Projekt Audio- & Videoprogrammierung

Wintersemester 2019

„MALICA“

Machine **L**earning Mus**i**c **C**omposing **A**pplication

Projektbericht

Name: Sabrina Göllner

Jannes Albers

Mat.-Nr.: 2343758

2308646

Bei Herr Prof. Dr. Andreas Plaß & Jacob Sudau

Inhaltsverzeichnis

1. Einleitung

Im Rahmen des Faches Audio- & Videoprogrammierung wurde die Aufgabe gestellt, eine Anwendung zu programmieren, welche die in der Vorlesung erlernten Komponenten beinhaltet. Zu diesen Komponenten gehören die Verarbeitung von Videobildern und das Arbeiten mit Audio-Content. Ferner wie man Audio- und Video-Content bearbeitet oder manipuliert. Wir haben uns außerdem die Aufgabe gesetzt, die Machine-Learning-Bibliothek TensorFlow in unsere Anwendung zu implementieren.

2. Grundlagen

Grundsätzlich wurde uns der Umgang mit den Bibliotheken openCV und WebAudio beigebracht. OpenCV basiert auf der Programmiersprache Python und WebAudio ist mit JavaScript realisiert.

2.1 TensorFlow

TensorFlow ist eine Machine-Learning Bibliothek welche eine intelligente Objekterkennung ermöglicht. Dabei wird ein lernfähiges neuronales Netz verwendet, welches schon eine Vielzahl an zu erkennenden Objekten beinhaltet. Es ist jedoch auch möglich TensorFlow mit eigenen Objekten zu trainieren. Die genauere Funktionsweise wird später erläutert.

2.2 OpenCV

Da sich OpenCV hervorragend zur Verarbeitung von Videobildern eignet, kam eine Verwendung dieser Bibliothek sehr gelegen. In unserem Projekt verarbeitet OpenCV das Videobild der Webcam und ist direkt an TensorFlow geknüpft.

2.3 WebAudio

Die WebAudio Bibliothek ist in Malica ein wesentlicher Bestandteil. Es lädt, steuert, manipuliert und gibt den Audio-Content unseres Projektes aus. Möglich sind hier zum Beispiel auch des einfügen von vielen Effekten, welche auf die Tonspuren angewendet werden.

3. Projekt

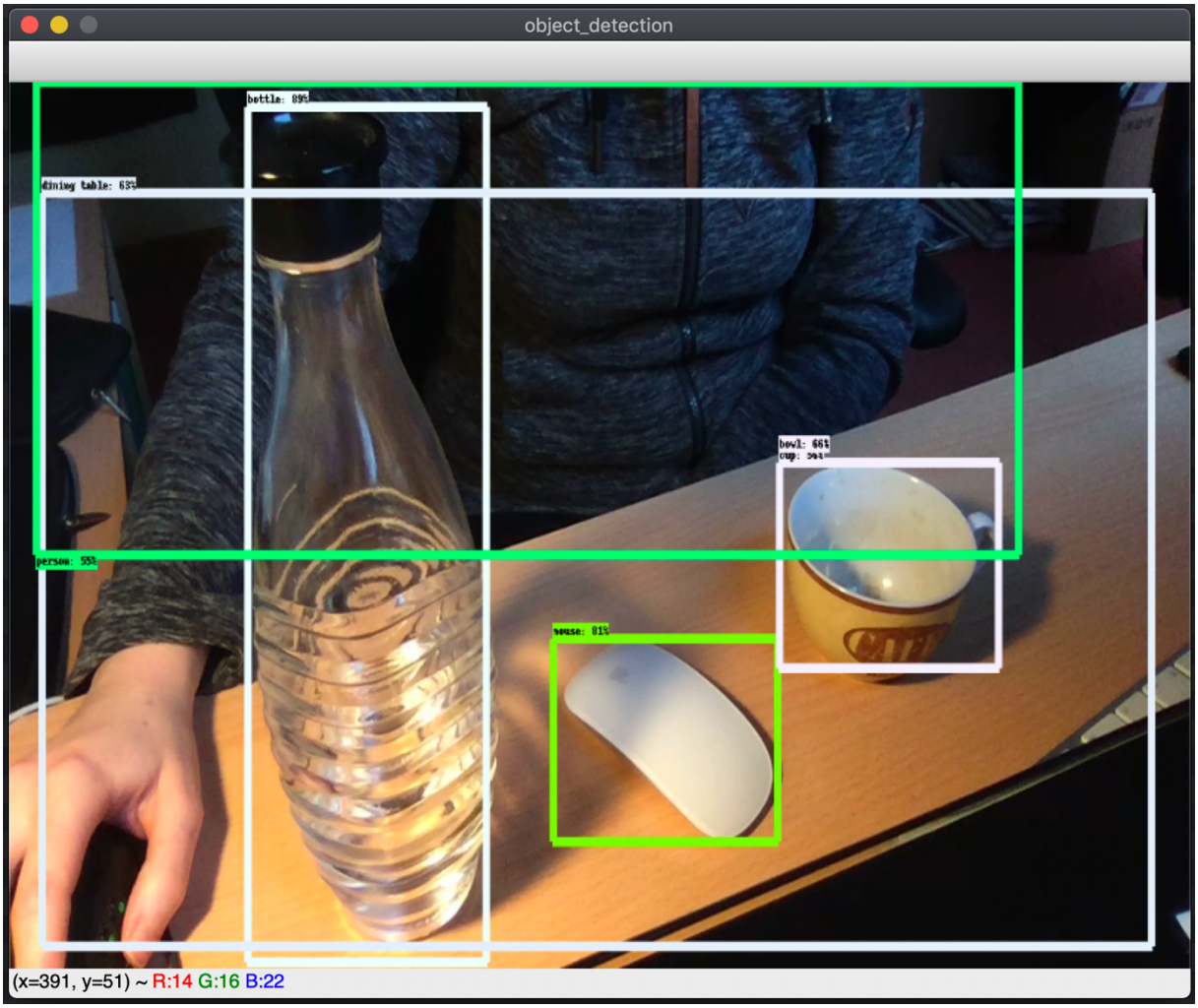
In diesem Kapitel werden zunächst unsere Visionen und Ziele vor Projektbeginn vorgestellt. Anschließend wird auf das Systemverhalten des Projekts eingegangen.

