UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

UN MODELLO PER IL TRASFERIMENTO DELLE PERFORMANCE MULTIMEDIALI NEL DOMINIO DIGITALE. UNO STUDIO DI CASO: "IL TEMPO CONSUMA" DI MICHELE SAMBIN

Relatore: Prof. Sergio Canazza Targon

Laureando: Paolo Ostan

Correlatore: Dott. Alessandro Fiordelmondo

ANNO ACCADEMICO 2020-2021
Data di Laurea 22/09/2021

Sommario

In questo lavoro si presentano i passi e gli strumenti utilizzati per il trasferimento nel dominio digitale della performance multimediale dal vivo *Il tempo consuma* di Michele Sambin.
Vengono approfonditi gli aspetti riguardanti il ruolo della tecnologia nello sviluppo dell'opera originale, risalente al 1978, oltre alla ricerca artistica e tecnica per la realizzazione del sistema utilizzato durante le performance con il *videoloop*. Si passa quindi all'analisi degli elementi che compongono il sistema di elaborazione digitale per il recupero dell'opera, introducendo inizialmente la composizione hardware del sistema utilizzato. Successivamente vengono trattati i componenti software realizzati tramite *Max/MSP*, ambiente di sviluppo grafico ad oggetti utilizzato per la creazione di applicazioni di elaborazione musicale e multimediale. Dopo una breve introduzione al linguaggio saranno trattate nel dettaglio le componenti che hanno permesso la creazione del software utilizzato per la riproduzione. Vengono quindi analizzati gli oggetti necessari per l'elaborazione video tramite l'utilizzo dei componenti *Jitter*, gli oggetti utilizzati per la processazione audio tramite *MSP*, e infine l'interfaccia implementata grazie al linguaggio *OSC* per il controllo di alcuni parametri del progetto tramite controller esterno.

Ringraziamenti

Desidero ringraziare il Centro di Sonologia Computazionale e in particolare il professor Sergio Canazza per l'opportunità di collaborazione a questo progetto e per la disponibilità concessa.
Ringrazio inoltre Michele Sambin, che ha trasmesso, nel corso di questo lavoro, tutta la sua passione per l'arte e soprattutto per la sperimentazione artistica.
Un ringraziamento particolare va al dott. Alessandro Fiordelmondo, che oltre ad essere stato un confronto per le scelte progettuali e per la stesura dell'elaborato si è sempre reso disponibile per ogni dubbio o chiarimento.
Voglio inoltre ringraziare la mia famiglia e gli amici che in questi tre anni sono stati di supporto in ogni momento.

Indice

[1]Quanto riportato nei paragrafi 1.1 e 1.2 è un estratto di un'intervista svolta con *Michele Sambin* il giorno 10 Agosto 2021. Intervista integrale presente in [10]

Elenco delle figure

Introduzione

Le installazioni di arte multimediale, che siano basate su film, video o computer hanno caratteristiche estremamente diverse tra loro. Tuttavia, elementi quali ad esempio la variabilità, la riproduzione, la performance, l'interazione e l'interconnessione sono punti in comune di questo tipo di prodotti artistici. Queste opere non sono realizzate come oggetti unici e immutabili, sono invece composte in molti casi da una collezione di componenti hardware e software, che insieme creano un'esperienza basata sul tempo e sul processo che le contraddistingue. Ad oggi non esiste una risposta e una metodologia universale per la preservazione e la ri-esibizione di questi lavori. La soluzione a questo problema richiede una ricerca individuale per ogni opera, preferibilmente condotta con l'artista. Infatti, il metodo generale è quello di condurre studi di caso ed interviste con gli artisti e altre figure coinvolte nell'opera. Nel lavoro di preservazione e ri-esibizione dell'opera vanno presi in considerazione tre tipi di processi: conservazione, migrazione, emulazione e virtualizzazione. Questo tipo di processi è distinguibile in due approcci distinti, un approccio "purista" e uno di "adattamento e rinnovo". Il primo fa riferimento all'utilizzo della tecnologia originale e quindi alla conservazione del lavoro il più possibile analogo a come appariva originariamente. Questo criterio utilizza il processo di conservazione. Il secondo approccio si rifà invece all'utilizzo delle nuove tecnologie ed è noto per un aspetto dinamico dell'opera. In questo tipo di soluzioni sono essenziali i processi di emulazione e migrazione. Considerando un'eventuale perdita di autenticità e storicità in relazione alla funzionalità e al concetto originale dell'opera, in questo caso è importante un'attenta discussione relativa alle possibili strategie da applicare [7].

In questo elaborato si prende in considerazione la ricostruzione dell'opera *Il tempo consuma* di Michele Sambin, realizzata dal Centro di Sonologia Computazionale (CSC) dell'Università di Padova. Attraverso la discussione e l'interazione con l'artista, sui vari aspetti dell'opera originale, si è scelta la strategia di ricostruzione. Tale strategia si può includere in un approccio di "adattamento e rinnovo". La ricostruzione consiste nella migrazione dalla tecnologia analogica, utilizzata nell'opera originale, verso una tecnologia digitale. Tale scelta è data da un alto grado di difficoltà nella ricostruzione di un sistema analogico equivalente all'originale. Le problematiche principali nascono dall'estrema obsolescenza del materiale utilizzato all'epoca. Sebbene la preservazione sia l'obiettivo principale, un elemento che ha guidato la scelta della strategia di ricostruzione è stato anche il carattere sperimentale che contraddistingue la produzione artistica di Sambin. Si è quindi optato per lo sviluppo di un sistema che permettesse all'artista di andare oltre l'opera originale consentendo la produzione di risultati artistici innovativi. La scelta è stata compiuta dall'artista stesso, che preso atto delle possibilità offerte dal mezzo digitale, ha desiderato esprimere la propria creatività al di là della riproduzione dell'opera originale.

Capitolo 1

L'opera da ri-mediare: *Il tempo consuma* di Michele Sambin

Il tempo consuma è un'installazione multimediale del 1978, creata dall'artista padovano Michele Sambin. Michele Sambin è un musicista, pittore e regista che sin dalle sue prime esperienze cinematografiche degli anni '60 ha focalizzato la sua ricerca artistica sul rapporto tra immagine e suono. *Il tempo consuma* nasce dalla sperimentazione dell'artista con il *videoloop*, che diventerà lo strumento principale delle sue opere per alcuni anni. Tra le opere di Sambin che utilizzano il *videotape* ci sono *VTR&I* del 1978, *SAX* del 1979 e *Autointervista* del 1980. Tuttavia, *il tempo consuma* rappresenta il climax di questa esperienza, sia per l'utilizzo particolare della tecnologia sia da un punto di vista del linguaggio artistico. Infatti, l'esperienza di quest'opera influenzerà in modo decisivo tutta la sua produzione artistica successiva [11].

1.1 La tecnologia nell'opera d'arte[1]

Analizzando il repertorio artistico di Michele Sambin, oltre alle opere con il *videotape*, è evidente un particolare interesse per la tecnologia. Seppur si trova spesso l'utilizzo di strumenti semplici che non richiedono particolari competenze tecniche, quello che caratterizza il linguaggio di Sambin è un uso singolare della tecnologia. Per l'artista, la tecnologia è uno stimolo per il pensiero creativo, fonte di risultati artistici innovativi. Un esempio è il portare al limite le possibilità fornite da uno strumento. Questo si può notare nelle opere realizzate con la tecnica del *videoloop*, delle quali fa parte anche *Il tempo consuma*. Lo stesso approccio si può vedere anche nel mondo del digitale. In spettacoli più recenti (per esempio *Tutto è vivo* del 2008 e *Finito illimitato* del 2014 [11]), Sambin utilizza la pittura digitale in tempo reale per creare delle vere e proprie scenografie teatrali viventi, in continua trasformazione.

