COLOR PLAYER

Audio Video Programmierung | Media Systems WiSe 2015/2016

Projekt zum Thema Tangible Sound Control | ColorPlayer

Shayan Shamseddin | Matrikelnummer: 2103763

Nielab Schahrochie | Matrikelnummer: 2117408

Inhaltsverzeichnis

Idee

Eine Applikation, die durch Farberkennung Musik erzeugt.
Mithilfe von variieren verschiedenfarbiger Objekte vor der Kamera kann Musik erzeugt werden. Dabei spielt der Anwender mit roten, blauen, grünen und/oder gelben Objekten vor der Kamera, diesen kann man im Vorfeld Töne, Oktaven und Tonlänge zuweisen.

Beschreibung aus Nutzersicht

Startet der Nutzer der Programm, öffnet sich die Benutzeroberfläche. Er hat die Möglichkeit für die vier Farben rot, blau, grün und gelb jeweils einen Ton (c, c#, d, d#, e, f, f#, g, g#, a, b), die Länge des gewählten Tons (volle Note, halbe Note, viertel Note, achtel Note, sechzehntel Note) und eines aus drei Oktaven über die jeweiligen Dropdowns wählen.
Alles eingerichtet kann das Program über das Start-Button gestartet werden.
Die Kamera wird im Kamera Frame gestartet und die Farben werden gesucht. Sobald eine Farbe gefunden ist, wird der vom Nutzer eingestellte Ton gespielt. Um die Applikation zu beenden kann der Stop-Button gedrückt werden. Nun kann man das Programm endgültig schließen oder man wählt neue Töne für die einzelnen Farben und startet die Applikation erneut.

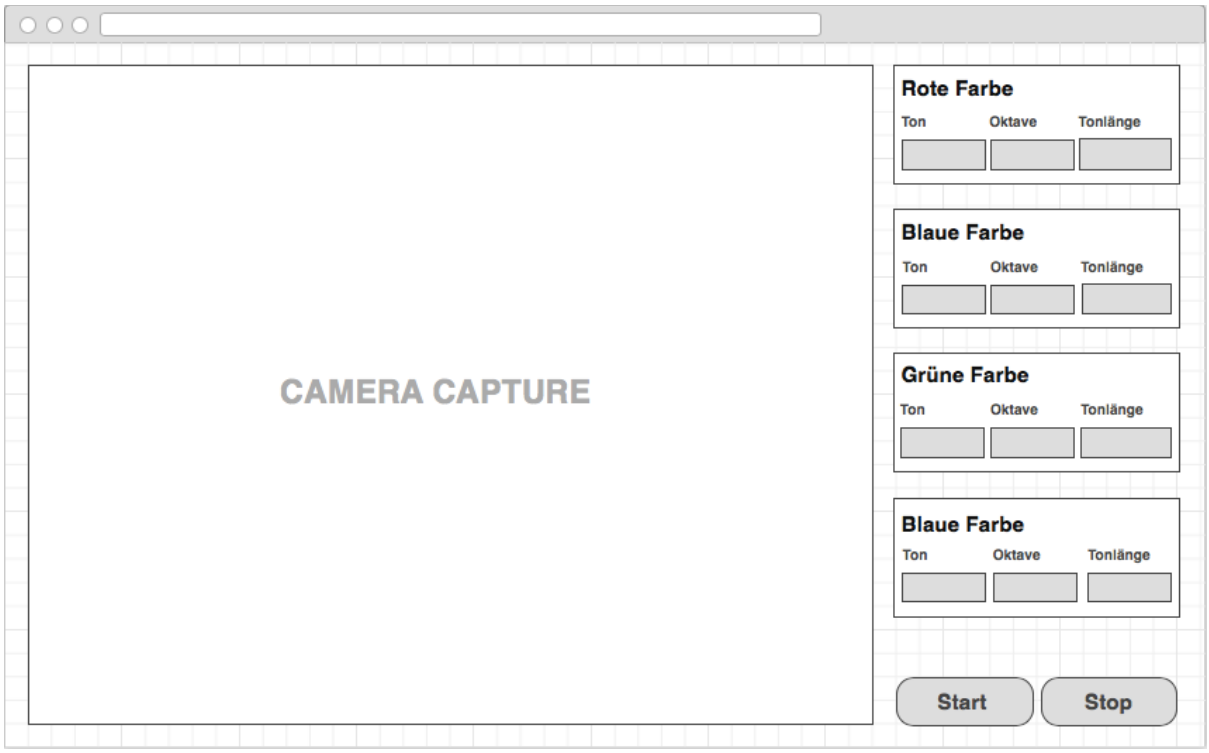

Abb. 1) GUI Mockup

Technisches Konzept

Die technische Umsetzung erfolgt mithilfe von OpenCV-Libraries und die Programmierung in C++ in Visual Studio 2012.

