Bauhaus-Universität Weimar
WS 2003/04
Fachkurs „Augen auf“
Leitung: Lars Wieneke
Student: Felix Obée
Matrikel: 10188

TimeBouncing

Diese Arbeit ist das Ergebnis des Fachkurses „Augen auf“ bei Lars Wieneke. Der Fachkurs beschäftigte sich mit der freien Software EyesWeb und erkundete die Möglichkeiten der Verarbeitung optischer Signale. Als Abschlussarbeit des Fachkurses habe ich eine Installation entworfen und das dazu benötigte Patch mit EyesWeb erstellt. Die Installation arbeitet auf zwei Ebenen. Dabei kann eine Person einmal ein Video aufnehmen, wobei die Aufnahme durch Gesten gesteuert wird, und dann in einem zweiten Schritt das aufgenommene Video durch Gesten beeinflussen.
Im folgenden werde ich kurz auf die Software EyesWeb eingehen und in einem längeren Teil die Installation und das Patch in ihren Funktionen erklären.

Inhaltsverzeichnis

1 Aufbau der Installation

Es werden zwei Kameras in einem Raum aufgestellt (Abb.1), die die agierende Person frontal (Kamera 1)und von der Seite (Kamera 2) beobachten. Im Raum sind zwei Zonen definiert.

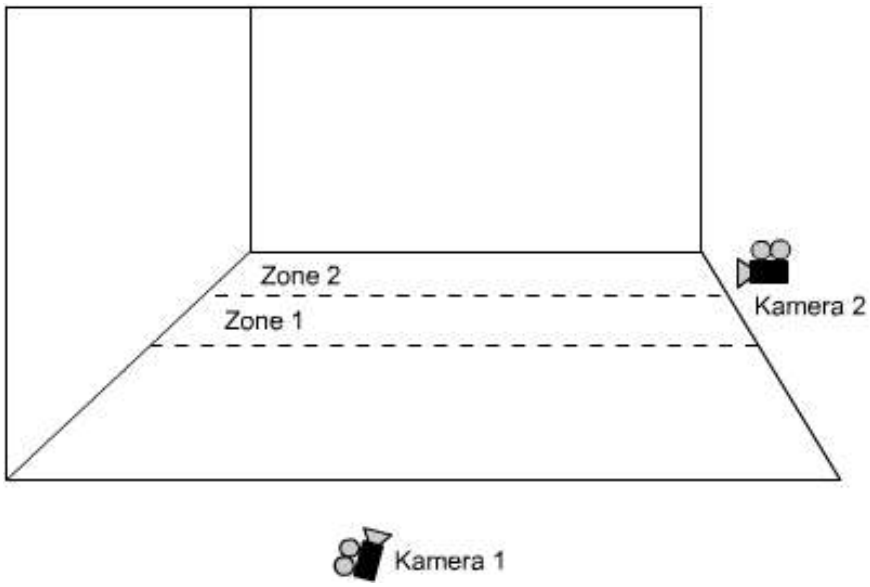

Abbildung 1: Aufbau

Befindet sich die Person in Zone 1, steuert sie das auf einem Monitor abgespielte Video durch ihren Umfang und ihre Position (Abb. 2). Das Video wird von einem bestimmten Startpunkt (In-Point) abgespielt, bis es einen Endpunkt (Out-Point) erreicht. Dann wird das Video rückwärts abgespielt, bis es den In-Point wieder erreicht hat und beginnt wieder von vorne. In- und Out-Point werden durch den Umfang der in Zone 1 befindlichen Person und deren Position definiert. Bewegt sie sich von links nach rechts, bewegt sie sich auf einer virtuellen Zeitleiste, wobei der äußerste linke Punkt ihres Körpers den In-Point und der äußerste rechte Punkt den Out-Point festlegt. Durch Veränderung ihres Umfangs, z.B. durch ausbreiten der Arme, kann die Länge des abgespielten Videos verändert werden.

