Eine kurze Einführung in softVNS

Inhalt

1. Einleitung

1.1 Ziel dieses Dokuments:

Dieses Dokument soll den Einstieg in die Arbeit mit softVNS vereinfachen, ohne zu ausführlich Informationen zu referieren, die, ist der Einstieg geglückt, leicht dem Manual zu entnehmen sind. Es richtet sich an Personen, die noch nicht mit softVNS gearbeitet haben, aber schon einige Erfahrungen mit Max/MSP (oder ähnlichen Umgebungen wie etwa pd oder PWGL) sammeln konnten.

Dazu soll erstes Wissen zur Bedienung, aber auch zu Videoformaten etwa vermittelt werden, darüberhinaus soll die Lektüre in die Lage versetzen, sich selbständig weiteres Wissen anzueignen und praktisch umzusetzen.

Dabei liegt der Schwerpunkt - ganz im Sinne der Genealogie von softVNS - auf Bild-Klang-Interaktion: ein detailliert vorgestelltes Patch soll verdeutlichen, auf welche Weise softVNS-Objekte genutzt werden können, um andere Max/MSP-Objekte anzusteuern.

Gleichzeitig wird die Beschäftigung der Arbeitsgruppe "Kamerabasiertes Motiontracking 2" (Systematische Musikwissenschaft, Universität Köln, Wintersemester 2006/2007) mit softVNS dokumentiert.

1.2 Systemvorraussetzungen:

Seit Version 2 läuft softVNS ausschließlich auf Rechnern mit G4-/G5-Prozessoren (bzw. auf vielen G3-Rechnern mit G4-Prozessorkarte).

Die Software wird als Teil von Max/MSP installiert und benötigt als Umgebung mindestens Max/MSP Version 4.

1.3 Weitere Voraussetzungen:

Abgesehen von einer installierten und authorisierten Version von Max/MSP 4 setzt diese Einführung grundlegende Kenntnisse im Umgang mit Max/MSP voraus, Max/MSP-Objekte, die im vorliegenden Text zur Anwendung kommen, werden jedoch erläutert.

Ebenso vorausgesetzt wird ein installiertes und authorisiertes softVNS.

Zur Installation und Authorisation sei auf die mitgelieferte Dokumentation der entsprechenden Software verwiesen (s. auch 1.4 'Weiterführende Informationen').

1.4 Weiterführende Informationen:

softVNS Manual:
Dem Programm liegt ein pdf bei (softVNS 2.19d Manual.pdf), das detaillierte Informationen v.a. zu den einzelnen Objekten bietet.

Im Internet:

http://www.cycling74.com/products/maxmsp (Herstellerseite Max/MSP)
http://homepage.mac.com/davidrokeby/softVNS.html (Herstellerseite softVNS)
http://www.maxobjects.com/ (Max Objects Database; umfangreiche Datenbank zu den einzelnen Objekten von Max/MSP, softVNS und weiterer Software)
http://www.dactilde.com/index.php?title=Main_Page (ein wiki zu Max/MSP)

Max/MSP Tutorials:
Max/MSP beinhaltet einige sehr hilfreiche Tutorials, die in grundlegende Fähigkeiten und den Umgang mit der Software einführen. Sie befinden sich im Max/MSP-Verzeichnis unter /Documentation/Tutorial Patches/ und sind einzelne Patches, die einfach in Max/MSP geöffnet werden.

Hilfedateien:
Einen weiteren wertvollen Bestandteil der beiden Programme stellen die Hilfedateien dar. Sie sind über die Menüleiste zugänglich. Darüberhinaus öffnet ein Mausklick auf ein Objekt bei gehaltener alt-Taste in vielen Fällen die Hilfedatei zu dem entsprechenden Objekt.

Die Hilfedateien sind funktionsfähige Patches, in denen zusätzlich zu Informationen über das Objekt seine Funktionsweise direkt ausprobiert werden kann und die auch Links zu den Hilfedateien ähnlicher Objekte bieten.

2. Grundlagen

2.1 Was ist softVNS?

David Rokebys softVNS ist eine Sammlung zusätzlicher Max/MSP-Objekte zur Echtzeit-Videoverarbeitung. Das bedeutet einerseits, dass Max/MSP um Videofähigkeiten erweitert wird, andererseits aber vor allem, dass die bereits vorhandenen Max/MSP-Objekte zur Weiterverarbeitung von softVNS-Daten zur Verfügung stehen, um so etwa Bild-Klang-Interaktion zu realisieren. softVNS bietet Objekte, die Videodaten einer Kamera oder einer Quicktime-Videodatei zugänglich machen, solche, die Videodaten weiterverarbeiten und solche, die Videobilder ausgeben.

softVNS ist eine Adaption von Rokebys interaktiver Klanginstallation 'Very Nervous System', die er seit 1986 entwickelte. Nachdem er seit 1993 mehrere Versionen auch zum Kauf anbot, entstand 1999 softVNS, dem 2002 die erweiterte Version softVNS 2 folgte.

2.2 Videodaten in softVNS:

Grundlegende Einstellungen und Informationen bieten die Fenster softVNS 2 Preferences und softVNS 2 status (s. Abb. 1). Sie öffnen sich über die Menüleiste unter Extras/.

