HOCHSCHULE OSNABRÜCK

UNIVERSITY OF APPLIED SCIENCES

Fakultät Ingenieurwissenschaften und Informatik

Studiengang Medieninformatik

Modul Audio- und Videotechnik

Praktikumsbericht

Meilenstein 5
Video- und Audiokompression, Multimedia-Container

Wintersemester 2020/2021

Dozent:

Prof. Dr. Julius Schöning

Verfasser:

Patrick Felschen, Julian Voß

Matrikelnummer:

932056, 934505

Datum der Abgabe:

07.01.2021

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Aufgabe: Diskrete Kosinustransformation (DCT)

In dieser Aufgabe wird eine diskrete Kosinustransformation mittels MATLAB programmiert, um ein PNG-Bild zu komprimieren. Zunächst wird ein Bild eingelesen und zu double Werten konvertiert. Nachdem das Bild in 32x32 Pixel-Blöcke eingeteilt wurde, wird auf alle Blöcke eine diskrete Kosinustransformation angewendet (in MATLAB mittels dct2(<block>)). Zudem wird eine 32x32 Pixel Quantisierungsmatrix erstellt, mit welcher die Blöcke punktweise dividiert und anschließend gerundet werden. Um das Bild zu rekonstruieren, wird punktweise mit der zuvor erwähnten Quantisierungsmatrix multipliziert und die inverse Funktion idct2(<block>) blockweise angewendet.

Es handelt sich bei der diskreten Kosinustransformation um eine Quellenkodierung, da durch die Quantisierung Informationen verloren gehen.

Durch die Transformation wird die Dateigröße des rekonstruierten Bildes 75 Kilobyte kleiner als die Größe des originalen Bildes.

Diese Methode unterscheidet sich darin, dass die Farbwerte erhalten bleiben. Anstatt fehlerhafter Farbübergänge, treten hier, je nach Qualität, mehr oder weniger Kacheln auf.

2 Aufgabe: Digitalisierung von Videosignalen

In diesem Aufgabeteil wird die Auswirkung des Nyquist-Shannon-Abtasttheorem auf einer Rad-Rotation in einem Video untersucht. Zunächst wird mittels MATLAB ein „VideoWriter“ erstellt, mit welchem es möglich ist eine *.mp4-Datei zu erstellen. Durch die gegebene Framerate von 25 fps (Bilder pro Sekunde) und der Videolänge von 3 Sekunden ergibt sich eine Gesamtbildanzahl von 75 Bildern. Pro Bild werden in einer For-Schleife sechs unterschiedlich rotierte Räder angeordnet und ins Video hinzugefügt.

Rotation pro Bild (°)	Beschreibung
$-\frac{360}{25}$	1 Umdrehung pro Sekunde
$-\frac{360}{25} \cdot 2$	½ Umdrehung pro Sekunde
$-\frac{360}{25} \cdot 26$	Mehr als 1 Umdrehung pro Frame/Bild
-360	1 Umdrehung pro Frame
$-\frac{360}{25} \cdot 24{,}5$	Fast 1 Umdrehung pro Frame
$-\frac{360}{25} \cdot 24$	Weniger als 1 Umdrehung pro Frame

Tabelle 1: Rad Rotation

Dadurch, dass sich das Bild des 5. Rads pro Frame fast einmal ganz dreht, wirkt es wie eine geringe Verschiebung in die entgegengesetzte Richtung. Die Abbildungen unten zeigen eine Drehung um 352,8° pro Frame. Pro Sekunde dreht sich das Rad also um 180° in die negative Richtung. (s. Abbildung 1-3)