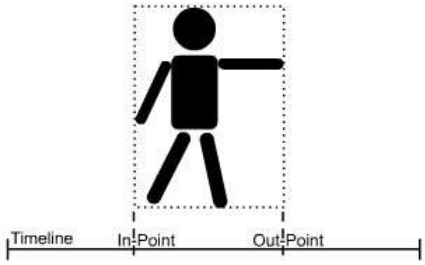

Abbildung 2: Steuerung der In/Out-Points auf der Timeline

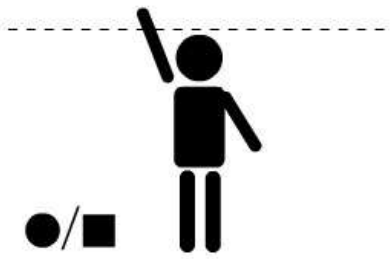

Abbildung 3: Auslösen und Anhalten der Aufnahme

In Zone 2 kann ein neues Video aufgenommen werden. Die Aufnahme wird durch einen virtuellen Schalter gestartet und gestoppt (Abb. 3). Dieser befindet im oberen Teil des Bildes in etwa 2 Meter Höhe über dem Boden und ist so durch heben der Arme zu erreichen. Verbleibt ein Körperteil einen vorgegebenen Zeitraum (500 Millisekunden) auf dem Schalter, wird die Aufnahme jeweils gestartet oder angehalten. Die Aufnahme beendet sich beim Betreten von Zone 2 oder nach Überschreiten der Maximallänge von selbst.

2 Aufbau und Funktionsweise des Eyesweb-Patches

Das Patch kann in verschiedene Aufgabenbereiche (Abb.4) aufgeteilt werden. Diese habe ich jeweils einzeln zusammengestellt und dann zu einem großen Patch zusammengefügt (siehe auch: 3.1 Arbeiten mit EyesWeb).
Eine Gruppe bildet der Player/Recorder-Switch, die feststellt, in welcher Zone sich eine Person aufhält. Die Gruppe teilt sich auf in Bildverarbeitung und Logikschaltung. Die Bildverarbeitung teilt das Bild von Kamera 2 in die zwei Zonen auf und stellt fest, ob sich jemand darin aufhält. Die Logikschaltung schaltet die Funktion der aktiven Zone ein, bzw die der inaktiven aus. Befindet sich jemand in beiden Zonen, werden beide Funktionen ausgeschaltet. Die Player-/Recorder-Sektion unterteilt sich in in die zwei verschiedenen Funktionen Player und Recorder auf. Beide haben eine eigene Bildverarbeitung, die allerdings mit dem selben, bereits bearbeiteten Bild von Kamera 1 gespeist wird. Die Ergebnisse der Bildverarbeitung werden dann als Steuerdaten an Player und Recorder weitergegeben.

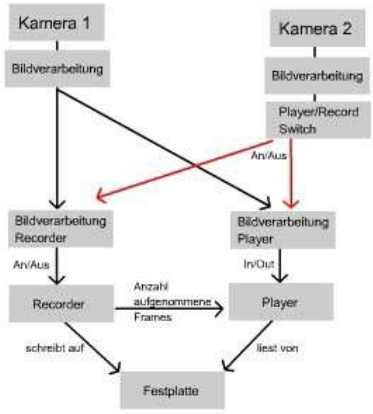

Abbildung 4: Funktionsschema des EyesWeb-Patches

2.1 Player/Recorder-Switch

2.1.1 Bildverarbeitung

An dem Bild, das von Kamera 2 (vgl. Abb. 1) geliefert wird, wird zuerst einmal eine Background-Substraction (Abb. 2) vorgenommen. Aus dem resultierenden Schwarzweiß-Bild werden die zwei Zonen extrahiert. Die jeweiligen Bildteile werden in eine Matrix konvertiert, deren Werte ausgelesen und summiert werden. Übersteigt die Summe der Helligkeitswerte einen bestimmten Wert, wird anstelle einer Null eine Eins an die Logikschaltung weitergegeben. Der Schwellwert dafür kann über den Regler „Empfindlichkeit“ eingestellt werden.

