitere Anwendungsmöglichkeiten	60
	Literatur	61
	Anhang	62

Abbildungsverzeichnis

2.1	Leistungsdichtespektren verschiedener Rauscharten	3
2.2	Gleichlautstärke-Kurven	4
2.3	Frequenzabhängige Maskierung	5
3.1	Aufbau Schwarzweißfilm	6
3.2	Entstehung des latenten Bildes	7
3.3	Belichtungsdreieck	10
3.4	Beispielbild ISO 100/21°	11
3.5	Beispielbild ISO 200/24°	11
3.6	Beispielbild ISO 400/27°	12
3.7	Beispielbild ISO 500/28°	12
3.8	Beispielbild ISO 3200/36°	13
4.1	Kornextraktion per Sobel-Algorithmus, Grenzwert 0,15	16
4.2	Kornextraktion per Hochpass-Filter, $\sigma = 8$, Kernelgröße 21	17
4.3	Ergebnis der 2D-DFT	19
4.4	Multiplikation des transformierten Bildes mit der Maske	19
4.5	Ergebnis der Inversen 2D-DFT	20
4.6	Abgetasteter Block	21
4.7	Gezählte Verteilung des Korns im ersten Block	22
4.8	Interpolierte Werte mit logarithmischer Verteilung	23
4.9	Interpolierte Werte mit linearer Verteilung	23
4.10	Matrix als Spektrogramm visualisiert	24
4.11	sinTable visualisiert	26
4.12	Darstellung der Audioframes mit Phasenreset 1	28
4.13	Darstellung der Audioframes mit Phasenreset 2	28
4.14	Darstellung der Audioframes ohne Phasenreset 1	30
4.15	Darstellung der Audioframes ohne Phasenreset 2	30
5.1	Frequenzanalyse Nr. 1	33
5.2	Frequenzanalyse Nr. 2: Gaußsches Rauschen	34
5.3	Frequenzanalyse Nr. 3	35
5.4	Frequenzanalyse Nr. 4	36

5.5	Frequenzanalyse Nr. 5: Gaußsches Rauschen mit kontinuierlichem Phasengang	37
5.6	Spektogramm 1	38
5.7	Spektogramm 2	39
5.8	Spektogramm 3	40
5.9	Spektogramm 4	41
5.10	Spektogramm 5	42
5.11	Histogramm und Spektogramm von weißem Rauschen	43
5.12	Histogramm und Spektogramm von rosa Rauschen	44
5.13	Histogramm und Spektogramm von rotem Rauschen	45
5.14	Frequenzhistogramm vom Rauschen	47
5.15	Frequenzhistogramm von Musikbeispiel	48
5.16	Addition von Rauschen und Musik	49
5.17	Multiplikation von Rauschen und Musik	50
5.18	Addition des Produkts von Rauschen und Musik	51
5.19	Addition des Produkts von Rauschen und Musik als Spektogramm	52
5.20	Frequenzgang Bandrauschen	53
5.21	Histogramm-Vergleich von künstlichem Rauschen und Schallplatte	54
5.22	Spektogramm-Vergleich von künstlichem Rauschen und Schallplatte	55
5.23	Frequenzgang Schallplattenrauschen	56
5.24	Spektogramm Schallplattenrauschen	57
5.25	Reduktion des Perzentils	58
5.26	Verändern der Framedauer	59

1 Einleitung

Akustisches Rauschen ist in vielen Bereichen im digitalen Umfeld anzutreffen. Sei es die Emulation von analogem Vintage-Sound oder ein „Comfort-Noise“, um zum Beispiel beim Mobilfunk die Stille zu füllen. Diese Rauschsignale werden jedoch in der Regel algorithmisch erzeugt und wirken daher künstlich. In dieser Arbeit werde ich ein alternatives Verfahren vorstellen und untersuchen: Das akustische Rauschen soll aus dem Bildrauschen bzw. Filmkorn von analogen Bildern generiert werden. Auf diese Art entsteht ein Rauschen, das auf echten zufälligen Parametern basiert und einem natürlichen Rauschen näher kommt. Es werden verschiedene Verfahren untersucht - sowohl beim Extrahieren des Bildrauschens als auch beim Erzeugen des akustischen Rauschens. Das auf diese Art erzeugte Rauschsignal wird sowohl einem natürlichen akustischen Rauschen als auch einem klassisch algorithmisch erzeugten Rauschen gegenübergestellt.

2 Algorithmische Erzeugung von Rauschen

2.1 Generieren von Zufallszahlen

Echte Zufallszahlen kann ein Computer im Normalfall nicht bestimmen. Computer arbeiten logikbasiert und haben den Anspruch, zuverlässige Vorhersagen treffen zu können. Dies widerspricht dem Konzept von Zufall grundlegend. Meistens werden daher sogenannte „Pseudo-Zufallszahlen“ verwendet, die algorithmisch berechnet werden. Dabei dient als Grundlage ein Startwert, der sogenannte Seed, von dem aus weitergerechnet wird. Hier ein Beispiel-Algorithmus, der auf einem linearen Kongruenzgenerator basiert: Es gibt vier natürliche Zahlen als Parameter; Das Modul $m \in \{2, 3, 4, \dots\}$, den Faktor $a \in \{1, \dots, m - 1\}$, das Inkrement $b \in \{1, \dots, m - 1\}$ und den Startwert $y_0 \in \{0, \dots, m - 1\}$. Das Modul m gibt die maximale Periodenlänge an. Aus dem Startwert werden dann die Werte für $i \geq 1$ mit folgender rekursiver Formel berechnet:

$$y_i = (a * y_{i-1} + b) \bmod m$$

Ein kleines Beispiel: Sei $m = 10, a = 2, b = 1, y_0 = 0$. Dann ist die erzeugte Zahlenfolge die immer wiederkehrende Abfolge $1 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow \dots$

Verschärft man die Bedingungen, sodass b zu m teilerfremd ist, jeder Primfaktor von m $a - 1$ teilt und aus der Teilbarkeit von m durch 4 die Teilbarkeit von $a - 1$ durch 4 folgt, kommt jede Zahl von 0 bis $m - 1$ tatsächlich in der erzeugten Zahlenfolge vor. Aber auch in diesem Fall wird jede Zahl genau einmal ausgegeben, bevor sich dieselbe Folge exakt wiederholt. Derartige Zufallsgeneratoren sind für viele Zwecke ausreichend; allerdings wird überall dort, wo es auf Sicherheit ankommt, auf hardwarebasierte Systeme gesetzt. Dabei werden physikalische Prozesse und Werte wie beispielsweise Wetterdaten oder gemessene Hintergrundstrahlung entweder direkt als Zufallsvariable oder als Startwert für einen Kongruenzgenerator verwendet. Das Sicherheitsunternehmen Cloudflare benutzt für seine kryptographischen Einsatzzwecke eine Wand voller Lavalampen, deren Helligkeit und Farbe mittels einer Kamera erfasst und analysiert werden („How Do Lava Lamps Help with Internet Encryption?“, [n.d.](#)). Bei meiner Methode mit dem Filmkorn werden nicht bei jedem Programmdurchlauf neue zufällige Werte erstellt - ein bestimmter Scan liefert stets die selben

Werte. Daher handelt es sich um eine physikalisch entstandene Zufallsstruktur, nicht um eine Echtzeit-Zufallszahlen-Generierung.

2.2 Leistungsspektren unterschiedlicher Rauscharten

Bei der künstlichen Erzeugung beziehungsweise Simulation von analogem Rauschen wird meistens mit mehreren Pink- und Brown-Noise-Signalen gearbeitet, die mit unterschiedlichen Parametern tiefpassgefiltert und anschließend überlagert werden. Weißes Rauschen (White Noise) zeichnet sich dadurch aus, dass die Leistungsdichte $S(f)$ über alle Frequenzen konstant ist. Bei rosa Rauschen (Pink Noise) ist die Leistungsdichte gegeben durch $S(f) = \frac{1}{f}$. Brownsches bzw. rotes Rauschen hat einen noch deutlicheren Abfall der Leistungsdichte bei steigender Frequenz von $S(f) = \frac{1}{f^2}$. Die spektrale Verteilung der unterschiedlichen Rauscharten ist in Abbildung 2.1 zu sehen. Es gibt noch weitere Rauscharten mit anderen Leistungsspektren, die allerdings für die weitere Betrachtung und Untersuchung keine Rolle spielen.

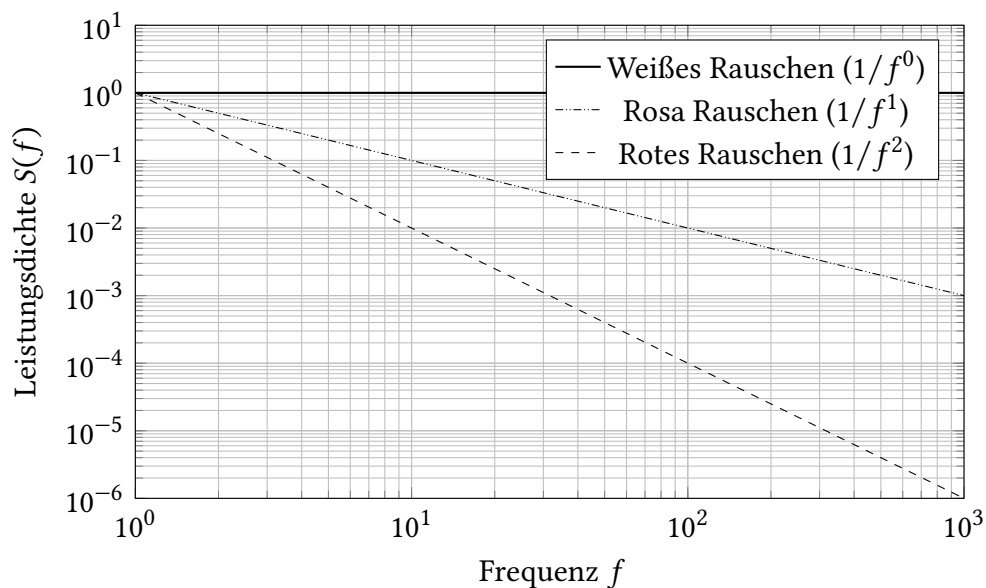


Abbildung 2.1: Schematische Leistungsdichtespektren verschiedener Rauscharten.

2.3 Psychoakustische Wahrnehmung von Rauschen

Die technische Beschreibung eines Rauschsignals erlaubt nur bedingt, eine Aussage bezüglich der persönlichen Wahrnehmung zu treffen. Auch bei gleichen Frequenzspektren können

sich Signale unterschiedlich anhören und eine andere Wirkung auf zuhörende Personen auslösen. Die Psychoakustik untersucht den Zusammenhang zwischen technischer, objektiver Basis und persönlicher Wahrnehmung. Für die in dieser Arbeit betrachteten und erzeugten Rauschsignale sind die spektrale Verteilung der Frequenzen aber auch deren zeitlicher Verlauf relevant. Das menschliche Gehör kann Frequenzen von 20 Hz bis 20 kHz wahrnehmen, wobei die Wahrnehmung von besonders hohen Frequenzen mit zunehmendem Alter abnimmt. Die stärkste Empfindlichkeit haben wir allerdings im Bereich von 2 kHz bis 5 kHz („ISO 226:2023 Acoustics – Normal equal-loudness-level contours“, 2023). Dies sorgt unter anderem dafür, dass weißes Rauschen eher als zischend wahrgenommen wird, wohingegen sich rosa Rauschen angenehmer und ausgeglichener anhört (Doyle & Evans, 2018). Bei rosa Rauschen verringert sich die Leistungsdichte mit steigender Frequenz und die Frequenzen werden über das gesamte Spektrum als ähnlich laut wahrgenommen. Abbildung 2.2 zeigt schematische Gleichlautstärke-Kurven für unterschiedliche Lautstärkepegel. Jeder Kurvenverlauf verbindet Töne, die trotz unterschiedlicher Frequenz als gleich laut wahrgenommen werden. Bei einer Frequenz von 1 kHz entspricht der Schalldruckpegel in Dezibel dem jeweiligen Lautstärkepegel in phon. Aus den Kurven wird ersichtlich, dass das menschliche Gehör insbesondere im Bereich zwischen ungefähr 2 kHz und 5 kHz empfindlich ist. Tiefe und sehr hohe Frequenzen benötigen hingegen einen höheren Schalldruckpegel, um denselben Lautstärkeindruck hervorzurufen (Genuit, 2014).

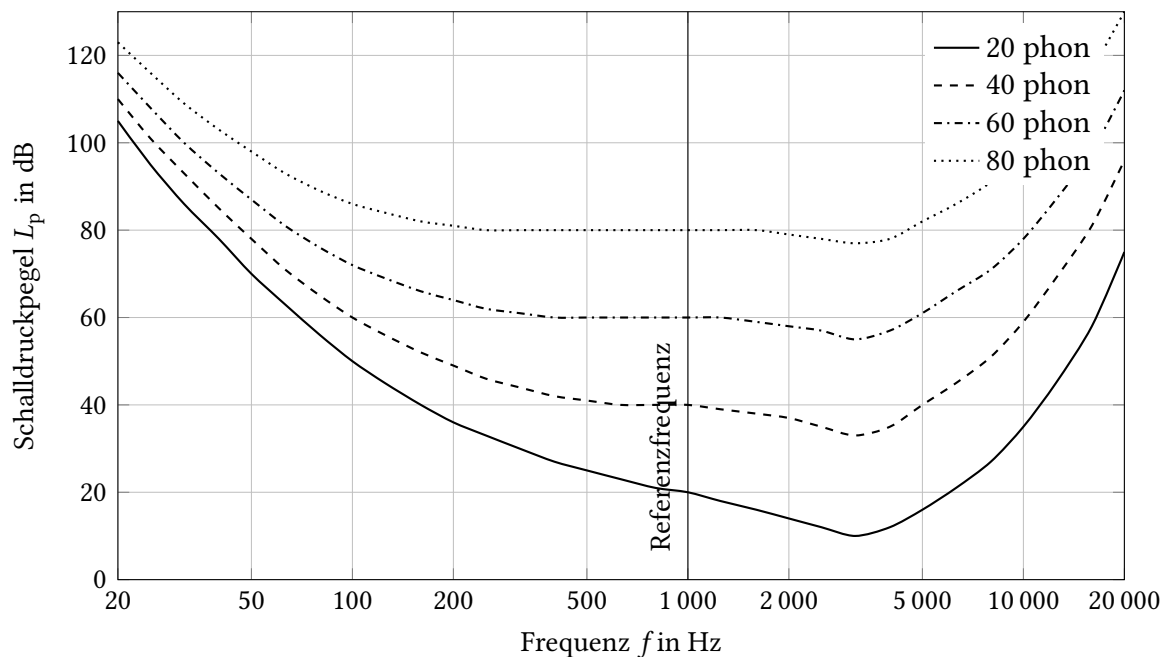


Abbildung 2.2: Schematische Gleichlautstärke-Kurven für verschiedene Lautstärkepegel

Neben der spektralen Verteilung spielt auch die zeitliche Struktur eines Signals eine wichtige Rolle. Rein synthetisch erzeugte Rauschsignale basieren häufig auf statistisch idealen Zu-

fallsprozessen. Obwohl diese mathematisch zufällig sind, besitzen sie oft eine sehr konstante spektrale Verteilung über die Zeit. Natürliche Rauschquellen wie Wind, Wasser, Magnetbandrauschen oder Filmkorn weisen hingegen kleine lokale Schwankungen auf. Diese Variationen entstehen durch physikalische Prozesse und führen dazu, dass das Signal lebendiger und weniger statisch wirkt. Für die in dieser Arbeit entwickelte Methode ist insbesondere die Annahme interessant, dass die aus Filmkorn gewonnene Frequenzverteilung stärkere lokale Variationen besitzt als konventionell erzeugtes weißes Rauschen. Dadurch könnte ein Rauschsignal entstehen, das von den hörenden Personen als natürlicher wahrgenommen wird. Ob dies tatsächlich der Fall ist, müsste jedoch durch geeignete Hörversuche überprüft werden. Die in Abschnitt 5.1.2 vorgeschlagenen Verfahren bieten hierfür geeignete Ansätze. Ein weiterer Aspekt ist, dass das Gehör nicht alle Frequenzen einzeln wahrnimmt, sondern diese in so genannten *kritischen Bändern* zusammenfasst. Eine Möglichkeit dies zu visualisieren und zu beschreiben ist die *Bark-Skala*, die den Hörbereich in 24 einzelne Frequenzbänder aufteilt. Frequenzen innerhalb eines solchen Bandes werden vom Gehör nicht vollständig unabhängig voneinander verarbeitet. Stattdessen beeinflussen sich die enthaltenen Signalanteile gegenseitig. Die absolute Breite der kritischen Bänder nimmt mit steigender Mittenfrequenz zu. Wichtig für die Verrechnung von Rauschen mit Musik sind zudem Maskierungseffekte, die innerhalb dieser Bänder auftreten. Ein besonders lauter Ton überdeckt bzw. maskiert die restlichen Frequenzen im Frequenzband hörbar. Abbildung 2.3 zeigt dabei schematisch die Maskierung eines Tons innerhalb des Frequenzbandes.

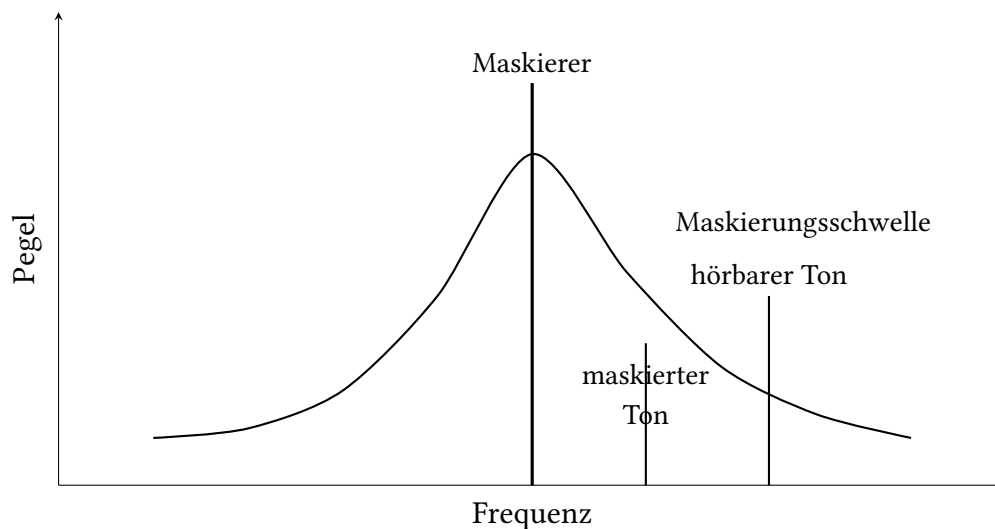


Abbildung 2.3: Schematische Darstellung der frequenzabhängigen Maskierung

Relevanz hat dies für die Verrechnung von Rauschen mit Musik, da nicht nur der reine Pegel relevant ist, sondern ebenso die Verteilung innerhalb der Frequenzbänder. Ein hoher Rauschanteil in einem bestimmten Frequenzbereich kann benachbarte Bestandteile der Musik verdecken, obwohl diese technisch weiterhin im Signal vorhanden sind.

3 Grundlagen analoger Fotografie

3.1 Aufbau fotografischen Films

3.1.1 Schichtaufbau

Ein fotografischer Film besteht aus mehreren Schichten, die alle verschiedene Aufgaben erfüllen. In Abbildung 3.1 ist der schematische Aufbau eines Films zu sehen. Am relevantesten für die Bilderzeugung ist die Fotoemulsion. Diese wird in Abschnitt 3.1.2 noch genauer besprochen. Die Schutzschicht bewahrt die Emulsion vor mechanischen Beschädigungen und die Haftschrift sorgt für die Verbindung der Emulsion mit dem Trägermaterial. In den Kindertagen der Fotografie war die Trägerschicht noch eine Glas- oder Metallplatte, mit der Erfindung des Rollfilms dann der heute bekannte flexible Kunststoff. Die Lichthofschutzschicht verhindert, dass Licht durch das Trägermaterial dringt, reflektiert wird und somit erneut auf die Emulsion trifft.

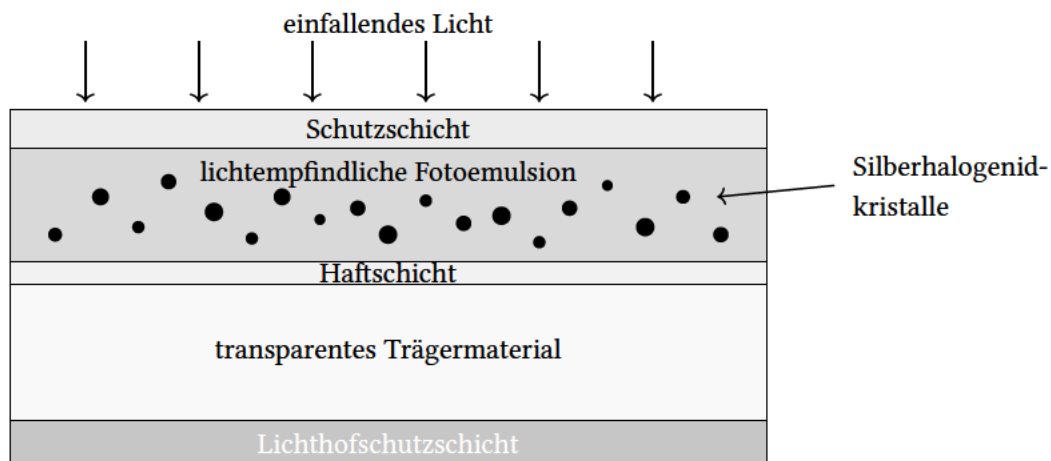


Abbildung 3.1: Schematischer Schichtaufbau eines fotografischen Schwarzweißfilms

3.1.2 Fotoemulsion

Die lichtempfindliche Fotoemulsion besteht aus Silberhalogenidkristallen, die in Gelatine gelöst sind und auf der Trägerschicht verteilt werden. Diese Kristalle reagieren bei Lichteinfall zu metallischem Silber und sind die bei naher Betrachtung sichtbaren Körner. Das Bild ist jedoch zunächst nicht fixiert und muss in einem weiteren chemischen Prozess entwickelt werden. Das beschriebene Verfahren erzeugt ein schwarz-weißes Bild.

3.2 Entstehung des latenten Bildes

Abbildung 3.2 zeigt schematisch die Entstehung des latenten Bildes. Die einzelnen Silberhalogenidkristalle bestehen aus positiv geladenen Silberionen. Treffen Photonen auf den Kristall, werden Elektronen freigesetzt, die am so genannten Empfindlichkeitszentrum mit den Silberionen zu Silberatomen reduziert werden. Diese kleinen Ansammlungen von nur wenigen Silberatomen werden Silberkeime genannt; diese enthalten bereits die Bildinformationen, sind aber durch ihre geringe Größe mit bloßem Auge nicht erkennbar. Diese Gesamtheit der Silberkeime auf dem Film wird als latentes Bild bezeichnet. Die Entwicklung verstärkt und fixiert dieses anschließend.

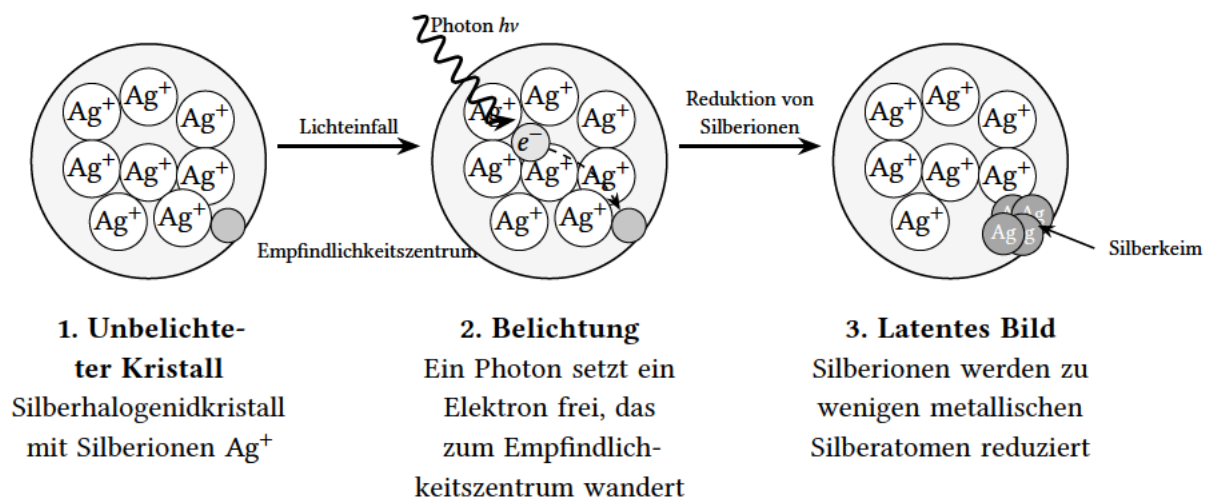


Abbildung 3.2: Schematische Darstellung der Entstehung des latenten Bildes in einem Silberhalogenidkristall. Die Darstellung ist nicht maßstabsgetreu und zeigt den chemischen Prozess stark vereinfacht.

3.3 Entwicklung des Films

Die Behandlung mit diversen Chemikalien sorgt dafür, dass die latenten Bilder auf dem Film fixiert werden und die Filmstreifen gefahrlos Licht ausgesetzt werden können. Es gibt bei der Entwicklung vier wesentliche Schritte, die mit unterschiedlichen Chemikalien durchgeführt werden:

1. Entwickler: durch diese Chemikalie werden die Silberhalogenide in Silberkristalle umgewandelt, wodurch das Bild erstmals sichtbar wird.
2. Stoppbad: Hat das Bild die gewünschte Schärfe erreicht muss der Entwicklungsprozess gestoppt werden; andernfalls würden alle Halogenide zu Silberkristallen und das Bild wäre nicht mehr erkennbar.
3. Fixieren: Hierbei werden die unbelichteten Silberhalogenid-Kristalle aus der Emulsion ausgewaschen. Dadurch werden der Film und das sich darauf befindliche Bild dauerhaft haltbar und sichtbar. Erst jetzt kann der Film normalem Licht ausgesetzt werden.
4. Wässern und Trocknen: Hierbei werden die letzten Chemikalien abgewaschen.