Durch einen Loop-Zugriff der Webcam ermöglichen wir das Live-Tracking, um dann durch den Color-Detector die BGR-Farbwerte des Bildes zu analysieren und sie in den HSV-Farbraum (Hue, Saturation, Value) zu konvertieren.
Über die Angabe der minimalen und maximalen HSV-Wertebereiche bestimmen wir 4 Farben (rot, blau, gelb, grün) und wir haben gleichzeitig alle Variationen dieser Farben abgedeckt.

Der Nutzer kann den einzelnen Farben über die Benutzeroberfläche Töne angeben, indem er über die Dropdown-Menüs eine Einstellung für Ton, Oktave und Tonlänge macht. Bei Erkennung einer Farbe wird der entsprechend gesetzte Ton gespielt.
Die Töne entstehen aus der Frequenz und der Tonlänge in Millisekunden.

	c	c#	d	d#	e	f	f#	g	g#	a	a#	b
Oktave 1	131	139	147	156	165	175	185	196	208	220	233	247
Oktave 2	262	277	294	311	330	349	370	392	415	440	466	494
Oktave 3	523	554	587	622	659	698	740	784	830	880	932	988

Tabelle 1) Frequenzen und Oktaven der Töne

	Millisekunden
ganzer Ton	3200ms
Halber Ton	1600
Viertel Ton	800
Achtel Ton	400
Sechzehntel Ton	200

Tabelle 2) Tonlänge in Millisekunde

Dokumentation des technischen Teilaspekts: Farberkennung

Die Farberkennung erfolgt mithilfe von OpenCV.
Es wird auf die Webcam zugegriffen und das aktuelle Bild analysiert (*cap.read(imgOriginal)*).
OpenCV nimmt Bilder und Videos im BGR Format auf, welches das Originalbild in drei Matrizen teilt: Blau, Grün und Rot. Da dieser Farbraum für die farbbasierte Segmentierung über die Kamera keinen Einfluss auf die Lichtverhältnisse nimmt, ist der HSV Farbraum geeigneter.
Der HSV Farbraum setzt sich zusammen aus Farbton (Hue, die dominante Wellenlänge der Farbe) Farbsättigung (Saturation, Zumischen von Weiß) und Hellwert (Value, maximale Amplitude des Lichts).
Durch die Trennung des Farbtons von der Sättigung und des Hellwertes ist der HSV-Farbraum wesentlich unempfindlicher von Lichtquellen bei der Identifizierung einer Farbe als der BGR oder RGB Farbraum.
Daher wird das aufgenommene Bild vom BGR Farbraum in den HSV Farbraum konvertiert: *cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV);*
Um nun die einzelnen Farbwerte im HSV Farbraum zu erkennen wird ein Schwellenwertbild benötigt. Um nach den einzelnen Farben im Schwellenwertbild zu filtern wird eine Trackbar erzeugt.

```
// Fenster „Control“ erzeugen
namedWindow("Control", CV_WINDOW_AUTOSIZE);

 int iLowH = 0;
 int iHighH = 179;
 int iLowS = 0;
 int iHighS = 255;
 int iLowV = 0;
 int iHighV = 255;

 //Trackbar in Control erstellen
 cvCreateTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
 cvCreateTrackbar("HighH", "Control", &iHighH, 179);
 cvCreateTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)
 cvCreateTrackbar("HighS", "Control", &iHighS, 255);
 cvCreateTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 255)
 cvCreateTrackbar("HighV", "Control", &iHighV, 255);
…
// Schwellenwerbild erstellen
inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded);
```

Für die Farben rot, blau, grün und gelb wurden die Farbwerte wie folgt gefunden:

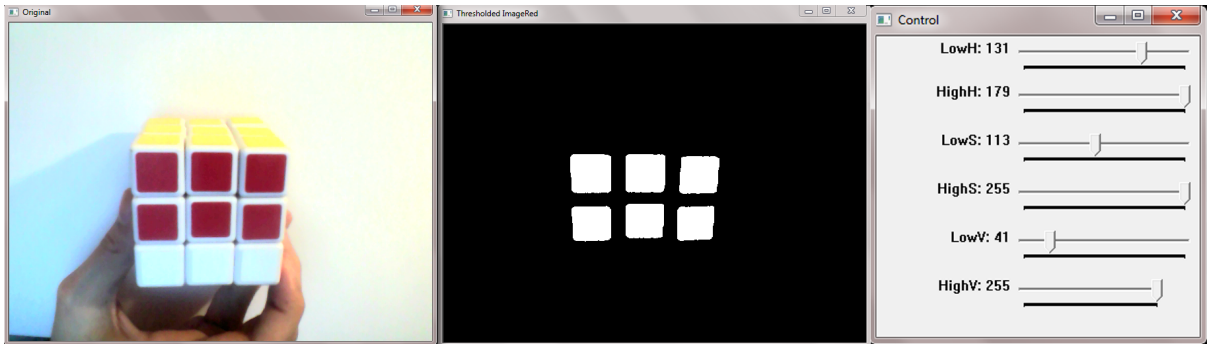

Abb. 2) der Rote Farbton liegt im Bereich 131 - 179

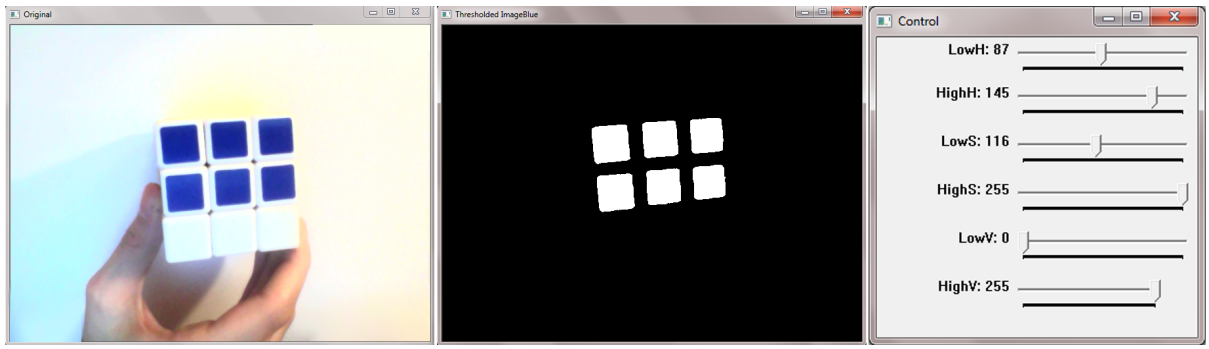

Abb. 3) der Blaue Farbton liegt im Bereich 87-145

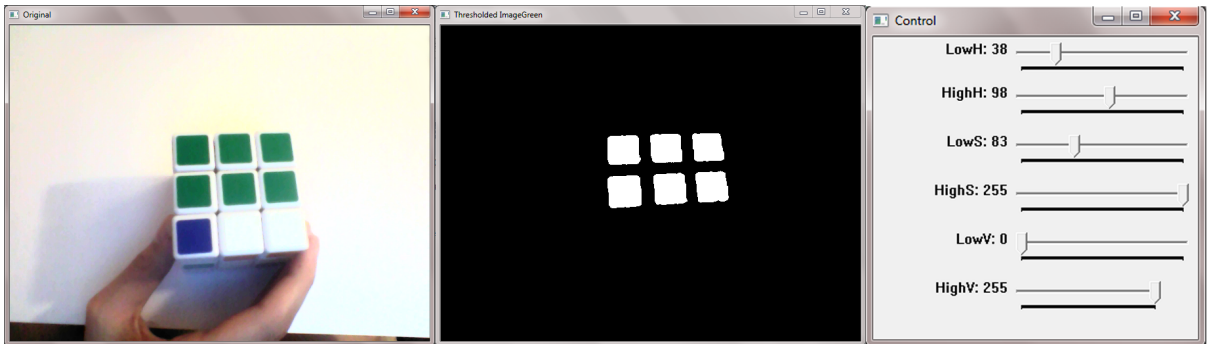

Abb. 4) der Grüne Farbton liegt im Bereich: 38 - 98

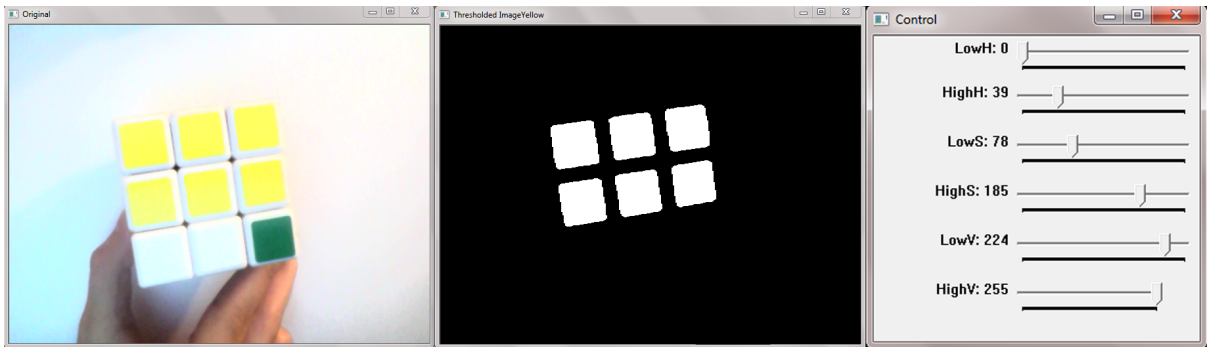

Abb. 5) der Gelbe Farbton liegt im Bereich 0 - 39

Saturation und Value können entsprechend der Lichtverhältnisse der Umgebung, sowie der Oberfläche des Objekts variieren.

Um die obigen Ergebnisse zu erhalten wurden folgende OpenCV Methoden verwendet:

- *void cvtColor(InputArray src, OutputArray dst, int code)*

Diese Methode konvertiert den Farbraum eines Bildes (src) in einen anderen Farbraum (in Code z.B. COLOR_BGR2HSV) und gibt diesen als Zielbild (dst) aus.

- *void inRange(InputArray src, InputArray lowerb, InputArray upperb, OutputArray dst);*

Überprüft, ob jedes Element des Quellbildes (src) zwischen 'lowerb' (untere Grenze) und 'upperb' (obere Grenze) liegt. Wenn ja, wird die entsprechende Lage im Zielbild (dst) zugeordnet.

Im Schwellenwertbild treten oft kleine Objekte auf, entweder durch Rauschen oder anderen kleinen Objekten im Bild, die dieselbe Farbe aufweisen, wie das zugfilternde Objekt. Diese kleinen Störungen im Bild kann man durch die morphologische Öffnung durch eine Erosion, gefolgt von der Dilatation mit dem gleichen strukturierenden Element entfernen.

Auf der anderen Seite können ebenfalls kleine weiße Löcher in den zu filternden Objekten auftreten. Diese kann man durch eine morphologische Schließen durch eine Dilatation, gefolgt von der Erosion mit dem gleichen strukturierenden Element entfernen.

- *void erode(InputArray src, OutputArray dst, InputArray kernel, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDER_CONSTANT, const Scalar& borderValue=morphologyDefaultBorderValue())*