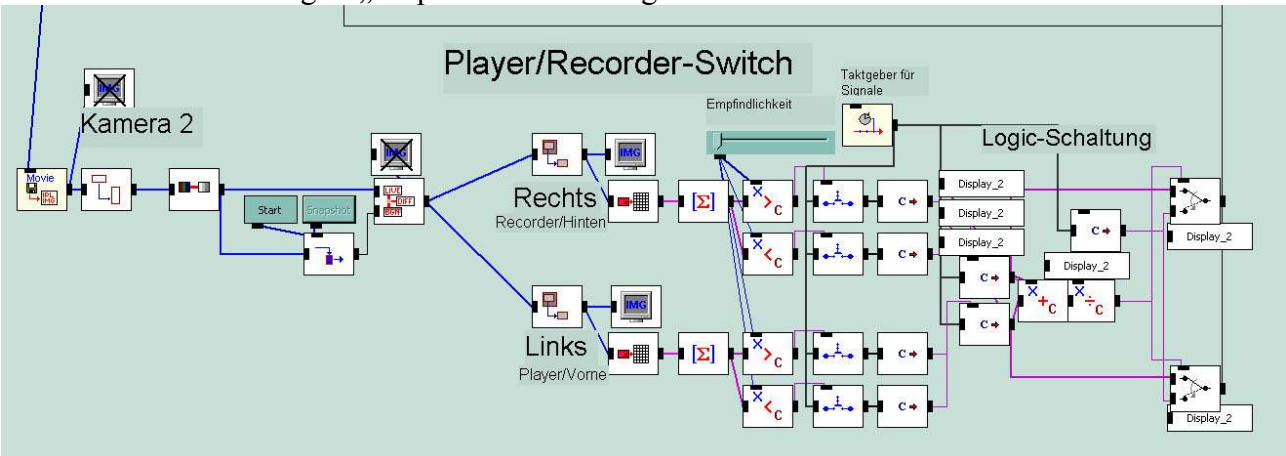

Abbildung 5: Kamera 2, Bildverarbeitung, Logikschaltung

2.1.2 Logikschaltung

Die Logikschaltung (Abb. 5) vergleicht die Werte der beiden Zonen. Bei Anwesenheit in einer Zone wird das dazugehörige Interface an- bzw das andere ausgeschaltet. Senden beide Zonen ein Anwesenheitssignal, werden beide abgeschaltet. Die beiden in 2.1.1 erläuterten Prozesse geben ihr Ergebnis als 1 (Anwesenheit) oder 0 (keine Anwesenheit) an die Logikschaltung weiter. Die beiden Werte werden addiert und durch 2 geteilt. Da es sich um imaginäre Zahlen handelt, wird ein Ergebnis von 0,5 auf 0 abgerundet. Bei Anwesenheit in beiden Zonen liefert uns die Logikschaltung also eine 1 ([1+1]:2=1]), wenn nur eine oder gar keine Zone besetzt ist eine 0 ([1+0]:2=0,5=0). Dieses Ergebnis wählt den Eingang von zwei InputSwitch-Objekten. Bei einer 1 schalten beide auf einen Eingang, der konstant eine 0 sendet. Bei einer 1 schalten sie auf den Eingang, der direkt das jeweilige Ergebnis der Schaltung aus 2.1.1 weiterleitet. Die InputSwitch-Objekte sind mit Switch-Objekten verbunden, die, je nachdem, ob sie eine 1 oder 0 erhalten, das Videosignal für das Recorder- und das Player-Interface zu- oder wegschalten.
Der für das Recorder-Interface zuständige Schalter beendet außerdem auch die Aufnahme, sobald das Videosignal weggeschaltet wird. Er ist mit dem Off-Schalter des Recorders verbunden. Damit der Off-Schalter nur ein einmaliges Signal bekommt, wenn der Recorder ausgeschaltet werden soll, musste ich noch eine kleine Konstruktion davor hängen. Theoretisch wäre ein ThreshCrossing-Objekt ausreichend, das den Übergang von 1 nach 0 meldet. Leider funktioniert dieses Objekt nicht, wenn der Schwellwert nur erreicht und nicht eindeutig überschritten wird. Deswegen konvertiere ich das Eingangssignal erst in reale Zahlen, damit ich eine Schwellwert von 0,5 angeben kann, der dann eindeutig unter- und überschritten wird.

2.2 Player/Recorder

Um das Bild von Kamera 1 (Abb. 6) wird zu Anfang eine Maske gelegt, die an jeder Seite des Bildes einen Rand aus 4 schwarzen Pixeln erzeugt. Die hat zwei Gründe: erstens funktioniert das für das Player-Interface eingesetzte ManParamsCalc-Objekt nur, wenn die getrackte Person nicht den Rand des Bildes berührt. Zweitens rauschte die von mir eingesetzte Kamera an den Rändern ein wenig. An die Background-Substraction wie bei Kamera 2 habe ich noch einen Median-Filter angehängt, um Bildrauschen zu entfernen, das bei dem ManParamsCalc-Objekt zu Störungen führt. Das Bild wird dann an die Bildverarbeitungen von Player und Recorder weitergegeben.