3.4 Scannen und Digitalisieren

Das Ergebnis ist zunächst ein Negativ, bei dem helle Objekte dunkel erscheinen und dunkle Bereiche hell. Mit den Negativen können nun vergrößerte Abzüge gemacht werden bzw. sie können gescannt werden, um sie digital weiterzuverbreiten. Die Scans, die ich angefertigt habe, haben eine Auflösung von 3600-5000 *dpi* (Pixel pro Zoll). Die Negative sind $36 * 24 \text{ mm}$ groß wodurch sich eine Auflösung von bis zu ungefähr 33,5 Megapixeln ergibt. Dies berechnet sich folgendermaßen: Zuerst muss man die Abmessungen der Negative in Zoll bzw. Inch umrechnen. 1 Zoll entspricht 2,54 *cm* oder 25,4 *mm*. Somit ergibt sich für die Breite $36 \text{ mm} = \frac{36}{25,4} \text{ in} \approx 1,417 \text{ in}$ und für die Höhe $24 \text{ mm} = \frac{24}{25,4} \text{ in} \approx 0,945 \text{ in}$. Geht man von einer Auflösung der Scans von 5000 dpi aus, ergibt sich:

$$1,417 \text{ in} * 5000 \frac{\text{Pixel}}{\text{in}} = 7085 \text{ Pixel}$$

$$0,945 \text{ in} * 5000 \frac{\text{Pixel}}{\text{in}} = 4725 \text{ Pixel}$$

$$7085 \text{ Pixel} * 4725 \text{ Pixel} \approx 33,5 \text{ Megapixel}$$

Dies ist teilweise mehr, als der eigentliche Bildinhalt hergibt, allerdings wird dadurch die Körnung sehr gut sichtbar. Je nach Lichtempfindlichkeit des Films sind die Körner größer oder kleiner.

3.5 ISO-Empfindlichkeit und Korngröße

Die ISO-Zahl gibt an, wie lichtempfindlich ein Film ist. Die Werte der DIN sind logarithmisch in Grad angegeben, wohingegen die heute allgemein gebräuchliche ASA-Zahl linear aufgebaut ist. Somit entspricht eine Erhöhung des DIN-Werts um 3° DIN einer Verdoppelung der Lichtempfindlichkeit und auch eine Verdoppelung des ASA-Werts. Ein Wert von ASA 100 entspricht 21° DIN. Die offizielle ISO-Zahl enthält beide Werte und wird auf der Verpackung eines Films beispielsweise als ISO 100/21° angegeben. Allgemein kann man sagen, dass mit steigender Lichtempfindlichkeit auch die Größe der Filmkörner wächst. Je größer die Körner, desto mehr Licht kann von jedem einzelnen eingefangen werden und somit ist weniger Lichtstärke notwendig, um die chemische Reaktion in Gang zu setzen. Diese bilden dann nach der Entwicklung stärker sichtbare Strukturen. Moderne Filmemulsionen können diesen Effekt allerdings relativieren. Es sind drei Werte für die korrekte Belichtung des Films wichtig: *Filmempfindlichkeit*, *Belichtungs-/Verschlusszeit* und *Blende*. In Abbildung 3.3 ist dieser Zusammenhang aufgezeigt. Bis auf die Filmempfindlichkeit können die Parameter vor jeder Aufnahme verändert werden. Verringert man beispielsweise die Belichtungszeit, um weniger Bewegungsunschärfe zu haben, muss man zwingend die Blende weiter öffnen, um den Film nach wie vor korrekt zu belichten.

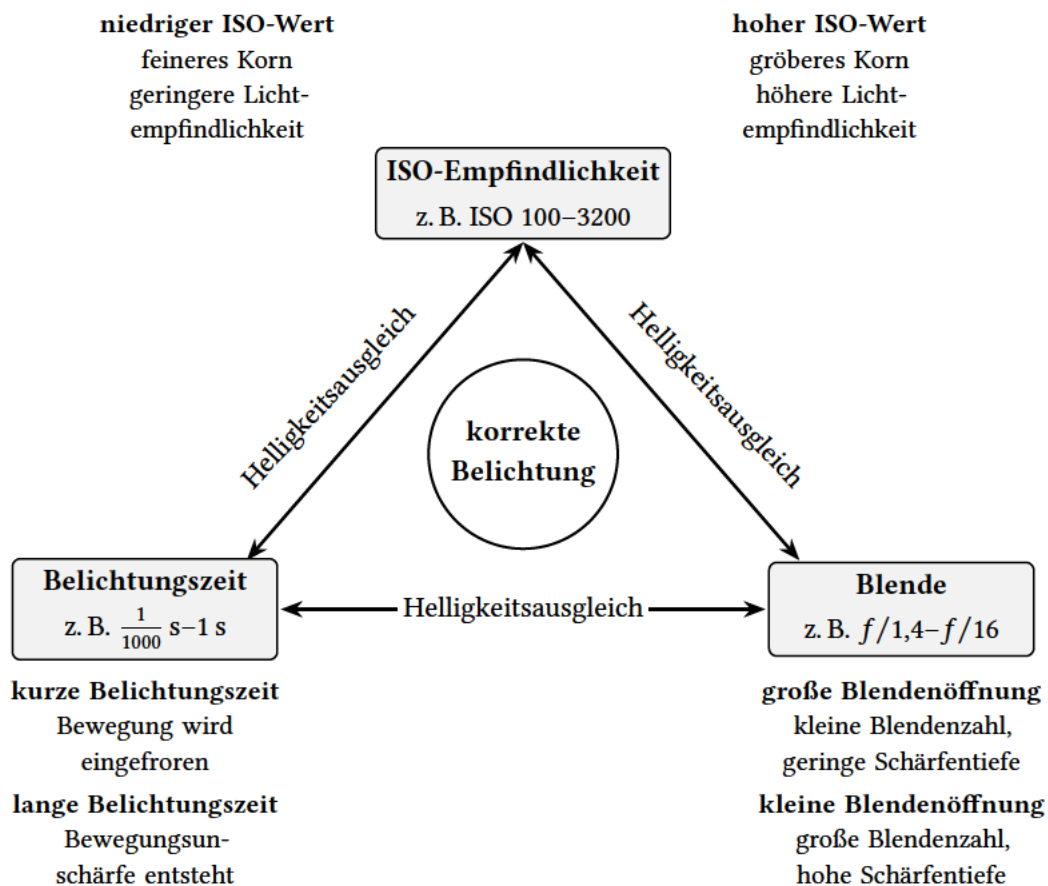


Abbildung 3.3: Zusammenhang zwischen ISO-Empfindlichkeit, Belichtungszeit und Blende im Belichtungsdreieck. Die Veränderung eines Parameters kann durch die Anpassung mindestens eines weiteren Parameters ausgeglichen werden.

Um den Zusammenhang zwischen ISO-Zahl und Korngröße zu verdeutlichen, habe ich einige Fotos mit verschiedenen Filmen aufgenommen und anschließend mit jeweils 3600 dpi gescannt. Die Bilder wurden nicht nachbearbeitet, die beiden Farbfotos in Abbildung 3.5 und 3.7 wurden allerdings auf die Helligkeiten reduziert, um einen besseren Vergleich zu den Schwarzweißfotos ziehen zu können. Zugeschnitten wurde auf eine Größe von 2635 * 2635 *Pixel*, was auf der Filmebene einer Fläche von 1,86 cm * 1,86 cm entspricht. Dieses Maß von Vergrößerung auf den folgenden Seiten ist noch deutlich im Rahmen dessen, für das diese Filme entwickelt wurden.



Abbildung 3.4: AgfaPhoto APX 100 - ISO 100/21°. Schwarzweiß-Negativfilm. Feines Korn, auch kleine Strukturen sind sehr gut erkennbar.



Abbildung 3.5: Kodak Gold 200 - ISO 200/24°. Farb-Negativfilm.



Abbildung 3.6: Ilford HP5 - ISO 400/27°. Schwarzweiß-Negativfilm. Deutlich sichtbares Korn, allerdings trotzdem noch eine gute Darstellung der Motivdetails.



Abbildung 3.7: Kodak Vision3 500T - ISO 500/28°. Farb-Negativ-Kinofilm. Verwendet als ISO 800/30°, also leicht unterbelichtet um Kornstrukturen zusätzlich sichtbar zu machen. Optimiert auf geringe Körnung trotz großer Lichtempfindlichkeit für Bewegtbild-Nachtaufnahmen



Abbildung 3.8: Ilford Delta 3200 - ISO 3200/36°. Schwarzweiß-Negativfilm. Optimiert auf Aufnahmen mit wenig Licht und damit sehr ausgeprägte Körnung

Im Vergleich zwischen all diesen verschiedenen Filmtypen ist klar zu sehen, dass mit steigender Lichtempfindlichkeit auch das Korn und das Rauschen zunimmt. Ein bisschen sticht der Film in [Abbildung 3.7](#) heraus, welcher ursprünglich nicht für Fotos sondern fürs Kino entwickelt wurde, um auf der großen Leinwand ein besonders scharfes, rauscharmes Bild zu liefern.

3.6 Unterschied zu digitalen Fotos

Der größte Unterschied in der Charakteristik von analogen und digitalen Fotos ist, dass ein analoges Foto zufallsverteilte Bildpunkte besitzt, wohingegen ein digitales Foto fest gerastert ist. Geht man nur von einem Schwarz-Weiß-Sensor aus, gibt es mehrere Millionen Fotozellen, die bei einfallendem Licht eine winzige Spannung erzeugen. Diese wird von der Kamera gemessen und sorgt dafür, dass der zugehörige Pixel den jeweiligen Helligkeitswert zugewiesen bekommt. In den Anfangstagen der Digitalfotografie hatten Sensoren noch nicht die Auflösung von Film, inzwischen haben sie diese zumindest im Profibereich weit überholt.

4 Erzeugung von Rauschen aus Filmkorn

Die Umsetzung meines Vorhabens erfolgte in zwei Schritten: zunächst die Extraktion des Filmkorns aus dem analog aufgenommenen Bild und die anschließende Umwandlung dieser Struktur in ein akustisches Signal. Zur Umsetzung habe ich Matlab benutzt, da es schon viele für mich nützliche Funktionen bereithält. Zudem werden sowohl Bild- als auch Tondateien beim Einlesen in Matrizen übersetzt. Dadurch können die bekannten Werkzeuge der linearen Algebra für die Umwandlung verwendet werden.

4.1 Extraktion des Filmkorns

Um das Filmkorn zu extrahieren habe ich mehrere Methoden untersucht, die unterschiedliche Herangehensweisen verwenden.

4.1.1 Kantendetektion

Ein möglicher Algorithmus zur Erkennung des Filmkorns ist der *Sobel-Algorithmus*, der eigentlich als Kantendetektions-Algorithmus erdacht wurde. Da bei sehr genauer Betrachtung jedes Filmkorn von Kanten umgeben ist, kann er aber auch für diesen Zweck verwendet werden. Der Sobel-Algorithmus benötigt eine Matrix aus Helligkeitswerten. Diese steht zur Verfügung nachdem die Bilddatei von Matlab eingelesen und in Graustufen umgewandelt wurde. Nun sind lediglich Helligkeitswerte vorhanden, mit denen gut gerechnet werden kann. Hier ein Beispiel für ein kleines Bild mit den Abmessungen 5x5 Pixel:

10	12	11	13	14
11	13	12	14	15
12	14	13	120	125
13	15	14	122	128
14	16	15	124	130

Nun kommen die beiden Sobel-Filter ins Spiel; zum einen der Filter für die horizontale Änderung:

$$\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$$

Zum anderen der Filter für die vertikale Änderung:

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{pmatrix}$$

Dann werden diese Filter mit dem Bild gefaltet; hierzu werden die Filter über das Bild geschoben und für jedes Pixel ein Gradient berechnet. Als Beispiel soll hier das mittlere Pixel

dienen. Wird der Ausschnitt $\begin{bmatrix} 13 & 120 & 125 \\ 14 & 122 & 128 \\ 15 & 124 & 130 \end{bmatrix}$ mit $\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}$ gefaltet ergibt dies

$$\begin{aligned} & (-1 * 13) + (0 * 120) + (1 * 125) \\ & + (-2 * 14) + (0 * 122) + (2 * 128) \\ & + (-1 * 15) + (0 * 124) + (1 * 130) \\ & = -13 + 0 + 125 \\ & \quad -28 + 0 + 256 \\ & \quad -15 + 0 + 130 \\ & = 455 = G_x \end{aligned}$$

Die Änderung in vertikale Richtung wird analog berechnet und ergibt

$$G_y = 15$$

Aus diesen beiden Werten lässt sich die Stärke der Kante berechnen. Je größer der Wert G ist, desto stärker ist sie. G berechnet sich wie folgt:

$$G = \sqrt{G_x^2 + G_y^2} = \sqrt{455^2 + 15^2} = \sqrt{207250} \approx 455,25$$

Es ist auch möglich, die Richtung bzw. den Winkel der Kante zu bestimmen. Dies ist für meinen Einsatzzweck allerdings nicht relevant. Zur finalen Kantendetektion muss zwingend noch ein Schwellenwert angegeben werden, ab dem ein berechneter Wert als Kante erkannt und gewertet wird. Die Festlegung dieses Wertes war ebenfalls Teil meines Prozesses und führte je nach Bild zu unterschiedlichen Ergebnissen. Dass durch diesen Algorithmus nicht nur Korn sondern auch Kanten erkannt werden spielt für meinen Einsatzzweck keine Rolle, da auch die Charakteristiken des Motivs zufällig sind und somit für die weitere Verarbeitung verwendet werden können. Für den Code ist Matlab gut geeignet, da man auf vorgefertigte Faltungs- und Wurzelfunktionen zurückgreifen kann.

```

1 sx = [-1 0 1; -2 0 2; -1 0 1]; %Sobel-Filter horizontal
2 sy = sx'; %Sobel-Filter vertikal
3 Gx = conv2(I, sx, 'same');
4 Gy = conv2(I, sy, 'same');
5 G = sqrt(Gx.^2 + Gy.^2);
6 korn = G > gradThreshold;

```

Der Schwellenwert `gradThreshold` wird bei der neuesten Version automatisch über ein Perzentil festgelegt. Dabei kann angegeben werden, wie viel Prozent der Kanten als Korn erkannt wird. Somit wird der Schwellenwert für jedes Bild einzeln festgelegt und hängt nicht mehr vom Bildinhalt ab. Die Funktion, die ich genutzt habe, war `prctile(A, pct)`. Für `A` wird dabei die Liste aller Gradienten angegeben und für `pct` der Wert des Perzentils. Ich habe meistens 80 angegeben, was dazu führt, dass 20% der stärksten Kanten erkannt bzw. verarbeitet werden. Abbildung 4.1 zeigt die Anwendung des Sobel-Algorithmus. Während auf der linken Seite das originale Analogfoto in Graustufen zu sehen ist, ist das Bild auf der rechten Seite binär und zeigt lediglich die Körnung. Dabei sind die Filmkörner schwarz dargestellt, die Zwischenräume weiß.

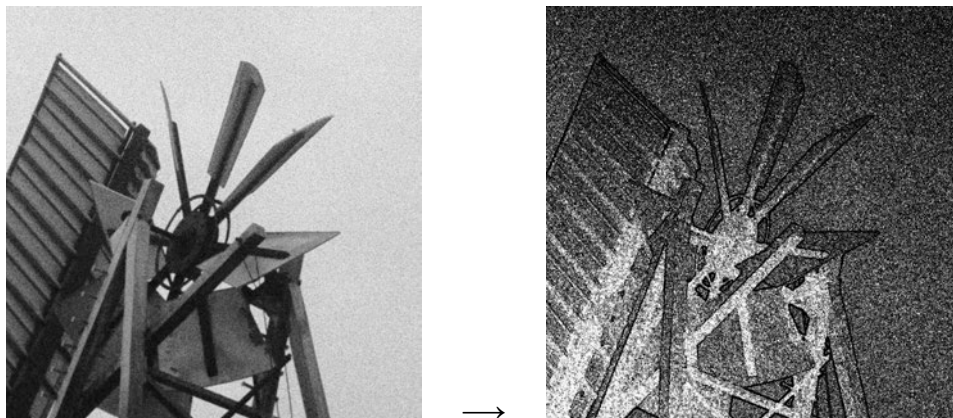


Abbildung 4.1: Kornextraktion per Sobel-Algorithmus, Grenzwert 0,15

4.1.2 Hochpassfilter

Bei dieser Methode benötigt man zunächst eine weichgezeichnete Version des zu analysierenden Bildes. Ein Weichzeichner ist auf analytischer technischer Ebene nichts anderes als ein Tiefpassfilter, der die hohen Frequenzen des Bildes entfernt. Da sich das Korn durch kleine, zufällige Strukturen auszeichnet und sich die Helligkeiten dabei schnell verändern, ist es im hohen Frequenzspektrum zu finden. Das extrahierte Korn erhält man, indem man das weichgezeichnete Bild vom Originalbild abzieht. Direkt hochpassfiltern kann man nur über eine vorherige FFT (vgl. Kapitel 4.1.4). Ein Weichzeichner kann jedoch nicht nur auf der

Frequenz-, sondern auch auf der Bildebene realisiert werden. In Abbildung 4.2 sieht man das weichgezeichnete Bild auf der linken und das extrahierte Korn auf der rechten Seite. Das Ursprungsbild ist dasselbe wie in Abbildung 4.1.

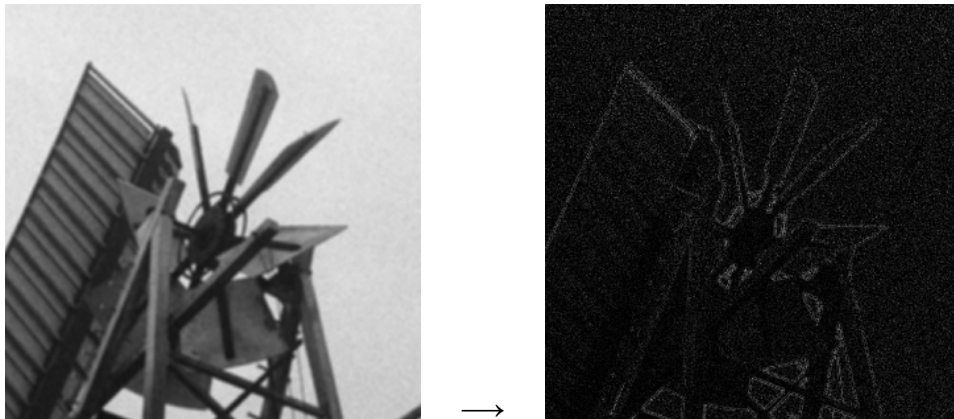


Abbildung 4.2: Kornextraktion per Hochpass-Filter, $\sigma = 8$, Kernelgröße 21

Der Programmcode für die Umsetzung dieser Methode ist weniger komplex als der im vorherigen Kapitel beschriebene, allerdings hat er beim Ausprobieren verschiedener Parameter das Korn nicht so zuverlässig extrahiert wie die in 4.1.1 beschriebene Methode. Es bleibt entweder sehr viel Bildinhalt stehen oder es entsteht nur eine schwarze Fläche. Durch die zwei vorhandenen Parameter σ und Kernelgröße ist es zudem schwierig, das Ergebnis zu justieren. Dadurch, dass der Sobel-Algorithmus nur *einen* veränderlichen Parameter hat, ist das Ergebnis leichter anzupassen.

4.1.3 Differenz der Gauß-Filter

Auch bei dieser Methode wird mit Weichzeichnern gearbeitet. Anstatt jedoch ein weichgezeichnetes Bild vom Originalbild abzuziehen, werden zwei unterschiedlich stark weichgezeichnete Bilder des Originals voneinander subtrahiert. Es gibt also einen zusätzlichen Rechenschritt. Die zugrunde liegende Formel ist $DoG = G_{\sigma_1}(I) - G_{\sigma_2}(I)$ mit $\sigma_1 < \sigma_2$. Das Beispielbild, das ich schon bei den vorherigen Methoden verwendet habe, hat zwei grundlegende Bildinhalte (Himmel und Mühle) und das Filmkorn, welches extrahiert werden soll. Die erste Weichzeichnung mit kleinem σ ist kaum vom Originalbild zu unterscheiden, lediglich die feinsten Strukturen des Korns werden etwas schlechter sichtbar. Stärkere Details werden dann bei der zweiten Weichzeichnung entfernt, sodass nur noch grobe Umrisse und Strukturen der Mühle und eine einheitliche Fläche des Himmels zurückbleiben. Nach der Subtraktion verschwinden die groben Strukturen von Mühle und Himmel und nur die feinen Strukturen von Kanten und Filmkorn bleiben zurück. Das Ergebnis entspricht im Wesentlichen dem von

Kapitel 4.1.2, daher kann hier auch Abbildung 4.2 zur Visualisierung dienen. Daraus ergibt sich auch, dass diese Methode für das, was sie leistet, zu aufwändig ist.

4.1.4 FFT-basierte Extraktion

Eine weitere Möglichkeit zur Extraktion des Filmkorns ist die Analyse des Bildes im Frequenzraum mittels einer zweidimensionalen Fourier-Transformation. Während bei den bisher vorgestellten Verfahren die Helligkeitswerte direkt im Bild ausgewertet werden, erfolgt hier zunächst eine Umwandlung in eine Frequenzdarstellung. Der Vorteil dieses Ansatzes besteht darin, dass sich großflächige Bildstrukturen und feine Details wie Filmkorn anhand ihrer Frequenzanteile voneinander trennen lassen. Die Grundlage bildet die zweidimensionale diskrete Fourier-Transformation. Für ein Bild $I(x, y)$ mit Breite M und Höhe N berechnet sich diese durch

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y) * e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

Dabei sind x und y die Koordinaten eines Pixels im Bild, u und v die räumlichen Frequenzen. Das Ergebnis der Transformation $F(u, v)$ enthält somit keine Helligkeitswerte mehr, sondern ausschließlich Frequenzen. Das Bild wird somit als Überlagerung vieler unterschiedlich großer Strukturen betrachtet. Eine hohe Frequenz bedeutet eine schnelle Änderung der Helligkeit, also hat beispielsweise der Himmel eine niedrige Ortsfrequenz, das Filmkorn eine sehr hohe. Für die folgenden Berechnungen und Darstellungen habe ich das Online-Tool [2D Inverse Fourier Transform Playground](#) vom Entwickler monman53 genutzt. In Abbildung 4.3 sieht man das Ergebnis der 2D-DFT visualisiert. Die tiefen Frequenzen befinden sich in der Mitte des Bildes, die hohen am Bildrand. Je heller die Pixel des transformierten Bildes sind, desto ausgeprägter sind die jeweiligen Frequenzen im Ursprungsbild. Das Bild ist in vier Quadranten aufgeteilt, da es sich bei den transformierten Werten um komplexe Zahlen handelt. Wie aus der Mathematik bekannt, ist auf der horizontalen Achse der Realteil und auf der vertikalen Achse der Imaginärteil aufgetragen. So erklärt sich auch die Verteilung der Frequenzbeträge um die Mitte des Bildes bzw. den Ursprung herum.

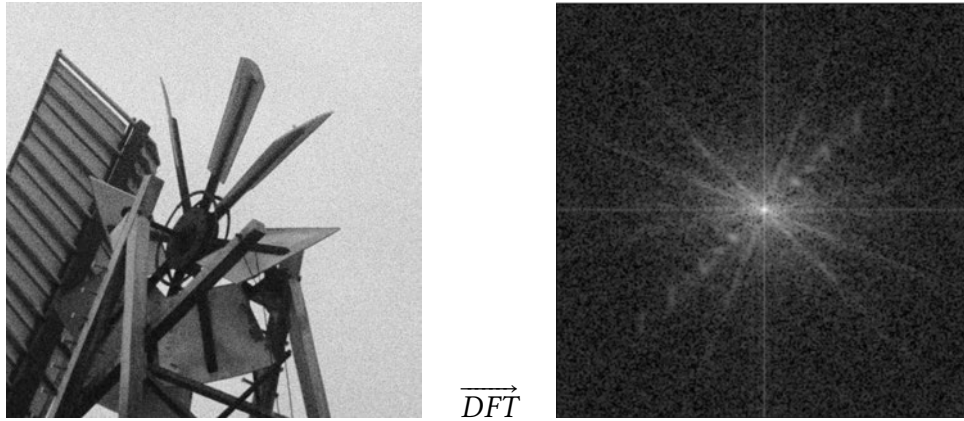


Abbildung 4.3: Ergebnis der 2D-DFT des linken Bildes

Da sich das Korn in den hohen Frequenzbereichen befindet, müssen nun die tiefen Frequenzanteile entfernt werden. Das Frequenzspektrum $F(u, v)$ multipliziert man dafür mit einer Filtermaske $H(u, v)$, also ist das hochpassgefilterte Spektrum $F_H(u, v) = F(u, v) * H(u, v)$. Für eine einfache, binäre, kreisförmige Hochpassmaske mit Radius r_0 gilt

$$H(u, v) = \begin{cases} 0, & r < r_0 \\ 1, & r \geq r_0 \end{cases}$$

mit

$$r = \sqrt{(u - u_c)^2 + (v - v_c)^2}$$

(u_c, v_c) ist dabei die Position des Zentrums vom Spektrum. Alle Frequenzanteile innerhalb der Maske werden dabei entfernt. In [Abbildung 4.4](#) sieht man links das transformierte Bild $F(u, v)$, in der Mitte die Maske $H(u, v)$ und rechts das Ergebnis der Multiplikation $F_H(u, v)$.