3.1 Vision und Ziel

Unsere Vision, bzw. das Ziel war eine Webanwendung zu entwickeln, welche es dem Benutzer erlaubt, mit einfachsten Mitteln ein Musikstück zu komponieren. Da ein Musikstück normalerweise mit Instrumenten und elektronischen Effekten erstellt wird, ist es für normale Benutzer nicht möglich so etwas zu realisieren. Gründe dafür sind zum Beispiel, dass Instrumente sehr teuer sein können und oft die Fähigkeit, Instrumente zu spielen, fehlt. Außerdem muss ein Musikstück im Anschluss geschnitten und die Tonspuren angeglichen werden. Malica hat in der jetzigen Form vorgefertigte Tonspuren, welche im Prinzip eine Gruppe von Instrumenten entspricht. Alle Tonspuren ergeben ein komplettes Lied. Ein weiteres Ziel war die Implementierung einer Objekterkennung. Es sollte möglich sein, durch erkannte Objekte, welche vor die Webcam gestellt werden, Tonspuren miteinander zu kombinieren und schließlich zu einem kompletten Lied nach dem Geschmack des Benutzers zusammenzumischen. Jede Tonspur enthält Instrumente der gleichen Familie und natürlich auch Vocals. Die Objekte, welche vor der Webcam platziert werden können, müssen vorher definiert werden und werden anschließend mit einer Tonspur gekoppelt. Ferner wurden die Tonspuren je mit einer Palette von Effekten versehen. Somit kann der Benutzer zum Beispiel die Gitarren-Sounds mit einem Hall versehen, verschiedene Filter/ Pässe anwenden oder einfach den Gain einer Tonspur anpassen.

3.2 Systemverhalten

Die Grundlage für die Objekterkennung ist ein trainiertes TensorFlow-Modell. Dieses wurde mit dem COCO-Datensatz trainiert und wird bei Beginn der Anwendung geladen. Der zugrundeliegende Algorithmus ist ein neuronales Netz, welches im Bereich Deep Learning (eine Machine Learning Methode) Anwendung findet, um aus Bildern Informationen zu erhalten. Anschließend wird eine Liste mit Labels geladen, die für die erkennbaren Objekte möglich sind. Erst dann wird der Video-Input, also die Webcam ins Spiel gerufen. Jedes Frame wird mittels der Objekterkennung darauf geprüft, ob es bekannte Objekte enthält. Um den Video Input überhaupt abgreifen zu können, wird OpenCV genutzt. Wenn die Genauigkeit des erkannten Objektes über 50% liegt, wird ein farbiger Rahmen um das Objekt gezogen, sowie das zugehörige Label dazu angezeigt.