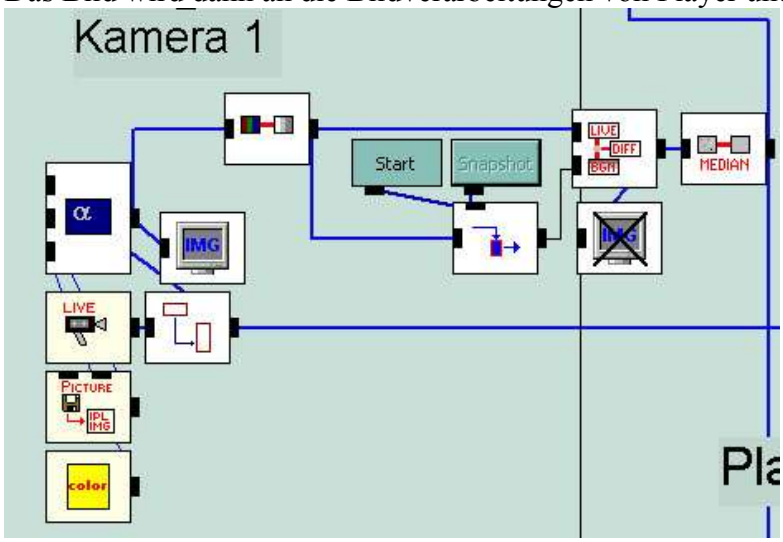

Abbildung 6: Kamera 1, Bildverarbeitung

2.2.1 Recorder

Die Bildverarbeitung des Recorders (Abb. 7) funktioniert anfangs genauso wie die des Player/Recorder-Switches. Ein Teil des Bildes, in diesem Fall ein Streifen über Kopfhöhe (Abb. 3), wird extrahiert, konvertiert und summiert und ein Signal weitergegeben, wenn ein einstellbarer Schwellwert überschritten wird. Wird der Schwellwert erreicht, setzt sich ein Zähler in Gang. Dieser zählt, solange der Schwellwert überschritten ist. Hat er eine bestimmte Zahl erreicht, die man über einen Regler einstellen kann, gibt er ein Signal an den Recorder weiter, das diesen aktiviert oder deaktiviert. Der Regler steuert die Zeit, die man mit der Hand in dem als virtueller Knopf definierten Bereich (Abb. 3) verbringen muss, bevor der Recorder aktiviert oder deaktiviert wird. Dadurch wird ein versehentliches An- oder Ausschalten verhindert.

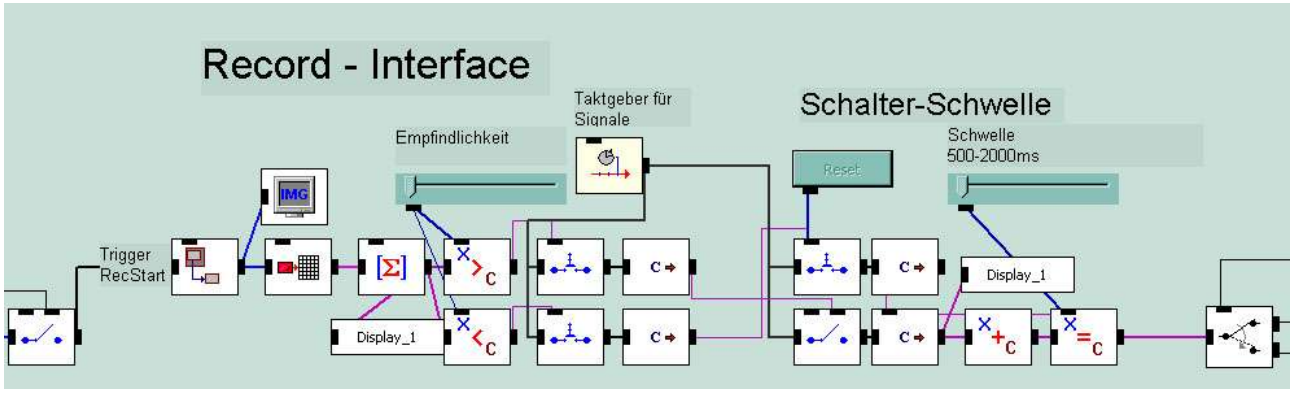

Abbildung 7.1: Recorder-Interface

Wird also der virtuelle Knopf betätigt, aktiviert oder deaktiviert er über einen in 2.2.2 beschriebene Schalter-Konstruktion ein PictureWrite-Objekt. Dieses ist direkt mit Kamera 1 verbunden und schreibt die Bilder als jpeg auf die Festplatte. Gleichzeitig läuft ein Counter, der die Anzahl der aufgezeichneten Frames festhält. Überschreitet die Anzahl der Frames einen bestimmten Wert, wird der Recorder automatisch gestoppt. Der Endwert wird dann an den Player weitergegeben. Während der Aufnahme leuchtet ein kleines rotes Quadrat auf einem separaten Display (Record-Indicator). Ist die Aufnahme beendet, wird es wieder schwarz.