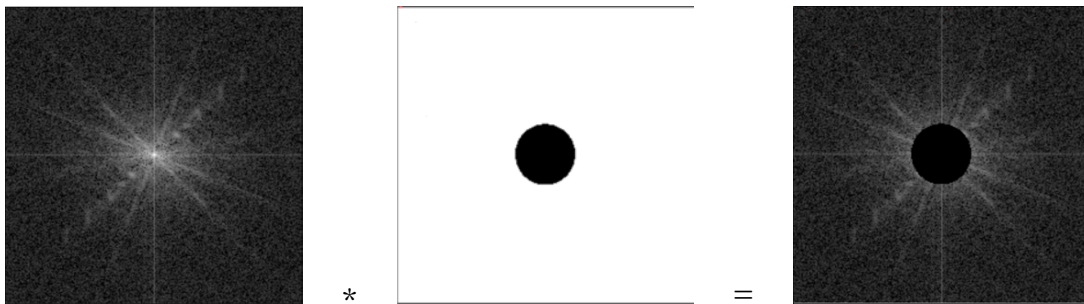


Abbildung 4.4: Multiplikation des transformierten Bildes mit der Maske

Möchte man das extrahierte Filmkorn bzw. die Teile des Bildes mit hohen Frequenzen erhalten, muss man eine inverse Transformation durchführen. Dies passiert mit der Formel

$$I_H = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_H(u, v) * e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

In Abbildung 4.5 ist das Ergebnis der Rücktransformation zu sehen. Das Bild auf der rechten Seite enthält nur noch die hohen Frequenzen.

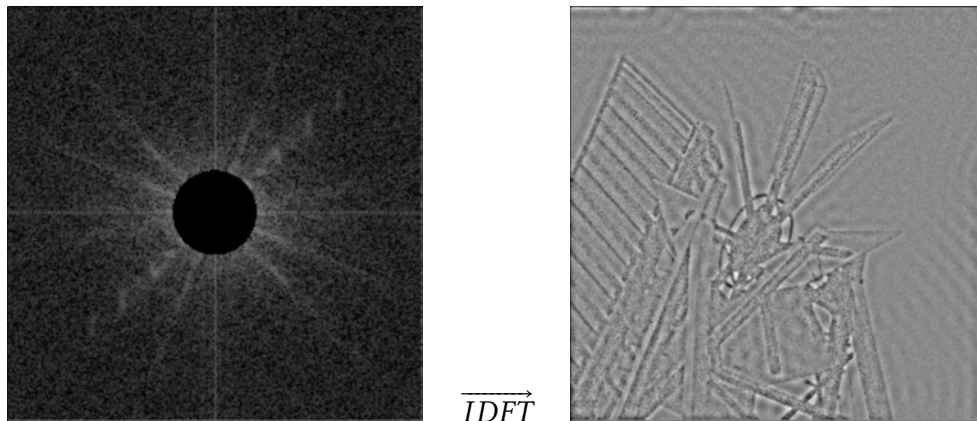


Abbildung 4.5: Ergebnis der Inversen 2D-DFT

Da bei diesem Ansatz schon direkt mit Frequenzspektren gearbeitet wird, liegt es nahe, diese auch für die weitere Klangerzeugung zu nutzen. Allerdings benötigt man bei der von mir zur Rauscherzeugung gewählten Methode zwingend ein binäres Bild bzw. eine binäre Matrix als Ausgangspunkt, die von der FFT nicht direkt geliefert wird. Es müsste also zusätzlich noch über einen Schwellenwert gefiltert bzw. ein Kantendetektionsalgorithmus angewendet werden. Daher entschied ich mich letztlich dazu, auf den Schritt der Fourier-Transformation zu verzichten und Rechenaufwand zu sparen. Die Ergebnisse per Sobel-Algorithmus sind in jedem Fall gut genug und das leichte Vorhandensein von Objektkanten zusätzlich zum Bildinhalt kann für die Erzeugung des Rauschens vernachlässigt werden. Die zusätzlichen Rechenschritte der Hin- und Rücktransformation sind also nicht notwendig.

4.2 Erzeugen des akustischen Rauschens

4.2.1 Lineares Interpolieren von Frequenzen

Als Basis für die Erzeugung des akustischen Rauschens dient die binäre Matrix bzw. Bilddatei des extrahierten Korns (vgl. Abbildung 4.1). In Blöcken wird die Menge des Korns ermittelt und anhand dieser werden Frequenzen ausgespielt. Dabei wird ein quadratischer Block pixelweise über die binäre Matrix geschoben und an jedem Punkt die Anzahl schwarzer und weißer Pixel im Block gezählt. Ein Beispiel für einen dieser Blöcke ist in Abbildung 4.6 zu sehen.

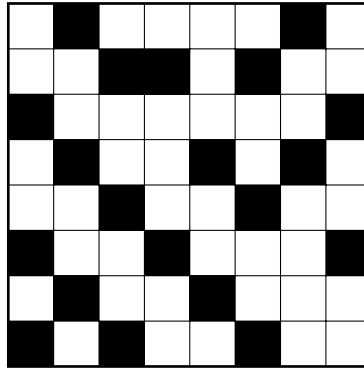


Abbildung 4.6: Abgetasteter Block

In einer neuen zweispaltigen Matrix wird die Anzahl schwarzer und weißer Pixel gespeichert. Bei jedem Verschieben des Blocks um einen Pixel entsteht eine neue Zeile mit den entsprechenden Werten. Auf Grundlage dieser Matrix wird nun eine Interpolation durchgeführt. In meiner ersten Version (`imageToNoise.m`) habe ich diesen Prozess folgendermaßen umgesetzt:

```

25 for y = 1:(h - blockSize)
26     for x = 1:(w - blockSize)
27         y2 = min(y+blockSize-1, h);
28         x2 = min(x+blockSize-1, w);
29         block = korn(y:y2, x:x2);
30         % Zählt die schwarzen Pixel pro Block:
31         counts(y,2) = histcounts(block(:), [-0.5 0.5]);
32         % Zählt die weißen Pixel pro Block
33         counts(y,1) = (blockSize^2)-counts(y,2);
34     end
35 end

```

h und w sind dabei Höhe und Breite des abzutastenden Korns, $blockSize$ die Kantenlänge des abtastenden Blocks. Die entstehende Liste bzw. Matrix beginnt in diesem Beispiel folgendermaßen:

$$\begin{pmatrix} 44 & 20 \\ \vdots & \vdots \end{pmatrix}$$

In Abbildung 4.7 ist die Verteilung für den in Abbildung 4.6 gezeigten Block visualisiert. Sie entspricht der ersten Zeile der obigen Matrix.

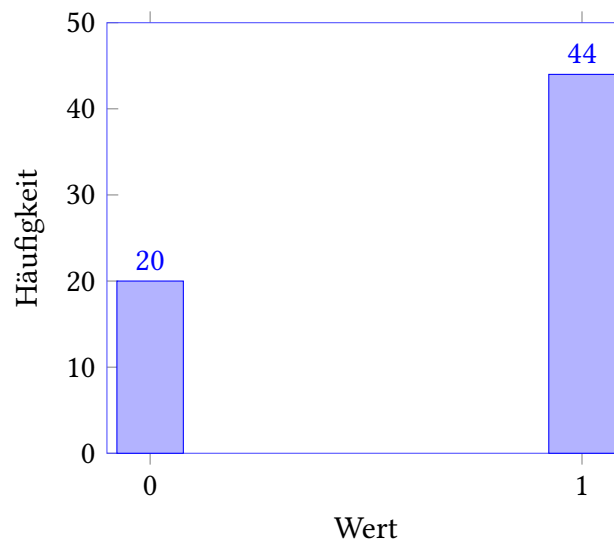


Abbildung 4.7: Gezählte Verteilung des Korns im ersten Block

Würde ich nur diese beiden Werte verwenden, gäbe es auch nur zwei Frequenzen, die ausgespielt werden. Für ein realistisches Rauschen werden allerdings deutlich mehr Frequenzen benötigt. Um diese zu erhalten, wird dem bisherigen Wert 0 der Wert 20 Hz zugewiesen und dem Wert 1 der Wert 20 kHz. Anschließend wird eine Geradengleichung erstellt, die die beiden Werte verbindet und dazwischen in regelmäßigen Abständen die Funktionswerte für die jeweiligen Frequenzen berechnet. Die Frequenzachse kann dabei entweder logarithmisch (vgl. Abbildung 4.8) oder linear (vgl. Abbildung 4.9) skaliert werden. In der ersten Version habe ich mit der `interp1`-Funktion in Matlab gearbeitet, die eine frei wählbare Anzahl an Zwischenwerten berechnen kann. Alle hier visualisierten Frequenzen werden hierbei für die Dauer eines Audioframes gleichzeitig ausgespielt. Die ausgespielten Frequenzen sind, sofern die selbe Anzahl gewählt wurde, immer die selben. Der Klangcharakter der einzelnen Frames unterscheidet sich nur durch die Lautstärke bzw. den Pegel der Einzelfrequenzen. In dieser Hinsicht entspricht das System fast dem Erstellen einer linearen Filterkurve. Es sind reine Sinusschwingungen, die übereinandergelegt werden.

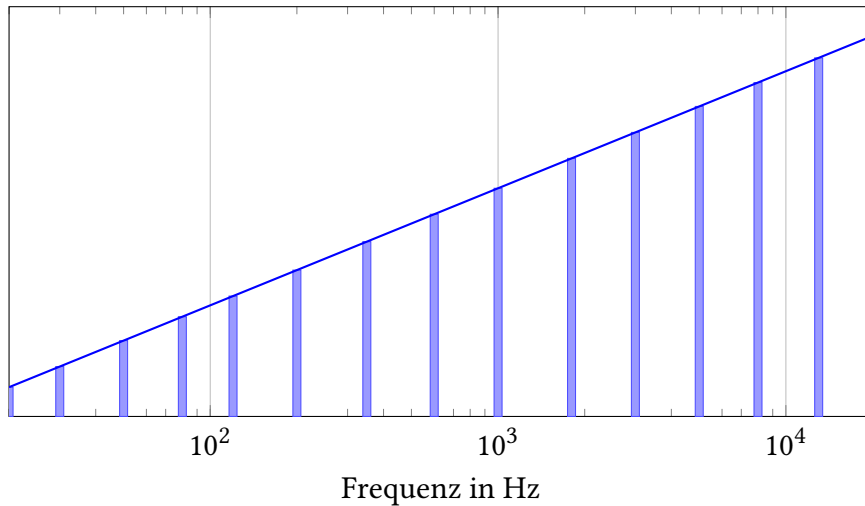


Abbildung 4.8: Interpolierte Werte mit logarithmischer Verteilung

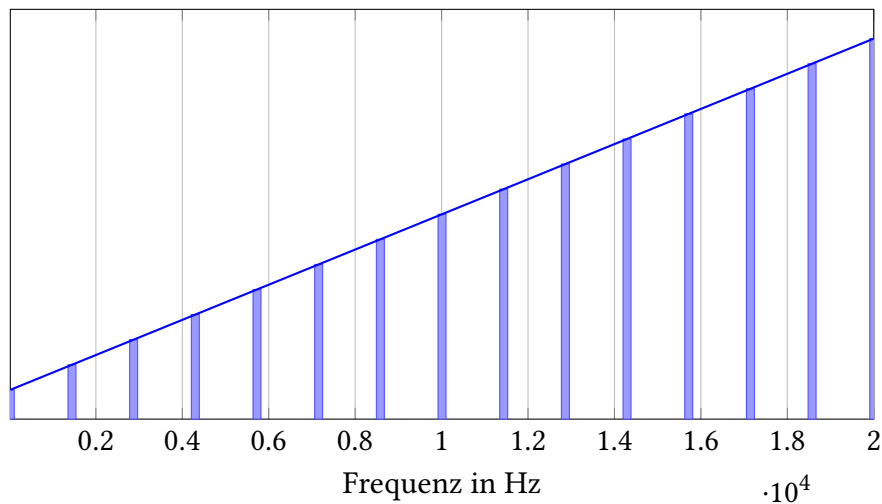


Abbildung 4.9: Interpolierte Werte mit linearer Verteilung

Obwohl durch die hier beschriebene Methode die Anzahl der Frequenzen deutlich erhöht wurden, stachen bei ersten Tests mit bis zu 100 Zwischenwerten einzelne Frequenzen immer noch klar hervor. Der akustische Eindruck entsprach dabei eher dem eines Klingelns und nicht dem eines Rauschens. Dieses Problem war ab einer Anzahl von 1000 behoben, allerdings klang es noch etwas steril und künstlich. Außerdem war in regelmäßigen Abständen zwischen den Frames ein Knacken zu hören.

4.2.2 Verwenden der Kornverteilung als Spektrogramm

Aufgrund der beim vorherigen Ansatz aufgetretenen Probleme beschloss ich, noch eine andere Methode zu verfolgen. Die zugrunde liegenden Daten waren erneut die abgetasteten Blöcke mit dem Korn wie in Beispiel 4.6. Allerdings wurde die Anzahl der Körner in einer neuen Matrix mit den Abmessungen (Bildhöhe minus Blockhöhe) * (Bildbreite minus Blockbreite) abgespeichert. Durch die Normierung auf Werte zwischen 0 und 1 stellt anschließend jeder Eintrag der Matrix den prozentualen Anteil des Korns in den abgetasteten Blöcken dar. Hier ein stark verkleinertes Beispiel: Nach dem Zählen entsteht die folgende Matrix:

$$\begin{bmatrix} 0,2 & 0,5 & 0,7 & 0,1 \\ 0,4 & 0,5 & 0,9 & 0,6 \\ 0,4 & 0,0 & 0,5 & 0,3 \\ 0,1 & 0,8 & 0,2 & 0,4 \end{bmatrix}$$

Visualisiert man diese Matrix durch Helligkeitswerte, hat sie schon viel von einem Spektrogramm - es fehlen nur noch die Zeit- und Frequenzachse. In Abbildung 4.10 ist zu sehen, dass diese Verteilung sehr zufällig ist. Das so entstehende Rauschen gibt daher die chaotische und zufällige Verteilung des Korns deutlich besser wieder. In der Bilddarstellung entspricht 0 Schwarz und 1 Weiß.

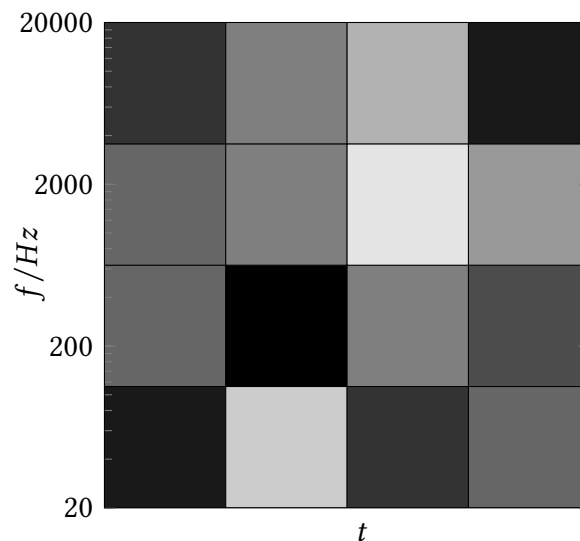


Abbildung 4.10: Matrix als Spektrogramm visualisiert

Die vertikale Auflösung des Bilds legt hierbei die maximal mögliche Anzahl der generierten Frequenzen fest. Die horizontale Auflösung des Bildes legt die maximal mögliche Dauer des Rauschens fest. Bei einem Bild von einer horizontalen Auflösung von 7000 Pixeln ergibt sich bei einer Sample- / Framedauer von 10ms eine Dauer von 70s Rauschen. Ein grundlegender

Unterschied dieser Methode ist somit, dass nicht ein Block pro Frame die Frequenzen festlegt sondern viele Blöcke kombiniert. Die ausgespielten Frequenzen sind nach wie vor festgelegt, allerdings ist die Frequenzverteilung nicht mehr linear sondern zufällig und hat somit einen realistischeren Charakter. Dies ist für die meisten Zwecke nicht ausreichend lang. Es gibt zwei Ansätze, um die Dauer des Rauschens auf eine gewünschte Dauer zu verlängern - entweder man wiederholt das Audio so oft wie es die Dauer verlangt oder man interpoliert auf der Zeitachse zwischen den Frames um das Ergebnis zu strecken. Hat ein Bild beispielsweise in der Mitte eine extrem verrauschte Stelle, ist dieses Rauschen immer bei der Hälfte der Zeit zu hören.

4.2.3 Erzeugen und Mischen der Frequenzen in Matlab

Als Basis diene in beiden Fällen eine Tabelle, in der die verschiedenen Sinusschwingungen mit unterschiedlicher Frequenz eingetragen wurden.

```
1 freqs = logspace(log10(20), log10(20000), numFreqBands);
2 ...
3 t = (0:samplesPerFrame-1) / fs;
4 sinTable = sin(2*pi*freqs(:)*t);
5 audio = zeros(1,numFrames*samplesPerFrame);
```

Hierbei stellen `freqs` die ausgespielten Frequenzen dar. Sie werden durch die `logspace`-Funktion zwischen 20 und 20000 logarithmisch gleichmäßig verteilt. `numFreqBands` ist dabei die Anzahl der Frequenzen, die ausgespielt werden. Diese kann bei dem in Abschnitt 4.2.1 beschriebenen Verfahren frei gewählt werden. Wird wie in Abschnitt 4.2.2 das Korn als Spektrogramm genutzt, legt die vertikale Auflösung deren Anzahl fest. `fs` ist die gewünschte Abtastrate des abgespeicherten Audios, `samplesPerFrame` berechnet sich durch die gewählte Dauer eines Frames multipliziert mit der Abtastrate. In Zeile 5 wird der Audiostream bereits im Speicher angelegt um bei den folgenden Schritten Ressourcen zu sparen. Das Programm war unter anderem durch diese Optimierung um Faktor 60 schneller als meine erste Version, bei der innerhalb einer Schleife der Audiostream jeweils einmal neu angelegt und verlängert wurde.

```
1 for k = 1:numFrames
2     % Lautstärken aller Frequenzen dieses Zeitpunkts:
3     amplitudes = spectrum(:,k);
4     % Frequenzen mischen:
5     frame = amplitudes.' * sinTable;
6     % Übersteuerung verhindern:
7     frame = tanh(frame);
```

```

8     idx = (k-1)*samplesPerFrame + (1:samplesPerFrame);
9     audio(idx) = frame;
10 end
11 %Normieren des Audios um Clipping zu verhindern:
12 audio = audio / max(abs(audio));

```

Im obigen Codeabschnitt der in 4.2.2 beschriebenen Methode ist zu sehen, wie die Frequenzen gemischt werden. Das Bild bzw. Spektrum wird entlang der x-Achse per Schleife durchlaufen. numFrames ist dabei die Breite des Bildes. Die Matrix spectrum enthält die normierten Helligkeitswerte (vgl. Abbildung 4.10). Für jeden Schleifendurchlauf wird dem Vektor amplitudes die jeweilige Zeile von spectrum zugewiesen und anschließend mit den Sinus-Werten des jeweiligen Zeitpunkts verrechnet.

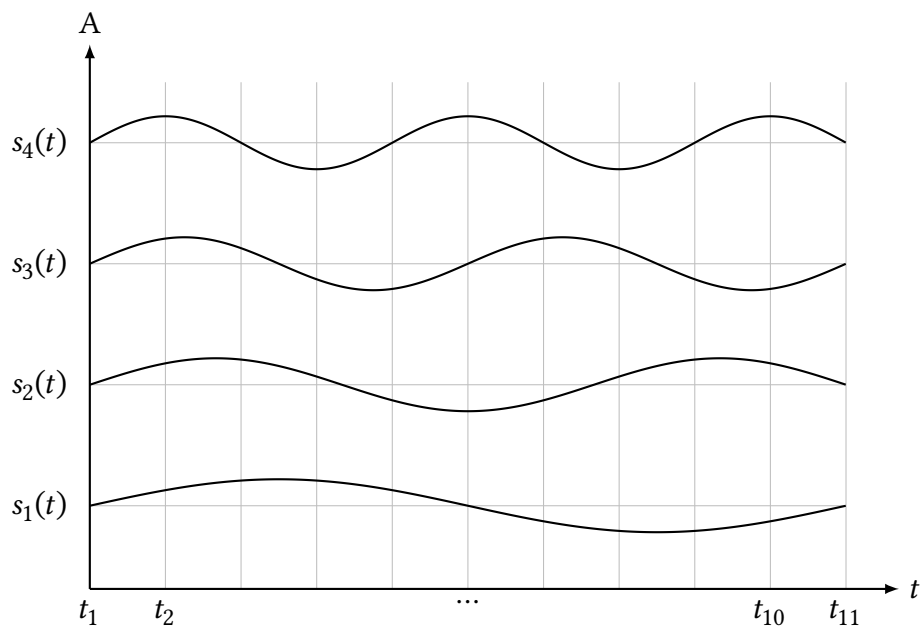


Abbildung 4.11: Schematische Darstellung der sinTable

Die Matrix sinTable funktioniert folgendermaßen: Sei numFreqBands = 4. In Abbildung 4.11 sind die Sinuskurven zu sehen, mit denen später gerechnet wird. Jede vertikale Linie des Rasters beschreibt dabei einen Sample-Zeitpunkt. Beim oberen Beispiel sind es insgesamt 11 Samples, die am Schluss pro Frame ausgespielt werden. Jede Sinusschwingung hat eine Amplitude von 1, sie sind nur zur besseren Übersichtlichkeit übereinander angeordnet. Die Frequenz nimmt nach oben hin zu. Jede Schwingung startet zum Zeitpunkt t_1 bei der Amplitude 0. In der Matrix wird dann zu jedem gerasterten Zeitpunkt die Amplitude der jeweiligen Sinusschwingung eingetragen. Zur vereinfachten Darstellung wird zum Beispiel $s_1(t_1)$ als s_{11}

dargestellt. Für das obere Beispiel würde die sinTable folgendermaßen aussehen:

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{111} \\ s_{21} & s_{22} & \cdots & s_{211} \\ s_{31} & s_{32} & \cdots & s_{311} \\ s_{41} & s_{42} & \cdots & s_{411} \end{bmatrix} = \begin{bmatrix} 0 & 0,588 & 0,951 & \cdots & -0,951 & -0,588 & 0 \\ 0 & 0,809 & 0,951 & \cdots & 0,951 & 0,809 & 0 \\ 0 & 0,951 & 0,588 & \cdots & -0,588 & -0,951 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

Zur besseren Beschreibung und zum Weiterrechnen wurde die Matrix vertikal gespiegelt, sodass die Werte von $s_1(t)$ in der obersten Zeile stehen und die von $s_4(t)$ in der untersten. Nun kommt der Vektor `amplitudes` ins Spiel. Sei $k=1$, wir betrachten also den ersten Frame, der zur Verfügung steht.

$$A_1 = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0,5 \\ 0,25 \\ 1 \end{pmatrix}$$

Mit dem Code `frame = amplitudes.' * sinTable;` passiert das eigentliche Mischen. Beim Verrechnen entsteht der folgende Vektor

$$frame = [fr_1 \quad fr_2 \quad fr_3 \quad \cdots \quad fr_{10}]$$

Die einzelnen Einträge berechnen sich dann wie folgt

$$\begin{aligned} fr_1 &= a_1 * s_{11} + a_2 * s_{21} + a_3 * s_{31} + a_4 * s_{41} \\ fr_2 &= a_1 * s_{12} + a_2 * s_{22} + a_3 * s_{32} + a_4 * s_{42} \\ &\vdots \\ fr_{11} &= a_1 * s_{111} + a_2 * s_{211} + a_3 * s_{311} + a_4 * s_{411} \end{aligned}$$

Diese Version hatte jedoch noch das Problem, dass die Phase der Schwingungen bei jedem Frame von 0 aus läuft. Dadurch hörten sich die einzelnen Frames immer sehr ähnlich an und auch bei der Verwendung von verschiedenen Bildern gab es keine sehr großen Unterschiede, da sobald ein bisschen Varianz vorhanden war, wieder auf 0 zurückgesetzt wurde. In [Abbildung 4.12](#) ist dies schematisch für 5 Frames und einer Einzelfrequenz zu sehen.

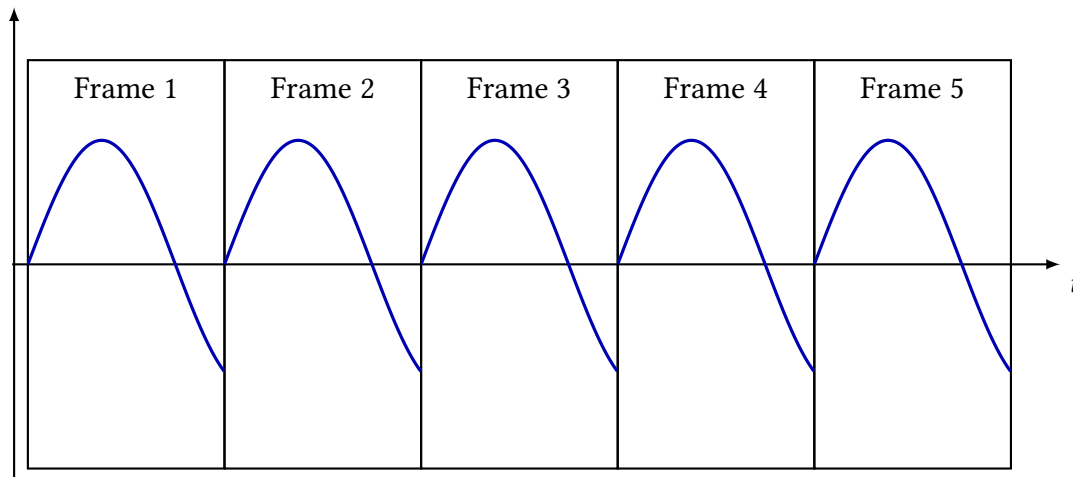


Abbildung 4.12: Schematische Anordnung der Frames mit Phasenreset in jedem Frame und gleichbleibender Amplitude

Im Realfall ist natürlich die Amplitude nicht in jedem Frame identisch, sondern wird durch die Multiplikation mit den Amplitudenvektoren angepasst. Trotzdem bleibt, wie in [Abbildung 4.13](#) zu sehen, das Problem mit den ähnlichen Klangcharakteristiken bestehen.

