https://www.colliseum.eu/wiki/AVIsynth

AVIsynth

Das skriptgesteuerte Programm AVIsynth erlaubt die nahezu beliebige (insbesondere auch nichtlineare) Bearbeitung von Videos unter Windows. Dabei liest AVIsynth das Video ein, führt die im Skript definierten Bearbeitungsschritte durch und reicht einen Strom unkomprimierter Einzelbilder an ein beliebiges Programm – meist VirtualDub – weiter. AVIsynth ist damit ein sogenannter *Frameserver*. Das Programm läuft (nur) unter Windows und kann von Haus aus nur AVI-Dateien lesen. Über etliche PlugIns und Sonderfunktionen können jedoch nahezu alle anderen Formate (insbesondere MPEG, MOV, Flashvideo) gelesen werden. AVIsynth selbst arbeitet im Hintergrund; es gibt kein Programmfenster; die Ergebnisse der Bearbeitung können nur über ein anderes Programm sichtbar gemacht werden.

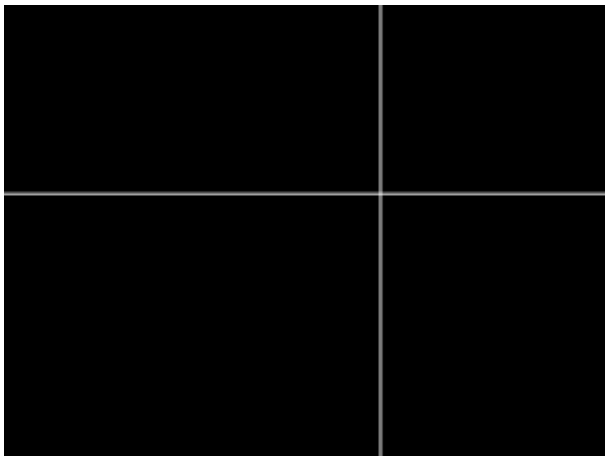
Die Funktionsweise von AVIsynth ist im Hugemann: Unfallrekonstruktion im Kapitel "Bild- und Videobearbeitung" beschrieben. Auf der Website zum Buch finden registrierte Benutzer etliche Beispielskripte samt Erläuterungen.

Als Editor für AVIsynth-Skripte ist AvsP das Werkzeug der Wahl. Ein einfaches Skript sieht z.B. etwa folgendermaßen aus:

```
AVIsource("C:\Test\MeinVideo.avi")
# Halbbilder zu doppelter Framerate auseinanderziehen
BoB()
# Höhen-Breiten-Verhältnis (Aspect-Ratio) auf 4:3 korigieren
LanczosResize(720,540)
```

Dieses Skript speichert man in einer Textdatei mit der Endung *.AVS und öffnet es anschließend mit VirtualDub.

Linie(n) an fester Position



Zwei Hilfslinien

Hlifslinien erzeugt man über die Funktion *BlankClip()*. Dabei werden alle Eigenschaften des "Bezugsvideos" übernommen, bis auf diejenigen, die man explizit neu setzt, nämlich *height* im Falle der horizontalen Linie und *width* im Falle der vertikalen Linie. Die Abmessungen des Videos müssen bei den meisten Codecs ein Vielfaches von 2 oder sogar 4 sein, wegen des Color-Subsamplings.

Platziert werden die Linien mit der Funktion *Overlay()*, wobei entweder x- oder y-Offset anzugeben sind.

Ein Pixel breite Linien sind möglich, wenn man das Video zuvor in mittels *.ConvertToRGB()* in RGB konvertiert.

```
video=BlankClip(length=100, width=640, height=480, color=$000000)
hline=BlankClip(video, height=2, color=$FFFFFF)
vline=BlankClip(video, width=2 , color=$FFFFFF)
Overlay(video, hline, y=200)
Overlay(last , vline, x=400)
```

Zwei Videos synchronisieren und nebeneinander stellen

Radfahrer aus verschiedenen Perspektiven

Im VKU 02/2015 wurde auf der EVU-Doppelseite ein Fall besprochen, bei dem ein Schlangenlinien fahrender Radfahrer aus zwei Perspektiven gefilmt und die Videos synchronisiert und nebeneinander zu einem Video zusammengestellt wurden.