OpenCV visualisiert dies dann im Webcam-Bild sichtbar für den Benutzer. Anschließend folgt eine Überprüfung der vorher definierten Objekte, die später den Musikinstrumenten zugeordnet werden. Dazu wird geprüft, ob sich diese in der aktuellen Liste befinden. Durch die Überschreitung eines Schwellwertes, d.h. wenn das Objekt in mehreren nacheinander folgenden Frames enthalten war, wird dies über einen Websocket an den Webbrowser als “erkannt” gesendet. Umgekehrt ist es so, dass wenn das Objekt in mehreren aufeinander folgenden Frames fehlen sollte, wird es als “fehlend” an den Socket geschickt. Im JavaScript wird geprüft, welcher Zustand aktuell gesendet wurde. Dann wiederum manipuliert das Skript den HTML-DOM, d.h. es werden Objekte verändert. Erkannte Instrumente werden blau umrandet und die Tonspur ist zu hören. Der Nutzer kann nun die Tonspur durch Effekte manipulieren. Dies ist mit der WebAudio-API möglich, die über diverse Filter den Sound verändern kann. Sobald ein Objekt nicht mehr im Bild sichtbar ist, wird die Tonspur wieder deaktiviert, technisch gesehen stumm geschaltet. Neben den Effekten gibt es eine Visualisierung der Tonspur, um dem

Nutzer dabei zu helfen, den Ablauf der Sounds in den Tonspuren besser einzuschätzen. Dies geschieht über die Bibliothek “Wafesurfer.js”.