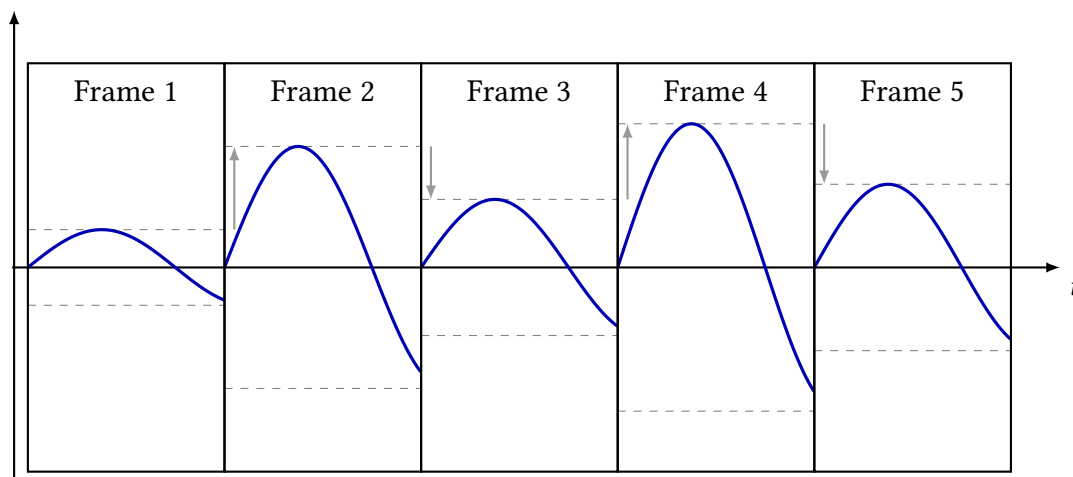


Abbildung 4.13: Schematische Anordnung der Frames mit Phasenreset in jedem Frame und wechselnder Amplitude

Werden wie in der echten Anwendung tausende Einzelfrequenzen miteinander verrechnet ist das Problem auch sehr auffällig. Zudem kommt es zu dem Phänomen, dass sich einzelne Frequenzen auslöschen können. Durchläuft eine Schwingung innerhalb eines Frames eine oder mehrere vollständige Perioden wird diese durch die Addition der positiven und negativen Halbwelle ausgelöscht. Da dies immer bei Vielfachen der selben Frequenz passiert entsteht eine Art „Fächer“ in der Frequenzdarstellung (vgl. [Abbildung 5.1](#)). Durch diese ersten

Frequenzanalysen in Abschnitt 5.1.1 ist mir dieser Fehler überhaupt erst aufgefallen. Behoben habe ich dieses Problem durch die Einführung einer globalen Zeit und das Berechnen einer neuen Sinus-Tabelle für jeden einzelnen Frame, die die Phasenverschiebung mit einberechnet. Der Rechenaufwand stieg ungefähr auf die sechsfache Zeit, dies ist allerdings verkraftbar und kann durch Optimierung des Codes sicherlich behoben werden. Hier ist der veränderte Codeblock:

```
1 audio = zeros(1, numFrames * samplesPerFrame);
2 for k = 1:numFrames
3     % Amplituden für den aktuellen Bildausschnitt
4     amplitudes = spectrum(:,k);
5     % Globale Sample-Indizes des aktuellen Frames
6     idx = (k-1)*samplesPerFrame + (1:samplesPerFrame);
7     % Globale Zeit: Phase läuft über Framegrenzen hinweg weiter
8     tFrame = (idx-1) / fs;
9     % Sinusmatrix für den aktuellen Zeitpunkt
10    sinTable = sin(2*pi*freqs(:)*tFrame);
11    % Frequenzbänder entsprechend der aktuellen Spalte mischen
12    frame = amplitudes.' * sinTable;
13    % Auf Anzahl aktiver Frequenzbänder normieren
14    frame = frame / max(sum(amplitudes), eps);
15    audio(idx) = frame;
16 end
```

Die ersten Zeilen dieses Code-Abschnittes sind identisch zur vorherigen Version. Auch Zeile 6 ist im vorherigen Code genau so zu finden, jedoch spielt idx hier noch eine weitere wichtige Rolle. idx wird genutzt um zusammen mit der Abtastrate den Samplezeitpunkt in Relation zur Gesamtzeit des Clips festzulegen. Dieser Vektor wird als tFrame abgespeichert und dient als Basis für die neue sinTable, die sich nun an den richtigen Zeitpunkten orientiert und die Phasenverschiebungen mitbeachtet. Es wäre auch möglich, eine große Matrix von Sinus-Werten über den gesamten Zeitraum bereits vor der Schleife anzulegen und dann innerhalb der Schleife nur auf Teile dieser Matrix zuzugreifen, allerdings ist der Speicherbedarf dabei viel zu hoch, da die Tabelle nicht nur 441 Spalten hätte, sondern schon bei einer Dauer des Rauschens von 1s 44100. So entstehen selbst bei wenigen Sekunden erzeugten Rauschens bereits mehrere Millionen neue Einträge. Dies ist selbst bei reinen Fließkommazahlen zu viel für eine effiziente Ausführung. Der Rest des Codes funktioniert analog zum vorherigen Beispiel. In Abbildung 4.14 ist diese Funktionsweise für eine Frequenz bei konstanter Amplitude, in Abbildung 4.15 bei wechselnder Amplitude aufgezeigt.

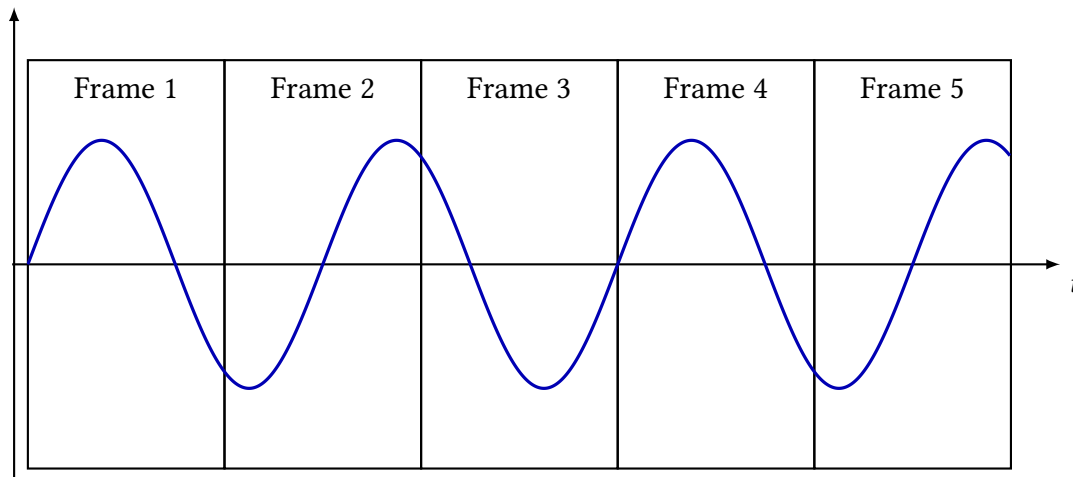


Abbildung 4.14: Schematische Anordnung der Frames mit kontinuierlicher Phase und konstanter Amplitude

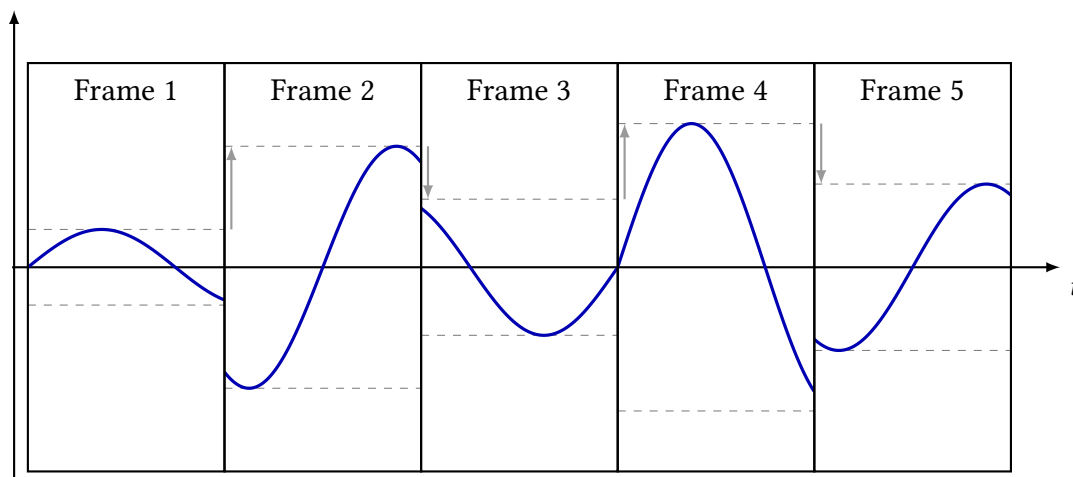


Abbildung 4.15: Schematische Anordnung der Frames mit kontinuierlicher Phase aber wechselnder Amplitude.

Durch diese Vorgehensweise gibt es zwar immer noch den Fall, dass sich einzelne Frequenzen auslöschen können, jedoch nie über die gesamte Dauer der ausgespielten Datei, sodass sowohl der Klangeindruck als auch das Frequenzhistogramm viel mehr dem eines breitbandigen Rauschens entspricht.

4.3 Verrechnen des Rauschens mit Musik

Liegt ein Rauschsignal und ein Musiksignal vor, so können diese auf verschiedene einfache Weisen in ein neues gemeinsames Audiosignal überführt werden. Diese sind:

1. Addieren: Das Rausch- und Musiksinal werden überlagert. Hierbei sind die beiden Anteile separat wahrzunehmen. Werden die Gewichte zwischen den zwei Signalen entsprechend gewählt, entsteht eine Art „Bandrauschen“, das im Hintergrund vorhanden ist und keinen direkten Einfluss auf das Nutzsignal hat.
2. Multiplizieren: Hierbei wird direkter Einfluss auf die Amplitude des Nutzsignals genommen. Das Rauschen ist hierbei nicht mehr unbedingt als ein solches wahrzunehmen, sondern beeinflusst die Aussteuerung gewisser Frequenzen des Nutzsignals. Ein Nachteil ist, dass es zur kompletten Auslöschung der Frequenzen kommt, die nicht in dem Rauschsignal enthalten sind.
3. Addieren von dem Produkt der beiden Signale zum Nutzsignal: Sei S_r das Rausch- und S_n das Nutzsignal. Dann berechnet sich das Resultat durch $S_n + (S_r * S_n)$. Dies hat den Vorteil, dass sowohl alle Frequenzanteile in der Musik erhalten bleiben als auch dass das Rauschen direkten Einfluss auf das Musiksinal hat.

Der folgende Code setzt alle drei Methoden um und speichert die Ergebnisse in separaten Dateien ab.

```

1      % Einlesen der Signale:
2      [rauschen, fsNoise] = audioread(noise);
3      [musik, fsMusic] = audioread(music);
4      % Abtastrate anpassen falls sie nicht identisch ist:
5      if fsNoise ~= fsMusic
6          musik = resample(musik, fsNoise, fsMusic);
7      end
8      musicValues = musik(:);
9      noiseValues = rauschen(:);
10     % Länge anpassen:
11     minLen = min(length(noiseValues), length(musicValues));
12     noiseValues = noiseValues(1:minLen);
13     musicValues = musicValues(1:minLen);
14     % Berechnen der neuen Signalkombinationen:
15     combinedSignalMult = musicValues .* (noiseValues/4);
16     combinedSignalAdd = musicValues + (noiseValues/4);
17     combinedSignalMultAdd = musicValues + combinedSignalMult;
18     % Normieren der Signale um Clipping zu verhindern:
19     combinedSignalAdd = combinedSignalAdd / max(abs(combinedSignalAdd));
20     combinedSignalMult = combinedSignalMult / max(abs(combinedSignalMult));
21     combinedSignalMultAdd = combinedSignalMultAdd / max(abs(combinedSignalMultAdd));
22     % Anschließend abspeichern

```

5 Fazit und Auswertung

5.1 Vergleich zu konventionell erzeugtem Rauschen

Es gibt mehrere Methoden, um mein Verfahren mit dem konventionell erzeugtem Rauschen zu vergleichen. Zuerst tat ich dies mit einer Frequenzanalyse. Diese bietet die Möglichkeit, gewisse Parameter auf einer technischen, objektiven Ebene zu vergleichen und zu bewerten. Die Untersuchung der subjektiven Wahrnehmung hätte den Rahmen dieser Arbeit gesprengt, allerdings habe ich in Abschnitt [5.1.2](#) eine Befragungs- und Bewertungsmethode entwickelt, die als weiterführende, gegebenenfalls auch interdisziplinäre Untersuchung durchgeführt werden könnte.

5.1.1 Frequenzanalyse

In Abbildung [2.1](#) ist zu sehen, wie die Leistungsdichte und somit auch das Spektrum von weißem Rauschen aussieht. Für den Vergleich mit dem von mir generierten Rauschen habe ich hauptsächlich mit Histogrammen gearbeitet um die Frequenzspektren zu analysieren. Abbildung [5.1](#) zeigt das Spektrum, das aus dem linken Bild entstanden ist. Dabei zeigt dies nicht den zeitlichen Verlauf sondern die Frequenzen über die gesamte Dauer des Rauschens. Am Histogramm kann man außerdem ablesen, wie verrauscht das Bild in den horizontalen Abschnitten ist. In diesem Fall ist das Rauschen sehr gleichmäßig verteilt. Dies spricht für den Algorithmus, da das Ursprungsbild zwei sehr unterschiedlich helle Bereiche hat und trotzdem kein klarer Unterschied der Amplituden ab ungefähr 1 kHz sichtbar ist. Es wird also tatsächlich unabhängig vom Bildinhalt das Filmkorn für die Rauscherzeugung genutzt.

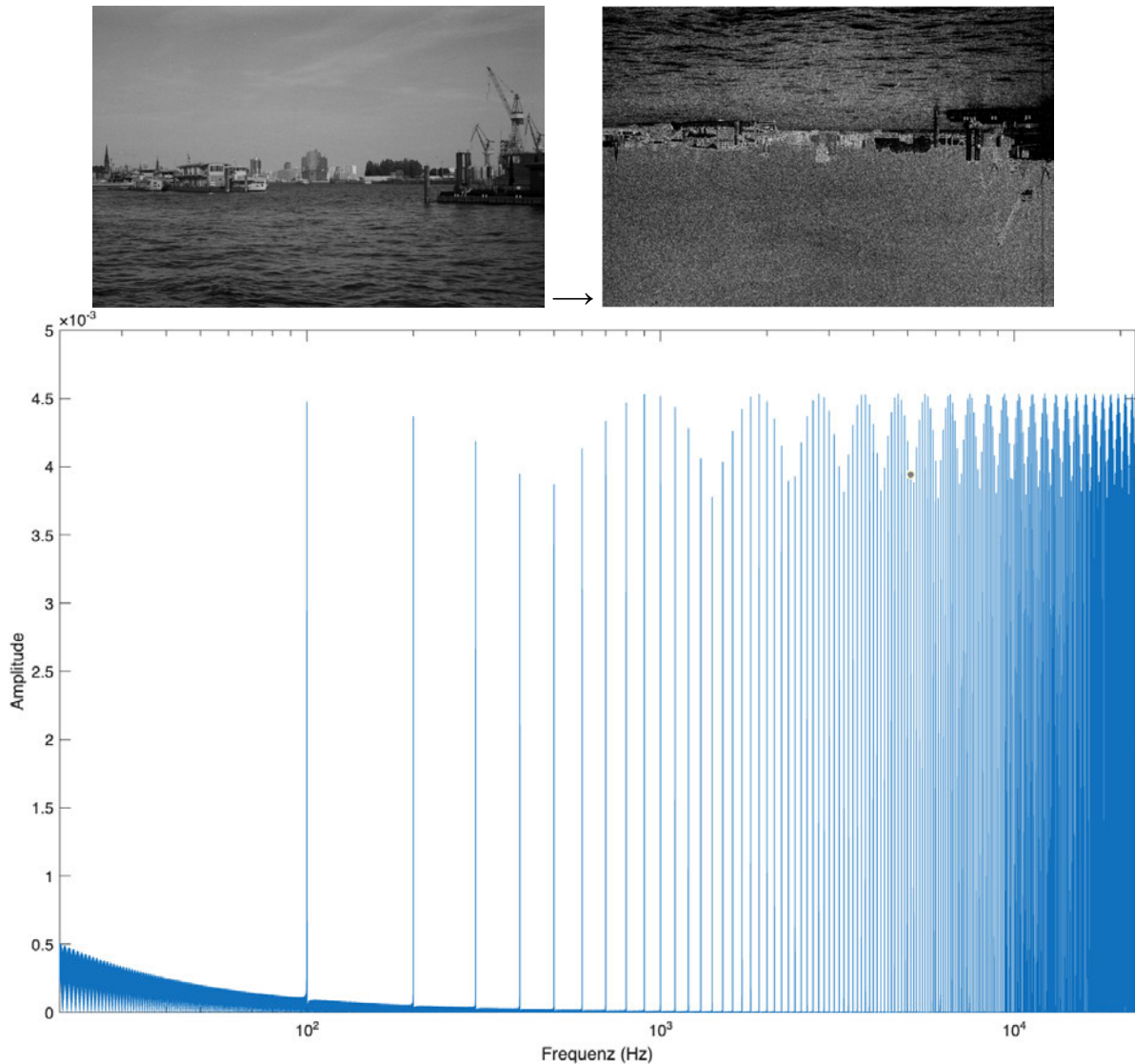


Abbildung 5.1: Extrahiertes Korn und Frequenzanalyse des mit der in Kapitel 4.2.2 beschriebenen Methode aus dem linken oberen Bild erzeugten Rauschens

Das oben rechts visualisierte Rauschen ist vertikal gespiegelt, da von Matlab aus die Frequenzen von oben nach unten generiert werden, ich allerdings gerne das Korn in den unteren Bildabschnitten für die Generierung der tiefen Frequenzen nutzen wollte. Dies geschieht mit dem Befehl `blockSums = flipud(blockSums)`.

Ich habe den Algorithmus auch getestet mit einem Bild, welches nur ein Gauß-verteilter Rauschen enthält. In [Abbildung 5.2](#) ist dieser Prozess zu sehen. Bei diesem Vergleich stößt die Histogramm-Darstellung an ihre Grenzen, da die beiden Abbildungen und somit die Frequenzverteilung nahezu identisch sind, sich das Audio aber trotzdem anders anhört. Eine Sache kann man hier allerdings sehen: die große Ähnlichkeit deutet darauf hin, dass das Korn in [Abbildung 5.1](#) sehr gleichmäßig verteilt ist, was auf einer molekularen, chemischen Ebene

auch Sinn ergibt.

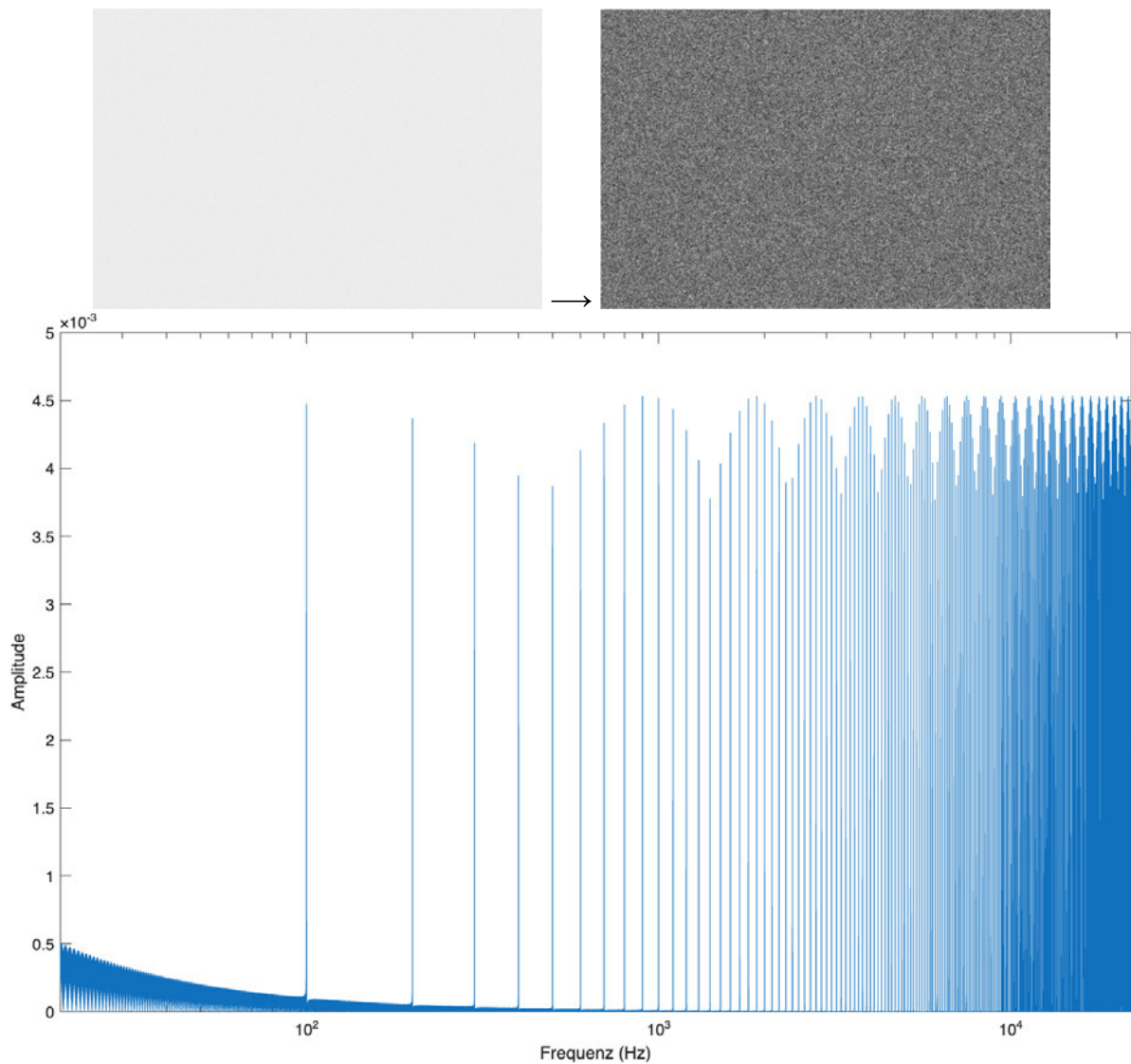


Abbildung 5.2: Künstlich erzeugtes Bild von Gaußischem Rauschen mit anschließender Frequenzanalyse

Bei erneuter Betrachtung dieser Frequenzdiagramme und meines Programmcodes ist mir der entscheidende Fehler aufgefallen, den ich bereits am Ende von Kapitel [4.2.3](#) beschrieben habe. Mit dem aktualisierten Programmcode ergibt sich das Frequenzspektrum, welches in [Abbildung 5.3](#) zu sehen ist.

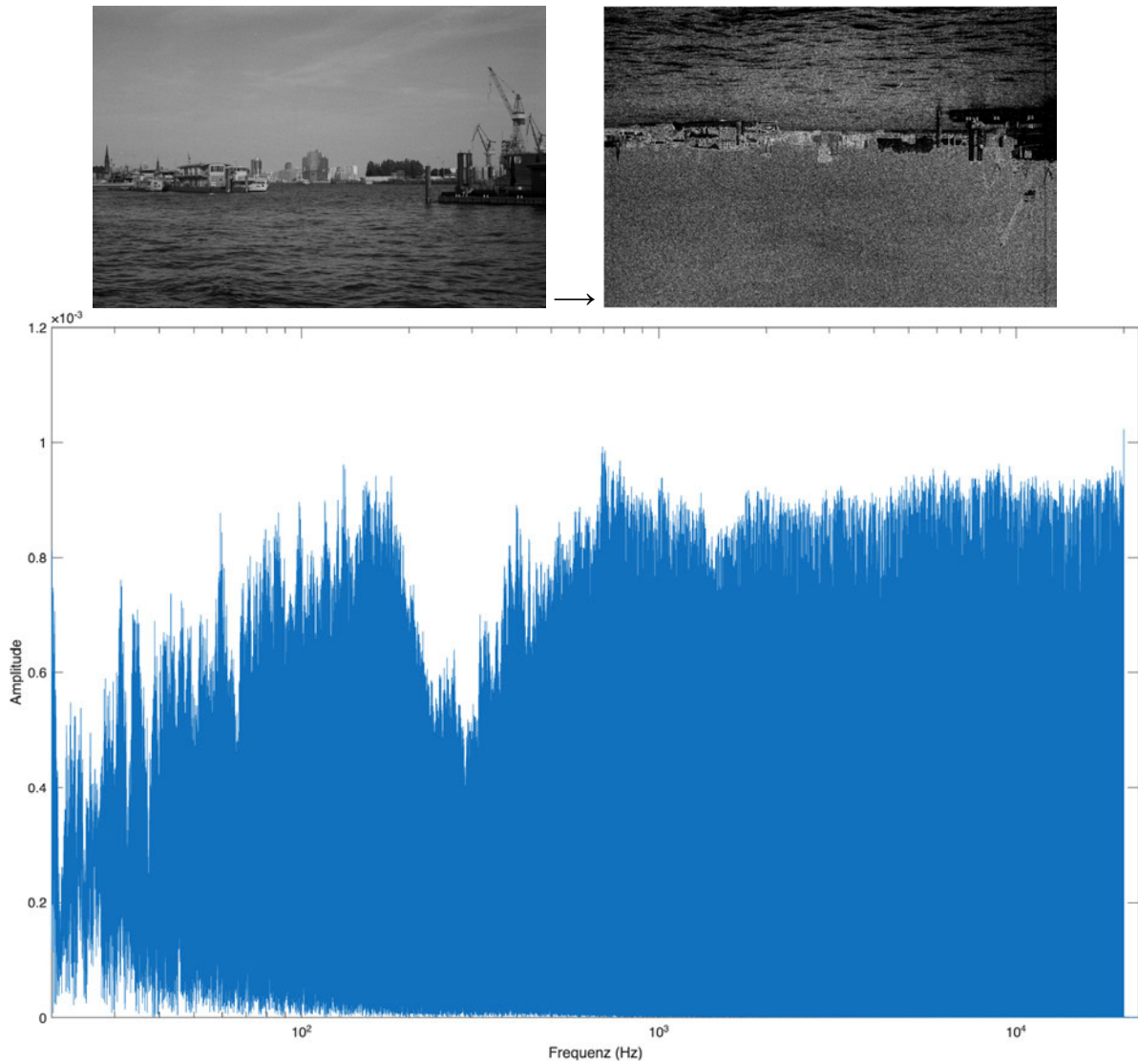


Abbildung 5.3: Frequenzanalyse des mit der in Kapitel 4.2.2 beschriebenen Methode und kontinuierlichen Frequenzphasen. Das Ursprungsbild ist das selbe wie in Abbildung 5.1

Dieses Spektrum ist deutlich dichter und willkürlicher bzw. zufälliger. Zudem ist auch nun ein deutlicher Unterschied zwischen verschiedenen Bildern vorhanden. Ein Beispiel für ein anderes Bild ist in Abbildung 5.4 zu sehen.