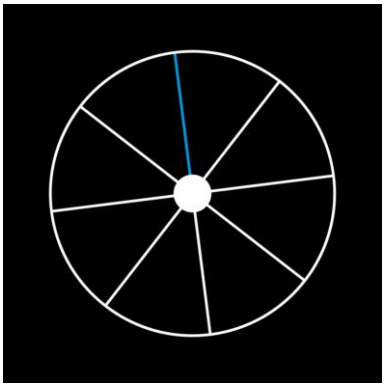

Abbildung 3: Frame 1

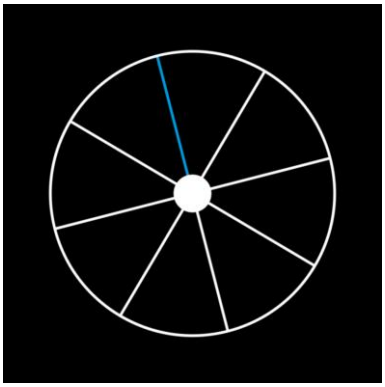

Abbildung 2: Frame 2

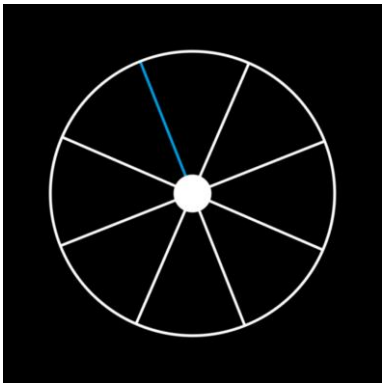

Abbildung 1: Frame 3

3 Aufgabe: Multimedia-Container

Mittels einer *.ass-Datei werden in dieser Aufgabe passende Untertitel zu den einzelnen Rädern erstellt. Mit der Software „MKVToolNix“ lassen sich die Video- und die Untertiteldatei zusammenfügen. Die *.ass-Datei beinhaltet Informationen über die Videodatei, wie die Bildhöhe und Bildbreite. Anschließend wird ein Untertitel Style erstellt, welcher im Video verwendet wird. Für den Style wird der Name, die Schriftart, Schriftgröße und die Primärfarbe festgelegt. Dieser Style lässt sich nun mit einem bestimmten Text in verschiedene Positionen einfügen. Der Untertitel wird hier über die gesamten 3 Sekunden angezeigt.

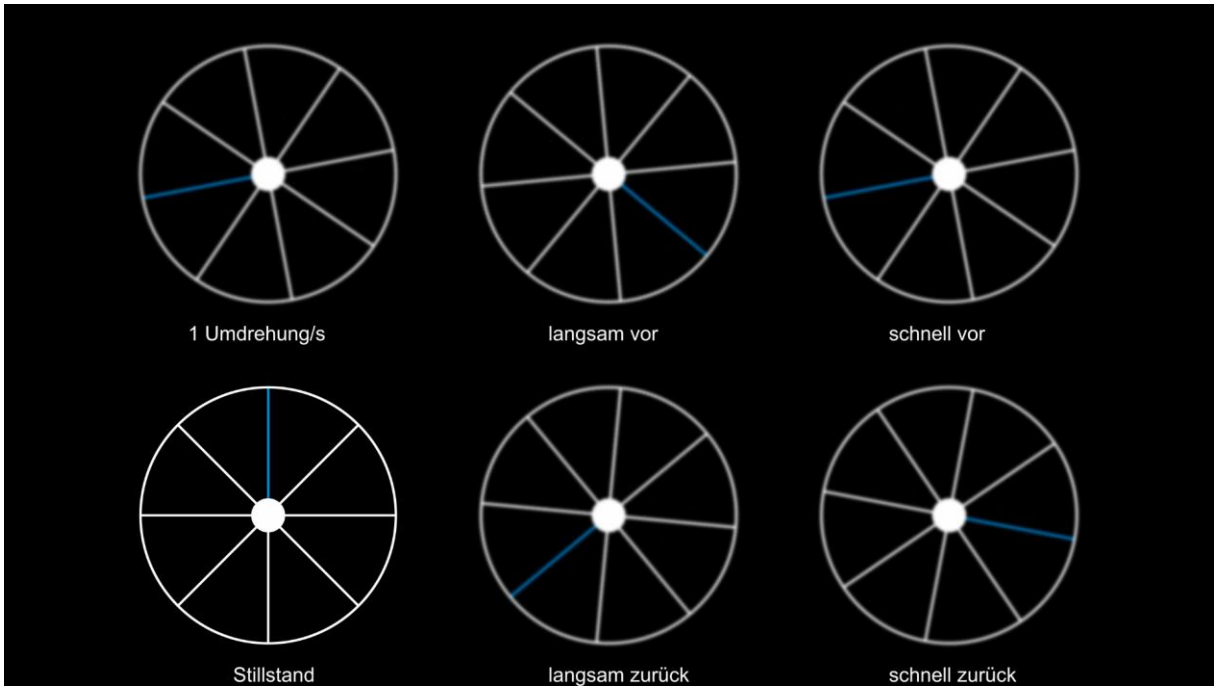

Abbildung 4: Videoausschnitt mit Untertitel

4 Aufgabe: Audiokompression

Zunächst wird in dieser Aufgabe mittels MATLAB eine Audiodatei eingelesen. Diese wird in 0,01 Sekunden Schritte in einer Matrix formatiert. Auf jeden dieser Abschnitte wird eine Fourier Transformation mittels FFT() angewandt. Das gleiche Verfahren wird mit zwei Vergleichstönen vorgenommen. Nach der Transformation lassen sich die einzelnen Abschnitte miteinander vergleichen.