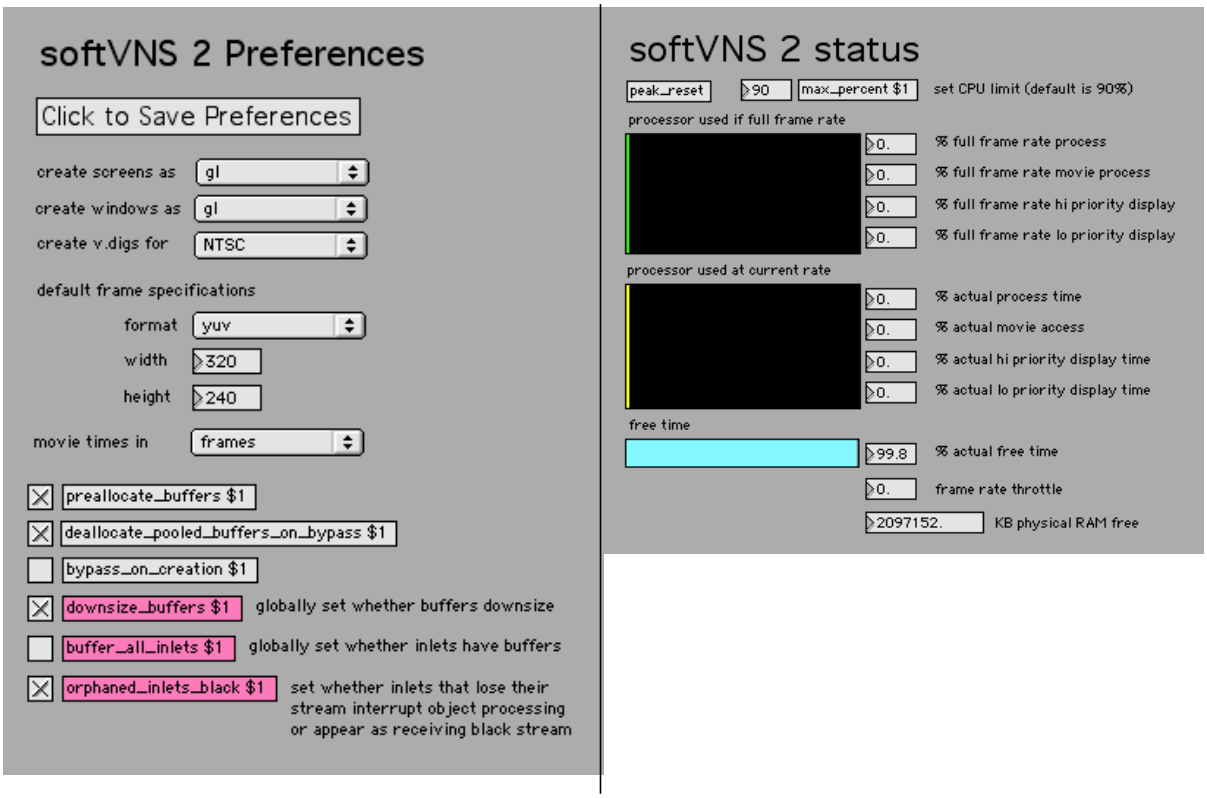

Abb. 1: Das Preferences- und das Status-Fenster

Anfangs genügt es zu wissen, dass softVNS mit drei verschiedenen Videoformaten umgehen kann. Dies ist besonders im Zusammenhang mit der Leistungsfähigkeit der Hardware wichtig: die Formate sind verschieden ressourcenintensiv. Das Graustufenformat grays beschreibt jeden Bildpunkt durch einen Helligkeitswert und verlangt am wenigsten Rechenleistung. Das Format yuv enthält zusätzlich zu Helligkeitswerten zwei Farbkomponenten, die sich jeweils zwei benachbarte Bildpunkte teilen. Das rgb-Format basiert auf drei Farbkomponenten je Bildpunkt.

Möchte man mit Farbbildern arbeiten, ist in der Regel yuv zu empfehlen, da es dem internen Format der meisten digitalen Videoquellen entspricht, dabei aber fast doppelt so schnell wie rgb verarbeitet wird.

2.3 Anmerkungen zu softVNS-Objekten und ihrer Syntax:

Alle softVNS-Objektnamen beginnen mit einen v gefolgt von einem Punkt. Eine Ausnahme stellt jit.softvns dar, das wie v.jit den Datenaustausch zwischen softVNS und der Max/MSP-Erweiterung Jitter ermöglicht.

Folgen einem Objektnamen Argumente, so werden diese durch jeweils ein Leerzeichen separiert.

3. Erste Schritte

3.1 Der Einstieg in softVNS:

Das Fenster softVNS 2.1 Object Overview:
Zum Einstieg in softVNS empfiehlt sich ein Blick auf die Datei softVNS_2_Overview (zu finden in der Menüleiste unter Extras/ oder im Max/MSP-Verzeichnis unter /patches/extras/) (s. Abb. 2).

Hier findet sich eine kategorisierte Übersicht der verschiedenen Objekte. Ein Mausklick auf ein Objekt öffnet die dazugehörige Hilfedatei.