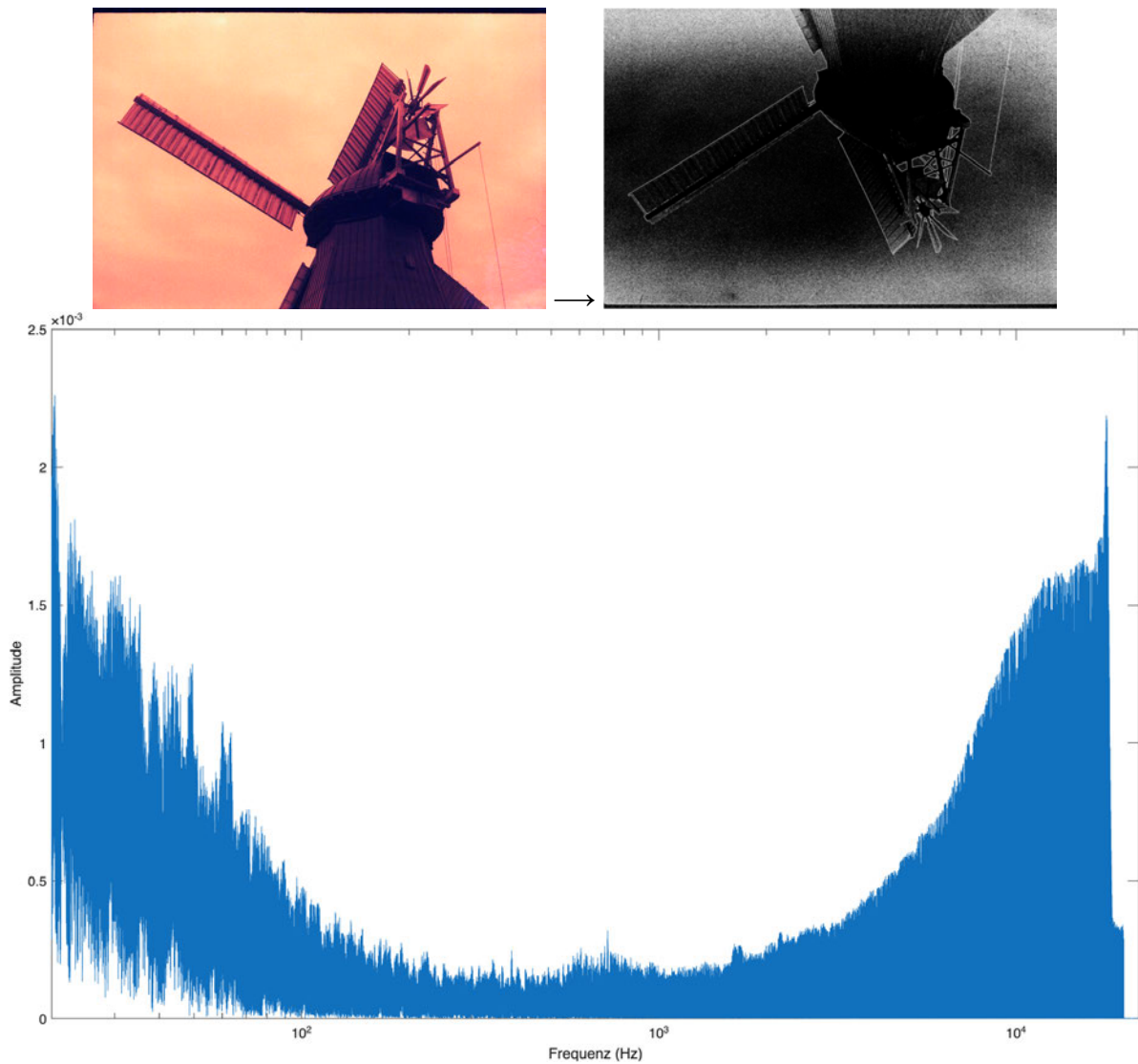


Abbildung 5.4: Extrahiertes Korn und Frequenzanalyse des mit der in Kapitel 4.2.2 beschriebenen Methode aus dem linken oberen Bild erzeugten Rauschens mit kontinuierlicher Phase

Sieht man sich erneut das künstlich erzeugte Bild mit Gauß-verteilterm Rauschen an, ergibt sich der Frequenzgang, der in Abbildung 5.5 zu sehen ist.

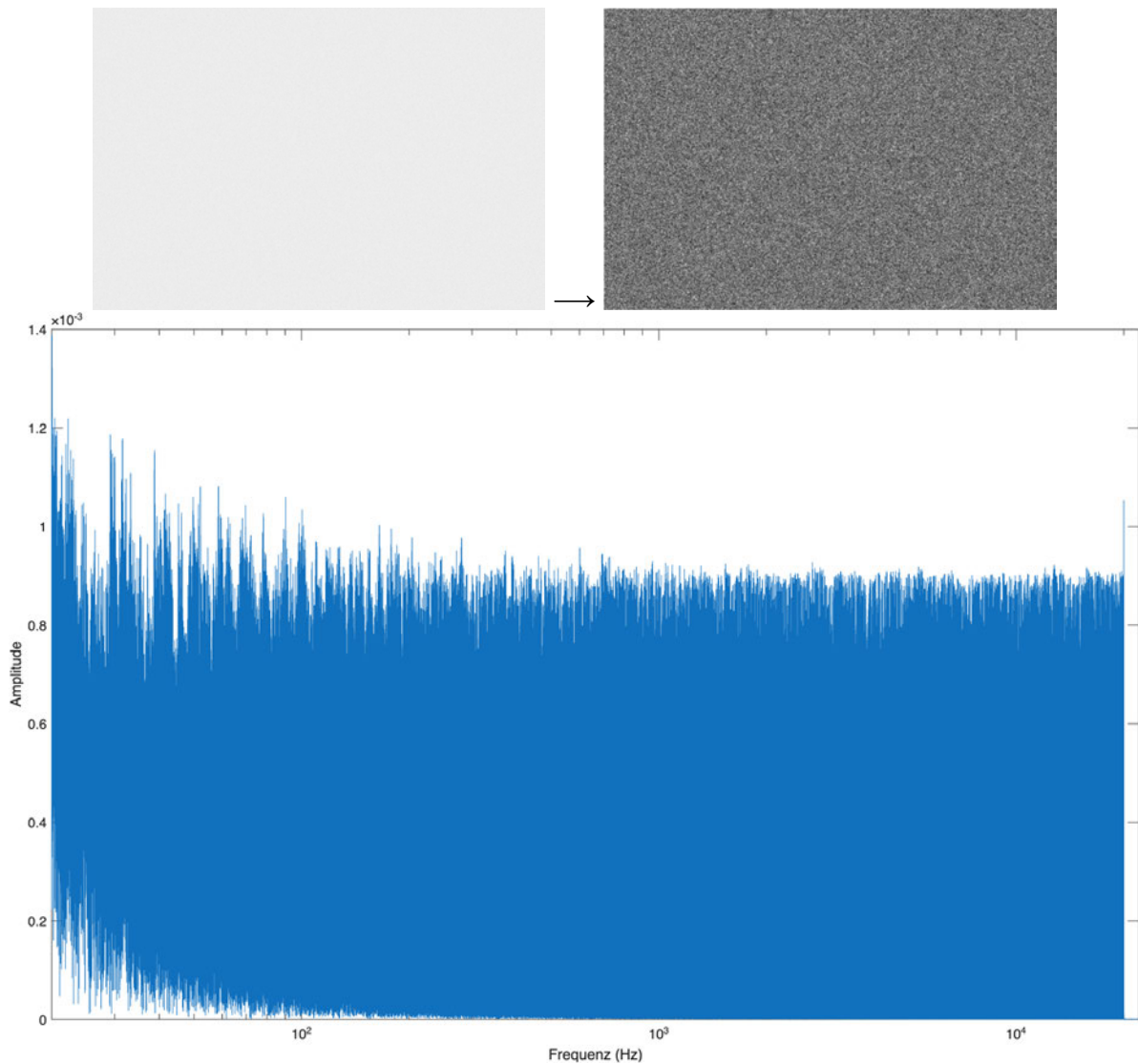


Abbildung 5.5: Künstlich erzeugtes Bild von Gaußischem Rauschen mit anschließender Frequenzanalyse. Verwendet wurde auch hier die verbesserte Rauscherzeugungsmethode

Im Vergleich zu dem in [Abbildung 5.2](#) zu sehenden Frequenzgang fällt sofort auf, dass sich durch die gleichmäßige Verteilung des Rauschens, bzw. des „künstlichen Korns“ eine relativ gleichmäßige, breite Frequenzverteilung ergibt. Dies passt deutlich besser zu der Charakteristik des Bildes, da nicht einzelne Frequenzen komplett eliminiert werden. Eine weitere Darstellungsmethode, um die Frequenzverteilung objektiv zu beurteilen, ist das Spektrogramm.

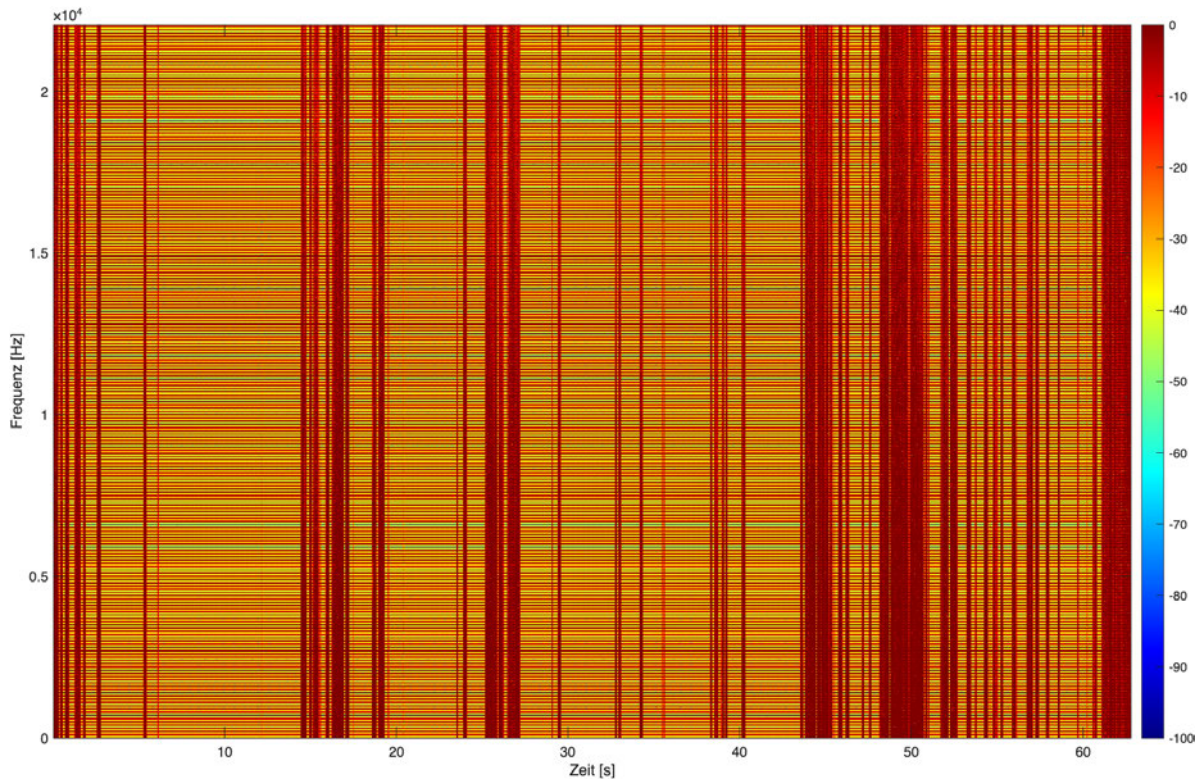


Abbildung 5.6: Spektrogramm eines Rauschsignals mit nicht kontinuierlicher Phase in den Frequenzen. Das zugrunde liegende Rauschsignal ist das selbe wie in [Abbildung 5.1](#)

In [Abbildung 5.6](#) ist ein solches Spektrogramm zu sehen. Dabei zeigt die x-Achse den zeitlichen Verlauf, die y-Achse die Frequenzen und die Farbe stellt die Intensität in dB dar. 0 dB entspricht dabei der maximal möglichen Aussteuerung. Die einzelnen Frames sind hierbei klar erkennbar, da sich eine Art Raster durch das gesamte Bild zieht. Sieht man sich das mit dem künstlich verrauschten Bild erzeugte Audiosignal in der Spektrogramm-Darstellung an, ist noch klarer erkennbar, dass sich durch den Phasenreset einzelne Frequenzen über die gesamte Länge auslöschen und auch sonst keine große Varianz vorhanden ist.

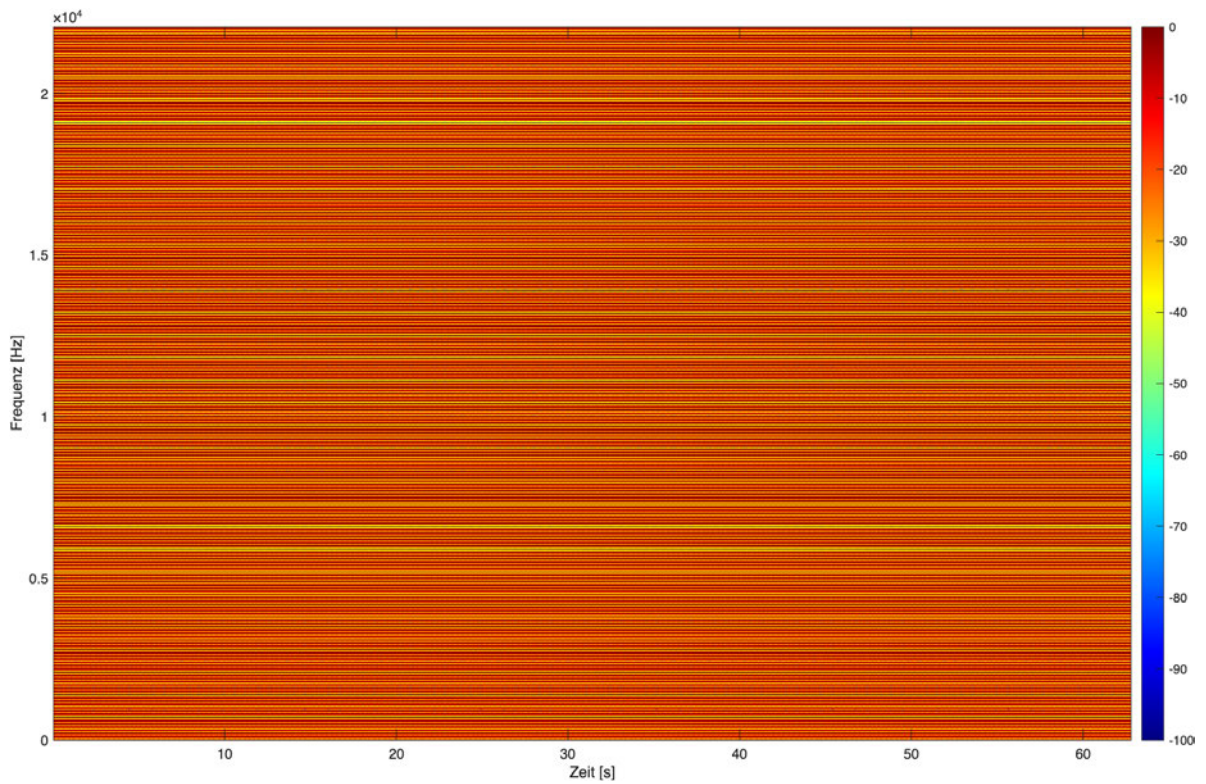


Abbildung 5.7: Spektrogramm eines Rauschsignals mit nicht kontinuierlicher Phase in den Frequenzen. Das zugrunde liegende Rauschsignal ist das künstliche Gauß-Rauschen von [Abbildung 5.2](#)

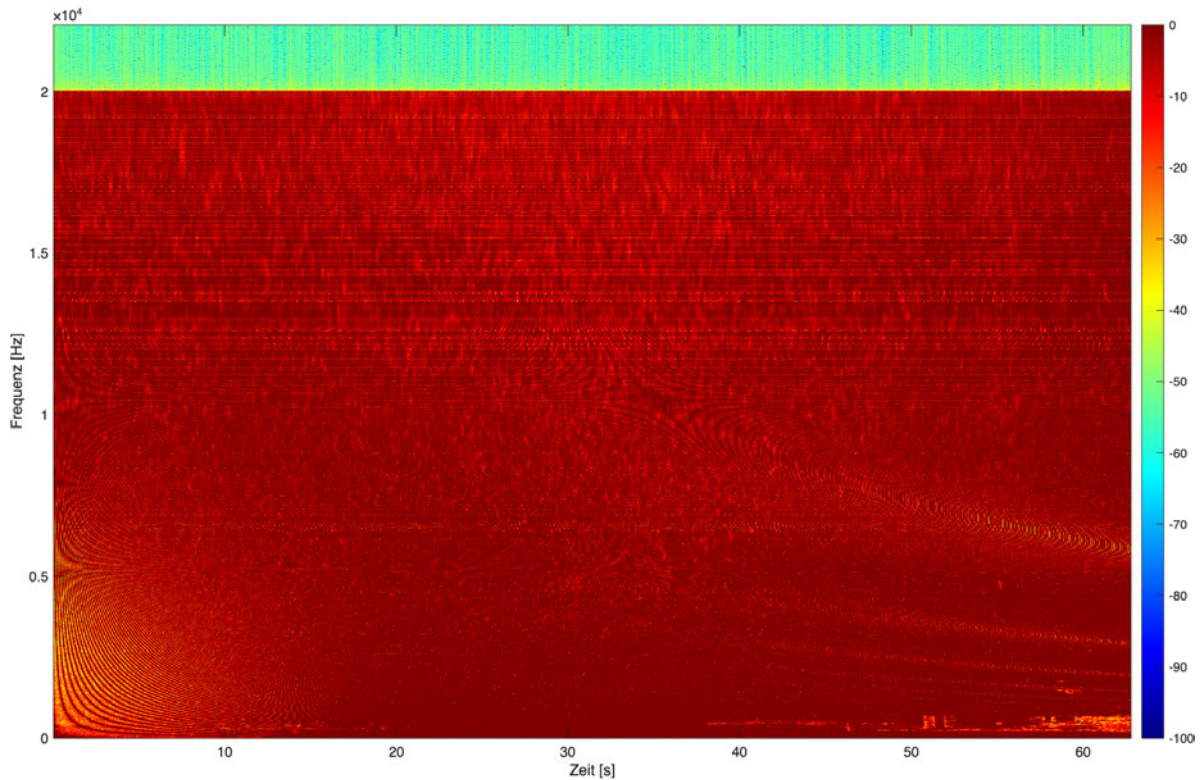


Abbildung 5.8: Spektrogramm eines Rauschsignals mit kontinuierlicher Phase in den Frequenzen. Die Version mit gleichem Ausgangsbild und Phasenreset ist in [Abbildung 5.6](#) zu sehen

In [Abbildung 5.8](#) sieht man das Ergebnis dieser Verbesserung. Es wirkt dichter und auch sehr zufällig verteilt. Außerdem kann man den am Anfang des Audios zu hörenden abwärts laufenden Frequenzsweep sehen. Erst ab ca. 10 Sekunden hat sich das System soweit eingependelt dass man von einem echten Rauschsignal sprechen kann. Da dieses Phänomen konstant für jedes Eingangssignal ist würde es ausreichen, entweder einen Teil wegzuschneiden oder im Programmcode einzuprogrammieren, dass nicht die Phase von allen Frequenzen bei $t = 0$ beginnt. Für die weitere Betrachtung der Ergebnisse spielt dies allerdings keine Rolle. Das Gauß-Rauschen in [Abbildung 5.9](#) zeigt ein ähnliches Bild, jedoch ohne den Abfall in den tieferen Frequenzen ab ca. 50 Sekunden.

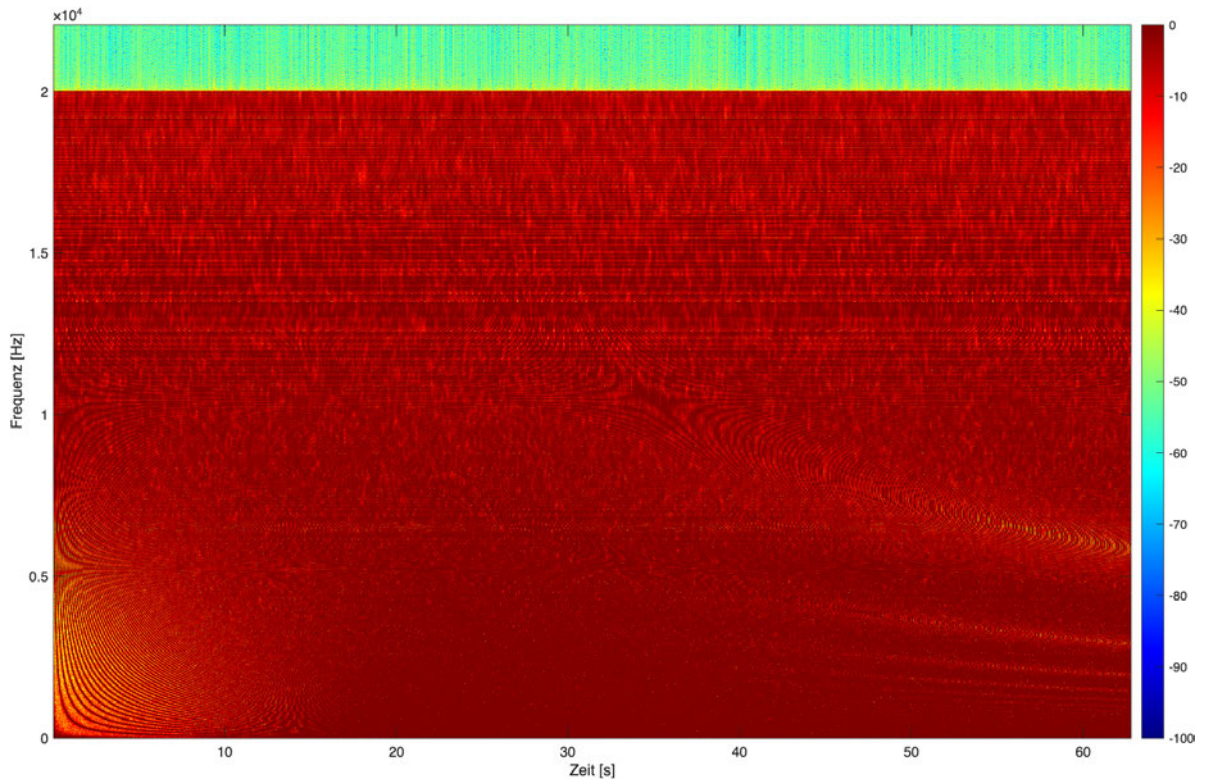


Abbildung 5.9: Spektrogramm eines Rauschsignals mit kontinuierlicher Phase in den Frequenzen. Die Version mit gleichem Ausgangsbild und Phasenreset ist in [Abbildung 5.7](#) zu sehen

Das „Einpendeln“ in den ersten Sekunden ist hier jedoch nach wie vor zu sehen. In manchen Ergebnissen zeigen sich tatsächlich in der Spektrogrammdarstellung Teile des ursprünglichen Bildinhalts. Ein Beispiel dafür ist [Abbildung 5.10](#).