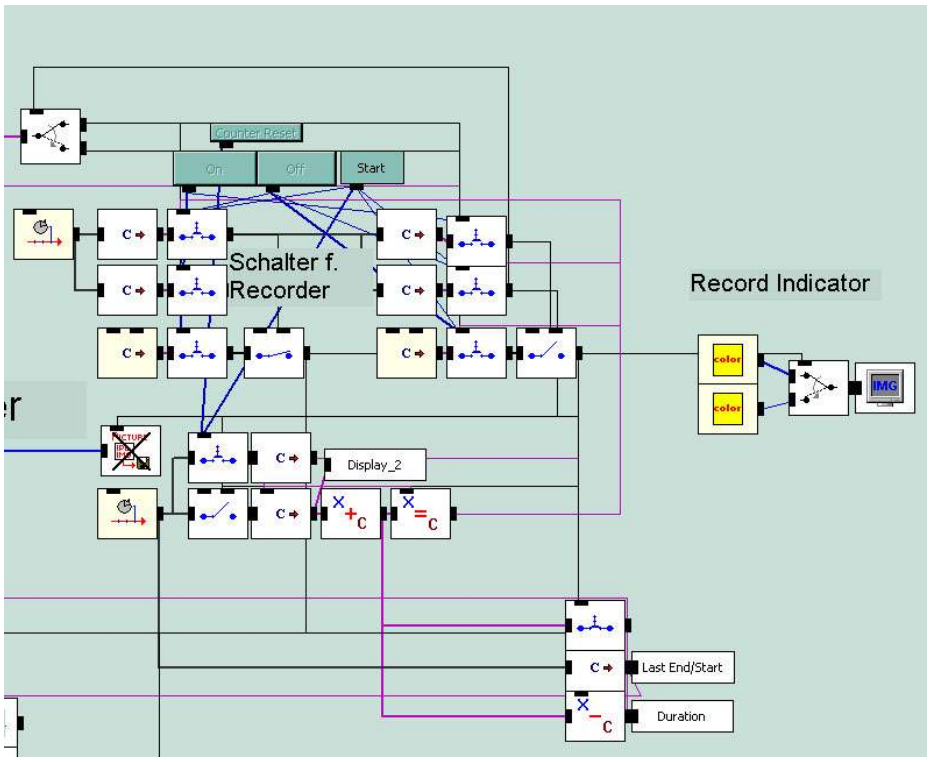

Abbildung 7.2: Schalter für Recorder, PictureWrite, Record-Indicator

2.2.2 Arbeit mit dem PictureWrite-Objekt

Das PictureWrite erwies sich in zweierlei Hinsicht als sehr problematisch. Als erstes musste ich feststellen, dass es keinen Eingang zum An- und Ausschalten besitzt, der ein explizites An- oder Aus-Signal z. B. in Form von 1 und 0 annimmt. Stattdessen kann man es nur durch ein beliebiges Signal aktivieren und deaktivieren. Dieses Problem hat es mit den meisten EyesWeb-Objekten gemein, nur dass es bei den anderen Objekten meist ausreicht, wenn man ihnen das Eingangssignal mittels eines Switch-Objektes wegnimmt, das wiederum mit 1 und 0 gesteuert werden kann. Das PictureWrite-Objekt würde ohne Signal aber weiterhin Bilder auf die Festplatte schreiben und diese nach und nach überfüllen. Diese Problem habe ich mit dem Schalter für den Recorder (Abb. 7.2) gelöst. Signale, die den Recorder anschalten sollen sind hier so verbunden, dass ein Puls in den activate/deactivate-Eingang des PictureWrite-Objekts gesendet wird. Ist dies passiert, wird danach die Verbindung zum PictureWrite unterbrochen und dafür eine Verbindung zu einem Schaltkreis hergestellt, bei dem Signale eintreffen, die PictureWrite ausschalten. Trifft hier ein Signal ein, wird wiederum die Verbindung zu den Anschalt-Signalen hergestellt und die zu den Ausschalt-Signalen unterbrochen. Auf ein Einschalten kann so nur ein Ausschalten folgen.
Ein weiteres Problem bei PictureWrite war das aufzeichnen der Bilder. Ich ging davon aus, das der Aufzeichnungs-Modus „SlideShow_Overwrite“ eine Serie von Bildern aufzeichnet und bei erneutem aktivieren des Blocks wieder von vorne überschreibt. Leider tut er das erst, wenn man das gesamte Patch neu startet, was bei dieser Installation nicht möglich ist. Stattdessen hängt er die Bilder der folgenden Aufzeichnungen einfach an die Reihe der vorherigen an, wie er es bei der Option „SlideShow_Append“ tun sollte. In der EyesWeb-Dokumentation war zu dem Block keine Erläuterung zu finden. Eine wirkliche Lösung für diese Problem habe ich noch nicht gefunden. Momentan schreibt das Patch so lange Bilder auf die Festplatte, bis sie voll ist. Sobald die Aufnahme beendet ist, wird der Zählerstand zu Beginn der Aufnahme auf den vom Player-Interface berechneten InPoint aufaddiert. Die Aufnahmedauer wird auf den berechneten OutPoint addiert.