1.2 *Il tempo consuma*

La sperimentazione in Michele Sambin copre tutti gli aspetti dell'opera. *Il tempo consuma* è un perfetto esempio di sperimentazione con il *videotape*, sia dal punto di vista tecnologico sia dal punto di vista compositivo.
L'opera in questione, del 1978, nasce dal desiderio di sperimentazione con il video in tempo reale. L'idea principale si sviluppa grazie ad un evento fortuito: il ritrovamento di un pezzetto di nastro adesivo argentato, già tagliato a 45 gradi e incollato sul nastro *open-reel* [2], utilizzato probabilmente per effettuare una giunta in caso di rottura. Questo, insieme alle influenze dei lavori di improvvisazione con il nastro di Terry Riley, porta alla nascita di *Il tempo consuma*, che lo stesso Riley ha modo di vedere a casa di Sambin, rimanendone entusiasta [4]. La

[1] Quanto riportato nei paragrafi 1.1 e 1.2 è un estratto di un'intervista svolta con *Michele Sambin* il giorno 10 Agosto 2021. Intervista integrale presente in [10]

[2] nastro magnetico in cui un'estremità viene lasciata libera, utilizzato in apparecchi di registrazione degli anni '70

costruzione e l'utilizzo del sistema per i *videoloop* porta a una nuova concezione ciclica del tempo, in cui la sovrapposizione di varie ripetizioni del singolo ciclo genera profondità. Questa esperienza, nata dalla possibilità tecnologica fornita dal sistema, dà vita a un nuovo linguaggio estetico del compositore. Questo lo caratterizzerà per il resto della propria esperienza artistica, non solo per le opere di *videoloop*.

1.3 Riproduzione dell'opera

La decisione di riproporre *Il tempo consuma* è in buona parte determinata dall'importanza che l'opera ha nel repertorio del compositore, ma anche dalla rilevanza che il sistema di *videoloop* ha nella videoarte. Inoltre, in coerenza con l'esperienza dell'artista, si è deciso di sperimentare il sistema *videoloop* nell'ambiente digitale.

1.4 Il sistema da recuperare

Il sistema originale era composto da una telecamera, due registratori (uno in registrazione e uno in lettura) e due schermi (uno utilizzato per la generazione del loop e uno per la riproduzione della performance). L'elemento fondamentale era un nastro video ad anello e quindi congiunto alle due estremità da un nastro adesivo. Il segnale della telecamera era affidato ad uno dei due registratori che supportava un nastro ½ pollice e consentiva il loop. Il registratore in lettura era collegato ad uno dei due monitor in diretta; l'altro schermo (di fronte al quale si posizionava Sambin) emetteva invece in differita. La distanza tra i due videoregistratori o, per meglio dire, tra la testina di registrazione del primo e quella di lettura e trasmissione del secondo, determinava il tempo del ritardo dell'immagine emessa dal secondo monitor; ciò permetteva a Sambin di realizzare il suo loop. (fig. 1.1) La telecamera inquadrava lo schermo in differita e, come già detto, tra i due oggetti si situava Sambin che, in questo caso, dondola il proprio corpo imitando un metronomo; il ritmo del movimento del busto scandisce le due

affermazioni dette dall'artista: "*il tempo consuma le immagini; il tempo consuma i suoni*". Dopo un determinato numero di secondi (circa 31, il tempo differito) ciò che è stato registrato è emesso dallo schermo in differita, di fronte al quale Sambin continua ad agire. Ciò che accade è che la telecamera, questa volta, riprenderà sia il Sambin "reale" di fronte allo schermo, che ciò che è emesso dal monitor stesso. Il risultato è una sovraimpressione dell'immagine - di natura diversa da quella filmica – e del suono. Dopo questo primo momento di registrazione, Sambin si scosta smettendo di interferire tra monitor e telecamera. La telecamera continua a registrare quanto avviene sul monitor in differita; poiché l'immagine emessa dal monitor ha una risoluzione minore, ogni volta che la telecamera ri-riprende quanto avviene sullo schermo, l'immagine risulterà sempre meno definita e allo spettatore parrà consumata; lo stesso procedimento avviene per il suono, che si distorce fino a rendere incomprensibile quanto detto dall'artista. (fig. 1.2)
[9]

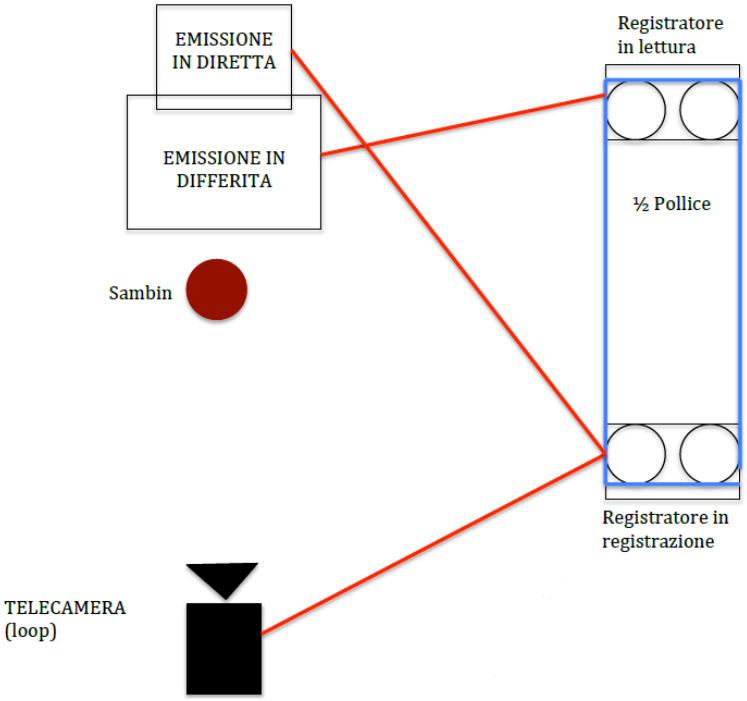

Figura 1.1: *Schema del sistema originale utilizzato durante le performance*

Figura 1.2: *Versione originale di Il tempo consuma*

Capitolo 2

Riproduzione digitale

Il recupero della performance è avvenuto mediante una trasposizione dal dominio analogico al dominio digitale. La scelta di tale trasposizione, avvenuta in accordo con l'artista, è data da un alto grado di difficoltà nella ricostruzione di un sistema analogico equivalente all'originale. Le problematiche principali nascono dall'estrema obsolescenza del materiale utilizzato all'epoca.

2.1 Descrizione sistema di riproduzione digitale

Di seguito viene presentata e descritta la lista dei componenti hardware che compongono il sistema di riproduzione digitale. (fig. 2.1)

- **Dispositivo di acquisizione video**: Telecamera accoppiata a scheda di acquisizione video.
- **Dispositivo di riproduzione video**: Schermo 32″ 16:9 alle spalle del performer
- **Dispositivo di acquisizione audio**: Convertitore analogico-digitale con microfono a condensatore supercardioide

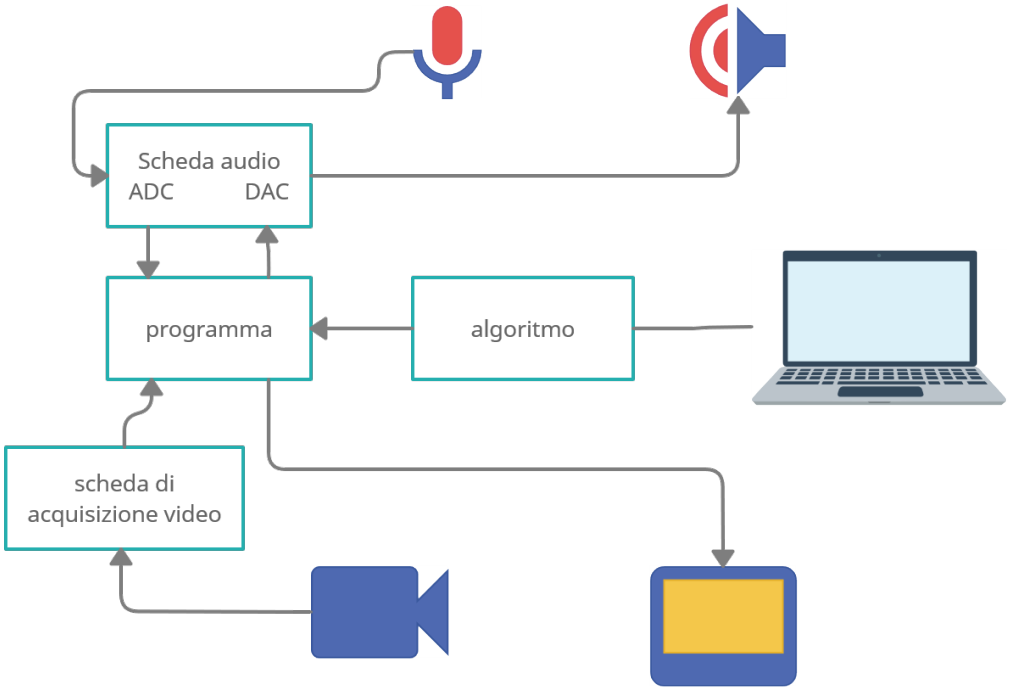

Figura 2.1: *Schema sistema acquisizione hardware*

- **Dispositivo di riproduzione audio**: Convertitore digitale-analogico con impianto di diffusione audio.
- **Computer**: Computer con *patch Max/MSP* per l'elaborazione di segnale video e audio

Il computer è il nucleo del sistema di processazione, di cui viene data una prima rappresentazione complessiva nella figura 2.2.