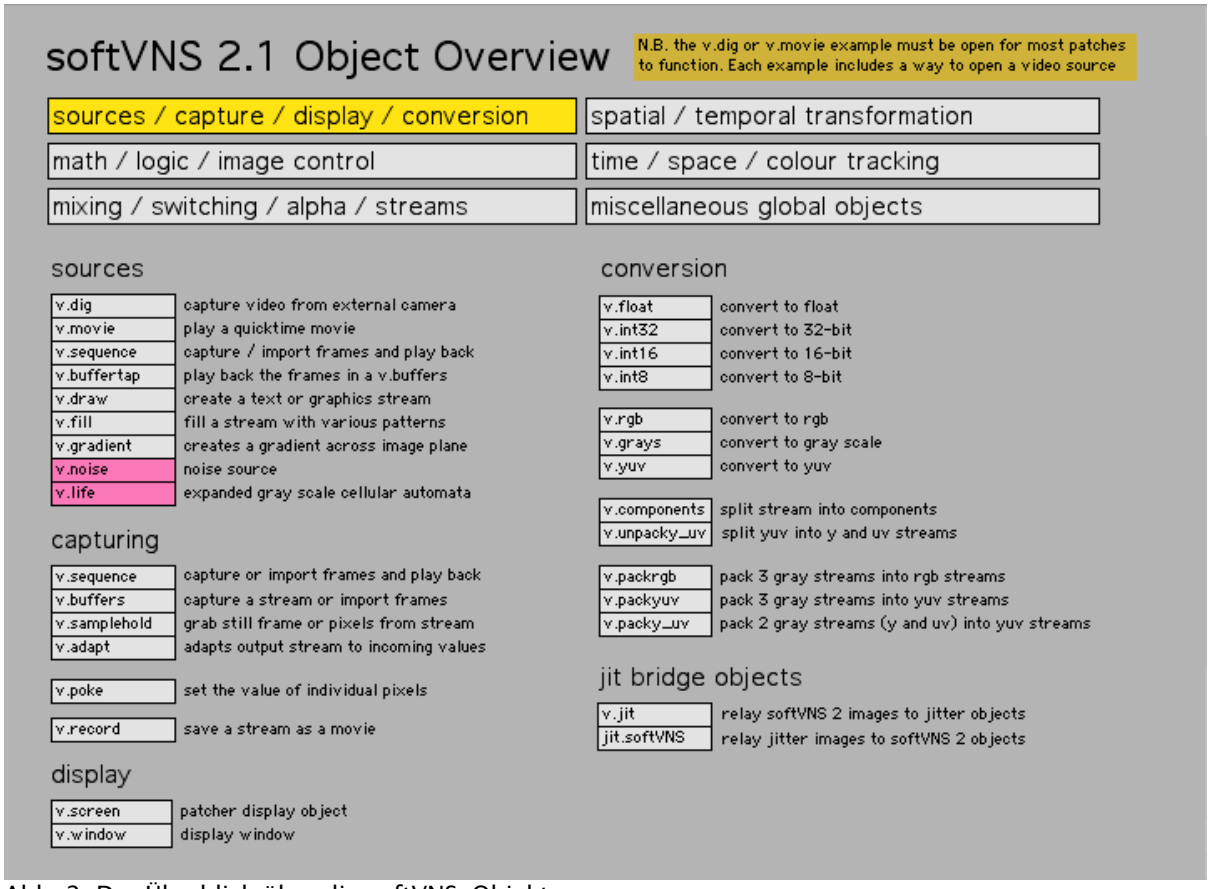

Abb. 2: Der Überblick über die softVNS-Objekte

Einen ersten Eindruck über einige Fähigkeiten von softVNS vermittelt auch die Datei softVNS.b+w examples (im Max/MSP-Verzeichnis unter /examples/softVNS additional examples/, s. Abb. 3). Dort werden grundlegende Objekte zur Bildverarbeitung vorgestellt.