2.2.3 Player

Die Bildverarbeitung (Abb. 8.1) des Players benutzt den ManParamCalc-Block, um ein Rechteck um die anwesende Person zu ziehen. Über die Koordinaten dieses Rechtecks kann gemessen werden, wie gross der Abstand zwischen linker und rechter Ecke ist, und wo es sich auf der X-Achse befindet. Zusammen mit den Daten über die Länge des aufgenommenen Videos werden dann die relativen Standpunkte der beiden Ecken auf der Timeline berechnet. Diese werden dann als In- und Out-Point an den Player weitergegeben. Der In- und Out-Point kann durch den Regler „Cropping“ noch einmal verschoben werden. Absicht dabei ist es, den Moment aus dem Video zu filtern, in dem der virtuelle Knopf betätigt wird.
Der Player (Abb. 8.2) besteht aus einem Zähler, der vorwärts zählt, bis sein Wert den des Out-Points erreicht hat. Dann wird er umgeschaltet und zählt rückwärts bis zum In-Point, wo er wieder in die andere Richtung umspringt. Dem Zählerstand wird im Zählwerk ein Dateiname angehängt und gemeinsam dann an ein PictureRead ausgegeben. Dieses liest das entsprechende Bild aus der aufgenommenen Sequenz auf der Festplatte, das dann auf einem Display sichtbar wird.