2.2 Introduzione al linguaggio MAX/MSP

Il lato software del progetto (fig. 2.2) è stato realizzato attraverso l'ambiente di sviluppo grafico (*Visual Programming Language* - VSL) *Max/MSP*. Questo linguag-

gio, di livello estremamente avanzato, è dedicato allo sviluppo di algoritmi per la musica e la multimedialità. Ideato e scritto, in origine, da Miller Puckette viene successivamente sviluppato ed esteso da David Zicarelli e commercializzato dalla sua compagnia: *Cycling '74* [2], che tuttora ne detiene la proprietà e lo mantiene aggiornato. È stato il primo linguaggio di programmazione pensato per musicisti, si basa infatti sulla metafora della programmazione del sintetizzatore analogico, con oggetti collegati tramite cavi che possono trasportare sia segnali di controllo, sia audio o video. [5]
Si divide in tre componenti principali:

- *Max*: Componente del linguaggio dedicata alla generazione e alla gestione grafica di segnali di come il MIDI, per il controllo di unità esterne dedicate alla sintesi audio. [13]

- *MSP*: Implementato successivamente alla componente originale Max, dedicato alla manipolazione audio tramite l'uso di algoritmi appositi per sintesi e elaborazione audio, offre inoltre la possibilità di interfacciarsi con DSP hardware. Gli oggetti *MSP* sono identificati da una tilde finale. [13]

- *Jitter*: Estensione del linguaggio dedicata alla generazione ed elaborazione grafica. L'insieme degli oggetti jitter consente la manipolazione di matrici di pixel ma anche processi di livello superiore quali, per esempio, l'acquisizione video, la manipolazione e la *chromakey*. Gli oggetti *Jitter* iniziano con il prefisso `jit.`.

Un file *Max/MSP*, composto da un insieme di oggetti collegati fra loro che svolge una determinata funzione viene definito *patch*.
Nel contesto di questo progetto, l'utilizzo di un ambiente di sviluppo grafico come *Max/MSP* ha permesso di condensare l'elaborazione del software risultante nei confronti dei processi audio e video, sollevando pertanto il programmatore dalla necessità di lavorare in maniera approfondita sui componenti di basso livello. Viene fornita nelle figure 2.2 e 2.3 una panoramica della *patch* principale del progetto. *Max/MSP* permette di impostare una modalità "presentazione" (fig.

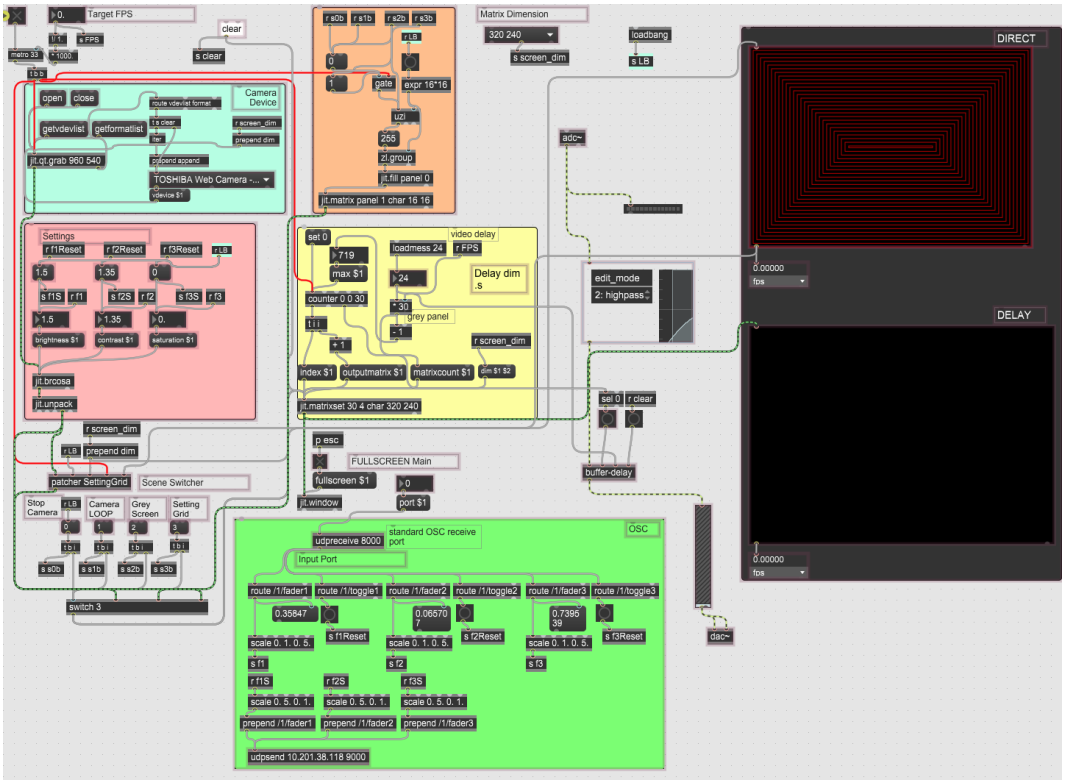

Figura 2.2: *Patch principale progetto*

2.3) con la quale decidiamo di visualizzare solamente gli elementi utili (escludendo dalla visualizzazione cavi, oggetti, ecc). La modalità di presentazione viene specificatamente utilizzata durante la performance per la gestione del sistema di controllo

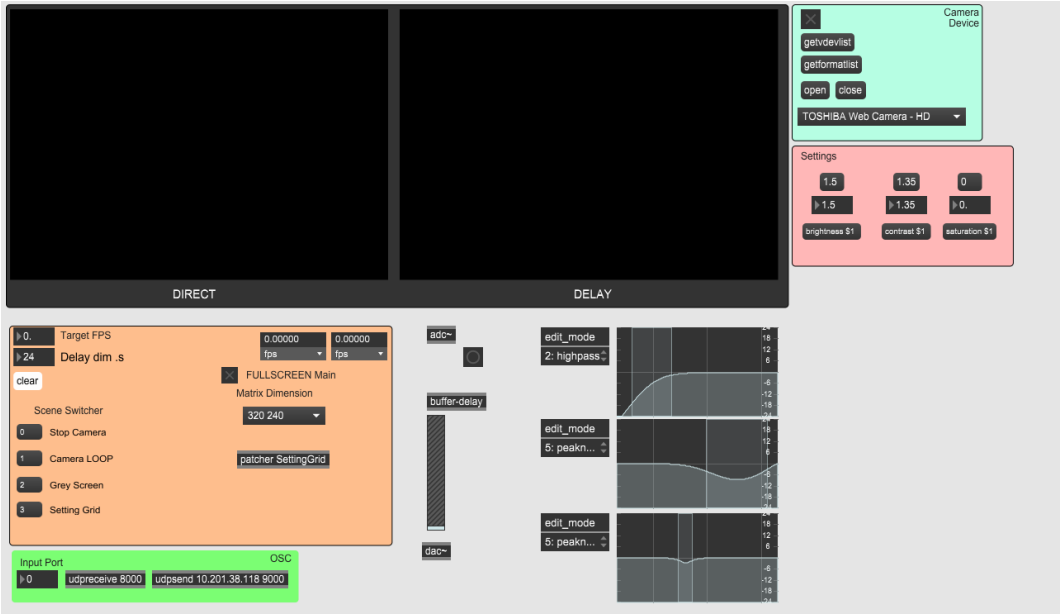

Figura 2.3: *Patch principale progetto in modalità presentazione*

Capitolo 3

Video

3.1 Acquisizione Sample Video

Il sistema di elaborazione inizia con l'acquisizione del flusso video all'interno dell'ambiente di sviluppo. Il sistema di acquisizione è rappresentato in figura 3.1. L'oggetto `jit.grab` [3], permette alla patch di interfacciarsi con i dispositivi di input video collegati al computer. Attraverso l'oggetto `metro` [3] è possibile determinare la velocità di acquisizione dei frame (*frame per senconds* - FPS) da parte del `jit.grab`. Sono presenti inoltre alcuni oggetti e messaggi utilizzati per il controllo dei parametri aggiuntivi come la visualizzazione dei vari dispositivi di acquisizione disponibili, la scelta del dispositivo di input da utilizzare, l'avvio e lo spegnimento del sistema di comunicazione.

