Technische Dokumentation
Feel the sound

Einleitung

Feel the sound ist Projekt, welches im Rahmen der Vorlesung Audio-Video-Programmierung entstanden ist.
Mit dem Endprodukt kann man, mit selbst gezeichneten Kurven, seine Musik verzerren.

Dabei nutzen wir eine Kamera, C++ in Kombination mit der OpenCV Bibliothek und JavaScript mit der Web Audio API.
Die Bilder der Kamera werden mittels OpenCV in eine Kurve umgewandelt, aus der dann ein Array generiert wird welches als JSON String an den Browser geschickt wird.
Die Kurvendaten werden im Browser noch verarbeitet und dann in die WaveShaper-Node eingefügt.

C++

Der C++ Teil unseres Projektes verarbeitet den Video Stream einer Webcam mittels der OpcenCV Bibliothek in Echtzeit und sendet die ausgewerteten Daten JSON formatiert mittels dem QWebChannel von Qt an das JavaScript und damit an die WebAudio API.

Das Programm öffnet zunächst den Kamera Stream und verarbeitet jeden einzelnen Frame einzeln.
Zur Verarbeitung gehören folgende Schritte die im Programm zu Demo zwecken durchgeschaltet werden können:

Farbfilter

Der Farbfilter reduziert das zu verarbeitende Bild zunächst auf die Farbe die wir auswerten wollen.
Dazu können in dem Programm 4 Parameter manuell eingestellt werden:

- Hue: Der Winkel der gewünschten Farbe auf der HSV Farbskala geteilt durch 2.
- Saturation: die Sättigung
- Value: die Dunkelstufe
- Treshold: Die Toleranz mit der der Hue filter arbeiten soll

Saturation, Value und Treshold sind voreingestellt mit den werten die beim testen die besten Ergebnisse geliefert haben.
Der Hue Wert kann entweder Manuell eingestellt werden, oder mit dem Klick auf den “Calibrate” Button wird die Farbe eingestellt die im aktuellen Bild die größte zusammenhängende Fläche ausmacht.

Bereichsfilter

Der Bereichsfilter reduziert das Bild nun auf die größte zusammenhängende fläche der Pixel die nach dem Farbfilter übrig sind um mögliche Störungen herauszufiltern und die Kurve die wir lesen wollen deutlicher zu machen indem wir kleinere ansammlungen von pixeln mit schwarz überschreiben bis nur noch die größte übrig bleibt.

Boxfilter

Der Boxfilter Berechnet nun die minimalen und maximalen X/Y werte die erkannte Pixel beinhalten um eine box um den auszuwertenden bereich zu ziehen, damit wir im nächsten schritt bei X = 0 auch direkt einen wert finden und keine lücke haben.

Kurvenfilter

Das bis hier hin verarbeitete Bild sollte nun nurnoch die zu erkennende Kurve als weisse pixel und alles andere als schwarze pixel beinhalten da die Kurve aber in der regel zu dick ist müssen wir sie noch reduzieren.
Um am ende ein Array mit eindeutigen X/X werten zu bekommen, gehen wir mit einer Schleife alle X werte durch und bilden für jeden X wert ein mittelwert der zugehörigen Y Werte. so erhalten wir eine Kurve die 1px dick ist und gleichzeitig ein Array von 0 bis maxX mit je einem wert für Y.

Auswertung

Nun haben wir einen Array bei dem der “key” nun unser X wert ist und die “value” unser Y wert.
Das ganze speichern wir in dem von QT bereitgestellten QJsonArray, welcher es uns erlaubt den Array ganz normal zu befüllen.
Wenn der User nun auf den “Send” button klickt wird der QJsonArray an das JsonTransmitter Objekt übergeben und in einem lokalen attribut des objektes abgespeichert.

Übertragung - QWebChannel

Mit der QWebChannel Library starten wir nun einen lokalen server in unserer C++ Anwendung der auf port 12345 lauscht und es externen Programmen erlaubt mittels websocket auf unser Objekt JsonTransmitter und seine Methoden zuzugreifen.
Das Javascript ruft nun periodisch (jede sekunde) die Methode “getJSON” auf welche den QJsonArray in einen QString umwandelt und mittels websocket an das JavaScript übermittelt.

JavaScript

Im Browser hat man ein kleines Interface, mit dem Musik geladen werden kann. Diese kann über einen Play Button abgespielt oder pausiert werden. Der Waveshaper-Effekt (Distortion Effekt) kann ebenfalls mit einem Button aktiviert und deaktiviert werden.

Musik laden

Die Musik muss auf dem Server im vorgegebenen Ordner abgelegt werden, sodass der Server auch Zugriff darauf hat. Dieser wird dann für die dauer der aktuellen Sitzung geladen und kann aus einer Dropdown Liste ausgewählt und abgespielt werden.

Wenn man einen anderen Song auswählt während der aktive noch spielt, wird der aktive automatisch gestoppt.
Die Liste existiert, weil dadurch ein flexibleres Wechseln des Songs zum Vergleichen möglich ist.

Interface

Playbutton:

Wie im Beispiel aus der Vorlesung. Einfaches pausieren durch die Web Audio API und Visualisierung im Browser durch Änderung des Button Textes.

Distortion Button:

Dieser schaltet die Distortion an und aus. In der Web Audio API wird dies Source Node jeweils getrennt und einmal wird ein Stack mit der Distortion Node aufgebaut und einmal ohne.

Wenn im Browser „Distortion On“ steht heißt es, dass die Distortion eingeschaltet ist und nicht eingeschaltet wird, wenn man draufdrückt.

Master Gain:

Ein einfacher Slider, falls die Musik zu leise oder zu laut ist.

Distortion

Die WaveShaper Node akzeptiert nur Werte zwischen -1 und 1. Dementsprechend wird das Array, welche von OpenCV rübergeschickt wird noch bearbeitet.
Im ersten Schritt, werden alle Werte zum 0 Punkt hingeschoben (alle Werte sind Positiv). Dann wird – um ausgeglichene Positive und Negative Werte zu bekommen – minus die Hälfte des maximalen Wertes gerechnet. Im letzten Schritt wird durch die Hälfte des maximalen Wertes geteilt, um nur Werte zwischen -1 und 1 zu haben.