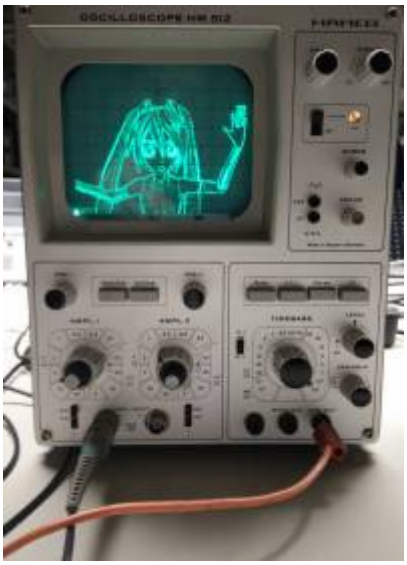
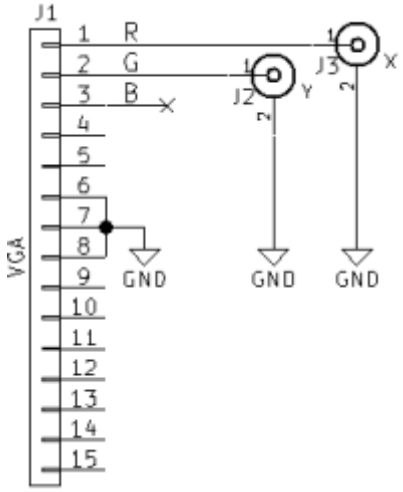


VGA: Vector Graphics Adapter

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. Der VGA-Port kann die Bilddaten auch mit sehr hoher Geschwindigkeit (> 100MHz) ausgeben, davon kann man bei Soundkarten nur träumen. Auch stöhrende Ausgangsfilter gibt es nicht. Damit ist er fast ideal für komplexe Vektor-Darstellungen geeignet.



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 / Software

- 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.
- Der ffmpeg edgedetect Filter liefert einen Intensitätswert, welcher die Ausgeprägtheit der Kante repräsentiert.
    - Bei der Umsetzung in das Vektorbild wird dies berücksichtigt indem der Anteil an der koordinaten an der Ausgabe entsprechend angepasst wird.
    - Je kleiner der Anteil desto schneller überstreicht der Elektronenstrahl die Kante und desto dunkler erscheint Sie.
    - Damit ist möglich die Helligkeit auch ohne Z-Eingang verändern.
- In der unteren linken Ecke (0,0) erscheint ein heller Punkt, dies ist Folge der Austastlücken („H/V-Blank“).
    - Um den Phosphor des Oszilloskop zu schonen sollte dieser außerhalb des sichtbaren Bereiches der Röhre bewegt werden.
- Beispiel „Vektor“-Bild zur Ausgabe auf dem VGA-Port:

Benutzung

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

```
xrandr --newmode scope 26.7  2048 2049 2060 2060 200 200 216 216 +hsync
+vsync
xrandr --addmode VGA1 scope
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
- Beispielaufruf:
  - 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=2
048:height=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.
  - Der Canny-Algorithmus ist sehr Rechenintensiv, deshalb wird das Bild zuerst mit `–vf scale:w:h` herunterskaliert.

Cheating at Vector Graphics - Rastergrafik

Der bei der Vektorgrafik benutzte Trick zur Helligkeitsänderung kann benutzt werden um Rastergrafik auszugeben.

Dabei werden die X/Y-Koordinaten (R/G-Farbkanäle) kontinuierlich so erhöht, das der Oszilloskop-Bildschirm zeilenweise überstrichen wird, wie bei einem Raster Monitor.

Helligkeitsunterschiede werden mittels Geschwindigkeitsänderung erzeugt. Helle Breiche werden langsamer überstrichen als dunkle indem die entsprechnden Koordinaten im „Vektorbild“ öfter wiederholt werden.

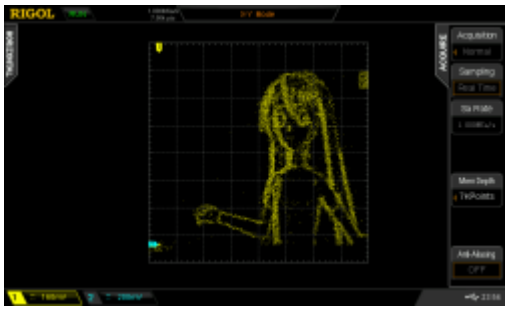
Benutzung

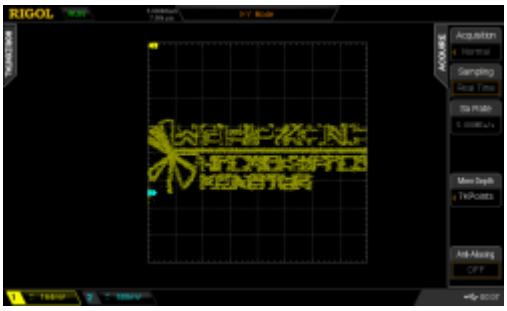
```
/pfad/zu/build/mpv --fs --geometry=<BREITE-HAUPTBILDSCHIRM>:0 --loop --vf
scale=256:256,vectorraster:width=2048:height=300 <VIDEO>
```

Digitaloszilloskop

Wer kein analoges Oszilloskop sein eigen nennt kann auch ein modernes digitales Oszilloskop verwenden, allerdings sind die Darstellungsergebnisse deutlich schlechter. Insbesondere der Rastergrafik-Modus ist kaum zu verwenden.

Die Ursache liegt darin, dass digitale Oszilloskope periodisch das Eingangssignal abtasten und dann an der zu den gemessenen Spannungen passenden Stelle einen Punkt zeichnen, welcher aber immer die selbe Helligkeit hat. Da das Abtasten nicht synchron zum Pixeltakt des VGA-Ports ist, werden auch viele Zwischenwerte im Übergang zwischen zwei Pixeln gemessen. Dies führt zu starkem „Rauschen“ auf dem Bild.





From:
https://wiki.warpzone.ms/ - **warpzone**

Permanent link:
https://wiki.warpzone.ms/projekte:vector_graphics_adapter?rev=1449963384

Last update: **01.03.2017**