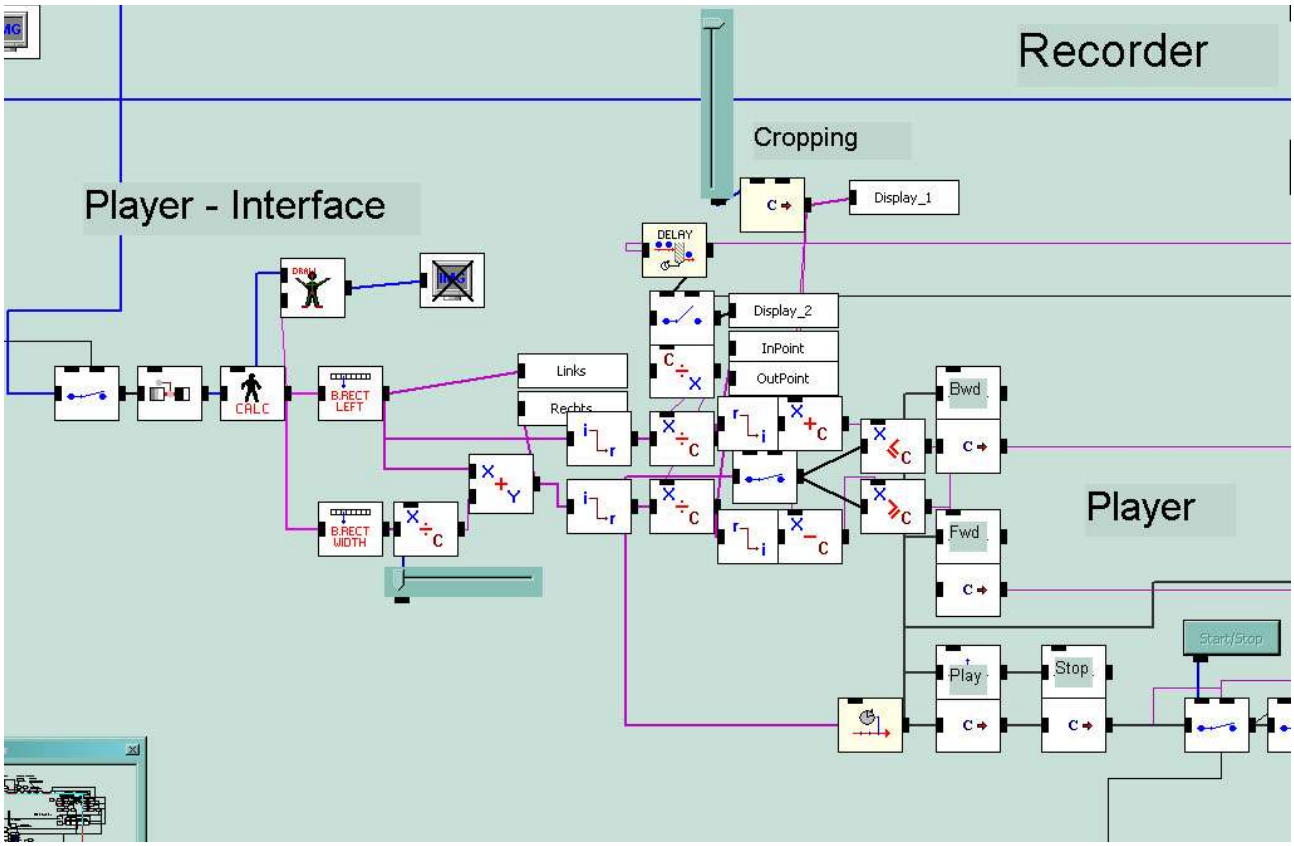

Abbildung 8.1: Player Bildverarbeitung, Cropping

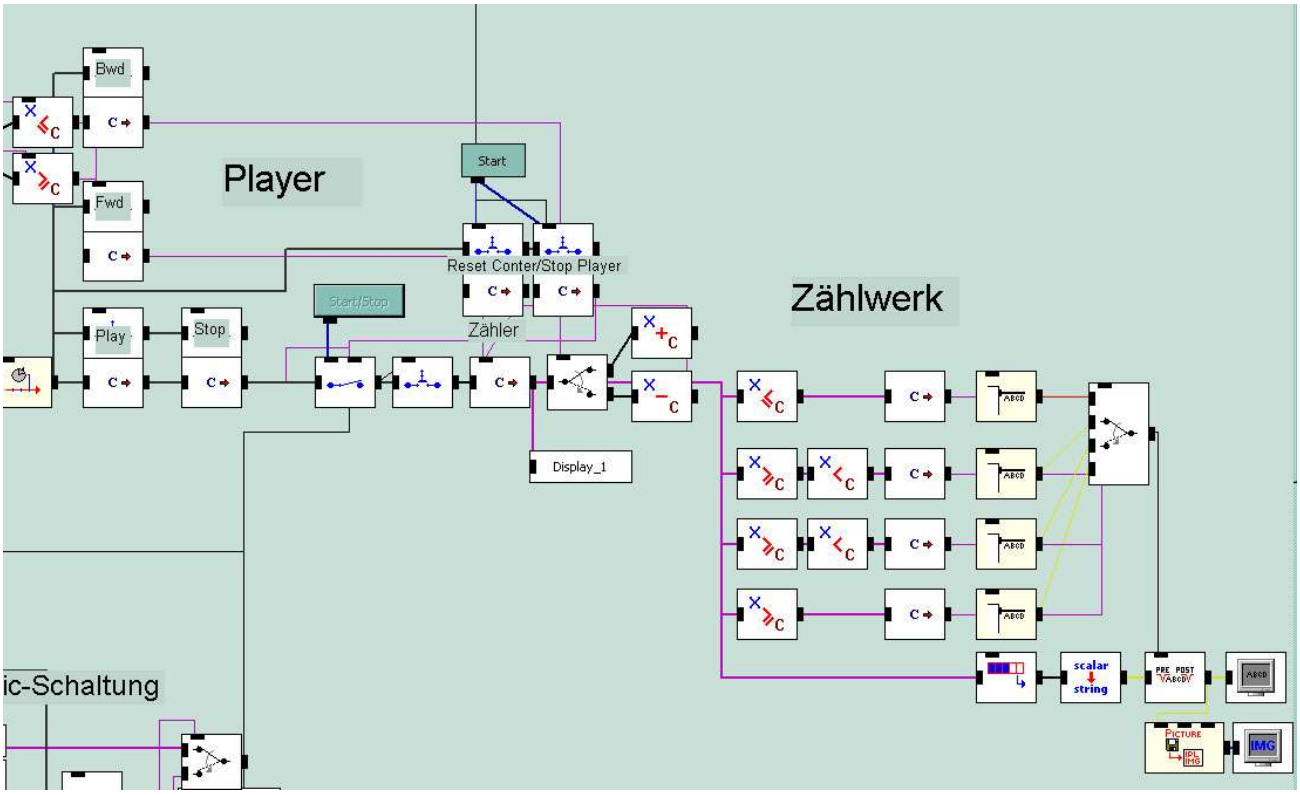

Abbildung 8.2: Player, Zählwerk für PictureRead

3 Arbeitsbericht

3.1 Arbeiten mit EyesWeb

Bei der Arbeit mit EyesWeb fiel mir auf, dass viele Vorgänge, die in ähnlichen Programmen kompliziert zu programmieren sind, wie etwa eine Background-Substraction, sehr schnell und einfach zu realisieren sind. Die Tücke des Programms liegt mehr bei einfachen logischen Vorgängen, die dann wiederum schwer zu lösen sind. Beispielsweise besitzt das PictureWrite-Objekt, das ich zur Aufnahme verwende, keinen Eingang, der ein An-/ oder Aus-Signal (z. B. in Form von 1 und 0) annimmt. Stattdessen schaltet es zwischen An und Aus durch Eingabe eines beliebigen Signals hin und her (siehe 2.2.2). Ich musste daher eine relativ aufwendige Schaltung bauen, die feststellte, ob PictureWrite gerade ein- oder ausgeschaltet war (Abb. 7). Ähnlich verhielt es sich mit dem Zählwerk zum auslesen und abspielen der Bilder (Abb. 8.2).
Die Tatsache, dass in EyesWeb 3.x keine Subpatches benutzt werden, macht besonders umfangreiche Patches schnell unübersichtlich. Insbesondere bei der Fehlersuche kann es extrem nervig sein, wenn man sich durch 6 aneinander hängende Bildschirme voller Objekte arbeiten muss. Die 4.x Version, die wohl Subpatches zulässt, war zum Zeitpunkt dieser Arbeit meines Erachtens noch nicht zuverlässig genug, um benutzt zu werden. Die von mir benutzte Version 3.2.0 stürzte zeitweise unter anderem wegen unzulässiger Vermischung von realen und imaginären Zahlen kommentarlos ab. Falsch angegebene Auflösungen beim Framegrabber-Objekt führen ebenfalls zum ungewollten Beenden der Software.
Insgesamt empfiehlt es sich, einzelne Elemente der Patches zuerst einmal separat aufzubauen und zu testen und dann später nacheinander zusammen zu fügen. Auf diese Weise bleibt das ganze übersichtlich, und Fehler sind schneller gefunden.