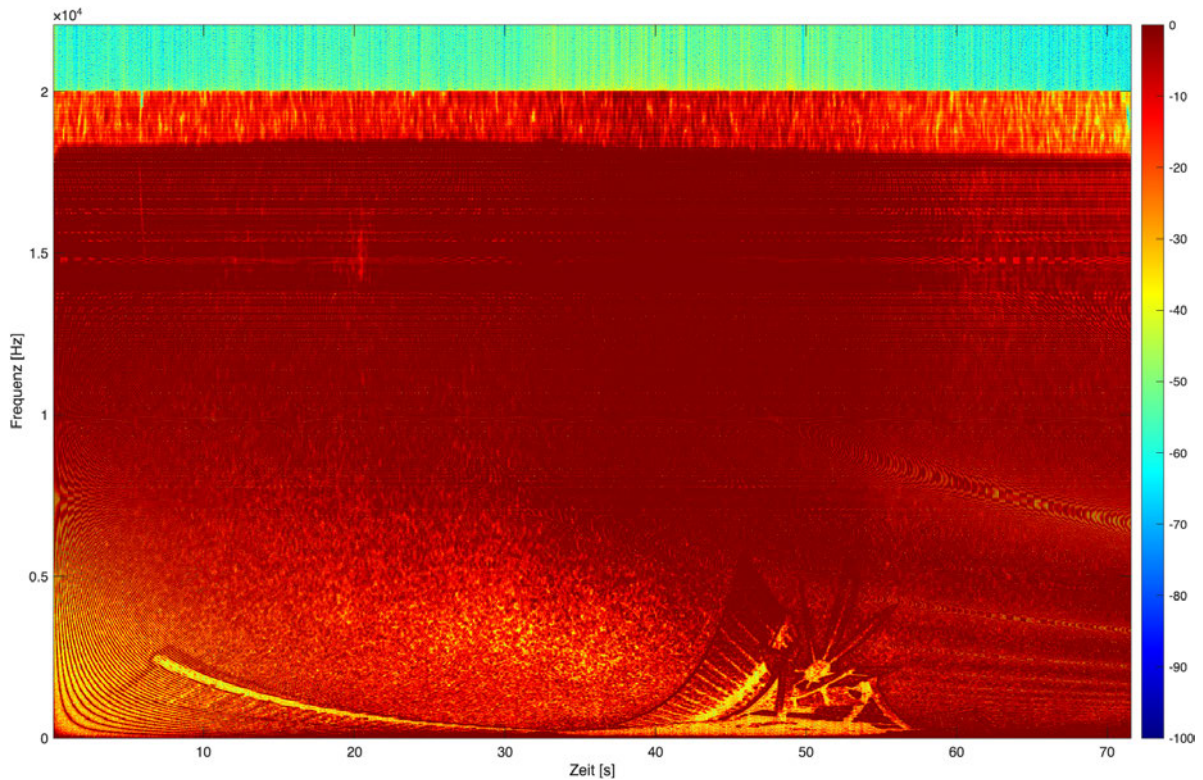


Abbildung 5.10: Spektrogramm eines Rauschsignals mit kontinuierlicher Phase in den Frequenzen. Ausgangsbild und Histogramm sind in [Abbildung 5.4](#) zu sehen. In den tiefen Frequenzen ist noch die Mühle als Hauptmotiv des Ursprungsbildes erkennbar. Die Verzerrung entsteht, da bei der Generierung der Frequenzen eine Logarithmische Skala gewählt wurde, aber die Spektrogrammsdarstellung linear ist

Besonders interessant ist natürlich zudem noch, wie im Vergleich zu den aus Filmkorn generierten Ergebnissen die klassisch algorithmisch erzeugten Rauschsignale aussehen.

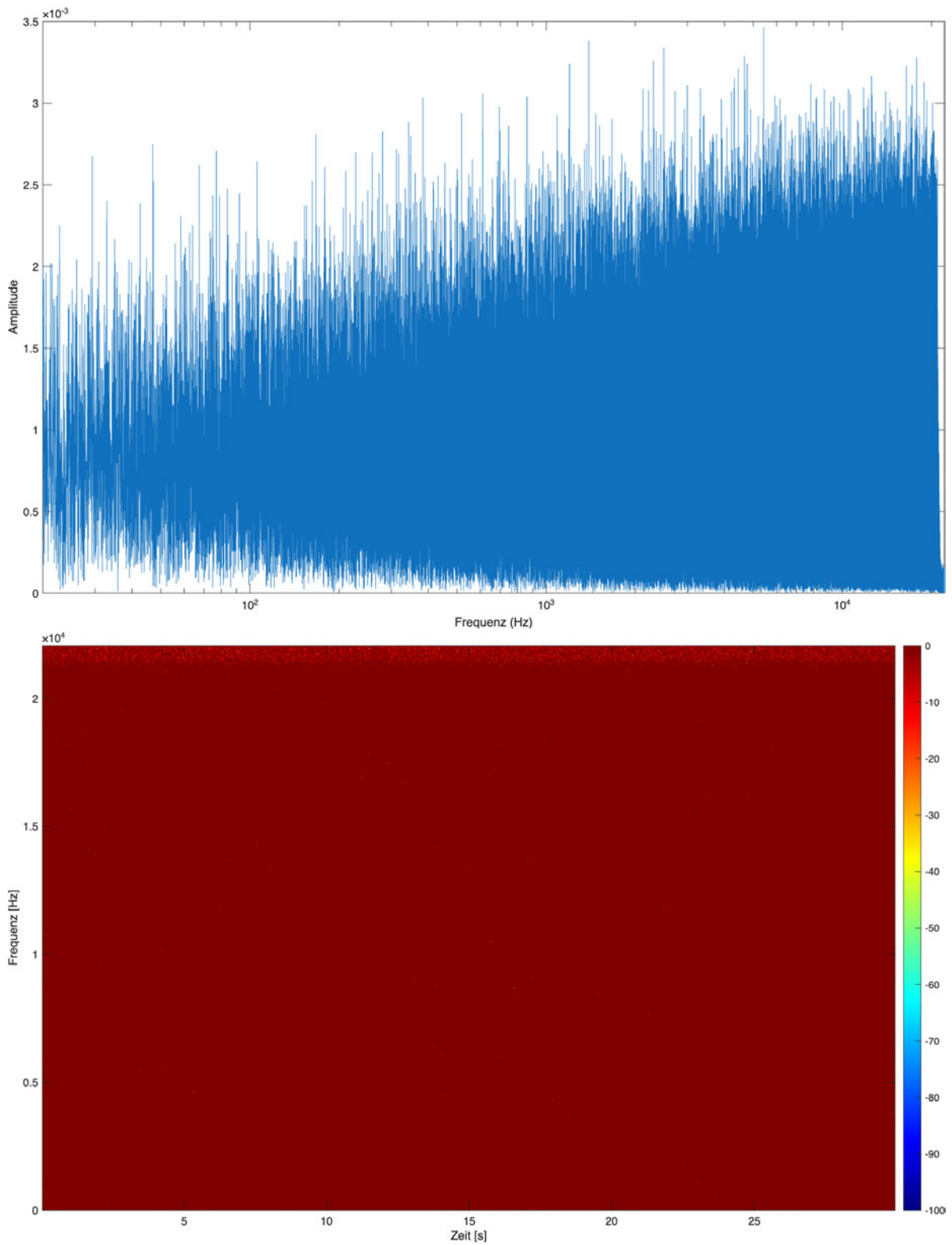


Abbildung 5.11: Histogramm und Spektrogramm von mit Audacity erzeugtem weißen Rauschen

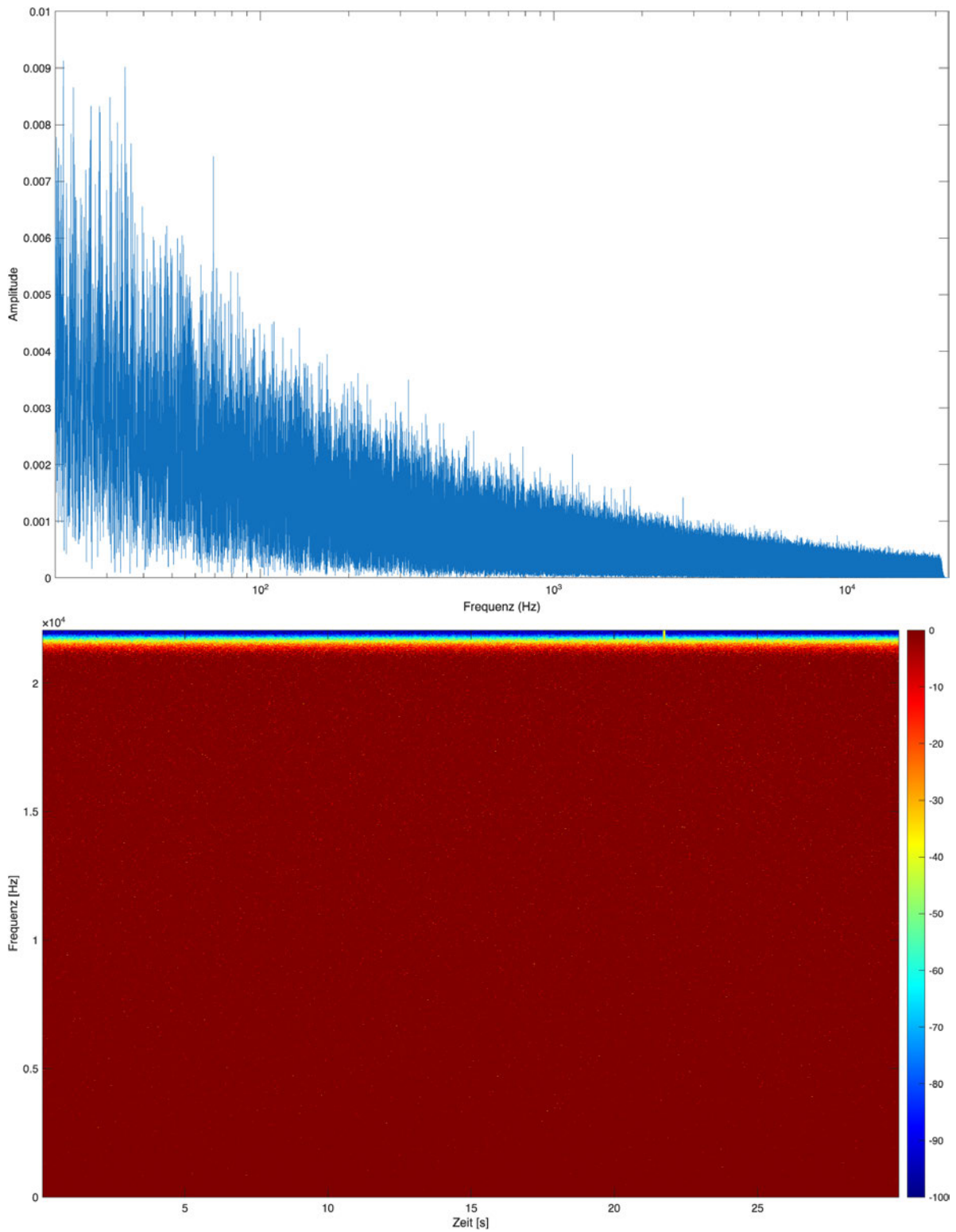


Abbildung 5.12: Histogramm und Spektrogramm von mit Audacity erzeugtem rosa Rauschen

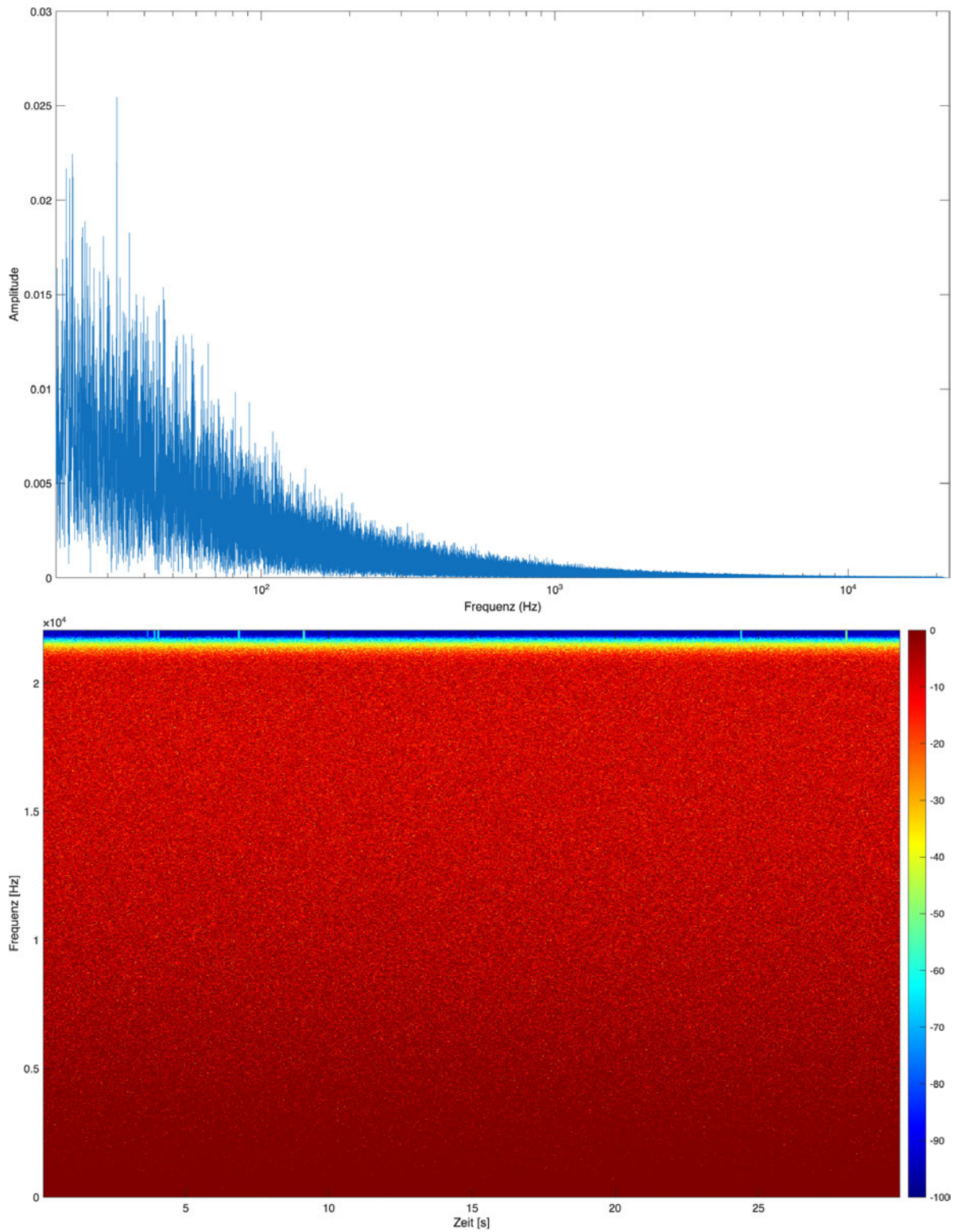


Abbildung 5.13: Histogramm und Spektrogramm von mit Audacity erzeugtem roten Rauschen

5.1.2 Subjektive Wahrnehmung

Für die subjektive Wahrnehmung kann mit verschiedenen Tests und Befragungen gearbeitet werden. Dafür habe ich folgende Optionen angedacht:

1. *ABX-Test*: Dieser Test ist in der Audioentwicklung beheimatet und prüft, ob die Probanden einen Unterschied wahrnehmen und wie gut bzw. treffsicher das Audiosignal zugeordnet werden kann. Dabei werden drei kurze Audioclips hintereinander abgespielt: Zuerst ein echtes, aufgenommenes natürliches Rauschen, als zweites das von meinem System generierte Rauschen und zum Schluss zufällig ein anderer Ausschnitt einer der beiden Signale. Der Proband muss nun bestimmen, ob das dritte Signal identisch zum ersten oder zum zweiten ist. Bei einer perfekten Simulation werden die Probanden nur noch raten und die Trefferquote wird sich bei 50% einpendeln. Für meinen Fall würde der Test ein zweites Mal durchgeführt werden mit einem synthetischen weißen Rauschen als Referenzsignal. Wenn das System gut funktioniert und sich das von mir generierte Signal deutlich von dem synthetischen Standard unterscheidet, sollte dabei die Trefferquote bei 100% liegen. Mittelt man anschließend diese Werte kann sowohl beurteilt werden wie natürlich das Signal klingt und wie sehr es sich von künstlichen Rauschsignalen abhebt.
2. *Semantisches Differential*: Dies ist eine vom amerikanischen Psychologen C. E. Osgood entwickelte Methode („Semantisches Differential“, [n. d.](#)). Dabei werden bipolare, also komplett gegensätzliche, Adjektivpaare vorgelegt, die dann auf einer beispielsweise 5-stufigen Skala eingeordnet werden müssen. Für einen Vergleich mit weißem Rauschen wären folgende Paare geeignet:
 - **synthetisch - natürlich**: Die eigentliche Kernfrage
 - **statisch - lebendig**: Natürliches Rauschen lebt wie bei meiner Generierung von kleinen Variationen, weißes Rauschen ist sehr starr
 - **gefärbt - neutral**: Bei einer unregelmäßigen Frequenzverteilung, wie sie bei natürlichem Rauschen auftritt ist eine Klangfärbung wahrzunehmen, bei weißem Rauschen nicht.
 - **unruhig - beruhigend**: Weißes Rauschen wirkt häufig stressig
 - **aggressiv - sanft**: Ähnlich zum vorherigen Punkt

Ein großer Vorteil bei dieser Methode ist, dass die emotionalen und psychologischen Aspekte messbar werden. Dadurch, dass nicht bei allen Begriffspaaren direkt ersichtlich ist, worauf die Antwort hinausläuft wird außerdem unvoreingenommen geantwortet. Gleichzeitig ist es noch klar und genau genug, um gegebenenfalls an Stellschrauben drehen zu können und das System zu verbessern.

Diese zwei Vorgehensweisen schließen sich nicht gegenseitig aus, sondern können gut in dieser Reihenfolge durchgeführt werden. Die quantitativen, starren Daten aus dem Hörtest könnten so mit dem semantischen Differential psychoakustisch untermauert und begründet werden.

5.2 Auswertung der Verrechnung von Rauschen und Musik

Für diese Untersuchungen habe ich erneut hauptsächlich mit Histogrammen gearbeitet. Abbildungen 5.16 bis 5.18 haben alle als Basis die in Abbildung 5.14 und 5.15 zu sehenden Ausgangssignale.

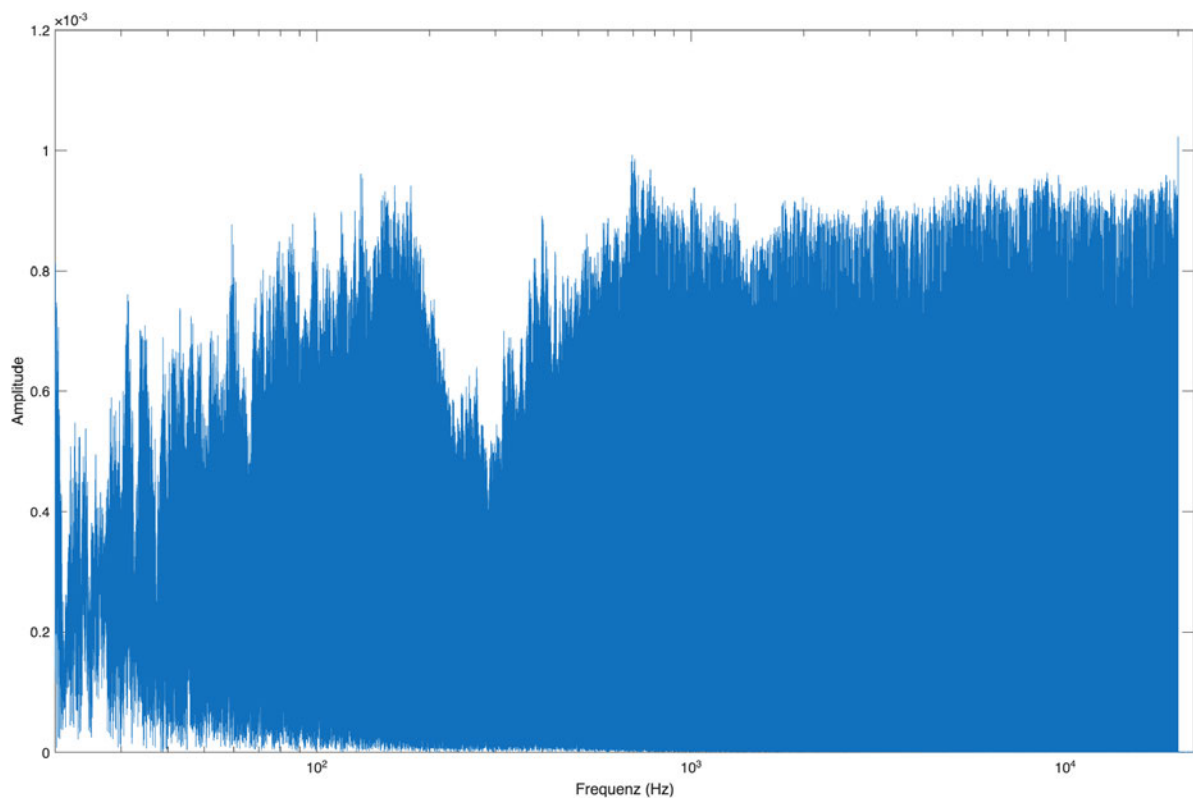


Abbildung 5.14: Frequenzanalyse vom verwendeten Rauschen

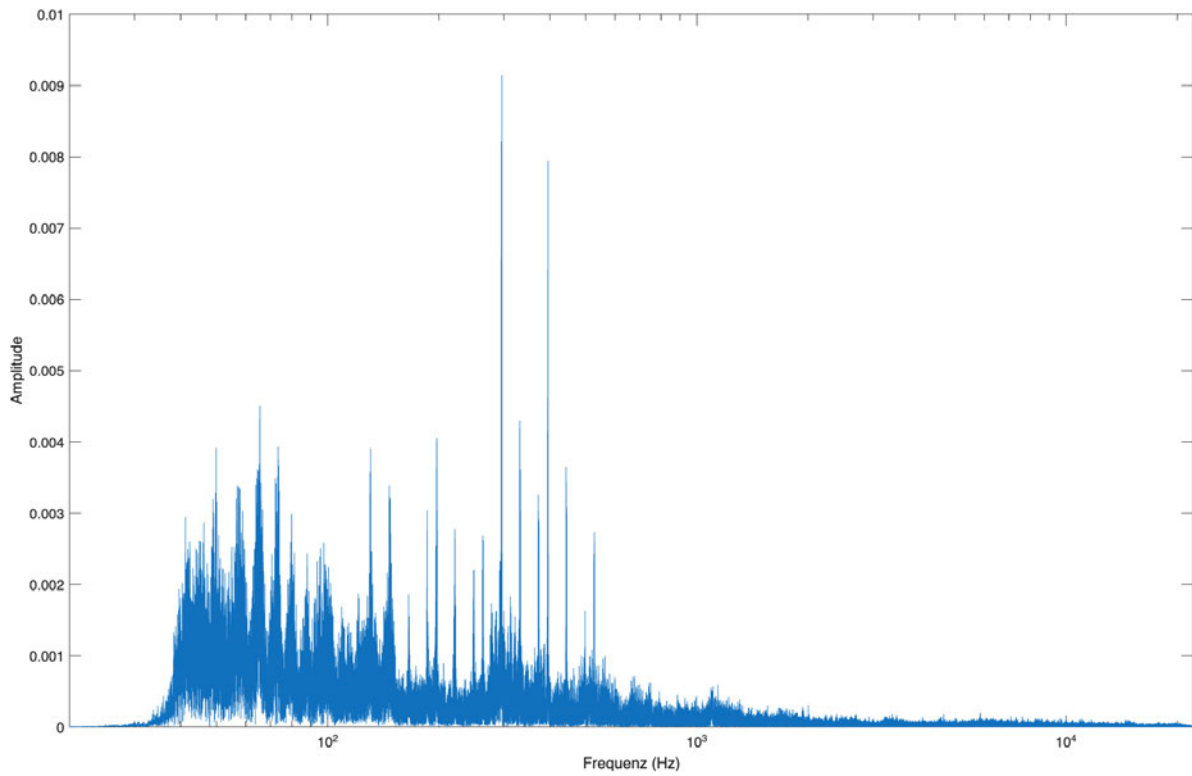


Abbildung 5.15: Frequenzanalyse vom unangetasteten Musikstück. Das Musiksignal wurde lediglich auf die selbe Länge des Rauschens zugeschnitten um zu verhindern, dass später vorkommende Frequenzanteile das Bild verfälschen.