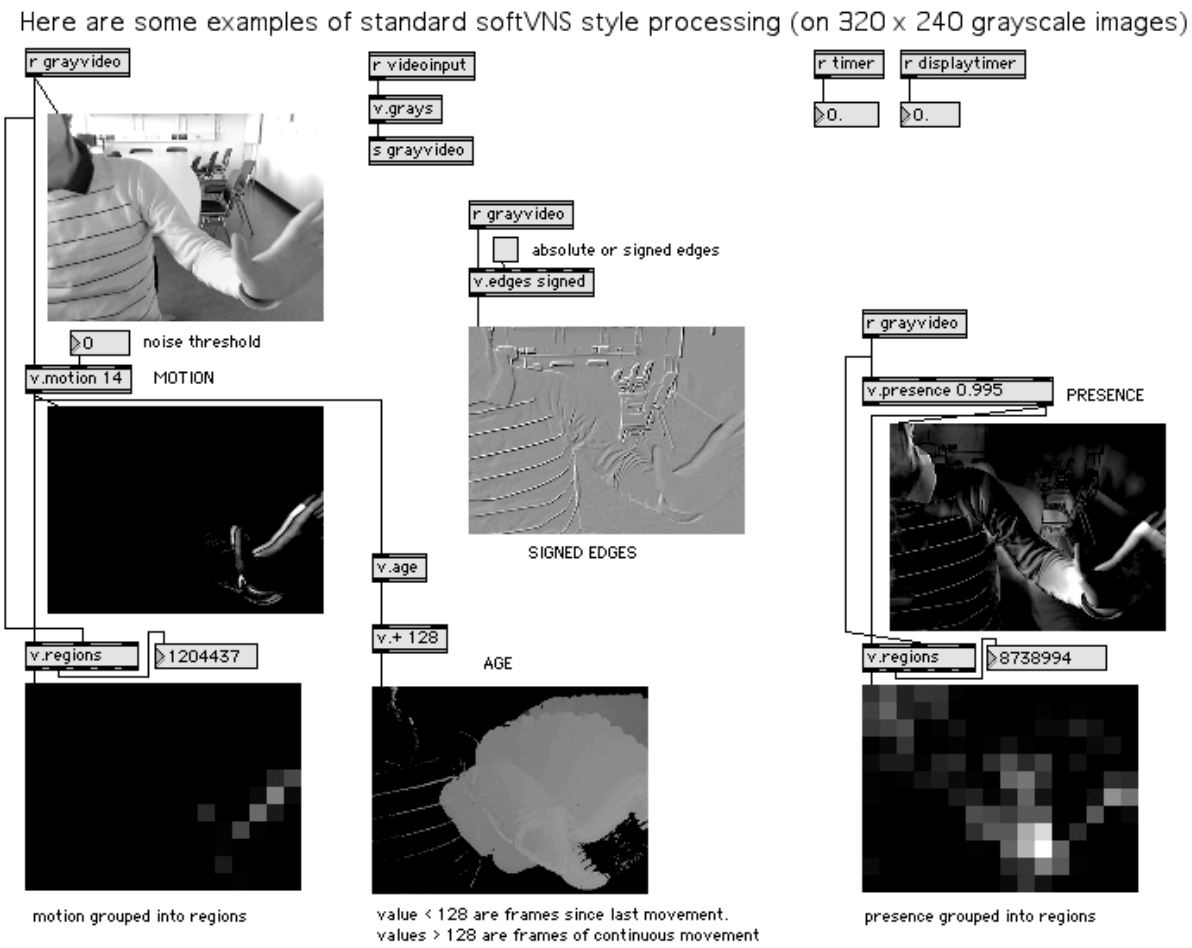

Abb. 3: Das softVNS.b+w examples-Patch

3.2 Nötige Vorbereitungen:

Zum Arbeiten mit softVNS muss in der Regel das v.dig oder das v.movie-Objekt aktiv sein (s. Abb. 4 und 5).

v.dig stellt das Signal einer angeschlossenen Kamera zur Verfügung und bietet Einstellungsmöglichkeiten für das Videosignal. Dies geschieht über das Fenster v.dig.help, das zugänglich ist durch einen Mausklick auf das v.dig-Objekt im Fenster softVNS 2.1 Object Overview, oder durch einen alt-Klick auf ein v.dig-Objekt in einem beliebigen anderen Patch.

Das Aktivieren des Objekts erfolgt durch Anklicken des gelb umrandeten Schalters. Das schwarze Videofenster unterhalb von v.dig sollte dann ein Videobild zeigen.

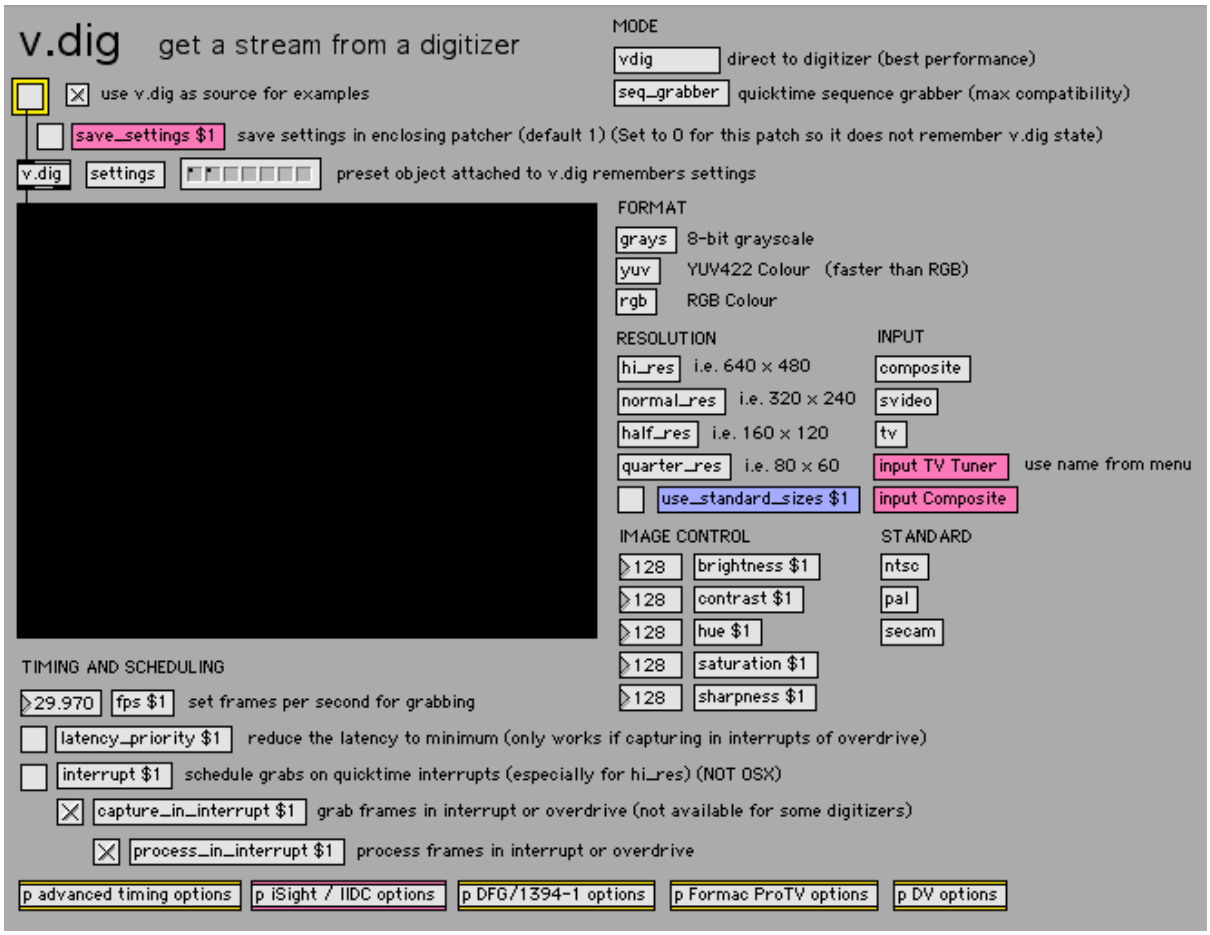

Abb. 4: Das v.dig-Fenster

Soll mit gespeicherten Videodaten gearbeitet werden, kann im Fenster v.movie.help eine Quicktime-Datei ausgewählt werden (s. Abb. 5).