Diese Funktion erodieren das Quellbild und speichert das Ergebnis in das Zielbild.

- *void dilate(InputArray src, OutputArray dst, InputArray kernel, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDER_CONSTANT, const Scalar& borderValue=morphologyDefaultBorderValue());*

Diese Funktion dilatiert das Quellbild und speichert das Ergebnis in das Zielbild.

In der Header Datei Color.h der Klasse Color.cpp werden die getter- und setter-Methoden für die HSV unter Grenze und HSV aber Grenze der Farben Deklariert:

```
public:
        Color(string name);

        Scalar getHSVmin();
        Scalar getHSVmax();

        void setHSVmin(Scalar min);
        void setHSVmax(Scalar max);

private:
        Scalar HSVmin, HSVmax;
```

Die Initialisierung erfolgt in der Klasse Color.cpp:

Die durch die Trackbar gefunden HSV-Werte für die einzelnen Farbtöne rot, blau, grün und gelb werden hier übergeben.

```
#include "Color.h"

Color::Color(string name){
        if(name ==„red"){ setHSVmin(Scalar(131, 113, 40)); setHSVmax(Scalar(190, 255, 255)); }
        if(name ==„blue"){ setHSVmin(Scalar(87, 116, 0)); setHSVmax(Scalar(145, 255, 255)); }
        if(name ==„green"){ setHSVmin(Scalar(38, 88, 0)); setHSVmax(Scalar(97, 198, 255)); }
        if(name ==„yellow"){ setHSVmin(Scalar(0, 43, 255)); setHSVmax(Scalar(37, 165, 255)); }
}
Scalar Color::getHSVmin(){ return Color::HSVmin; }
Scalar Color::getHSVmax(){ return Color::HSVmax; }

void Color::setHSVmin(Scalar min){ Color::HSVmin = min; }
void Color::setHSVmax(Scalar max){ Color::HSVmax = max; }
```

In der Main-Klasse erfolgt dann die Farberkennung. Durch die Methode *inRange()* und der getter-Methoden für die HSVmin und HSVmax werden die Farben erkannt.
Mit der *removeAndFill()* - Methode, werden ausschließlich die Objekte erkannt und kleinere Störungen entfernt.
Die Farberkennung erfolgt in der folgenden Reihenfolge: Rot, Blau, Grün, Gelb.

