Dokumentation Video Filtering, Alexander Ries

Benutzung

Das System ist eine HTML5 Seite, die eine Benutzeroberfläche zum Filtern eines Videos anzeigt. Sie besteht aus einem „Upload“ Button, einem „Apply Filter“ Button, ein Textfeld für einen Skalierungswert, einem Video Player, einer 5x5 Matrix aus Textfeldern und einem Bereich in dem das Resultat des gefilterten Videos angezeigt wird.
Vor dem Verwenden des Systems, muss ein Video vom Lokalen Dateiensystem ausgewählt werden. Das erfolgt Mittels Drücken des „Upload“ Buttons, der ein Fenster erscheinen lässt, in dem eine Video Datei ausgewählt werden kann. Nachdem ein Video ausgewählt wurde, kann das Video in dem Player abgespielt werden und ein Kernel Filter auf das Bild des Videos angewandt werden.
Durch Ändern der Werte in den Textfeldern der Matrix und durch anpassen des Skalierung Werts, kann ein Benutzerdefinierter Filter auf die Frames des Videos angewandt werden. Nach jedem Ändern der Matrix oder der Skalierung, muss der „Apply Filter“ Button gedrückt werden um einen Effekt hervorzurufen. Das Ergebnis wird unter dem Videoplayer angezeigt.

Strukturierung

Das Projekt beinhaltet, eine HTML Seite in der fast alle Elemente enthalten sind. Einzig die Textfelder der Matrix werden dynamisch durch JavaScript erstellt. Ferne ist die Logik des Projekts in einer main.js JavaScript Datei und zwei weiteren JavaScript Dateien, *KernelInput.js* und *KernelFilter.js*, verteilt, welche als Klassen verwendet werden. Beim Laden der Seite wird in main.js die *init()* Methode aufgerufen. In Ihr werden Variablen initialisiert und Verknüpfungen zu Elementen in der DOM hergestellt. Ferner wird ein FileInputChangeListener auf das Auswählen eines Videos gesetzt und ein weiterer Listener auf das Abspielen

des Videos im Player. Es wird die Klasse **KernelInput** initialisiert, welche die Textfelder der Matrix erstellt und in die DOM einfügt. Zuletzt werden Default Werte für die Matrix und die Scalierung aus der KernelInput Klasse ausgelesen und ein Click Listener für den „Apply Filter“ Button erstellt.
Ausgelöst durch das VideoPlay Event, wird beim Abspielen die draw() Methode aufgerufen. Hier wird der aktuelle Frame in einen Hilfs Kontext gerendert und die Pixel Daten ausgelesen. Danach wird ein Filter durch aufrufen der *applyKernelFilterToFrameDataWithMatrixAndScale(frame, kernel, scale)* Methode der **KernelFilter** Klasse auf die Pixel des Frames angewandt. Hinterher wird der neue Frame in den Kontext des sichtbaren Canvas gezeichnet. Die *draw()* Methode wird alle 20 Millisekunden wiederholt, während das Video abspielt. Das wird mit dem Setzen eines Timeouts am Ende der *draw()* Methode erreicht.

Die Klasse **KernelFilter** stellt eine öffentliche Methode *applyKernelFilterToFrameDataWithMatrixAndScale(frame, kernel, scale)* bereit. Als Parameter nimmt die Methode einen Video Frame, die Werte des Kernels, gespeichert in einem Array, und die Skalierung entgegen. Nach den Werten des Kernels und der Skalierung, werden die RGB- und Alphawerte der einzelnen Pixel des Frames in der Methode verändert.
Das erfolgt nach dem Prinzip, dass ein Pixel abhängig der umliegenden Pixelwerte modifiziert wird. Ein Kernel Filter multipliziert jedes der umliegenden Rot, Gelb, Blau, und Alpha Werte eines Pixels mit einem Faktor, definiert im Kernel. Danach werden alle Werte addiert. Die Summe der Rot, Gelb, Blau und Alpha Werte werden als neuer Rot, Gelb, Blau und Alpha Wert festgelegt. Das Kernel Array repräsentiert eine 5x5 Matrix, wobei nach 5 Werten eine neue Zeile in der Matrix beginnt. Alle Rot Gelb und Blau Werte werden durch den Divisor geteilt, bevor sie einem neuen Pixel zugeordnet werden.
Danach wird der modifizierte Frame zurückgegeben. Das Verändern der Pixelwerte geschieht wie folgt: Zuerst wird ein leeres Canvas Objekt erstellt in

das die neuen Pixelwerte gespeichert werden. Dann wird durch jedes Pixel iteriert. Bei jedem Pixel iteriert die Methode durch jede Position in der Matrix. Durch Addieren des X- und Y-Pixel Position im Frame und der jeweiligen X- und Y-Position im Kernel und durch Subtrahieren der Kernel Breite wird die Position eines umliegenden Pixels bestimmt. Daraufhin wird zuerst überprüft ob sich das Pixel nicht innerhalb des Frames befindet, da dies bei Pixel am Rand des Frames der Fall sein kann. Ist dies nicht der Fall, wird der Offset des Pixels in den Frame Daten bestimmt und der Faktor an der Stelle im Kernel ausgelesen. Danach wird der Rot, Gelb, Blau und Alpha Wert aus dem Pixel ausgelesen, mit dem Kernel Faktor multipliziert und zu der Summe der jeweiligen Rot, Gelb, Blau und Alpha Werte addiert. Nach dem Iterieren durch alle Positionen im Kernel wird die Summe der Rot, Gelb und Blau Werte durch die Skalierung geteilt und die Werte, einschließlich des Alpha Werts, werden als neue Farben des Pixels in der *outputFrame* Variable gespeichert.

Die **KernelInput** Klasse erstellt während der Initialisierung 25 Textfelder, welche die Werte der Matrix repräsentieren. Bei Änderung eines Textfeld Wertes, wird die ID des Textfeldes ausgelesen und der neue Wert in der Kernel Matrix gespeichert. Gleiches Prinzip gilt für das Skalierungstextfeld. Die Klasse stellt, zwei öffentliche Methoden *getKernelValues()* und *getScaleValue()* bereit. Diese geben, die aktuellen Werte für den Kernel und die Skalierung zurück und werden beim drücken des „Apply Filter“ Buttons in der main.js Datei abgefragt.