4.1 Töne eines Klaviers

Um ein richtiges Ergebnis beim Vergleichen der Klaviertöne zu erhalten, wird die Differenz eines Fourier transformierten Klaviertons (d.flac bzw. g.flac) mit einem Abschnitt der ausgangs Audiodatei verglichen. Der Betrag der Differenz darf hier nicht größer als die Toleranz sein, welche hier auf den Wert 2,0 eingestellt wurde. (s. Anhang 2)

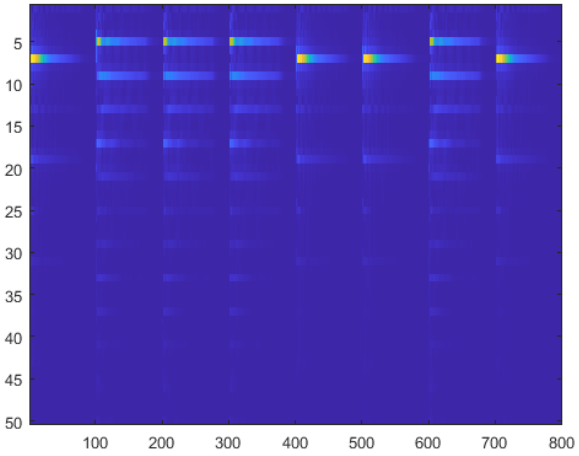

Abbildung 5: FFT von acht Klaviertönen

4.2 Gesprochener Text

Da die Länge der eingelesenen Audiodatei nicht durch die Anzahl der aufgeteilten Tonabschnitte (Abschnitte von 0,01 Sekunden) teilbar ist, werden hier die einzelnen Töne und die Audiodatei selbst mit Nullen aufgefüllt, sodass ohne Rest durch die Menge der Tonabschnitte geteilt werden kann. Der Vergleich der Töne in der Audiodatei verläuft ähnlich wie zuvor. Die Toleranz muss jedoch angepasst werden, um sich einem richtigen Ergebnis anzunähern. Bei einer Audiodatei mit einer synthetischen, weiblichen Stimme und dazu passenden „Ja“ und „Nein“ Töne, wird eine Toleranz von 15 verwendet. Werden nun männliche Töne benutzt, liefert eine Toleranz von 14 ein besseres Ergebnis. Eine Toleranz von 14 liefert bei einer menschlichen Stimme das beste Ergebnis, unabhängig von den verwendeten Tondateien. (s. Anhang 3)

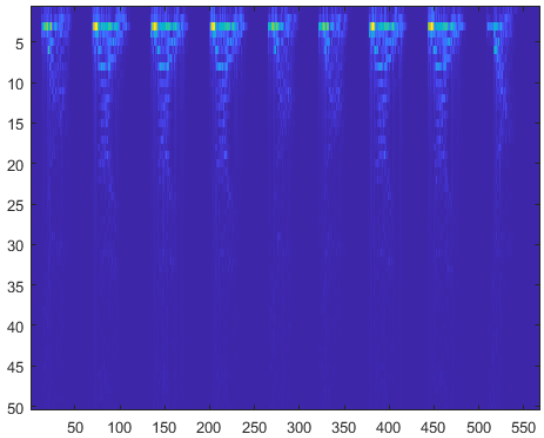

Abbildung 6: FFT von synthetischer Stimme

4.3 Probleme und Verbesserungen

Je unterschiedlicher die einzelnen Töne von eigentlich gleichen Informationen sind, desto schwieriger ist es diese zu identifizieren. Dies ist gerade bei der menschlichen Stimme ein Problem, da kein Ton gleich dem anderen ist.

Ein möglicher Ansatz zur Verbesserung der Erkennung von Stimmen, wäre der Einsatz einer auf menschliche Stimmen trainierten künstlichen Intelligenz, welche die leichten Abweichungen vom Gesprochenen erkennt.

