HOCHSCHULE OSNABRÜCK

UNIVERSITY OF APPLIED SCIENCES

Fakultät Ingenieurwissenschaften und Informatik

Studiengang Medieninformatik

Modul Audio- und Videotechnik

Praktikumsbericht

Meilenstein 6
Videokomprimierung, Spracherkennung

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: Interframe-Kodierung mit Forward Motion Estimation

In dieser Aufgabe wird zuerst eine Videodatei eingelesen. Die einzelnen Frames des Videos werden nacheinander durchlaufen und jedes fünfte als I-Frame genutzt. Es wird beim Durchlaufen jeweils die Differenz des aktuellen Frames berechnet. Jedes Frame wird in 40x40 Pixel Blöcke eingeteilt und jeweils wird die durchschnittliche Abweichung berechnet. Nun solle im Umkreis von 20 Pixel um den aktuellen Block eine geeignete Verschiebung kalkuliert werden. Die Umsetzung mittels zwei For-Schleifen wurde im Ergebnis-Script nicht mit richtigem Ergebnis gelöst. Zuletzt solle das Ergebnisvideo aus den zuvor erstellten P-Frames wieder zusammengesetzt werden.

1.1 B-Frames

B-Frames werden durch Forward Motion Estimation und Backward Motion Estimation aus I- bzw. P-Frames erstellt. Somit können aus P-Frames rückwirkend Bildern Informationen übergeben werden. Werden zusätzlich zu den P-Frames noch B-Frames verwendet, verringert sich die zu übertragende Bandbreite. Die Qualität ist hierbei gleichbleibend.

2 Aufgabe: Objektbasierte Kodierung

Zunächst wird der Quellcode des gegebenen MATLAB-Scripts, der objektorientierten Kodierung Zeilenweise beschrieben.

Zeile	Beschreibung
5	Mittels eines „VideoReader“ eine Videodatei (*.mp4) einlesen und in der Variable „v“ speichern.
6	Mittels eines „VideoReader“ eine Videodatei (*.mp4) einlesen und in der Variable „s“ speichern.
8	Eine 4D Matrix wird erstellt, die ersten 3 Dimensionen spiegeln das Video wider (Höhe, Breite, Farbkanäle).
10	Ein Schwellenwert mit dem Wert 0,6 wird festgelegt.
11	Eine 3x1 Matrix mit den Werten 0,1,0 wird erstellt. Der erste Wert steht für den roten, der zweite für den grünen und der dritte für den blauen Farbkanal.
12	Die Schleife läuft so lange, wie das Video Frames hat.
13	Aktuelles Frame wird als Double Matrix abgespeichert.
14	Die einzelnen Werte der Farbkanäle werden quadriert und von den Werten aus der „Values“ Matrix subtrahiert. Diese Werte werden addiert und anschließend wird die

	Quadratwurzel angewendet. Nun wird geprüft, ob der berechnete Wert größer oder kleiner als der Schwellenwert ist. Abhängig davon ist die Ausgabe eine 1080x1920 Matrix aus logical Werten (0,1).
15	Für jeden Wert in Img wird nun mittels der Mask entschieden, welcher Farbkanal-punkt auf 0 gesetzt wird. Enthält somit die Mask eine logische 1, wird der Farbkanal-punk an dieser Stelle 0. Diese Ergebnismatrix wird in der 4. Dimension vom Video abgespeichert.
16	Der Frame aus Zeile 15 wird angezeigt.
18	Ende der while-Schleife
20	Mittels „VideoWriter“ wird eine Ausgangsvideodatei (*.mp4) erstellt und in der Variab-le „videoOut“ gespeichert.
21	Die Framerate wird auf die des Videos aus Zeile 5 festgelegt.
22	Öffnen des Ausgangsvideos.
24	Schleife von 1 bis zur Größe der 4. Dimension des Eingangvideos.
25	Das t-te Frame der 4. Dimension wird in der Variable „im“ zwischengespeichert.
26	Maske erstellen aus logischen Werten. 1 = alle Farbkanäle an einem Punkt sind gleichzeitig 0. Die Maske enthält somit Informationen, wo sich schwarze Bildpunkte befinden.
27	Die Bildpunkte in der Scene s werden anhand der Mask Matrix schwarz gesetzt.
28	Das Ursprungsframe wird nun auf das in Zeile 27 erstellte Frame addiert.
29	Das in Zeile 28 erstellte Frame wird in das Ausgangsvideo abgespeichert.
30	Das in Zeile 28 erstellte Frame wird angezeigt.
31	Ende der for-Schleife
32	Schließen des Ausgangsvideo.