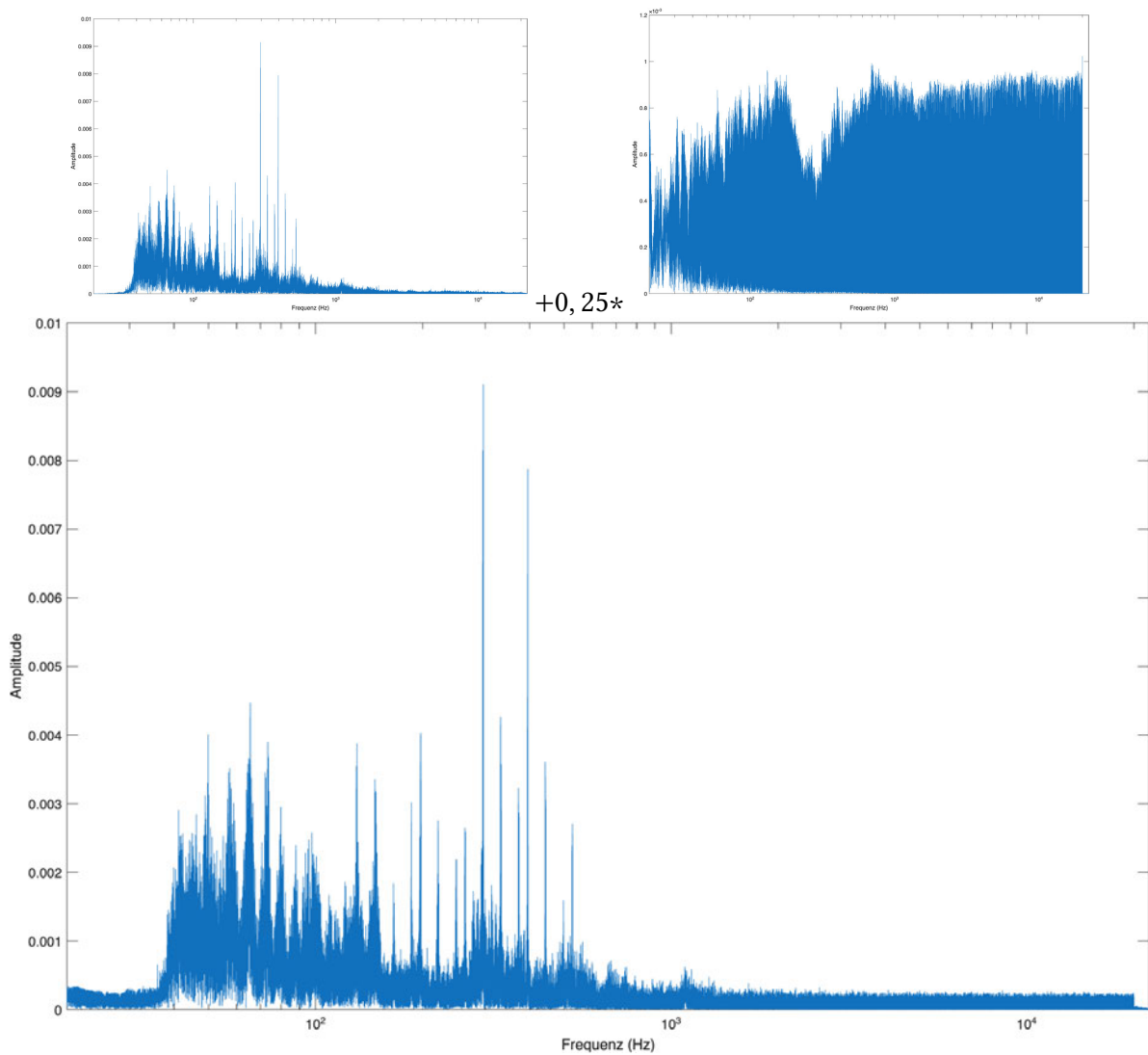


Abbildung 5.16: Addition von Rausch- und Musiksignal. Damit das Rauschsignal nicht Überhand nimmt, wurde es auf den Faktor 0,25 beschränkt

In [Abbildung 5.16](#) sieht man, wie sich die Frequenzverteilung ändert, wenn das Musik- und das Rauschsignal addiert werden. Insbesondere in den hohen Frequenzen ab ca. 10 kHz sieht man, wie die Amplitude angestiegen ist. In anderen Bereichen dominiert trotzdem noch das Musiksignal. Würde man bei der Addition nicht das Rauschsignal begrenzen, würde es das Musiksignal völlig überschatten und wäre kaum noch hörbar.

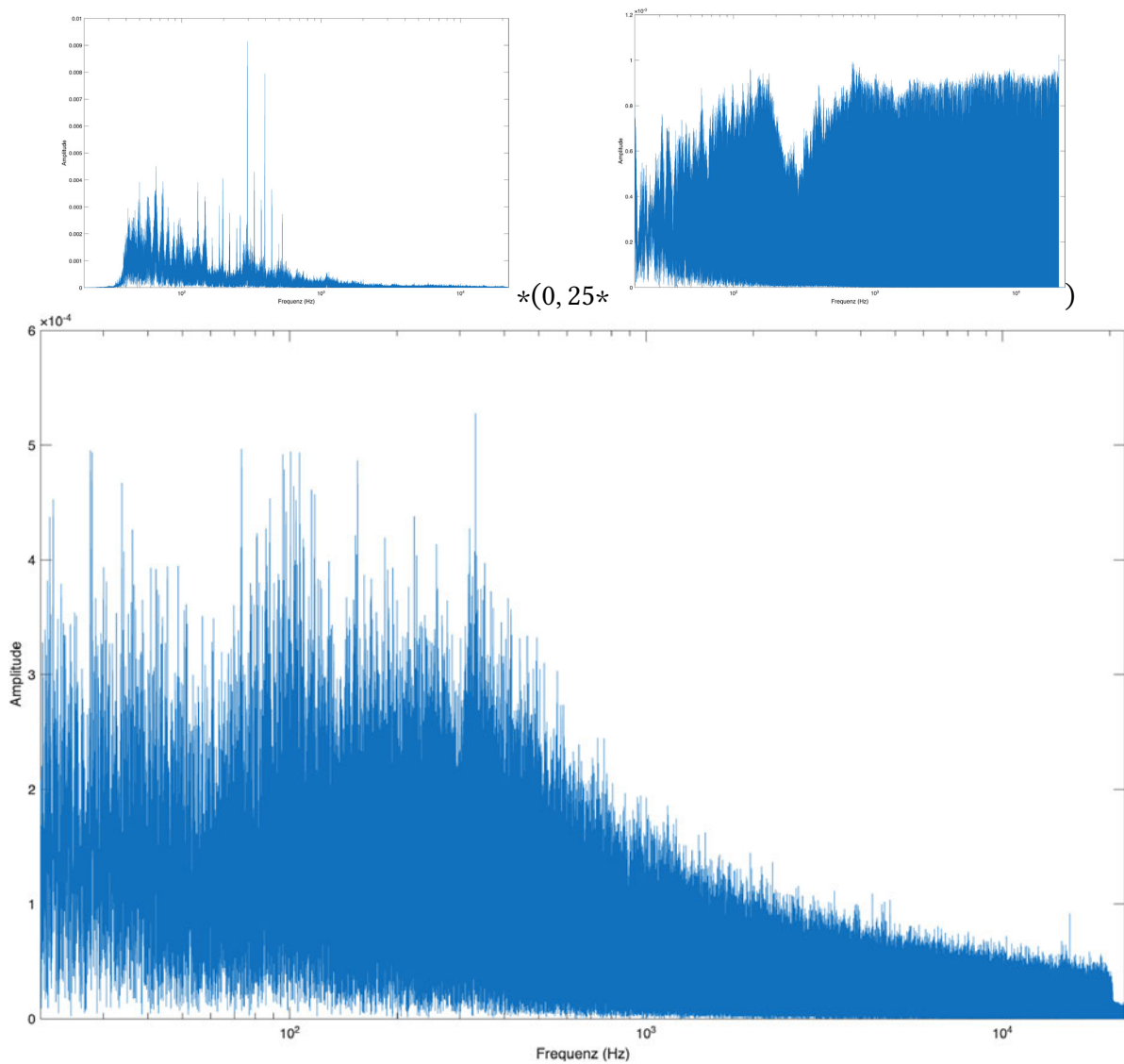


Abbildung 5.17: Multiplikation von Rausch- und Musiksignal. Das Rauschsignal wurde hier ebenfalls auf den Faktor 0,25 begrenzt

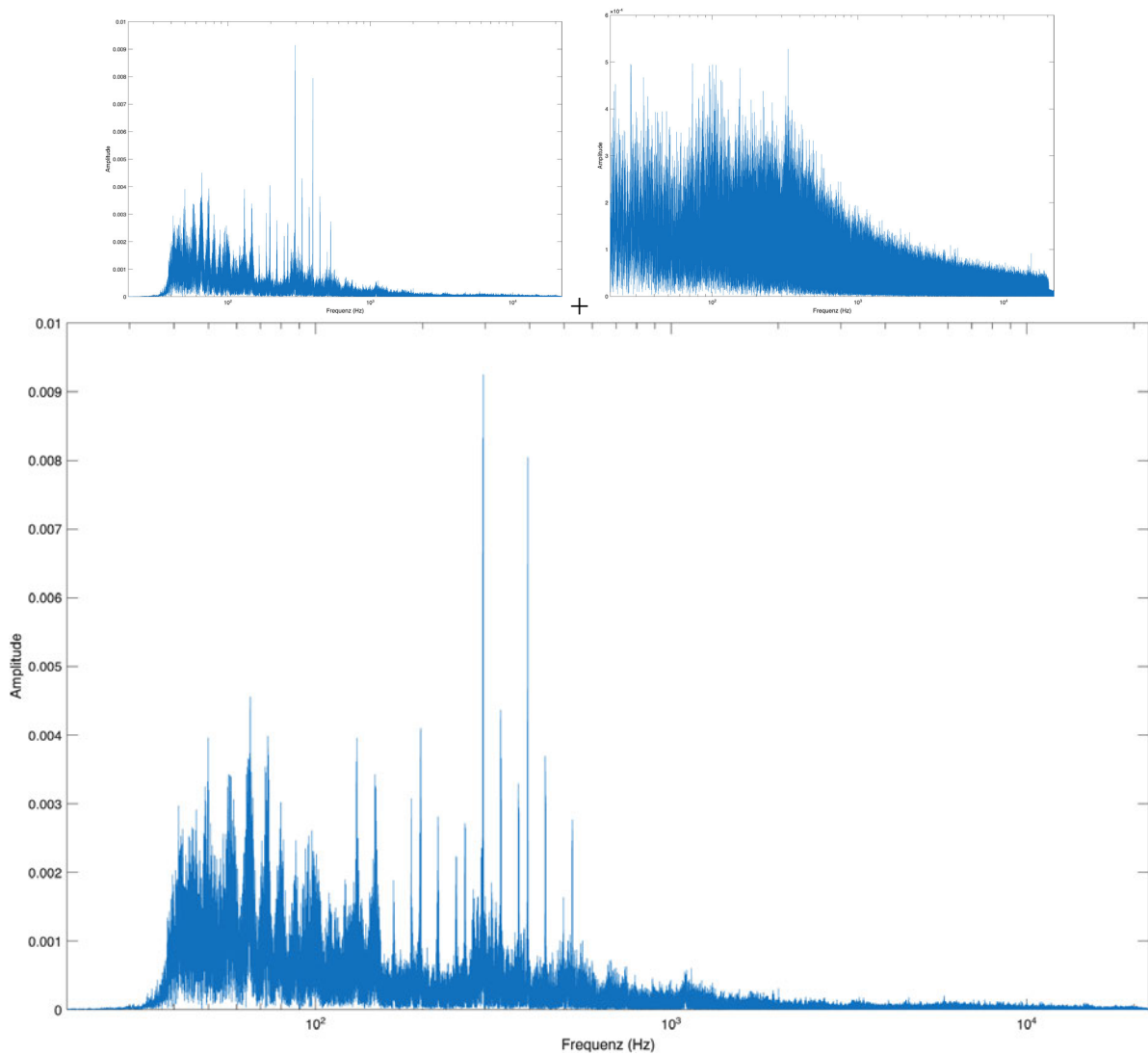


Abbildung 5.18: Addition des Produkts von Rausch- und Musiksignal

Das in [Abbildung 5.18](#) zu sehende Ergebnis sieht wieder sehr nach dem Ursprungssignal aus. Hier könnte zur genaueren Betrachtung die Spektrogrammdarstellung helfen.

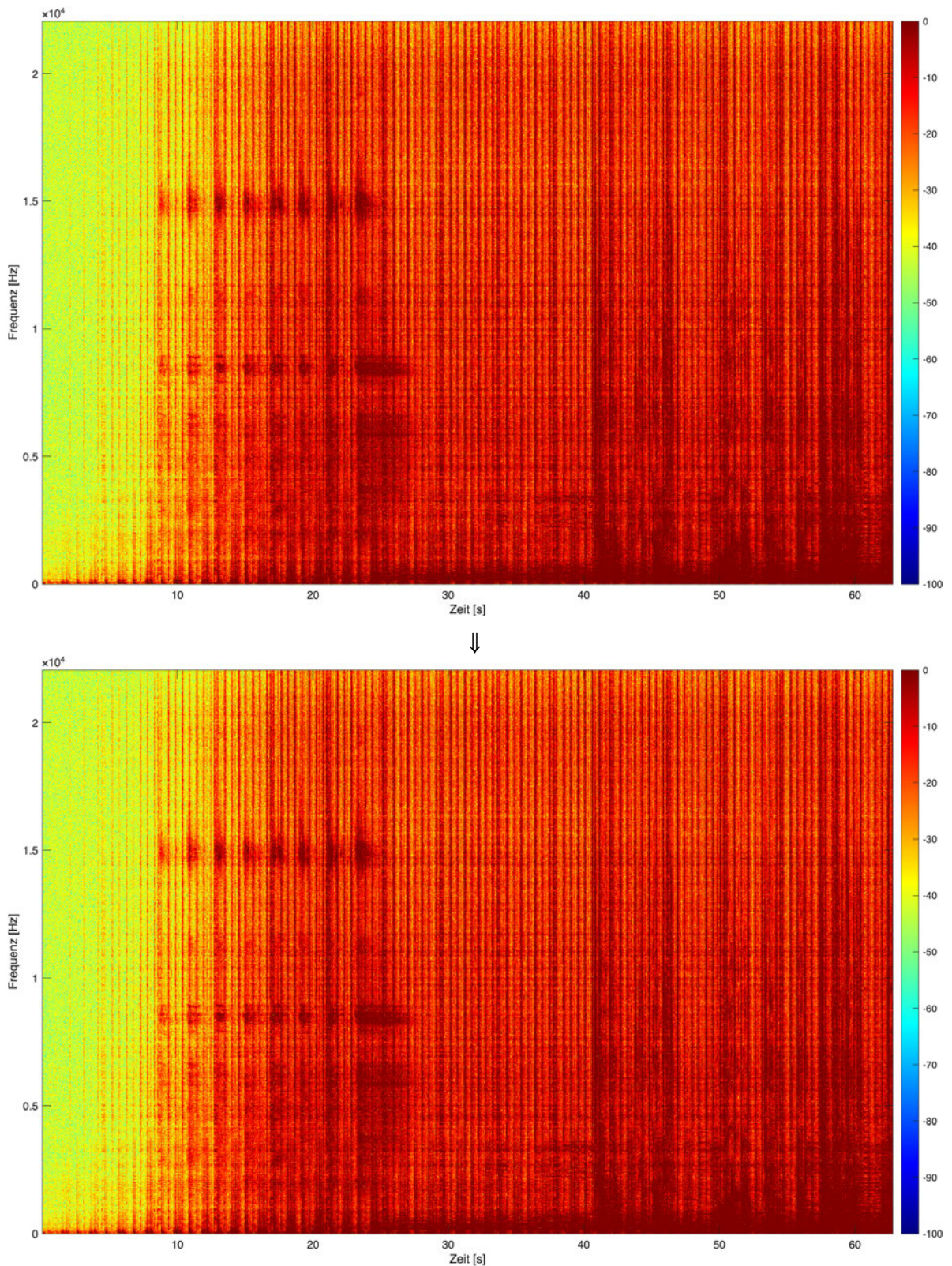


Abbildung 5.19: Ursprungssignal der Musik oben und das Ergebnis nach der Verrechnung ($\text{Musik} + (\text{Musik} * \text{Rauschen})$). Auch in dieser Darstellung ist der Unterschied nur bei sehr genauem Betrachten sichtbar, allerdings ist der Unterschied deutlich hörbar.

5.3 Vergleich zu natürlichem analogen Rauschen

Dieser Punkt ist besonders interessant um zu verstehen in wie weit meine Methode zur Simulation von analogen Tonträgern anwendbar ist. Dafür habe ich Aufnahmen von Bandrauschen und Schallplatten untersucht. Der Vergleich zu analogen Vinylaufnahmen war für mich besonders interessant. Für eine gute Analyse benötigte ich zum einen eine Aufnahme von der Schallplatte und zum anderen das selbe Musikstück als digitale Audiodatei. Somit konnte ich vergleichen, wie sehr sich der Klangeindruck und die Frequenzen nach dem Verrechnen mit dem Rauschen an die analoge Aufnahme annähern. In Abbildung 5.20 ist das Frequenzspektrum eines Bandrauschens zu sehen. Dieses ist deutlich dumpfer bzw. die tiefen Frequenzen sind stärker ausgeprägt als die in dem von mir erzeugten Rauschen.

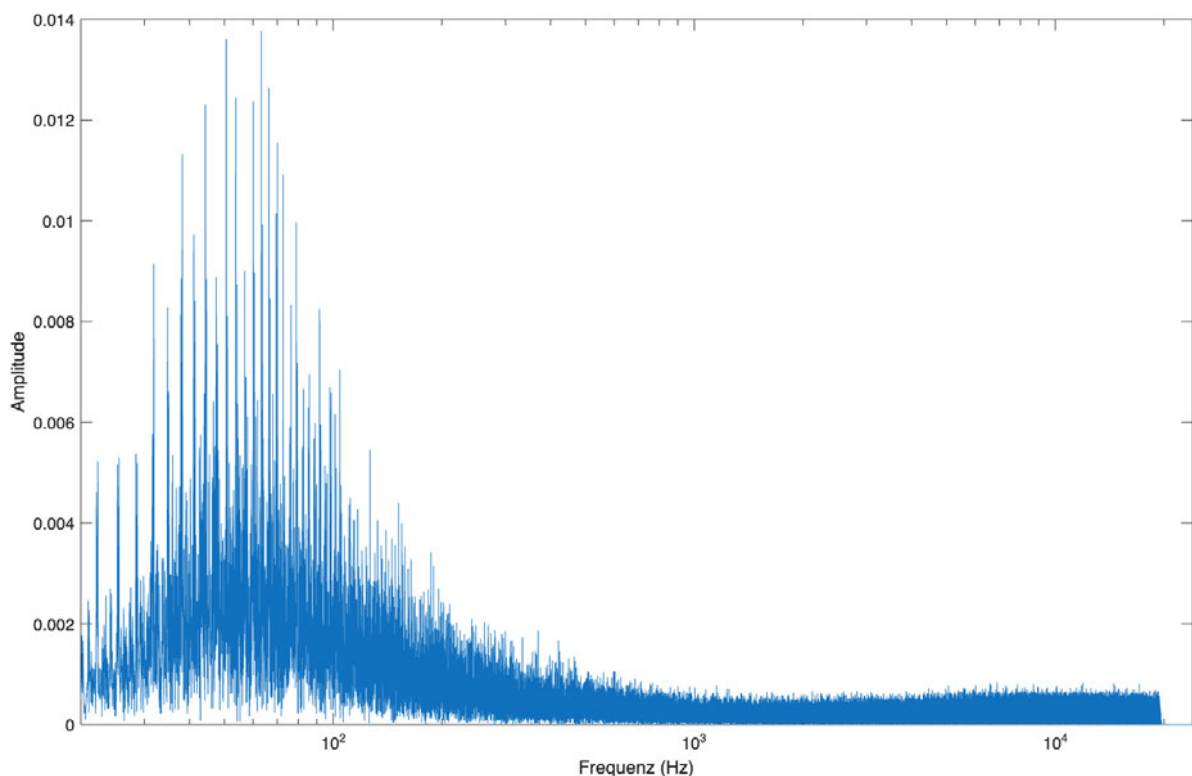


Abbildung 5.20: Histogramm eines aufgenommenen Bandrauschens

Für den Vergleich mit einer Schallplattenaufnahme habe ich den gleichen Song selber von der Schallplatte mit der gleichen Abtastrate wie die digitale Version aufgenommen. Anschließend habe ich den Clip soweit zugeschnitten, dass er die gleiche Länge wie das von mir bearbeitete und mit Rauschen versehene Audio hatte. Die Ergebnisse sind in Abbildung 5.21 und Abbildung 5.22 zu sehen. Verglichen wird mit den Ergebnissen von Abbildung 5.18 und Abbildung 5.19, da diese am ehesten einem analogen Höreindruck entsprachen.

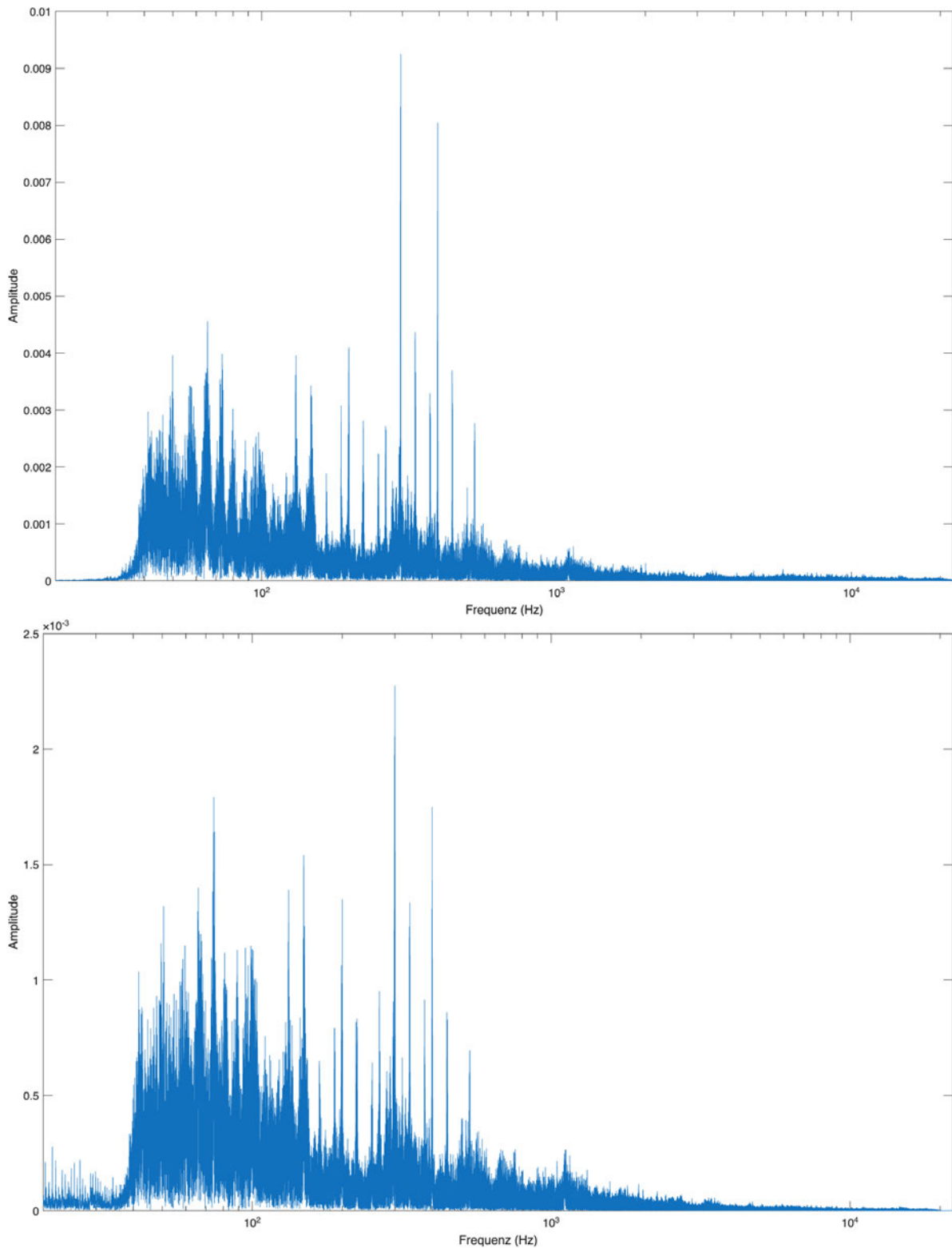


Abbildung 5.21: Histogramm von der digitalen Aufnahme mit hinzugefügtem künstlichem Rauschen oben und der Aufnahme von der Schallplatte unten. Die Ähnlichkeit ist sofort erkennbar, allerdings sind bei der Schallplattenaufnahme die tiefen Frequenzen sehr ausgeprägt (vgl. Abbildung 5.23) und durch die Aufnahme mit einer Abtastrate von 44,1 kHz sind die Frequenzen oberhalb von 22,05 kHz nicht vorhanden

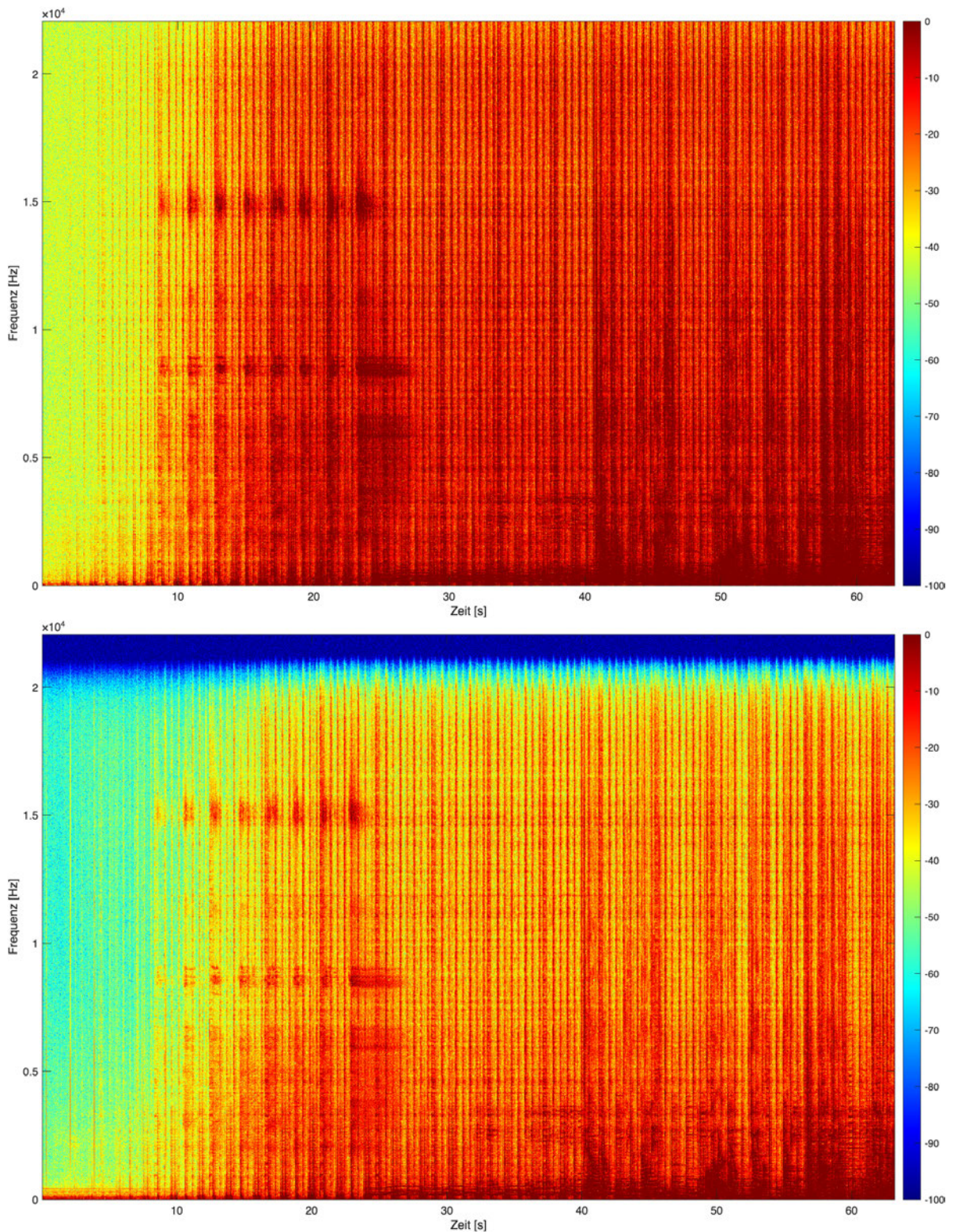


Abbildung 5.22: Spektrogramm von der digitalen Aufnahme mit hinzugefügtem künstlichem Rauschen oben und der Aufnahme von der Schallplatte unten. Das grundlegende Bild ist fast identisch, lediglich der Aufnahmepegel von der Schallplatte war etwas geringer und das künstliche Rauschen hat einen größeren Frequenzumfang. Das horizontale Raster ist auch in der Analogaufnahme vorhanden, daher hat es etwas mit der Musik und nicht mit dem künstlichen Rauschen zu tun.