3.2 Creazione Loop Video

Una volta acquisito il flusso video, questo viene inviato alla sezione di delay che consente la generazione di una sua copia ritardata nel tempo. Per questo scopo viene utilizzato l'oggetto `jit.matrixset` [3] il quale permette la registrazione di una collezione di frame che possono essere riprodotti in modo differito. È quindi possibile scandire la scrittura dei *frame* all'interno dell'oggetto e successivamen-

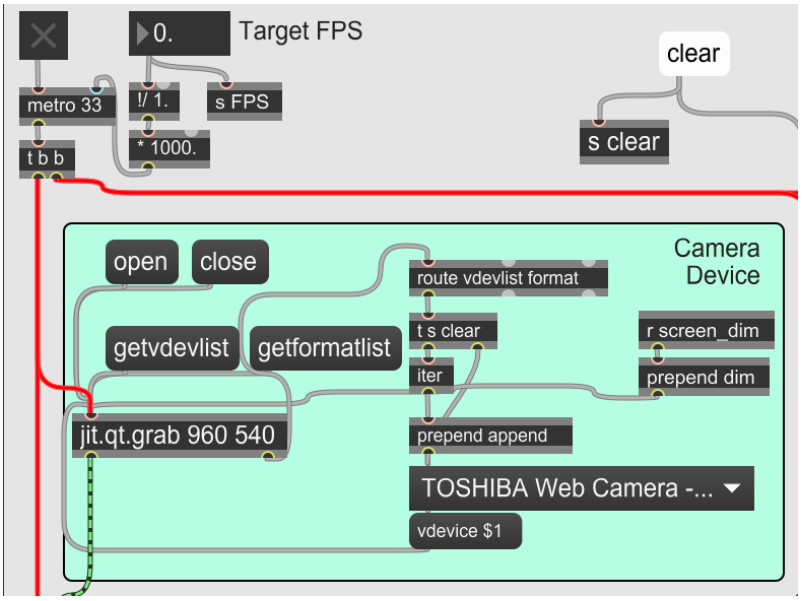

Figura 3.1: *Sistema di acquisizione*

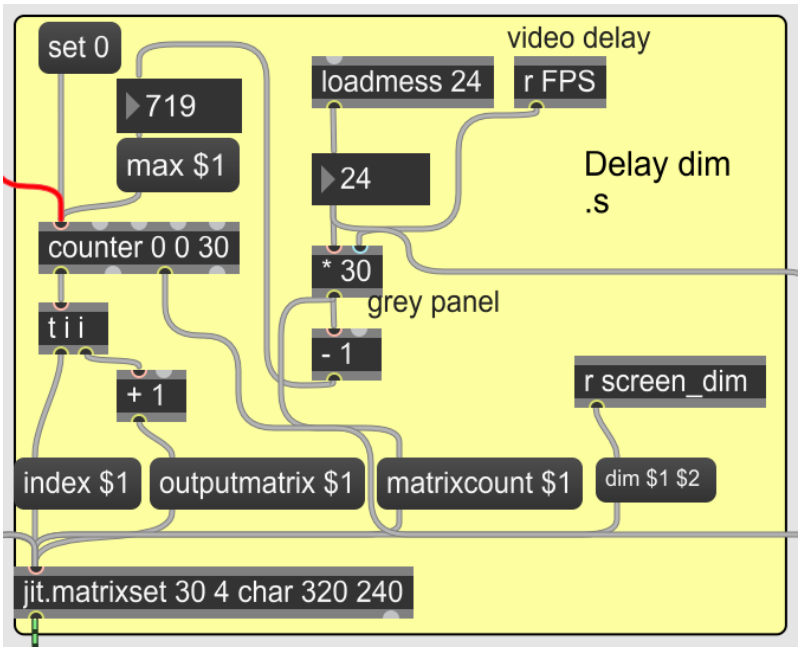

Figura 3.2: *Video delay*

te effettuarne la lettura dopo un tempo specifico. Viene utilizzato un oggetto `counter` [3] che permette la registrazione e la riproduzione dei singoli frame secondo un indice ascendente e un oggetto `metro` che attiva il `counter` e determina la frequenza di scrittura e lettura. Nella patch la dimensione del delay può essere scelta in secondi. Pertanto il counter registrerà una serie di matrici con un indice ascendente da 0 a $(\text{sec} * \text{fps}) - 1$.

3.3 Manipolazione parametri video

La patch consente la gestione del flusso video e di parametri specifici. La gestione del flusso video consente all'utente di accendere e spegnere l'acquisizione in tempo reale del video, di posizionare accuratamente la telecamera e di impostare una schermata iniziale per la performance. La patch è fornita di un selettore che consente di scegliere tra quattro diverse opzioni. (fig. 3.3)

- **Stop Camera**: Viene bloccata l'acquisizione di immagini da parte del programma, e di conseguenza è bloccata anche la riproduzione. Quest'impostazione viene selezionata di default all'avvio del programma.
- **Camera LOOP**: Impostazione che consente di utilizzare come fonte le immagini acquisite dalla sorgente di input.
- **Grey Screen**: Questa impostazione viene selezionata per generare un'immagine statica e monocromatica, utilizzata nello specifico come fonte iniziale delle performance. Il flusso video viene generato a partire da una matrice monocromatica di pixel. (fig. 3.4)
- **Setting Grid**: Viene inviata come sorgente del video loop l'immagine composita generata tramite il componente `setting grid`, discusso successivamente.

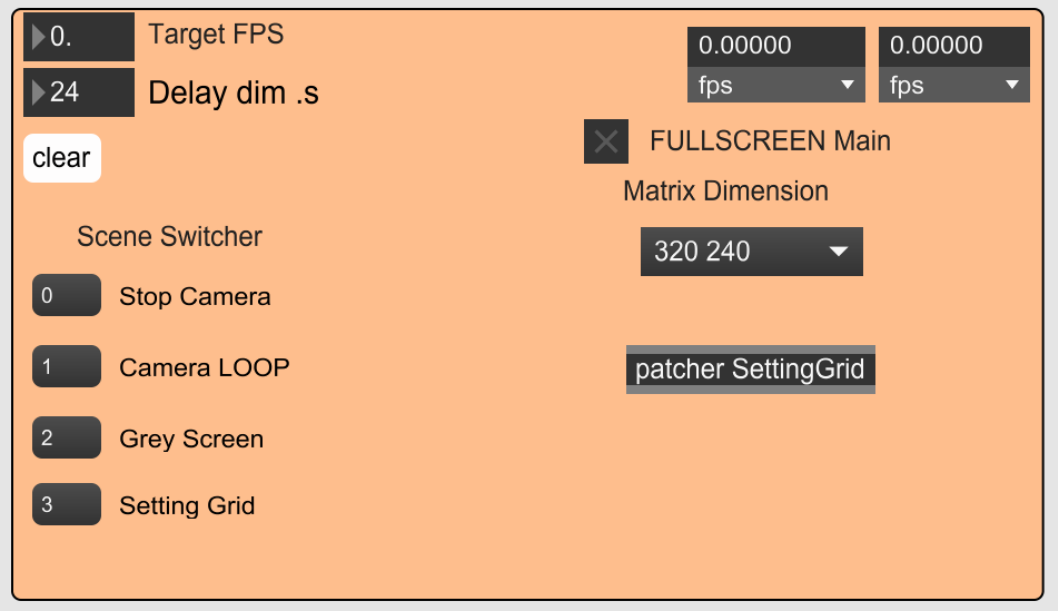

Figura 3.3: *Impostazioni video*

Il sistema implementato permette inoltre il controllo di alcuni semplici parametri dell'immagine processata, quali luminosità, contrasto e saturazione. Il controllo di tali parametri è possibile tramite l'utilizzo dell'oggetto `jit.brcosa` [3], (fig. 3.5)
L'oggetto viene posto all'uscita del componente di acquisizione, così da permettere la trasformazione dell'immagine utilizzata come sorgente per tutti i processi successivi. Come mostrato in figura 3.5, viene inoltre utilizzato l'oggetto `jit.unpack` [3] con il quale vengono separati i piani della matrice ARGB (*Alpha Red Green Blue*) e prelevato solamente il colore verde. L'utilizzo di una matrice ad un unico piano ci consente di ottenere un immagine in bianco e nero, fedele alla versione originale dell'opera.