Tabelle 1: Beschreibung des Scripts

2.1 Modifizierung des initialen Scripts

Das erste Frame wird zwischengespeichert, da sich auf diesem Bild noch kein Objekt befindet, somit steht der Hintergrund für sich allein. In der Berechnung der Maske wird nun jeder quadrierte Farbkanal vom aktuellen Frame mit dem Wert des entsprechenden Farbkanals des ersten Frames subtrahiert. Die Quadratwurzel der addierten Farbkanäle wird nun wieder mit dem Schwellenwert, welcher auf 0,9 geändert wurde, verglichen. Der nachfolgende Ablauf des Skripts wurde nicht geändert. (s. Anhang 1)

2.2 Komprimierung des Hintergrundes

Durch eine Komprimierung des Hintergrundes, wird die Videodateigröße, mit steigendem Faktor, des Hintergrundfilters geringer. Je höher die Komprimierung, desto unschärfer wird der Hintergrund. Bei einem Faktor von 16 ist nicht mehr viel auf dem Hintergrund zu erkennen. Vorteilhaft bei dieser Form der Komprimierung ist, dass der Vordergrund nicht verändert wird und somit scharf bleibt. (s. Anhang 1, 2)

3 Aufgabe: Spracherkennung mit künstlichen neuronalen Netzwerken

Im letzten Meilenstein werden zur Erkennung von Worten Spektren miteinander verglichen. Fallen diese unter einer gewissen Toleranzgrenze, wird nichts erkannt. Der Unterschied zum künstlichen neuronalen Netzen, liegt darin, dass der Spektren Vergleich von einer trainierten „Intelligenz“ übernommen wird. Je mehr das Netzwerk mit Daten trainiert wurde, umso besser kommt es zu Erkennungen.

3.1 Vor- und Nachteile von künstlichen neuronalen Netzen

Ein Nachteil für den Einsatz von künstlichen neuronalen Netzen ist, dass vor dem Einsatz ein langer Trainierprozess des Netzes, durchlaufen werden muss.

VI. Anhang

Anhang 1: Modifiziertes MATLAB Script

```
% Programmzeilen mit * gekennzeichnet muessen fuer die Aufgabe 2.a nicht im
% Projekbereicht erklaert werden.

clear %*
close all %*
v = VideoReader(fullfile('videos', 'combineColoredBackground.mp4'));
s = VideoReader(fullfile('videos','scene.mp4'));

video=zeros(v.Height,v.Width,3,floor(v.Duration/(1/v.FrameRate))-2,'uint8');
t=1;%*
threshold = 0.9 ;
first_frame = im2double(read(v, 1));

while hasFrame(v)
    Img = im2double(readFrame(v));
    mask = sqrt(first_frame(:,:,1) - Img(:,:,1).^2 + first_frame(:,:,2) -
Img(:,:,2).^2 + first_frame(:,:,3) - Img(:,:,3).^2) < threshold;
    video(:,:,:,t) = imoverlay(im2uint8(Img),mask,[0 0 0]);
    imshow(video(:,:,:,t));
    t=t+1;%*
end

videoOut = VideoWriter('result.mp4','MPEG-4');
videoOut.FrameRate=v.FrameRate;
open(videoOut);
moveX=0;%*

filter = fspecial('average', 16);

for t=1:1:size(video,4)
    im = video(:,:,:,t);
    mask = im(:,:,1)+im(:,:,2)+im(:,:,3)==0;

    background = imfilter(readFrame(s), filter);
    frame=imoverlay(readFrame(s),~mask,[0 0 0]);

    frame=frame+im;
    writeVideo(videoOut,frame);
    imshow(frame);
end
close(videoOut);
```