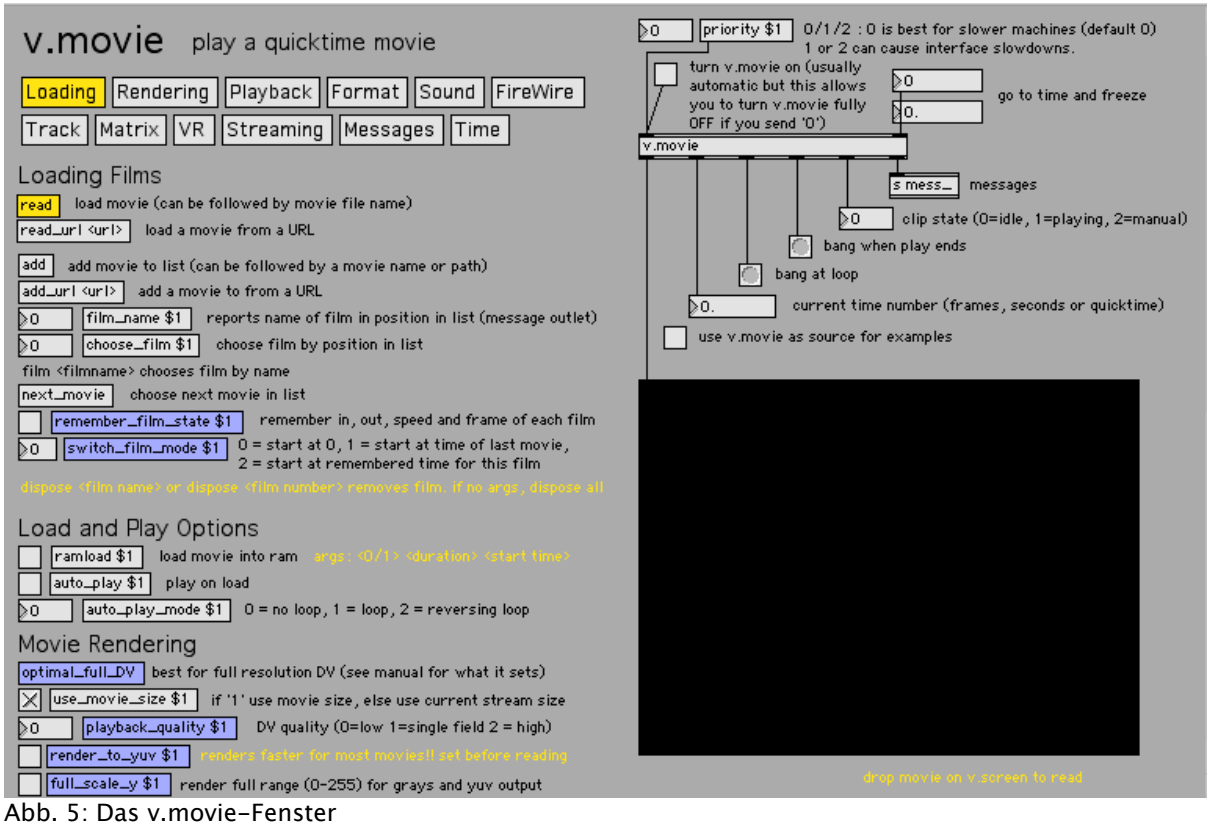

Abb. 5: Das v.movie-Fenster

Im Rahmen der Entstehung dieses Textes wurde ausschließlich mit Kamerainput gearbeitet, so dass hier auf eine Erläuterung der v.movie-Einstellungsmöglichkeiten verzichtet wird.

Zeigt die verwendete Hardware Performanceprobleme, kann etwa die Auflösung des Live-Kamerasignals im v.dig-Fenster verringert oder das Eingangssignal auf ein Graustufenbild umgeschaltet werden. Bei Patches, die wenige Videostreams verarbeiten und nicht sehr komplex sind, sollte das jedoch nicht notwendig sein.

Ist sowohl v.dig als auch v.movie geöffnet, so dient das zuletzt geöffnete als Quelle.

3.3 Ein softVNS-Patch erstellen:

Hat man ein neues, leeres Patch geöffnet, so muss zuerst ein eingehendes Videosignal verfügbar werden. Dies ist in softVNS durch send- und receive-Objekte realisiert, die zu Max/MSP gehören. Sie bestehen aus einem s bzw. einem r, gefolgt von einem Namen, in diesem Fall 'videoinput' – ein receive-Objekt empfängt automatisch die Daten von einem gleichnamigen send-Objekt, auch wenn sich letzteres in einem anderen Patch befindet, das jedoch geöffnet sein muss.

Die Fenster v.dig und v.movie geben das Videosignal prinzipiell über ein Objekt s videoinput aus. Dieses Signal kann dann in einem anderen Patch von einem Objekt r videoinput empfangen werden (s. Abb. 6).

Es sei noch angemerkt, dass s videoinput im v.dig- und im v.movie-Fenster nicht zu sehen ist. Es wird jedoch sichtbar, wenn man das entsprechende Fenster im Bearbeitungsmodus (Befehl-E) betrachtet.