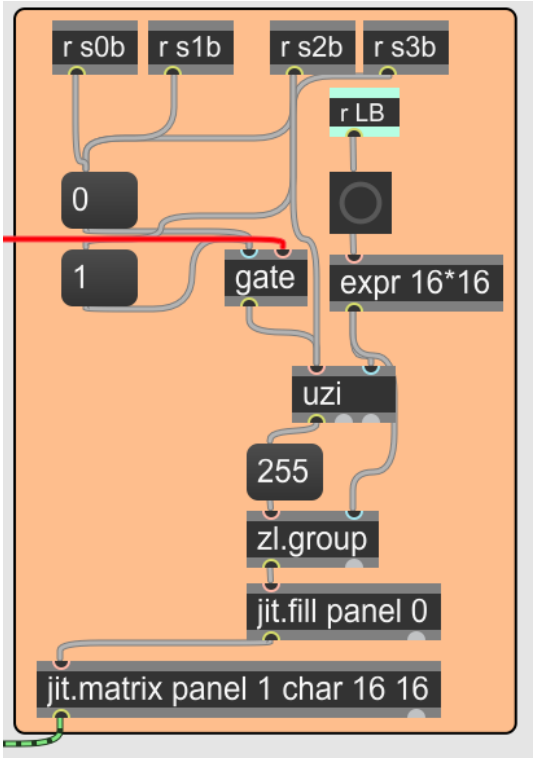

Figura 3.4: *Grey Screen*

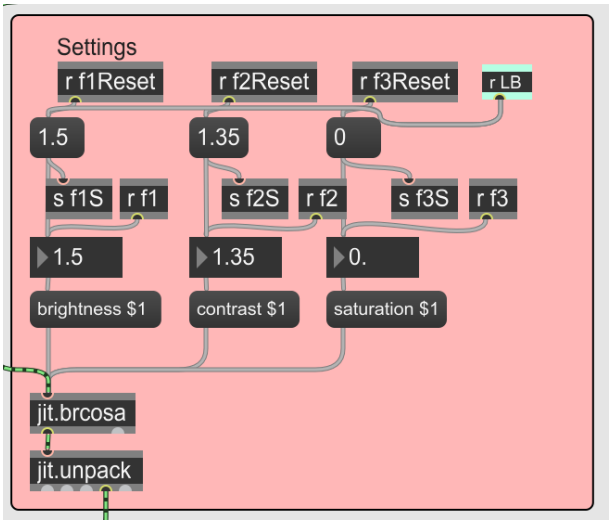

Figura 3.5: *Impostazione parametri*

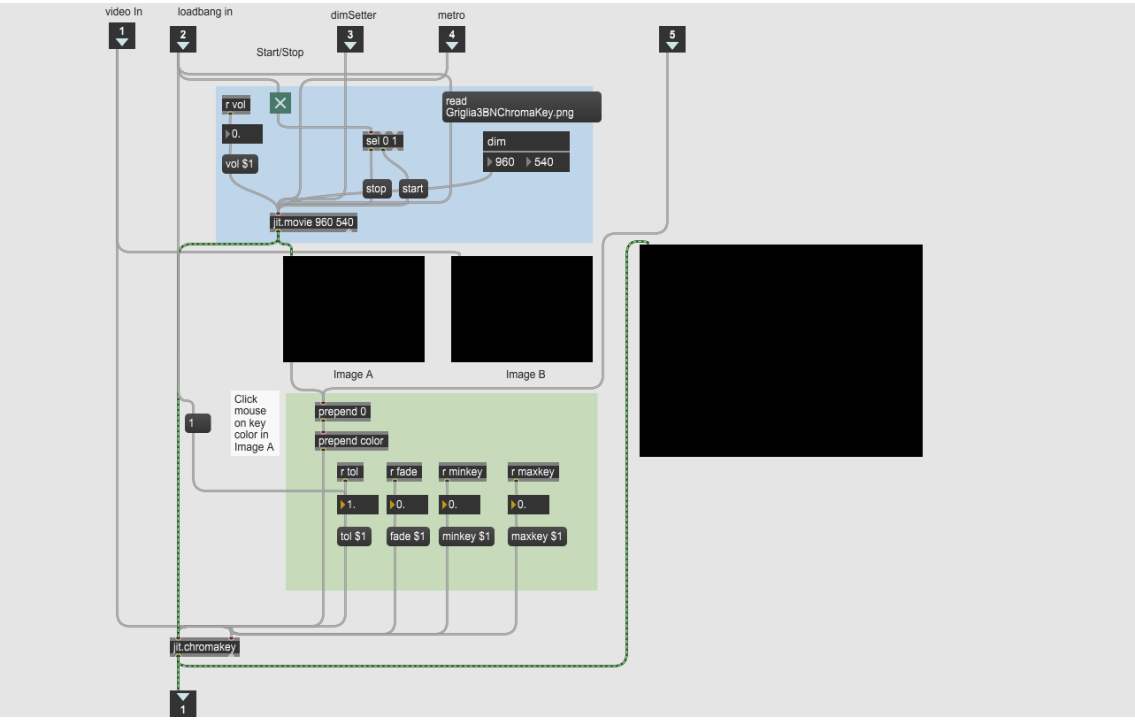

Figura 3.6: *SettingGrid*

3.4 Allineamento tramite Chroma-Key Jitter

Durante i primi test di riproduzione si è presentata la necessità di allineare precisamente il dispositivo di acquisizione con lo schermo utilizzato per la riproduzione, così da evitare distorsioni delle immagini generate, a causa dell'interazione fra i due dispositivi. Questo processo può essere effettuato solo manualmente a causa della composizione dell'apparato. Tramite un insieme di oggetti appartenenti a *Jitter*, è stato quindi creato uno strumento che permettesse la creazione di una griglia di allineamento con la quale velocizzare il processo.
Il sistema è contenuto all'interno di `SettingGrid`. (fig. 3.6) Presente all'interno del progetto come `bpatcher`. [1] [3]
Il cuore del sistema di allineamento è l'oggetto `jit.chromakey` [3], che esegue

[1] subpatch all'interno di una patch

l'omonimo effetto di composizione di immagini: la creazione di un'immagine in cui vengono uniti un soggetto in primo piano e uno sfondo. Nello specifico, il soggetto in primo piano viene posto su uno sfondo di colore uniforme, solitamente verde o blu, senza ombre o riflessi luminosi. Questo semplifica l'analisi e quindi il riconoscimento di soggetto principale e sfondo da parte del programma di elaborazione. Viene successivamente creata l'immagine composita andando a sovrapporre il soggetto in primo piano con lo sfondo ricavato dalla seconda fonte. [1]
L'oggetto è stato organizzato tramite l'utilizzo di aree colorate (fig. 3.6). In questo modo è possibile distinguere i vari componenti:

- **Area evidenziata in azzurro**: Partendo dall'immagine della griglia di allineamento (fig. 3.7), tramite l'oggetto `jit.movie` [3] viene generato un segnale video utilizzabile dall'oggetto `jit.chromakey`

- **Area evidenziata in verde**: Contiene tutti i parametri di controllo di `jit.chromakey`, utilizzabili per variare il prodotto della composizione delle due fonti video fornite.

Sono presenti inoltre tre oggetti `jit.pwindow` [3] utilizzati come anteprima, così da permettere la visualizzazione di errori in caso di necessità. È inoltre presente un oggetto `suckah` [3] che permette di generare il codice RGB (*Red Green Blue*) corrispondente alla gradazione di colore utilizzata come sfondo per effettuare il processo di *chromakey*.