Anhang 2: Hintergrund Komprimierung

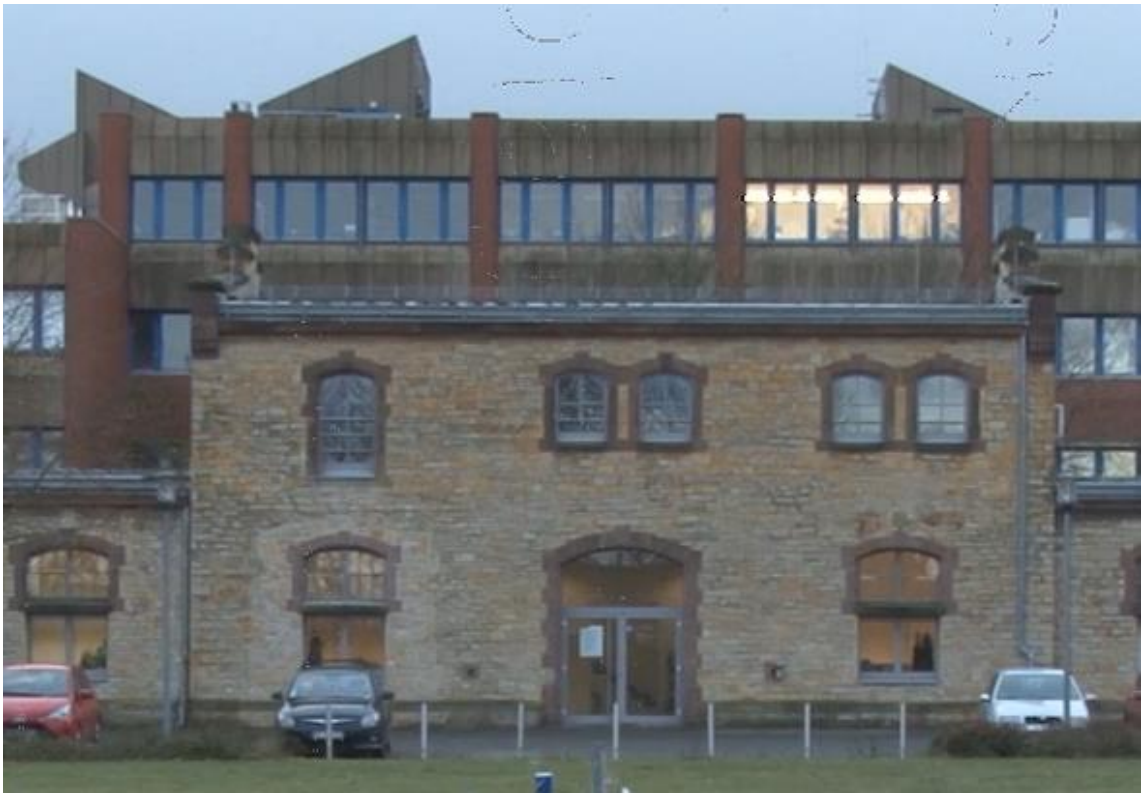

Abbildung 1: Original Videoausschnitt

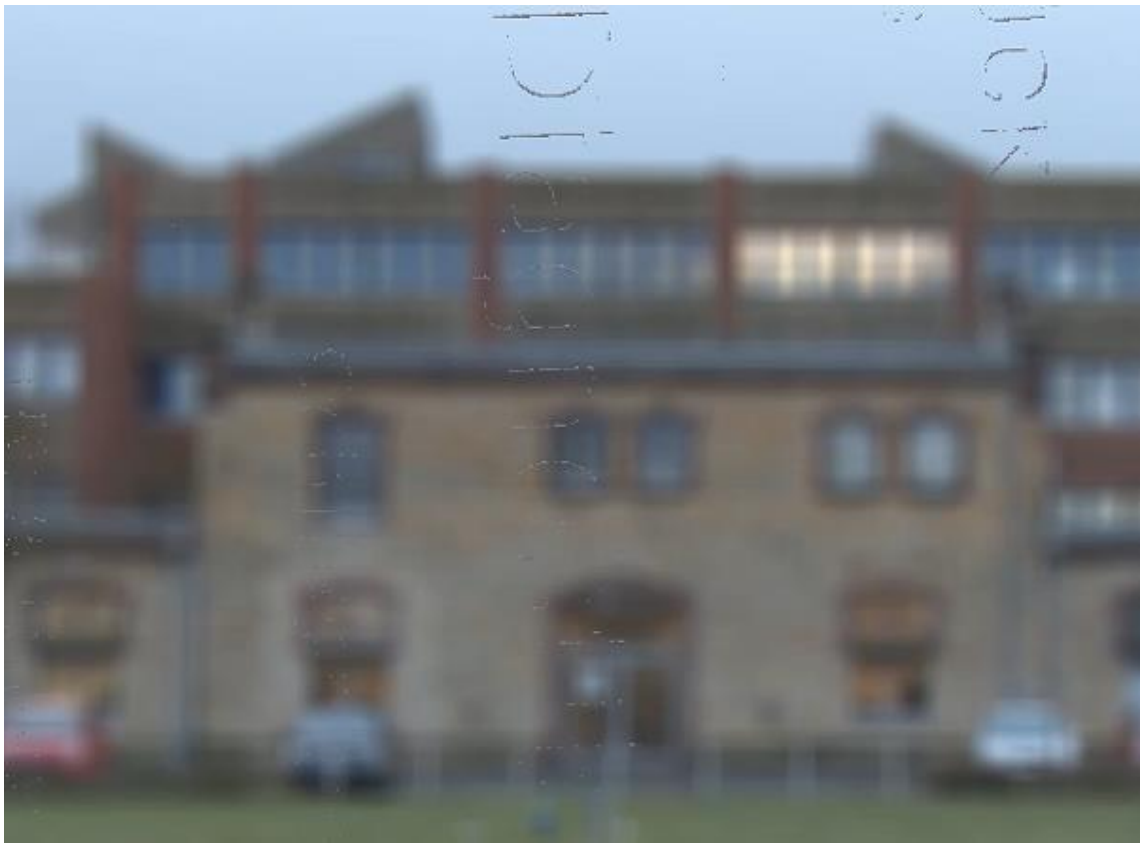

Abbildung 2: Videoausschnitt mit Faktor 16