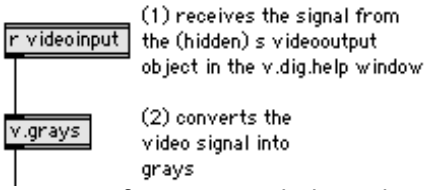

Abb. 6: So empfängt ein Patch das Videosignal

4. Eigene softVNS-Patches

Im folgenden wird ein Patch beschrieben, das Studierende der systematischen Musikwissenschaft an der Universität Köln im Rahmen der Arbeitsgruppe "Kamerabasiertes Motiontracking 2" erstellt haben. Abb. 7 zeigt ein funktionsfähiges Patch, das Bewegungen in veschiedenen Bildbereichen zur Generierung von Klängen verschiedener Tonhöhen nutzt; Abb. 8-12 zeigt vergrößerte Ausschnitte aus diesem Patch.

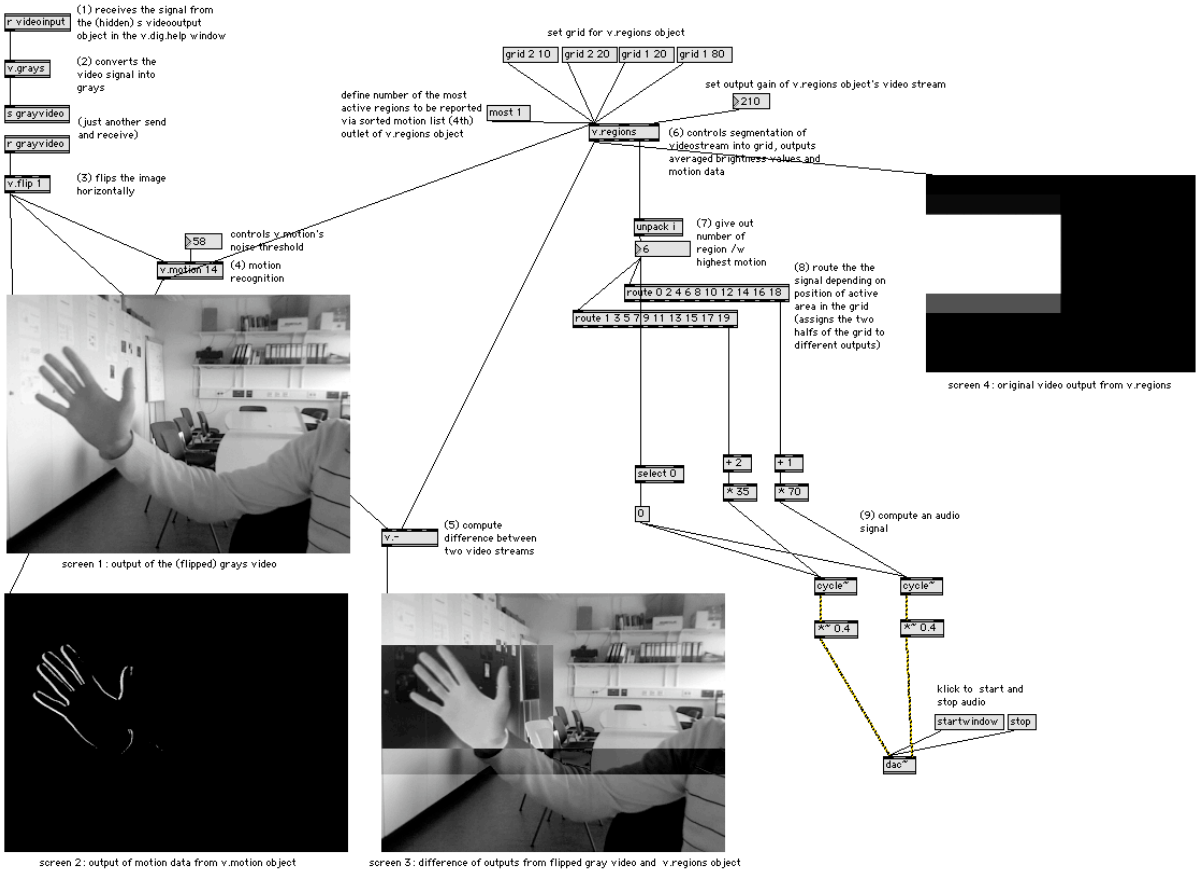

Abb. 7: Das vollständige Patch

Die Zahlen in Klammern beziehen sich auf die Numerierung innerhalb des Patchs.