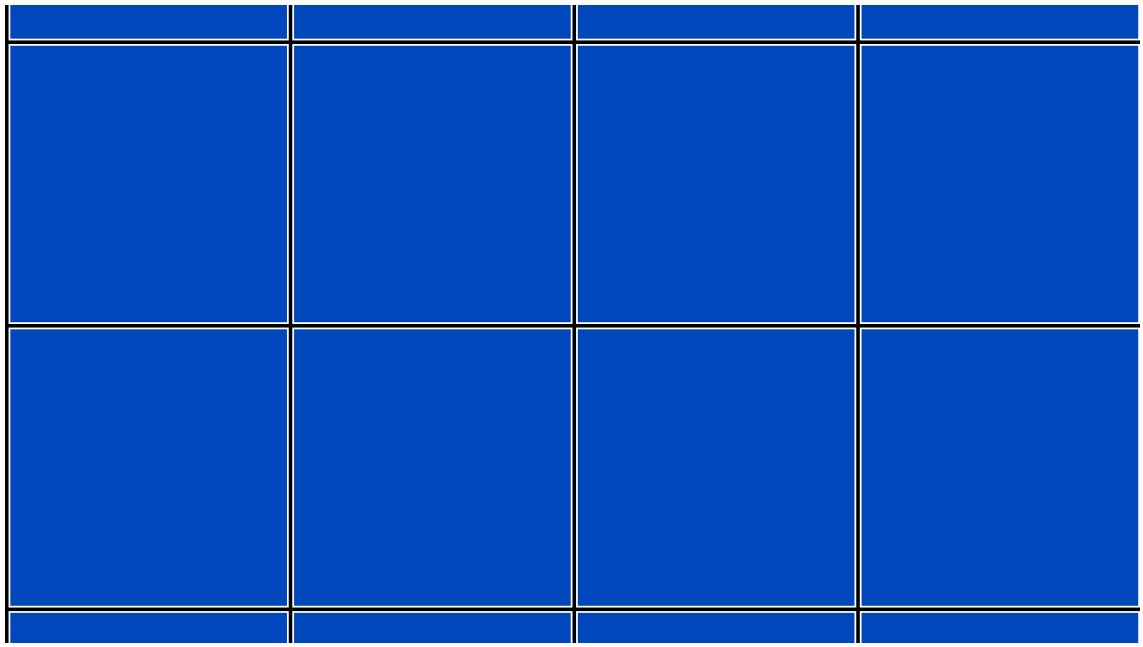

Figura 3.7: *Griglia Chroma-Key*

Capitolo 4

Audio

Il sistema di elaborazione audio, di cui viene rappresentata un panoramica in figura 4.1, è composto di una sezione di acquisizione audio, uno stadio di equalizzazione e la generazione del loop audio, poi riprodotto tramite l'utilizzo dell'oggetto `dac~`.

4.1 Acquisizione sample audio

Tramite l'oggetto *MSP* `adc~` [3] il segnale audio digitalizzato dal dispositivo hardware viene importato nell'ambiente di sviluppo per la successiva elaborazione.

4.2 Manipolazione Audio

Dopo la conversione digitale il segnale audio viene equalizzato tramite una serie di tre filtri. Tali filtri sono generati tramite oggetti `biquad` [3]. Nelle figure 4.2 e 4.3 è possibile vedere due differenti modalità di visualizzazione dei tre filtri. Nella modalità *presentazione* vediamo solamente i parametri modificabili dei tre filtri.
In questo caso i filtri vengono utilizzati per equalizzare la voce del performer e

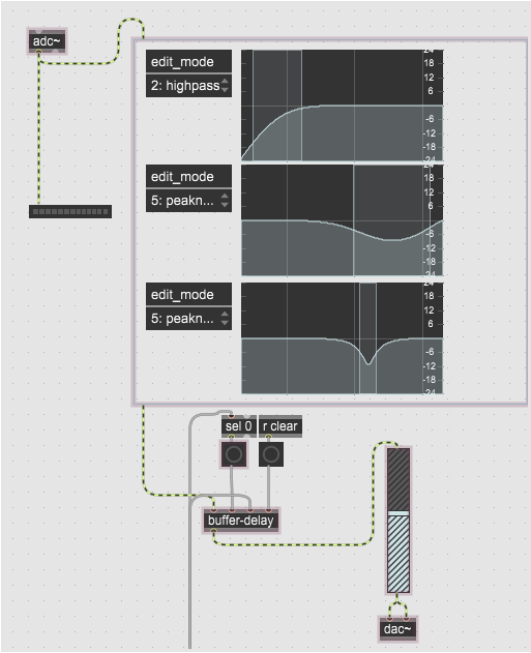

Figura 4.1: *Sistema elaborazione audio*

per attenuare leggermente le frequenze risonanti della stanza. I filtri utilizzati nel caso specifico sono:

- **un filtro passa-alto**: con frequenza di taglio attorno ai 100 Hz, utilizzato per eliminare le basse frequenze captate dal microfono e non utili alla comprensione della voce del performer o degli strumenti da lui utilizzati.
- **due filtri a campana**: utilizzati per eliminare le frequenze risonanti che compromettono il segnale e che possono generare dei feedback non controllati. La frequenza centrale e la profondità dei filtri vengono impostate in relazione all'ambiente in cui viene installato il sistema, alla tipologia di microfono utilizzato e alle necessità specifiche incontrate durante l'installazione.

Tuttavia, oltre alla possibilità di attenuare il contenuto spettrale del suono, è possibile andare ad accentuare alcuni intervalli di frequenze. Questo favorisce la generazione di feedback controllati e quindi la degradazione del segnale, elemento costitutivo della performance.

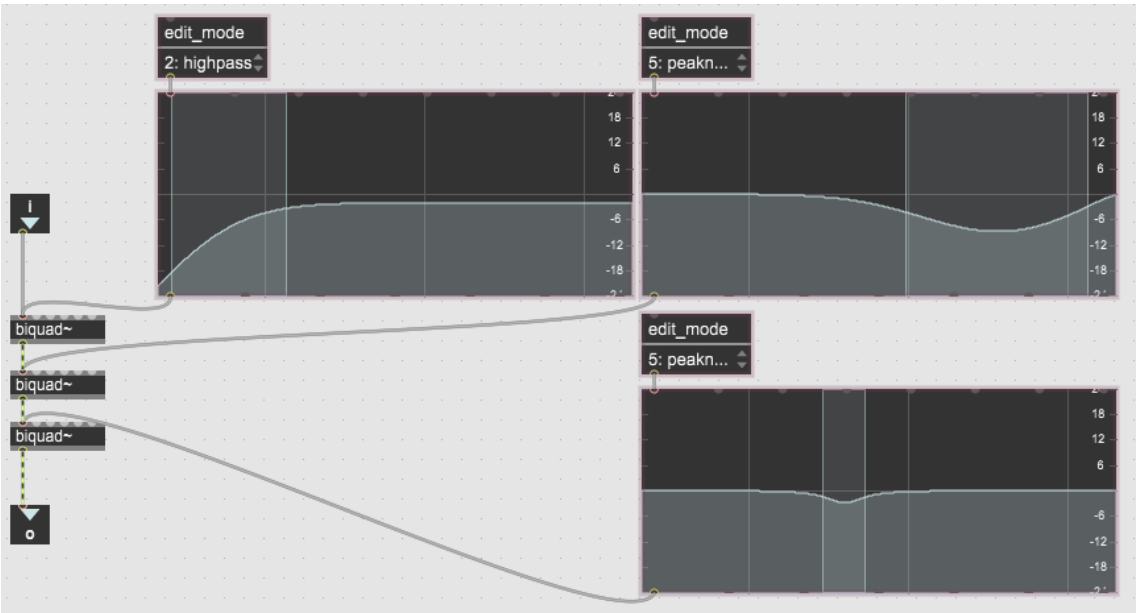

Figura 4.2: *Filter Section*

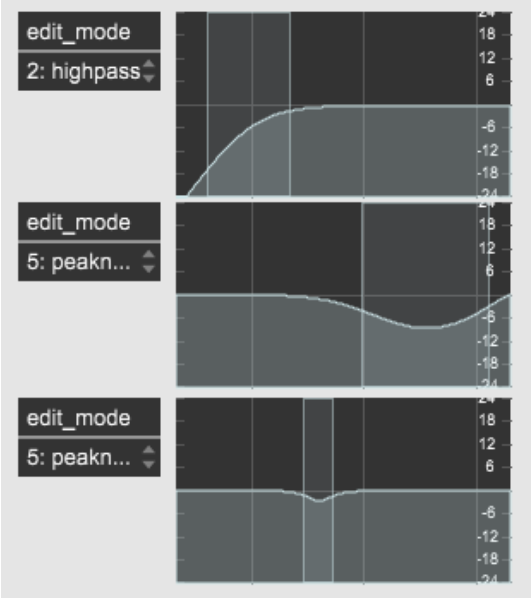

Figura 4.3: *Filter Section Presentation*

4.3 Creazione Loop Audio

Il flusso audio equalizzato, viene successivamente trasmesso alla sezione di delay, che consente la generazione di una sua copia ritardata nel tempo. È stato utilizzato l'oggetto *MSP* `buffer~` [3] per la memorizzazione e la riproduzione del segnale audio. Questo permette di salvare e riprodurre campioni di segnale tramite l'utilizzo di un array di dimensione variabile. Poichè all'interno del sistema è necessario che il buffer venga sincronizzato con il video acquisito durante ogni ripetizione, si è scelto di utilizzare una combinazione di tre buffer sincronizzati. Un buffer viene utilizzato per acquisire il segnale, un buffer per la riproduzione e un buffer viene ripulito e preparato per la successiva acquisizione. L'oggetto `buffer-delay`, di cui viene mostrata una panoramica nell'immagine 4.4, inserito nella `patch` tramite l'oggetto `patcher`, si occupa di tutta la gestione del segnale ritardato.
All'interno della `patch` in figura 4.4 sono presenti varie sezioni per la gestione dei buffer distinte tramite l'utilizzo di aree colorate.