Wie man in den Abbildungen 5.21 und 5.22 sehen kann ähneln sich die beiden Versionen sehr. Dies bestätigt sich auch beim Höreindruck. Somit kann man sagen, dass die Simulation eines Analograuschens für diesen Fall definitiv funktioniert. Der Code müsste jedoch noch so angepasst werden, dass automatisch für die gesamte Länge der Musik das Rauschen generiert wird. Aktuell sind alle Rauschsignale deutlich zu kurz dafür. Ich habe zudem das Knacken und Rauschen einer Schallplatte ohne zusätzliches Musiksinal analysiert. Es handelt sich um die ersten 10 s des Randes bevor das Musikstück beginnt. Eine Leerplatte zur längeren Analyse lag mir nicht vor. Ein Histogramm ist in Abbildung 5.23 zu sehen.

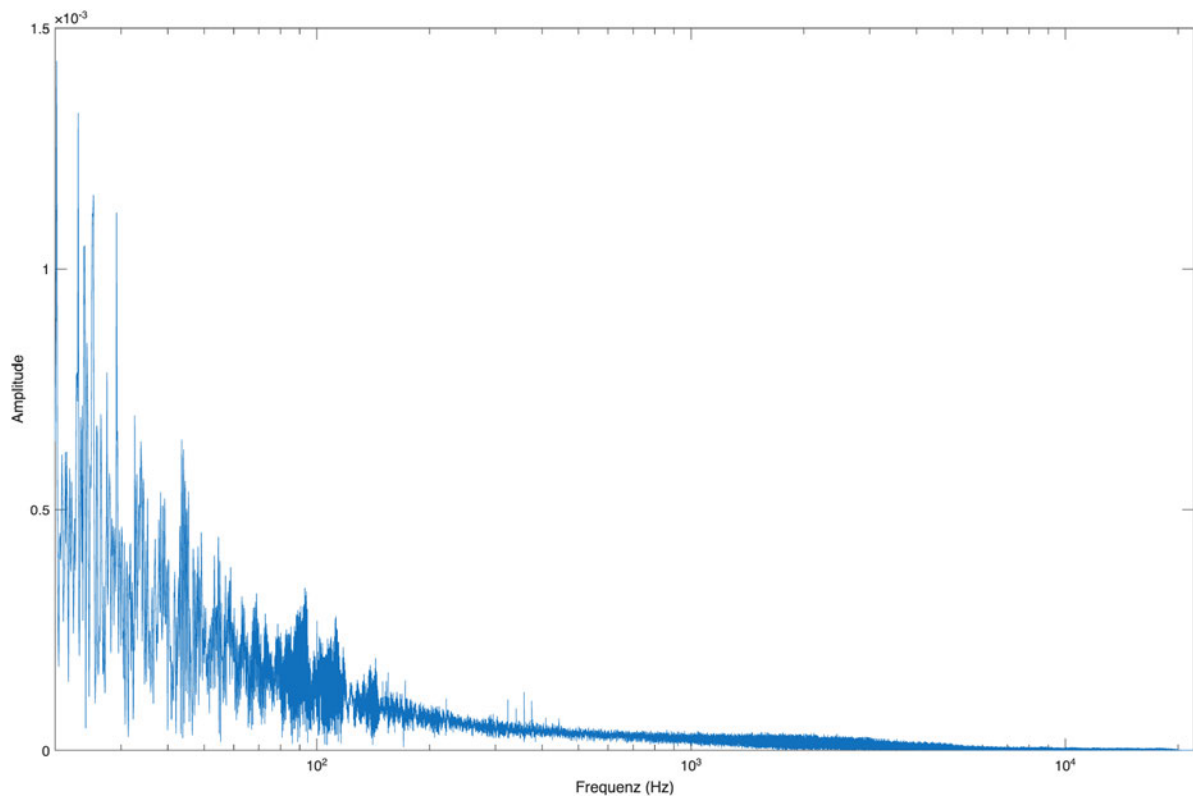


Abbildung 5.23: Histogramm des Rauschens einer Schallplatte

Dieses Histogramm zeigt zumindest in der Histogrammdarstellung vollkommen andere Charakteristika als das von mir erzeugte Rauschen. Die tiefen Frequenzen sind deutlich stärker ausgeprägt als die hohen Frequenzen. Am ehesten sind zumindest im Bereich bis ca 500 Hz Gemeinsamkeiten zum in Abbildung 5.4 gezeigten Rauschen zu sehen. Sinnvoll ist aber auch hier nochmal die Betrachtung des zeitlichen Verlaufs. In Abbildung 5.24 ist ein Spektrogramm der selben Aufnahme zu sehen.

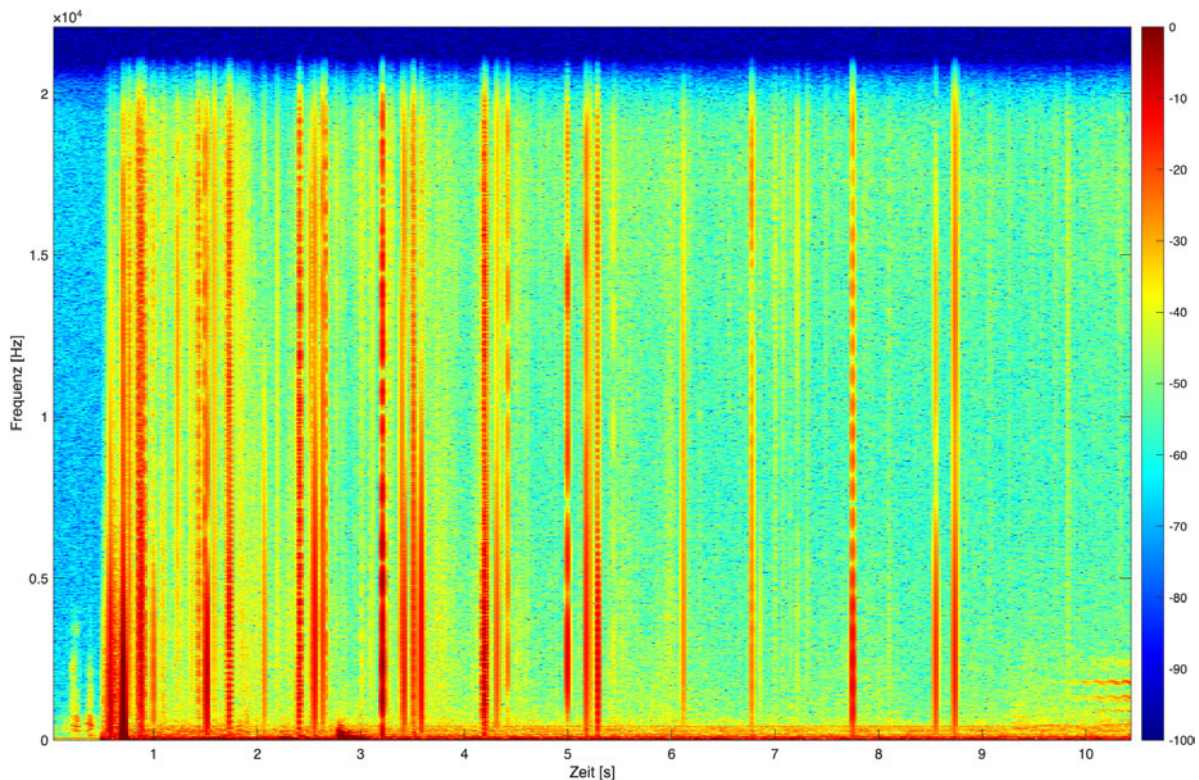


Abbildung 5.24: Spektrogramm des Rauschens einer Schallplatte

Hier ist deutlich das tiefe Grundrauschen zu erkennen, welches das Histogramm dominiert. Zudem sieht man als vertikale rote und orange Balken die kleinen Knackser durch Staubkörner oder Kratzer. Solch ein Knacken ist auch bei dem von mir generierten Rauschen ein bisschen hörbar, allerdings nicht nur abhängig vom Korn, sondern auch von dem Beginn des neuen Frames. Würde man noch mehr der Objektkanten im Bild zur Analyse nutzen würde sich dieser Höreindruck mit Knacksern verstärken.

5.4 Auswirkung von verschiedenen Parametern

5.4.1 Blockgröße

Die Blockgröße beim Abtasten der binären Matrix hat eine Auswirkung darauf, wie differenziert die Lautstärken der einzelnen ausgespielten Frequenzen wiedergegeben werden können. Bei einer Blockgröße von 2×2 gibt es beispielsweise nur die 5 Level 0 – 0,25 – 0,5 – 0,75 – 1. Ich habe meistens mit einer Blockgröße von 16 gearbeitet, also 256 möglichen Werten. Wirklich wahrnehmbar ist der Unterschied nicht, da genug Frequenzen in verschiedenen Intensitäten

vorhanden sind um ein variables Signal zu erzeugen. Auch die Histogramme unterscheiden sich nicht stark genug voneinander, daher habe ich auf eine Abbildung verzichtet.

5.4.2 Perzentil

Hierüber wird festgelegt, wie viel Korn in der Kornmatrix vorhanden ist. Abbildung 5.25 zeigt den Unterschied zwischen zwei verschiedenen Perzentilen. Der Höreindruck ist nicht grundlegend anders, die Frequenzverteilung ebenfalls nicht. Auf der rechten Seite sieht es nur durch die leicht unterschiedliche Achsenskalierung etwas dichter bzw. auch lauter aus. Es ist kein großer „Ausreißer“ mehr vorhanden, der die Achsenskalierung erweitert.

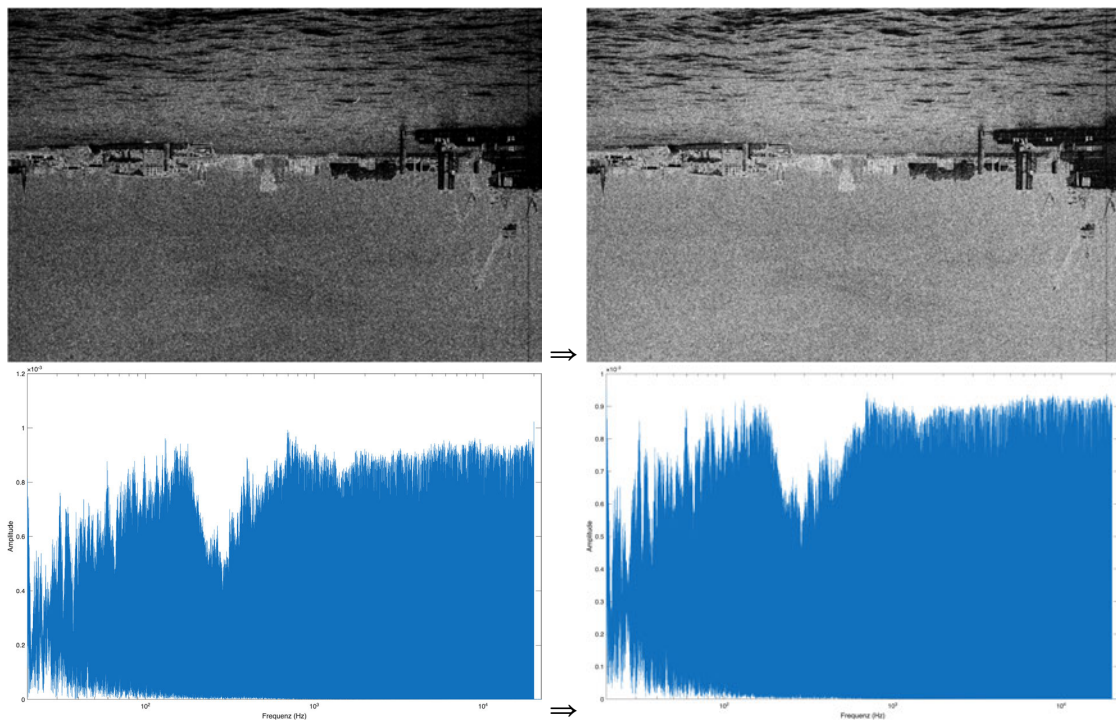


Abbildung 5.25: Reduktion von edgePercentile von 80 auf 60. Somit werden anstatt 20% 40% der vom Sobel-Algorithmus erkannten Helligkeitsunterschiede als Korn wahrgenommen

5.4.3 Anzahl Frequenzbänder

Über diesen Parameter wird direkt Einfluss darauf genommen, wie stark das ausgespielte Signal als Rauschen wahrgenommen wird. Festgelegt wird dies in der aktuellen Version über die vertikale Auflösung des Eingangsbildes. Verringern der Frequenzen wäre einfach möglich, indem Pixel zur Analyse ausgelassen werden, beim künstlichen Erhöhen der Frequenzbänder

müsste man wieder zwischen den Frequenzen interpolieren. In meinem verwendeten Programmcode ist dies nicht vorgesehen, da bei der von mir verwendeten Scanauflösung schon genügend Frequenzen vorhanden sind.

5.4.4 Framedauer

Bei dieser Untersuchung wurde das Perzentil wieder auf 80 gesetzt und die Blockgröße blieb bei 16×16 . Bei allen bisherigen Beispielen habe ich eine Framedauer von 10 ms verwendet. Was mir sofort beim Verändern der Framedauer aufgefallen ist, ist die sich deutlich unterscheidende Rechenzeit. Während ich bei meinen Standardeinstellungen eine Rechenzeit von knapp 55 s hatte, hat das Generieren von der Datei mit 100 ms Framedauer schon über 16 min gedauert. Beim Verringern der Framedauer auf 1 ms betrug die Rechenzeit nur noch knapp 10 s. Dies hängt mit dem großen Aufwand zusammen, für Tausende von Frequenzen die passenden Amplituden für jeden Zeitpunkt zu berechnen. In Abbildung 5.26 sieht man die Histogramme und Spektrogramme dieser beiden Extremfälle.

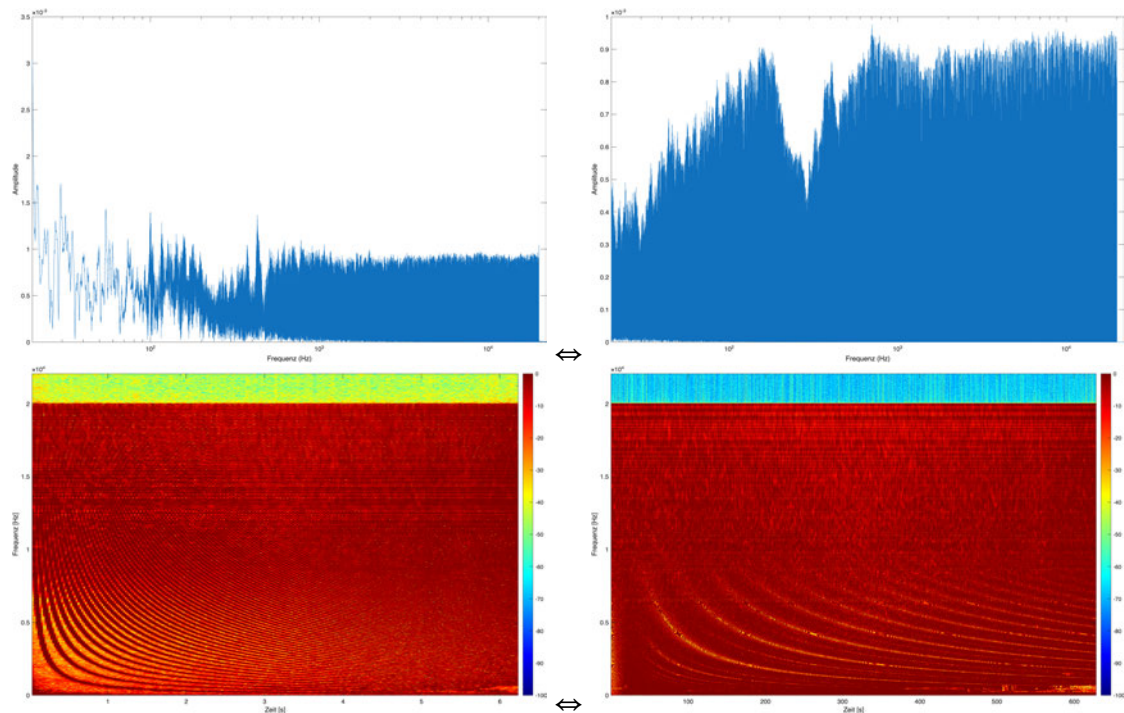


Abbildung 5.26: Histogramm und Spektrogramm bei der Verwendung von anderen Framedauern. Auf der linken Seite sieht man eine Framedauer von 1 ms, auf der rechten Seite sind es 100 ms.

5.5 Weitere Anwendungsmöglichkeiten

Zusätzlich zur Generierung von Rauschsignalen könnte die aus dem Filmkorn gewonnene Datenstruktur als Grundlage für die Erzeugung von Zufallswerten verwendet werden. Die räumliche Verteilung der Körner entsteht durch physikalische und chemische Prozesse bei der Herstellung, Belichtung und Entwicklung des Films. Sie folgt keinem regelmäßigen Pixelraster und wurde nicht durch einen im Computer ausgeführten Pseudozufallsalgorithmus erzeugt. Die aus einem Scan gewonnenen Werte können daher als physikalisch entstandene Zufallsstruktur betrachtet werden. Für die Erzeugung der Werte kann erneut die in Abschnitt 4.2.2 beschriebene Blockauswertung verwendet werden. Bei einem binären Block mit der Kantenlänge b wird die Anzahl der als Korn erkannten Pixel gezählt. Der mögliche Wertebereich reicht dabei von $0 \leq z \leq b^2$, wobei z die Anzahl der Kornpixel innerhalb des Blocks bezeichnet. Bei einer Blockgröße von 8×8 entstehen somit theoretisch 65 mögliche Werte zwischen 0 und 64. Die Anzahl der möglichen Ergebnisse ist deshalb $N = b^2 + 1$. Die Ergebnisse sind allerdings nicht zwangsläufig gleichverteilt. Da die Kornpixel innerhalb eines Blocks mit einer gewissen Wahrscheinlichkeit auftreten, sind mittlere Zählwerte meist wahrscheinlicher als die Extremwerte 0 und b^2 . Die bloße Anzahl der Kornpixel eignet sich daher nicht unmittelbar zur Erzeugung gleichverteilter Zufallszahlen. Vor einer weiteren Verwendung müssten die statistische Verteilung untersucht und mögliche Verzerrungen durch eine nachgelagerte Verarbeitung reduziert werden. Zu berücksichtigen ist außerdem, dass ein einmal digitalisiertes Negativ keine fortlaufend neue Zufallsquelle darstellt. Wird derselbe Bildausschnitt erneut mit denselben Parametern verarbeitet, entsteht dieselbe Zahlenfolge. Die Wiederholbarkeit kann für künstlerische oder technische Anwendungen von Vorteil sein, da sich ein bestimmtes Ergebnis rekonstruieren lässt. Für sicherheitskritische Anwendungen ist sie jedoch problematisch, sobald das Ausgangsbild oder ein Teil der erzeugten Folge bekannt wird.

Literatur

Doyle, J. A., & Evans, A. C. (2018). What Colour is Neural Noise? *arXiv*.

Genuit, K. (2014). Status of Psychoacoustics in Noise Analysis.

How do lava lamps help with internet encryption? (n. d.). Verfügbar 11. Juni 2026 unter <https://www.cloudflare.com/learning/ssl/lava-lamp-encryption/>

ISO 226:2023 Acoustics – Normal equal-loudness-level contours. (2023). International Organization for Standardization.

Semantisches Differential. (n. d.). Verfügbar 23. Juli 2026 unter <https://www.spektrum.de/lexikon/psychologie/semantisches-differential/14065>

Anhang

Bildquellen

Alle Histogramme zur Frequenzanalyse wurden mit dem Code `frequenzPlot.m` generiert. Die Spektogramme mit dem Code `Spektogramm.m`.

Alle Beispielfiguren wurden von mir persönlich auf Film aufgenommen und auch von mir gescannt.

Das Bild `Gauss_Rauschen.jpg` wurde mit Affinity Photo v. 1.10.8 erstellt und für die Analysen in Abbildung 5.2 und 5.5 verwendet.

Die restlichen Abbildungen mit Graphen, Beispielen und schematischen Darstellungen wurden von ChatGPT-5.5 erstellt. Dabei wurden meine Beschreibungen der Grafiken in PGF/TikZ-Code übersetzt und zusammen mit dem $\text{\LaTeX}$ -Code kompiliert.

Programmcode

Es sind mehrere Matlab-Skripte für verschiedene Teile der Untersuchung entstanden

- `frequenzPlot.m` und `Spektogramm.m`: Erzeugen von Histogrammen bzw. Spektogrammen
- `edgeDetection.m`: Kanten-/Filmkornerkennung per Sobel-Algorithmus wie in Kapitel 4.1.1 beschrieben.
- `filmkornHochpass.m`: Kanten-/Filmkornerkennung per Weichzeichner wie in Kapitel 4.1.2 beschrieben.
- `imageToNoise.m`: Rauscherzeugung aus einem Bild mit linearer Interpolation wie in Kapitel 4.2.1 beschrieben.
- `imageToNoiseSpectralAutoDuration.m`: Rauscherzeugung aus einem Bild wie in Kapitel 4.2.2 beschrieben. Diese Version hat den Phasenreset in jedem Frame.
- `imageToNoiseSpectralAutoDurationImproved.m`: Rauscherzeugung aus einem Bild wie in Kapitel 4.2.2 beschrieben. Diese Version hat den kontinuierlichen Phasengang und wurde für die finalen Untersuchungen und Vergleiche genutzt.

- `amplitudeModifier.m`: Verrechnen des Rausch- und Musiksignals. Es werden alle drei in Abschnitt 4.3 beschriebenen Varianten generiert und abgespeichert.

Audiobeispiele

Ordner „Rauschen“

- Es gibt mehrere Dateien die nach dem Muster `xjpg_Spektral_(Improved)_Blockx_Pctx` benannt sind. Dabei steht am Anfang der Name der Bilddatei, die zur Rauschgenerierung genutzt wurde, die Blockgröße und das Perzentil. Wenn noch ein Improved vorhanden ist, wurde es mit der neueren Version mit kontinuierlicher Phase generiert.
- `white_noise.wav`, `pink_noise.wav`, `brown_noise.wav`: rotes, rosa und weißes Rauschen welches in Audacity für die Abbildungen 5.11, 5.12 und 5.13 generiert wurde.
- `Platte_Knacken.wav`: Aufnahme des Teils der Schallplatte ohne Musik, welche für die Abbildungen 5.23 und 5.24 verwendet wurde.
- `tape-hiss.mp3`: Von [pixabay](#) geladene Aufnahme von Bandrauschen, die für Abbildung 5.20 verwendet wurde.
- `rauschenLinfewFreq.wav` und `rauschenLinmoreFreq.wav`: Generiertes Rauschen mit der in 4.2.1 beschriebenen Methode mit wenigen und mit vielen Interpolationsschritten

Ordner „Musik“

- `Musik_Digital_Cut.wav`: Digitale, gekürzte Version des in Kapitel 5.3 untersuchten Musikstücks
- `Musik_Platte.wav`: Von mir von Schallplatte aufgenommene Version desselben Musikstücks
- `rauschen+musik_add`: Mit `amplitudeModifier.m` kombinierte Version von `Musik_Digital_Cut.wav` und `6jpg_Spektral_Improved_Block16_Pct80.wav` nach dem Muster $S_r + S_n$
- `rauschen+musik_mult`: Mit `amplitudeModifier.m` kombinierte Version von `Musik_Digital_Cut.wav` und `6jpg_Spektral_Improved_Block16_Pct80.wav` nach dem Muster $S_r * S_n$
- `rauschen+musik_add`: Mit `amplitudeModifier.m` kombinierte Version von `Musik_Digital_Cut.wav` und `6jpg_Spektral_Improved_Block16_Pct80.wav` nach dem Muster $S_n + (S_r + S_n)$

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

Rauschsignalsynthese aus Filmkorn

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.



Hamburg, 27. Juli 2026