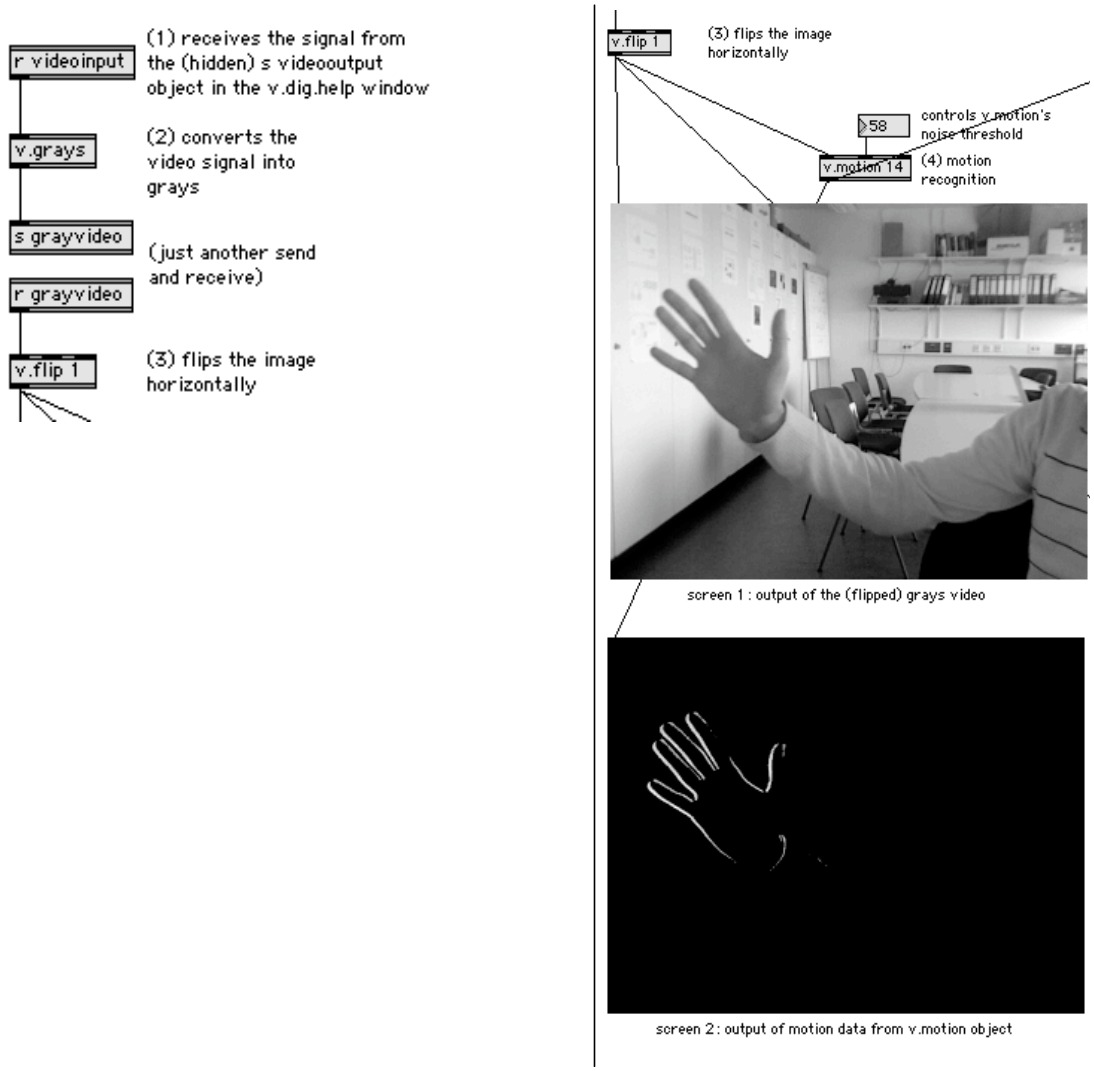

Abb. 8

(2) v.grays: Das von r videoinput empfangene Videosignal wird zunächst in ein Graustufenformat umgewandelt.

(3) v.flip spiegelt das Kamerabild horizontal, so dass man sich wie in einem Spiegel sieht. v.flip akzeptiert als Attribute zweimal den Wert 0 oder 1. Das erste Attribut ermöglicht eine horizontale, das zweite eine vertikale Spiegelung. Das Objekt gibt über seinen Ausgang das Videosignal an ein v.screen-Objekt (das stets als Videoscreen erscheint), an v.motion sowie an v.- aus.

(4) v.motion errechnet aus seinem Eingangssignal Bewegungsdaten, indem es die Differenz zwischen jedem Frame und dem ihm vorhergehenden berechnet. Diese Daten gibt es als Videosignal an Videoscreen 2 aus sowie an v.regions weiter. Praktisch bedeutet dies, dass nur noch Veränderungen im Videosignal, d.h. durch Bewegung hervorgerufene Videodaten vorliegen. Der mittlere Eingang akzeptiert als Eingabe einen Schwellenwert, so dass nur Differenzen, die größer als dieser Wert sind, weitergegeben werden. Reagiert das Patch unerwünscht auf kleine

Bewegungen (z.B. Zittern der Hand), kann durch Wahl eines größeren Werts eine Verbesserung erzielt werden.

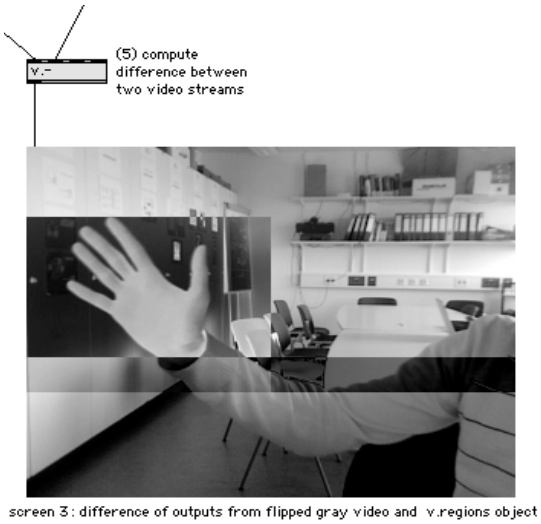

Abb. 9

(5) v.- überlagert durch Subtraktion den Videooutput von v.flip und v.regions (Videoscreens 2 und 4), so dass der Aktivierungsgrad der einzelnen Regionen im normalen Videobild sichtbar wird (Videoscreen 3, s. Abb. 9).