- **Area evidenziata in azzurro:** Tramite la combinazione dei comandi viene selezionato il tipo di comportamento assegnato a ciascun buffer:

  - registrazione
  - riproduzione
  - cancellazione

  In questo modo è possibile mantenere il sistema sincronizzato. Tramite un counter viene selezionata la sequenza di comandi da utilizzare per la manipolazione del buffer. Il sistema viene controllato da un segnale di *bang*, generato dall'esterno, utilizzato per far avanzare il `counter`, che definisce l'inizio di una nuova ripetizione del segnale audio/video.

- **Area evidenziata in rosso:** Contenente gli oggetti *MSP* `buffer~`, che allocano lo spazio in memoria per il salvataggio dei campioni utilizzati. I buffer

necessitano di essere ripuliti dal segnale registrato e riprodotto prima di poter acquisire altri campioni utilizzabili. Per questo vengono periodicamente ripuliti tramite un messaggio di *clear*, innescato dal `counter` di controllo.

- **Area evidenziata in giallo:** Viene specificato il tempo del delay in secondi e poi convertito in campioni audio per determinare la dimensione di ciascun buffer. Nella sincronizzazione tra delay audio e video possono avvenire dei disallineamenti. Per questo motivo sono stati aggiunti due secondi alla dimensione del buffer che consente di mantenere la riproduzione audio attiva fino alla conclusione della ripetizione.

- **Area evidenziata in verde:** Contiene i comandi utilizzati per il controllo della registrazione del segnale all'interno dei tre buffer. Nel momento in cui viene inviato il *bang* il `counter` seleziona il `buffer` abilitato alla registrazione. Da quel momento fino alla fine del ciclo di delay il segnale audio in input viene scritto nel buffer specificato.

- **Area evidenziata in viola:** Contenente i comandi utilizzati per la riproduzione del buffer corretto nella ripetizione specificata. Ogni segnale riprodotto, subisce, ad inizio ripetizione un aumento lineare del volume della durata di 100 ms tramite l'oggetto *MSP* `line~` [3]. È, in questo modo, possibile effettuare una transizione progressiva, così da non avere suoni o rumori improvvisi generati dal processo di registrazione.

Nella `subpatch` sono presenti tre istanze dell'oggetto `waveform~` [3], con i quali e possibile visionare lo stato dei tre buffer.

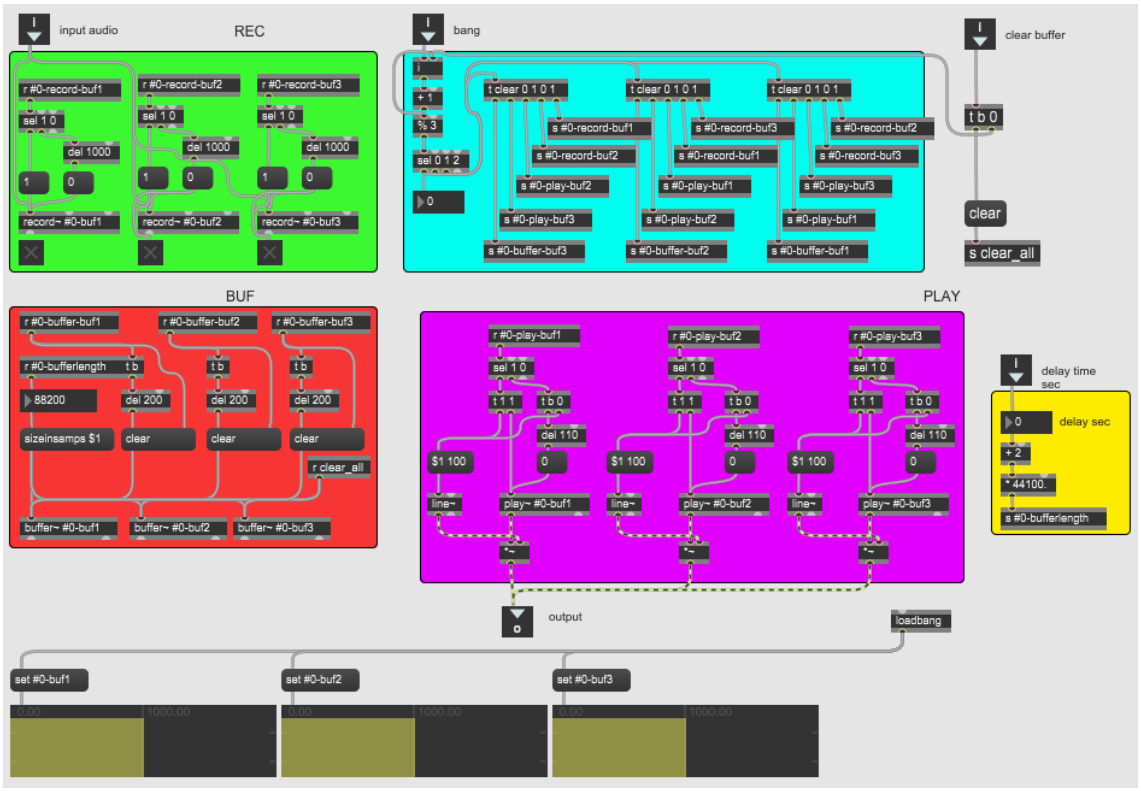

Figura 4.4: *Buffer-delay*

Capitolo 5

Controllo parametri tramite controller esterno

Vista la tipologia della performance, si è deciso, in accordo con l'artista, di dotare il programma di un sistema di controllo dei parametri. Nello specifico si è scelto di utilizzare il protocollo *Open Sound Control* (OSC) per il controllo di alcuni parametri video. Risulta quindi possibile l'interazione con varie tipologie di dispositivi come smartphone, tablet, controller fisici, senza l'utilizzo di interfacce cablate. Questo semplifica e rende più versatile il sistema di controllo, soprattutto durante le performance live.

5.1 Open Sound Control

Open Sound Control (OSC) viene sviluppato originariamente da Wright e Freed nel 1997, per semplificare la trasmissione di segnali di controllo fra piccoli *array* di sistemi di calcolo eterogenei senza l'utilizzo di sistemi di distribuzione complessi. [14] Nello specifico progetto si è scelto di implementare un'interfaccia tramite questo linguaggio data la sua semplicità di utilizzo e programmazione. *Open Sound Control* è un protocollo indipendente dal tipo di trasporto utilizzato. Questo fa si che i dati trasmessi possano viaggiare attraverso una varietà di network

differenti. [6] Nel progetto questa particolarità viene sfruttata per trasportare i messaggi di controllo tramite protocollo di rete su segnale wifi.

5.2 OSC Max

In *Max/MSP* i comandi *OSC* dall'esterno vengono ricevuti e tradotti in messaggi compatibili con il linguaggio di programmazione tramite l'oggetto `udpreceive` [3]. Successivamente l'oggetto `route` [3] confronta il primo elemento di ogni messaggio in input e, in caso di piena uguaglianza con il primo parametro dell'oggetto, genera un messaggio di output contenente la parte rimanente del messaggio di input. Nel caso specifico il primo elemento del messaggio inviato rappresenta il nome che identifica l'oggetto di controllo utilizzato, il secondo il valore di controllo a cui l'oggetto viene posto al momento dell'invio del messaggio. I valori ricevuti, solitamente in un intervallo fra 0 e 1, vengono scalati tramite l'oggetto `scale` [3], così da essere compatibili con i valori accettati dagli oggetti da controllare. Nel caso specifico si è scelto di controllare i parametri video di luminosità, contrasto e saturazione tramite dei controller a variazione continua come *fader o potenziometri*. Oltre alla possibilità di ricevere messaggi OSC si è implementata la possibilità di inviarli, così da poter assegnare e ripristinare i corretti valori di default ai controller utilizzati. Questo viene reso possibile tramite l'oggetto `udpsend` [3], che permette l'invio di messaggi *OSC* a un dispositivo definito specificando indirizzo ip e porta utilizzati dal dispositivo all'interno della rete in cui è inserito. Una panoramica degli oggetti utilizzati per la gestione dei messaggi OSC all'interno della *patch Max/MSP* è presente in figura 5.1.