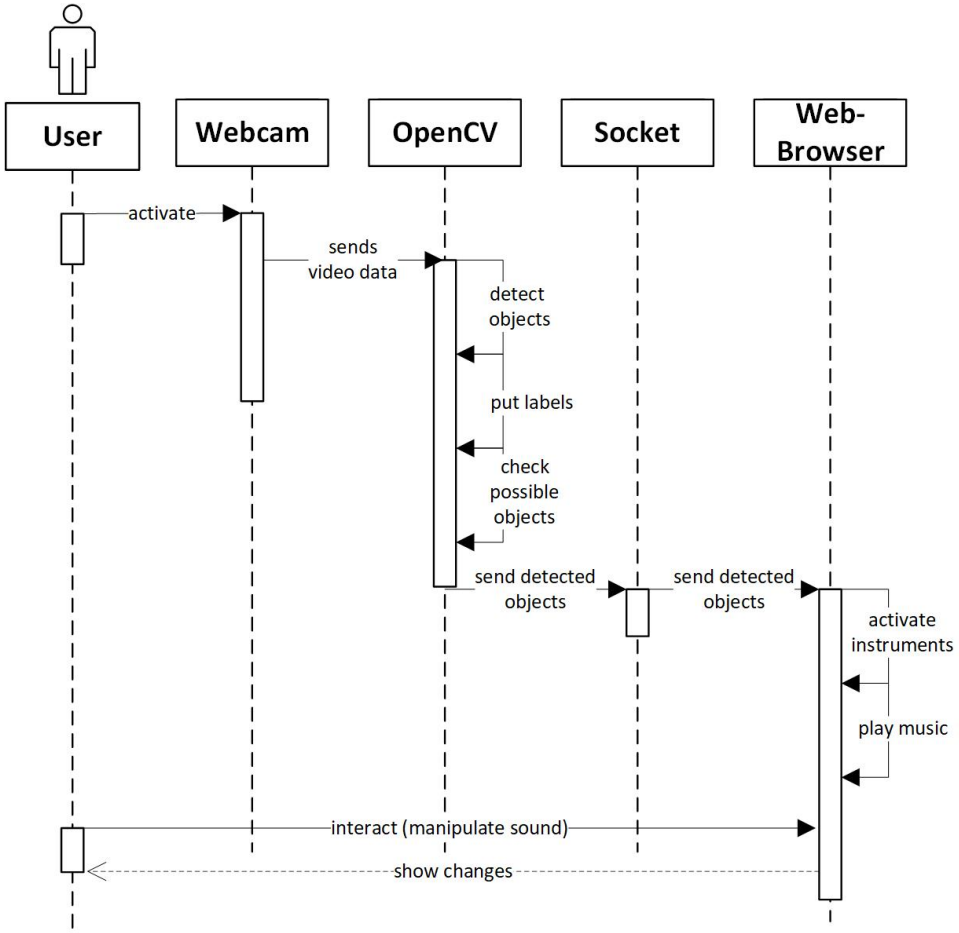


Abbildung 1: Systemverhalten

3.3 Systemkomponenten

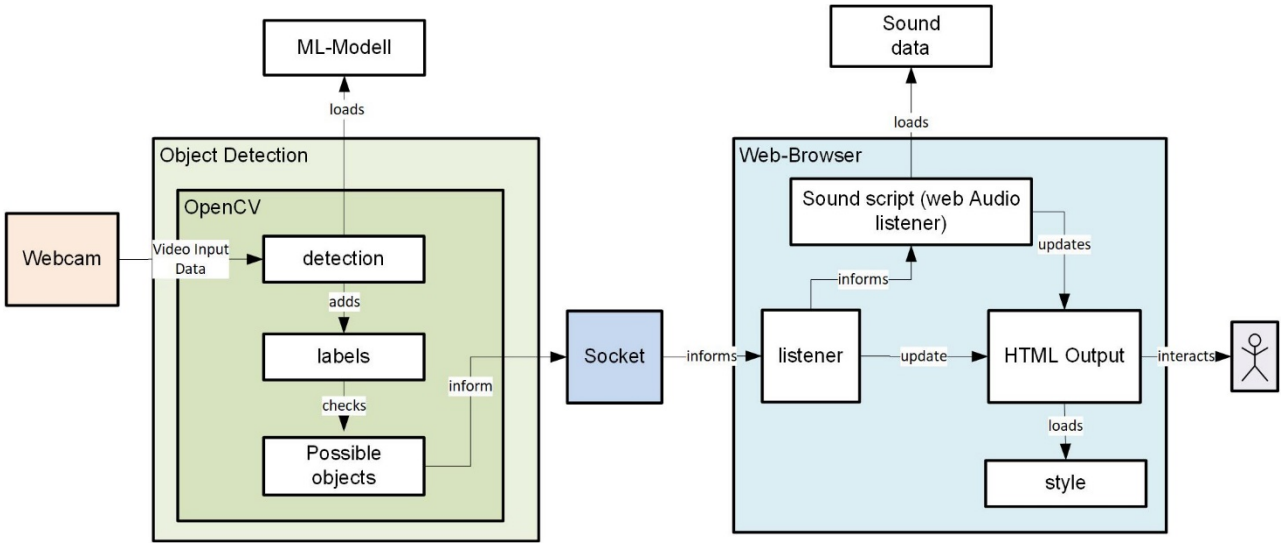


Abbildung 2: Systemkomponenten

3.4 Funktionen

- Objekterkennung mit der Webcam durch TensorFlow
- Weboberfläche zur Steuerung der Anwendung
    - Starten und stoppen der Tonspuren in der Weboberfläche
    - Aktuelle Zeit des Musikstückes
    - Erkennen, welche Tonspur durch die Objekterkennung aktiviert wurde
    - Struktur der Tonspuren einsehen können
    - Effekte für jede Tonspur einstellen
        - Gain (Lautstärke) anpassen
        - Filter einstellen
        - Frequenz der Filter anpassen
        - Reverb-Filter auswählen (Andere räumliche Gegebenheiten simulieren)
        - Etc.
    - Counter für erkannte Objekte
- Audioausgabe der aktiven Tonspuren

