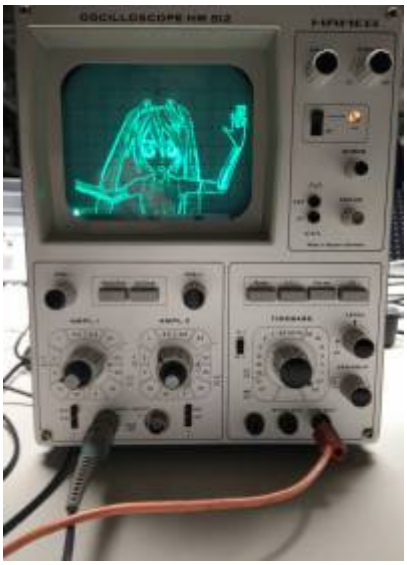

VGA: Vector Graphics Adapter

Vektorgrafik auf dem Oszilloskop dürfte so alt sein wie der X/Y-Modus. Neben Lissajous-Figuren erfreut sich die Soundkarte als Quelle für Oszilloskop-Vektorgrafik einiger beliebtheit. Dabei werden linker und rechter Kanal mit der X bzw. Y Auslenkung des Oszilloskops verbunden und bewegen so den Elektronenstrahl im analogen Oszilloskop.

Eine Soundkarte ist im Prinzip ein Digital-Analog Wandler, hat jedoch Tiefpassfilter welche zwar die Audio-Qualität erhöhen, aber die Geschwindigkeit mit welcher der Elektronenstrahl bewegt werden kann eng limitiert. Außerdem sind in der Regel Koppelkondensatoren verbaut, welche den für Lautspecher schädlichen Gleichstromanteil entfernen. Dadurch verliert die Grafik jedoch den Nullpunkt und wabert um den Mittelpunkt aller Vektoren falls sie nicht gut auf diese Gegebenheiten angestimmt wird.

Doch der PC besitzt ja meist noch einen weiteren Digital-Analog-Wandler: Den VGA Anschluss. Eigendlich für Raster-Grafik ge gedacht, lässt er sich prima zur Ausgabe von Vektorgrafik zweckentfremden. Von den 3 Analogen Kanälen (Rot, Grün und Blau). Werden zwei genutzt um den Elektronenstrahl des Oszilloskops in X bzw. Y Richtug auszulenken.

Hardware

- VGA Pin 1 (Rot) mit dem Horizontal Eingang (Meist Ch2) des Oszilloskops verbinden.
- VGA Pin 2 (Grün) mit dem Vertikal (Y) Eingang (Meist Ch1) des Oszilloskops verbinden.
- Massen von PC und Oszilloskop verbinden.

Funktionsprinzip

- Die Vektorisierung ist als mpv video filter implementiert.
- Das Video (oder Bild) wird mittels Canny Kantenerkennung in ein Kantenbild umgerechnet.
    - Dazu habe ich in einen mpv fork den OpenCV-Canny Algorithmus als Video Filter eingebaut (-vf canny)
    - Alternativ kann der edgedetect filter aus ffmpeg/libavfilter verwendet werden (-vf lavfi=edgedetect)

- Anschließend wird ein weiterer Videofilter nachgeschaltet, welcher das Kantenbild in eine Vektorfolge umrechnet. (-vf vector).
    - Dieser Filter basiert auf dem OpenCV FindContours funktion, welche die Kanten als Vektor-Pfad extrahiert.
- Der Vektorpfad wird anschließend als „Vektorbild“ ausgegben, dabei entspicht Rot dem X und Grün dem Y Wert des Vektors. Die im Vektorbild verfügbaren Pixel werden auf die Vektoren Anteilsmäßig aufgeteilt.

Software

- Zunächst das VGA Timing so einstellen das möglichst wenig störende H-Blanks erzeugt werden:

```
xrandr --newmode scope2 53.3952 2048 2049 2060 2060 200 200 216 216
xrandr --addmode VGA1 scope2
xrandr --output VGA1 --mode scope --right-of <HAUPTBILDSCHIRM>
```

- ffmpeg (libav besitzt den edgedetect filter nicht)
- opencv
- Modifizierter mpv media player:

```
git clone https://github.com/da1l6/mpv
```

Aufruf

- Mit ffmpeg Kantenerkennung:

```
/pfad/zu/build/mpv --fs --geometry=<BREITE-HAUPTBILDSCHIRM>:0 --loop --
vf
scale=576:512,lavfi=[edgedetect=high=0.04:low=0.03],vector:width=2048:h
eight=200 <VIDEO>
```

- Mit openCV Kantenerkennung:

```
/pfad/zu/build/mpv --fs --geometry=<BREITE-HAUPTBILDSCHIRM>:0 --loop --
vf scale=576:512,canny:t1=128:t2=130,vector:width=2048:height=200
<VIDEO>
```

Die optimalen werte für t1 (bzw. low) und t2 (bzw. high) können je nach material variieren. Einfach ausprobieren.