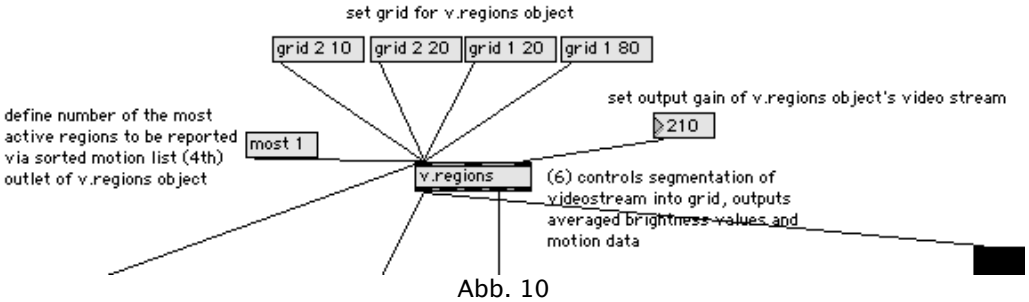

Abb. 10

(6) v.regions fasst daraufhin die eingehenden Helligkeitswerte (die nun dem Maß an Bewegung entsprechen) in einem Raster definierbarer Regionen zusammen. Es gibt dieses Videobild (Videoscreen 4) und u.a. eine Liste dieser einzelnen Regionen mit den entsprechenden Daten für das Maß an Bewegung aus.

Die Helligkeitswerte des Videobildes werden dabei bezogen auf jede einzelne Region gemittelt, der rechte Eingang akzeptiert ein Integer zur Einstellung der Verstärkung des ausgehenden Videobilds. Der linke Eingang empfängt nicht nur das Videosignal, sondern auch eine Reihe von messages: grid, gefolgt von einem oder zwei Integers, definiert die Anzahl der Spalten und Zeilen des Rasters (im Beispiel ein 2-10-Grid), die entstehenden Regionen werden von links nach rechts und von oben nach unten, beginnend mit dem Wert 0 numeriert; most gefolgt von einem

Integer definiert den Umfang der von v.regions am vierten Ausgang ausgegebenen geordeneten Liste der Regionsnummern und ihrer Helligkeitwerte, sortiert nach abnehmendem Aktivitätsmaß (Helligkeit). Diese Liste besteht also aus den Nummern der Regionen gefolgt vom Wert für das Maß an Bewegung in der entsprechenden Region; so erzielt etwa das Senden der Message most 1 die Ausgabe der Nummer nur der Region, in der gerade am meisten Bewegung stattfindet, gefolgt vom Wert, der den Grad an Bewegung repräsentiert.

Die Objekte im Patch, die auf v.regions folgen, sind Max/MSP-Objekte.

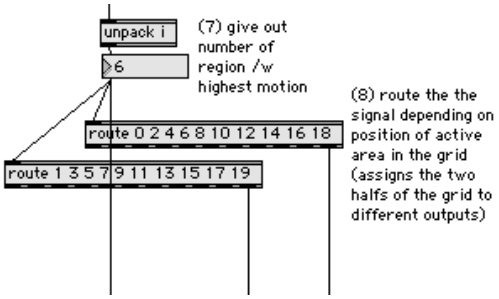

Abb. 11

Im vorliegenden Fall wurde die Entscheidung getroffen, zuerst einfache und nachvollziehbare Steuerungsparameter zu verwenden. Daher soll nur der erste Wert der Liste verwendet werden, d.h. die Nummer der Region höchster Aktivität.

(7) Dies wird durch das unpack-Objekt erzielt. Erhält es eine Liste an seinem Inlet, so gibt es das n-te Elemente der Liste über sein n-tes Outlet aus. Die Anzahl des Attributs i bestimmt die Anzahl der Outlets. Im vorliegenden Fall wird also einfach der erste Wert der Liste an das Outlet weitergegeben, das heißt: die number box zeigt stets die Nummer der Region, in der am meisten Bewegungsaktivität vorliegt. Dies ist der Wert, der zur Steuerung der Klangsynthese verwendet wird.

(8) Dazu sollen die beiden Hälften des Grids (d.h. die linken Regionen mit den Nummern 0, 2, 4, …, die rechten Regionen mit den Nummern 1, 3, 5, …) jeweils einem eigenen Syntheseprozess zugeordnet werden.

route gibt an seinem letzten Outlet alle Werte weiter, die nicht als seine Attribute auftauchen. So gibt also das linke route-Objekt alle geraden, das rechte alle ungeraden Werte weiter.

Auf deren Grundlage wird dann die Frequenz für des Syntheseprozess durch das cycle~-Objekt berechnet - nach der Addition und Multiplikation durch + und * erzeugt das linke cycle~ Klänge mit Frequenzen von 70, 140, 210, 280, 350, 420 … Hz, das rechte solche von 140, 280, 420, 560 … Hz. Die den cycle~-Objekten folgenden *~-Objekte verringern schließlich die Amplitude des Ausgangssignals.

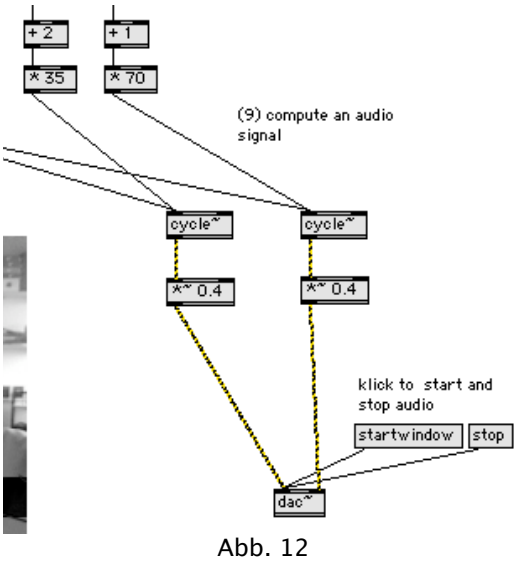

Abb. 12

Das Patch erzeugt also Klänge mit einer Frequenz in Abhängigkeit von der Nummer der Region, in der jeweils am meisten Bewegung stattfindet.