Die Zuverlässigkeit der Steuerung hängt sehr von der Qualität des Kamerabildes ab. Unter anderem habe ich festgestellt, dass es nicht unbedingt ratsam ist, mit einem weißen Hintergrund bei starker Beleuchtung zu arbeiten. Glanzpunkte und sehr helle Partien, wie zum Beispiel Hände, werden dann als Hintergrund heraus gefiltert. Bei der Arbeit mit dem ManParamsCalc-Objekt habe ich festgestellt, dass es kein Viereck um die Person zeichnet, wenn sie den Bildrand berührt. Ich habe deswegen mit AlphaBlend eine Maske auf das Kamerabild gelegt, die alle Pixel am Rand schwarz einfärbt.

3.2 Performance

Das vorliegende Patch ist sehr rechenaufwändig. Mein Computer (Centrino 1,5GHz, 1,2 Gbyte Ram) ist damit überlastet. Die Überlastung führt unter anderem zu Timing-Schwierigkeiten beim Record-Zähler und der Weitergabe von Signalen. Das liegt unter anderem daran, dass es in EyesWeb Unverträglichkeiten zwischen die beiden verfügbaren Webcams gab und ich ein Videosignal über DV empfangen musste, was mehr Leistung benötigt. Zum Zeitpunkt dieses Berichts versuche ich gerade die Leistungsaufnahme zu verringern, indem ich beim Abschalten der Interfaces nicht nur das Videosignal unterbreche, sondern auch die entsprechenden Blocks deaktiviere. Alternativ dazu kann man auch Rechenleistung auslagern. Man könnte die gesamte seitliche Beobachtung auf einem separaten Rechner laufen lassen und die Steuersignale über Netzwerk an die Interfaces weitergeben.

4 Ausblick

Die Installation hat eine Menge Potential. Zum Beispiel lässt sich leicht noch der Steuerungsfaktor Geschwindigkeit hinzufügen. Die Größe der Person könnte die Framerate des Videos beeinflussen. Die Performance lässt sich sicher noch verbessern, indem man einige Abläufe vereinfacht und eventuell unnötige Blocks entfernt oder ersetzt. Ich bin mir zum Beispiel sicher, dass es eine weniger aufwändige Lösung für das Problem des Bildrandes gibt, als den Alpha_Blend-Block. Die gesamte Installation ließe sich auch mit nur einer Kamera realisieren, was ebenfalls Performance einsparen würde. Der von mir entwickelte On/Off Schalter kann auch weniger umfangreich realisiert werden.

Statt Videodaten könnte auch Audiomaterial beeinflusst werden. Allerdings zeigte sich EyesWeb bisher bei der Verarbeitung von Audiodaten als sehr buggy, so dass hier die Auslagerung an ein anderes Programm empfehlenswert wäre.