5.3 Controller

Vista la compatibilità universale del protocollo OSC è possibile utilizzare varie tipologie di controller. Durante lo sviluppo si è implementato un controller tramite l'applicazione *TouchOSC* [8], che permette di generare interfacce facilmente

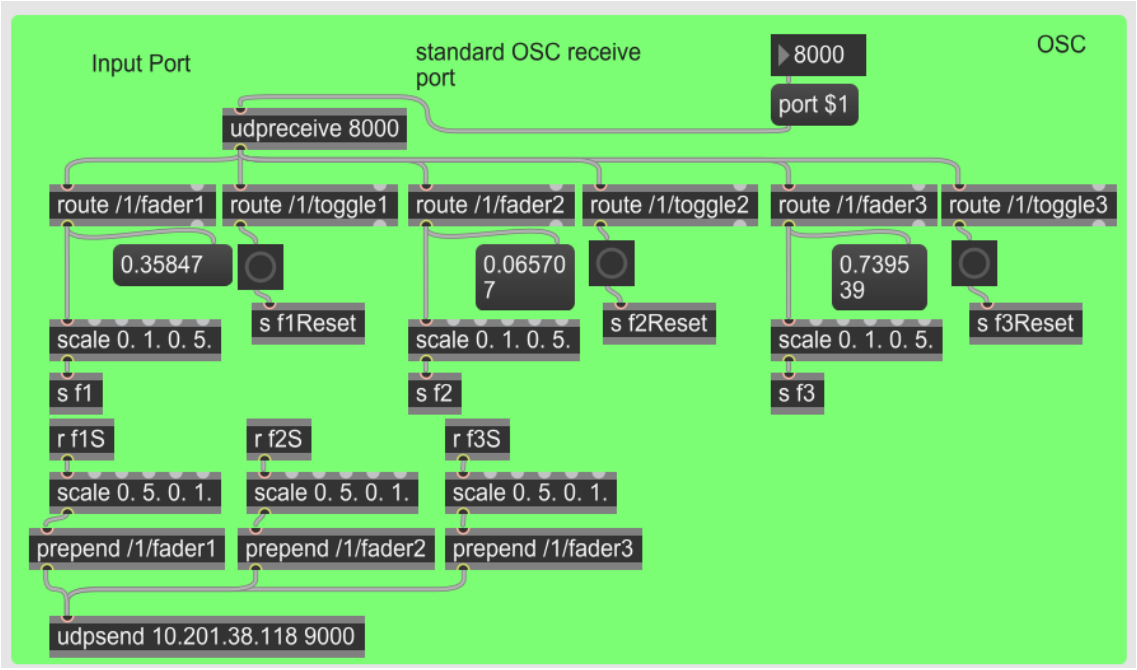

Figura 5.1: *OSC components*

utilizzabili in dispositivi come smartphone tablet e computer. (fig. 5.2) Nel corso degli sviluppi successivi sarà possibile lo sviluppo di controller *hardware* o *software* che possano fornire le migliori funzionalità in relazione alle necessità date dalle performance o dalle installazioni da effettuare.

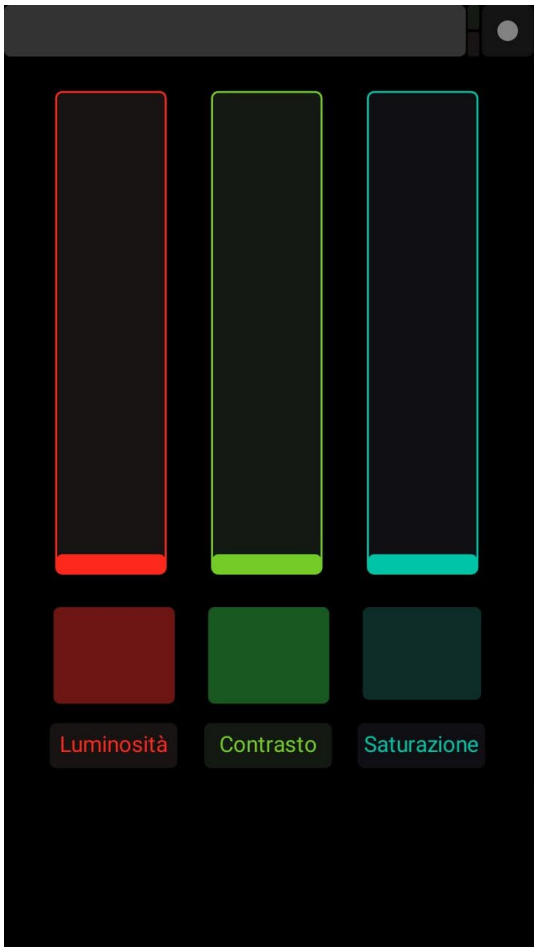

Figura 5.2: *TouchOSC controller*

Figura 5.3: *Test-Il tempo consuma- loop digitale*

Capitolo 6

Conclusioni

Per il recupero dell'opera *Il tempo consuma* si sono approfondite le possibilità di utilizzo del linguaggio *Max/MSP* per l'elaborazione audio/video in tempo reale. I risultati delle prove realizzate durante le varie sessioni di test del sistema, esemplificate dalla figura 5.3 [12], hanno permesso un risultato che ricalca l'estetica presente nell'opera risalente al 1978. Inoltre, dopo aver osservato le opportunità fornite dal sistema digitale in questione, oltre alla ricerca di un risultato analogo a quello originale, sarà possibile sperimentare differenti potenzialità compositive ed estetiche grazie all'utilizzo di immagini a colori. Questa nuova strada apre svariate opportunità quali la modifica di un gran numero di parametri e una maggiore profondità di manipolazione. Inoltre, grazie al trasferimento nel dominio digitale della componente di elaborazione, sarà possibile la nascita di differenti versioni del progetto, utilizzabili in contesti eterogenei come performance live, installazioni fisse o installazioni interattive.

Bibliografia

[1] M. A. BAGIWA, A. W. A. WAHAB, M. Y. I. IDRIS, S. KHAN, AND K.-K. R. CHOO, *Chroma key background detection for digital video using statistical correlation of blurring artifact*, Digital Investigation, 19 (2016), pp. 29–43.

[2] CYCLING74, *Cycling '74*. `https://cycling74.com`.

[3] ——, *Max reference*. `https://docs.cycling74.com/max8/vignettes/docrefpages`.

[4] S. L. E LISA PAROLO, *Michele Sambin performance tra musica, pittura e video*, CLEUP sc, Padova, first ed., 2014.

[5] A. C. E MAURIZIO GIRI, *Musica Elettronica e Sound Design*, ConTempoNet, Roma, second ed., 2009.

[6] A. FREED, *Open sound control: A new protocol for communicating with sound synthesizers*, in International Computer Music Conference (ICMC), 1997.

[7] V. HEDIGER, C. G SABA, B. LE MAITRE, AND J. NOORDEGRAAF, *Preserving and exhibiting media art: Challenges and Perspectives*, Amsterdam University Press, 2013.

[8] HEXLER, *Touchosc*. `https://hexler.net/touchosc`.

[9] L. PAROLO, *Il linguaggio artistico di michele sambin dal film alla video-performance musicale (1968-1982). ipotesi per la conservazione, il restauro e la ri-*

proposta attuale di looking for listening (1977), Master's thesis, Università di Padova, 2013.

[10] M. SAMBIN, *Intervista a michele sambin del 10 agosto 2021*. https://drive.google.com/drive/folders/1izeJHKMF0FD91nX9dE-MJAz7fVDuIyfX?usp=sharing.

[11] ——, *Michele sambin*. https://michelesambin.com.

[12] ——, *test-il tempo consuma- loop digitale*. https://michelesambin.com/projects/test-il-tempo-consuma-loop-digitale.

[13] G. WANG, *A history of programming and music.*, 2012.

[14] M. WRIGHT, A. FREED, A. LEE, T. MADDEN, AND A. MOMENI, *Managing complexity with explicit mapping of gestures to sound control with osc*, in ICMC, Citeseer, 2001.