VI. Anhang

Anhang 1: Advanced SubStation Alpha

```
[Script Info]
; created by Patrick Felschen, Julian Voss
; Lecture: Audio und Videotechnik
PlayResX: 2268
PlayResY: 1512

[V4 Styles]
Format: Name, Fontname, Fontsize, PrimaryColour
Style: Default,Arial,50,16777215

[Events]
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 1,0200,0000,0756,,1 Umdrehung/s
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 2,1000,0000,0756,,langsam vor
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 3,1756,0000,0756,,schnell vor
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 4,0300,0000,0000,,Stillstand
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 5,1000,0000,0000,,langsam zurück
Dialogue: 0,0:00:00.00,0:00:03.00,Default,Block 6,1756,0000,0000,,schnell zurück
```

Anhang 2: Audiokompression von Klaviertönen

```
%created by Patrick Felschen, Julian Voss
%Lecture: Audio und Videotechnik
clc;
clear;
close all;

%% Einlesen
ton_d = audioread('d.flac');
ton_g = audioread('g.flac');
audio = audioread('dgggddgd.flac');

sampleRate = 44100;
fragment_size = sampleRate / 100;

%% Einteilen
ton_d_mat = reshape(ton_d, fragment_size, []);
ton_g_mat = reshape(ton_g, fragment_size, []);
audio_mat = reshape(audio, fragment_size, []);

%% Fourier transformieren
fft_ton_d = abs(fft(ton_d_mat, [], 1));
fft_ton_g = abs(fft(ton_g_mat, [], 1));
fft_audio = abs(fft(audio_mat, [], 1));

%% Vergleichen

i = 1;
result = '';
tolerance = 2.0;

while i <= size(fft_audio, 2)

    diff2ton_d = abs((fft_audio(1:50,i:i+99)) - (fft_ton_d(1:50,:)));
    diff2ton_g = abs((fft_audio(1:50,i:i+99)) - (fft_ton_g(1:50,:)));

    if(max(diff2ton_d) <= tolerance)
        result = append(result, ' d');
    end

    if(max(diff2ton_g) <= tolerance)
        result = append(result, ' g');
    end

    i = i + 100;
end

%% Plot
result

figure;
plot(audio);
figure;
imagesc(fft_audio(1:50,:));
```

Anhang 3: Audiokompression von Gesprochenen

```
%created by Patrick Felschen, Julian Voss
%Lecture: Audio und Videotechnik
clc;
clear;
close all;

%% Einlesen

%ton_ja = audioread('ja_f.flac');
ton_ja = audioread('ja_m.flac');

%ton_nein = audioread('nein_f.flac');
ton_nein = audioread('nein_m.flac');

%audio = audioread('synthetic_Ja_Nein_Nein_Nein_Ja_Ja_Nein_Nein_Ja.flac');
audio = audioread('voice_Ja_Nein_Nein_jein_Ja_Ja_Nein_Nein_Ja_Nein.flac');

sampleRate = 44100;
fragment_size = sampleRate / 100;

%% Einteilen
ton_ja(end+1:fragment_size*ceil(numel(ton_ja)/fragment_size)) = 0;
ton_nein(end+1:fragment_size*ceil(numel(ton_nein)/fragment_size)) = 0;
audio(end+1:fragment_size*ceil(numel(audio)/fragment_size)) = 0;

ton_ja_mat = reshape(ton_ja, fragment_size, []);
ton_nein_mat = reshape(ton_nein, fragment_size, []);
audio_mat = reshape(audio, fragment_size, []);

%% Fourier transformieren
fft_ton_ja = abs(fft(ton_ja_mat, [], 1));
fft_ton_nein = abs(fft(ton_nein_mat, [], 1));
fft_audio = abs(fft(audio_mat, [], 1));

%% Vergleichen
i = 1;
result = '';
tolerance = 14;

ja_size = size(fft_ton_ja, 2);
nein_size = size(fft_ton_nein, 2);

while i+ja_size-1 <= size(fft_audio, 2)

    diff2ton_ja = abs((fft_audio(1:50,i:i+ja_size-1)) - (fft_ton_ja(1:50,:)));
    diff2ton_nein = abs((fft_audio(1:50,i:i+nein_size-1)) - (fft_ton_nein(1:50,:)));

    if(max(diff2ton_ja) <= tolerance)
        result = append(result, ' ja');
        i = i + ja_size;
    elseif(max(diff2ton_nein) <= tolerance)
        result = append(result, ' nein');
        i = i + nein_size;
    else
        i = i + 1;
    end

end

%% Plot
result

figure;
plot(audio);
figure;
imagesc(fft_audio(1:50,:));
```