3.5 User Interface

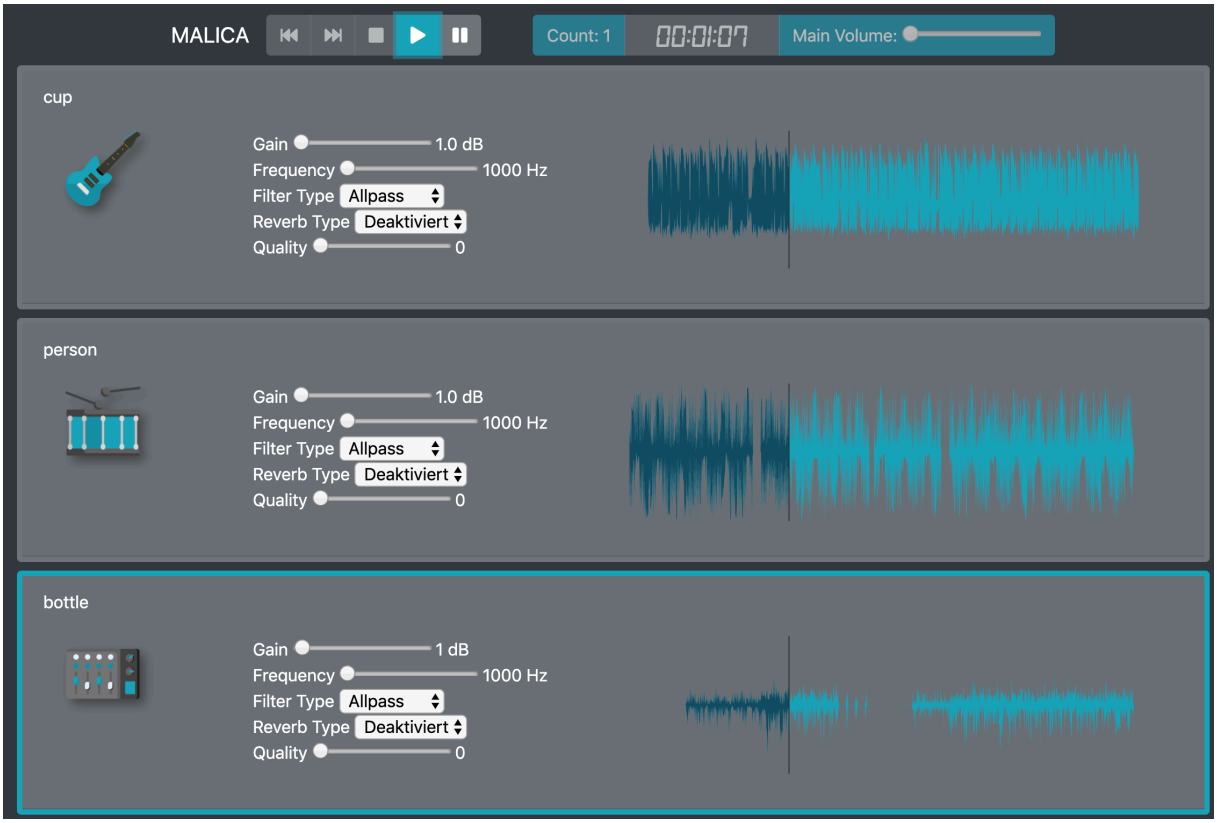


Abbildung 3: User Interface (Ausschnitt)

Um das User Interface zu verbessern wurde ein Timer, sowie ein Start und Pause-Button und diverse Grafiken der Instrumente ins HTML eingebettet. Jede aktive Tonspur ist blau umrandet, inaktive verbleiben grau.

4. Git-Repository

https://github.com/nayemi/malika

6. Ausblick

Unsere Anwendung ist in vielen Punkten erweiterbar.

- Audio:
  - Es können weitere vorgefertigte Lieder, welche in Tonspuren unterteilt sind, implementiert werden
  - Normale Musikinstrumente können eingefügt werden, welche nicht vorgefertigt sind. Somit wäre eine komplette eigene Komposition möglich.
  - Effekte könnten an die Instrumente angepasst werden
  - Weitere Effekte könnten eingebunden werden
- Video & Objekterkennung:
  - Durch weiteres Ausprobieren kann die optimale Umgebung herausgefunden werden, damit die Objekterkennung optimiert werden kann. Dabei spielt zum Beispiel das Licht und der Hintergrund eine bedeutende Rolle
  - Es könnte eine noch hochwertigere Kamera verwendet werden
  - Dem Benutzer könnte es ermöglicht werden, die Objekte eigenständig mit Tonspuren zu verknüpfen, oder selbst Objekte anlernen zu lassen
- Design:
  - Allgemein kann das Dashboard für die Benutzer weiterhin verbessert werden
  - Die Effekte könnten in ein Untermenü untergebracht werden
  - Außerdem kann die Übersichtlichkeit der Tonspuren verbessert werden