Demo: Synchronisation zweier Videos – erstes Bild

Das dazu notwendige Vorgehen in *AVIsynth* können wir hier nur im Prinzip verdeutlichen, weil die Videodateien zu groß sind, um sie ins *Colliseum* zu stellen. Im nachfolgenden Beispiel werden die Videos stattdessen im Skript selbst erzeugt. Kopiert man die folgenden Zeilen in *AvsP*, so ist das Ergebnis ein Video, das direkt in VirtualDub angezeigt wird:

```
# Das Skript stellt zwei Videos mit unterschiedlichen Auflösungen
# und unterschiedlichen Frameraten nebeneinander dar
#
# 4 Sekunden Video 720x576 mit 25 fps, Hintergrund rot
BlankClip(width=720, height=576, fps=25, length= 100, color=$FF0000)
RotesVideo=ShowFrameNumber(size=100)
#
```

```
# 3 Sekunden Video 720x576 mit 50 fps, hochkant mit blauem Hintergrund
BlankClip(width=576, height=720, fps=50, length= 150, color=$0000FF)
BlauesVideo=ShowFrameNumber(size=100)
#
# Die Framerate des blauen Videos halbieren und die Höhe auf 576 Pixel
zuschneiden
# Die Videos starten am Synchronistationspunkt:
# Blau: Frame 44
# Rot:  Frame 30
Links=SelectEvery(BlauesVideo,2,44).Crop(0,0,-0,576)
Rechts=SelectEvery(RotesVideo,1,30)
#
# Die Videos (mit einem hellgrauen Zwischenstreifen) nebeneinander stellen
StackHorizontal(Links.AddBorders(0,0,32,0,color=$999999),Rechts)
```

Das obige Skript dünnt das blaue Video aus, um es auf die geringere Framerate des anderen zu bringen. Man kann jedoch ebenso gut die Framrate des roten Videos herauf rechnen, indem man interpolierte Frames hinzurechnet:

```
# Das Skript stellt zwei Videos mit unterschiedlichen Auflösungen
# und unterschiedlichen Frameraten nebeneinander dar
#
# 4 Sekunden Video 720x576 mit 25 fps, Hintergrund rot
BlankClip(width=720, height=576, fps=25, length= 100, color=$FF0000)
RotesVideo=ShowFrameNumber(size=100)
#
# 3 Sekunden Video 720x576 mit 30 fps, hochkant mit blauem Hintergrund
BlankClip(width=576, height=720, fps=30, length= 150, color=$0000FF)
BlauesVideo=ShowFrameNumber(size=100)
#
# Das blaue Video skalieren, sodass es dieselbe Höhe wie das rote hat
# Die Videos starten am Synchronisationspunkt:
# Blau: Frame 50
# Rot:  Frame 30
# (Jetzt glatte Werte, damit man die Synchroniation besser verfolgen kann.)
# Das rote Video auf 30 fps hochrechnen (interpolieren)
# wenn stattdessen Frames nach Bedarf wiederholt werden sollen die Funktion
ChangeFPS(30) verwenden
Links =BlauesVideo.Trim(50,-100).Lanczos4Resize(576*576/720, 576)
Rechts=RotesVideo.Trim(30,-100).ConvertFPS(30)
#
# Die Videos (mit einem hellgrauen Zwischenstreifen) nebeneinander stellen
StackHorizontal(Links.AddBorders(0,0,32,0,color=$999999),Rechts)
```

Die Zahleneinblendungen im roten Video verschwimmen jetzt ineinander, weil sie sich schlecht interpolieren lassen. Will man keine Pseudo-Information hinzudichten, bietet sich stattdessen die Funktion *ChangeFPS()* an, die Frames nach Bedarf dupliziert. Auf diese Weise kann man dann z.B. 300 fps Highspeed-Video mit normalem PAL-Video synchronisieren.

Die Synchronisation findet in diesem Skript über die Funktion *Trim()* statt, welche die Videos jetzt bei Frame 50 bzw. 30 beginnen lässt. Über diese jeweiligen Startpunkte werden die Videos synchronisiert.