Aktueller Stand

Leider konnte die Gut bisher nicht fertiggestellt werden. Das Projekt wurde mit Visual Studio begonnen. Wie sich später herausstellte war die Erstellung eines GUI's nicht einfach umsetzbar, da Visual Studios Windows Forms (GUI Library) in C# zu programmieren wäre, das Projekt allerdings in C++ ist.
Die Entscheidung auf den Qt-Creator umzusteigen kam leider etwas spät. Hier dauerte die Einrichtung und Umarbeitung des Codes einige Zeit, sodass es Zeitlich nicht mehr geschafft wurde das GUI des Projektes umzusetzen.
Daher kann man die oben beschriebenen Einstellungen über die Konsole einstellen. Beim Start der Applikation erfolgt die Abfrage nach Ton, Oktave und Tonlänge der einzelnen Farben. Es gibt auch die Möglichkeit komplett frei zu sein indem man selbst eine Frequenz zwischen 131 und 988 wählt und eine Tonlänge zwischen 100 und 2000 Millisekunden. Zusätzlich kann man aus 6 anderen Themenbereichen wählen um keine eigenen Töne entstehen zulassen, sondern um Samples abspielen zu lassen.

AVPRG - Visual Studio 2012 Projekt
Inhalt:

main.cpp
Color.cpp
Sound.cpp
CreateSound.cpp

Color.h
Sound.h
CreateSound.h

Ordner: Sounds

Team & Aufgabenverteilung

Shayan Shamseddin
Matrikelnummer: 2103763

- Konzept
- SoundPlayer
- Sound Aufnahme

Nielab Schahrochie
Matrikelnummer: 2117408

- Konzept
- Farberkennung
- GUI
- Dokumentation
- Ausstellungsplakat
- SoundPlayer

Zeitaufwand

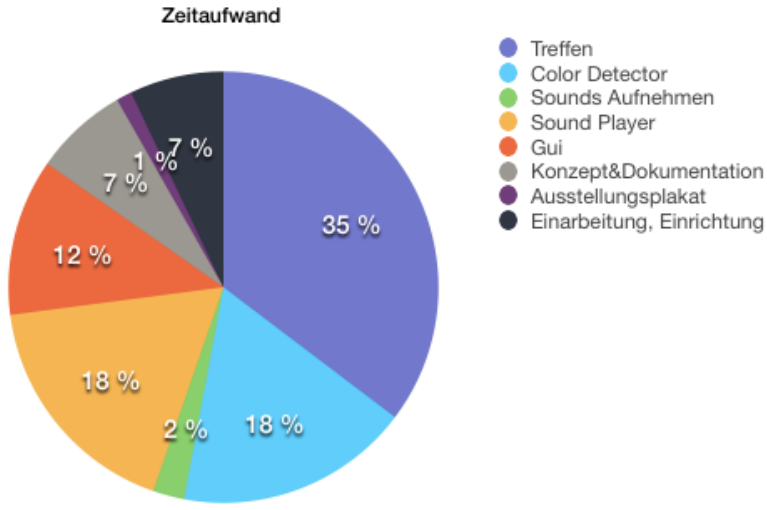

Abb. 6) Grafik Zeitaufwand

Abbildungsverzeichnis

Tabellenverzeichnis

Quellen

Farberkennung:

- Implementierung und Evaluierung einer Objekterkennung für einen Quadrocopter | Letzter Aufruf am 14. Dezember 2015

http://www8.informatik.uni-wuerzburg.de/fileadmin/10030800/user_upload/quadcopter/Abschlussarbeiten/Objekterkennung_Christian_Reul_BA.pdf

- Eine Processing-Erweiterung um kamerabasierte Gestenerkennung | Letzter Aufruf am 10. Dezember 2015

https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/schulze.pdf

- OpenCV in a Nuttshell | Letzter Aufruf am 10. Dezember 2015

https://www.matse.itc.rwth-aachen.de/dienste/public/show_document.php?id=7052

- Color Detection & Object Tracking| Letzter Aufruf am 30. November 2015

http://opencv-srf.blogspot.de/2010/09/object-detection-using-color-seperation.html

Soundwiedergabe:

- Audio-Aufnahme und Bearbeitungs-Programm : "Streaming Audio Recorder"

- Audioquellen : IOS-App "Big Bang"

- Audioquellen : "http://www.acoustica.com/sounds.htm"

- Audioquellen : Selbstaufnahme(Gitarre, Geräusche etc.)