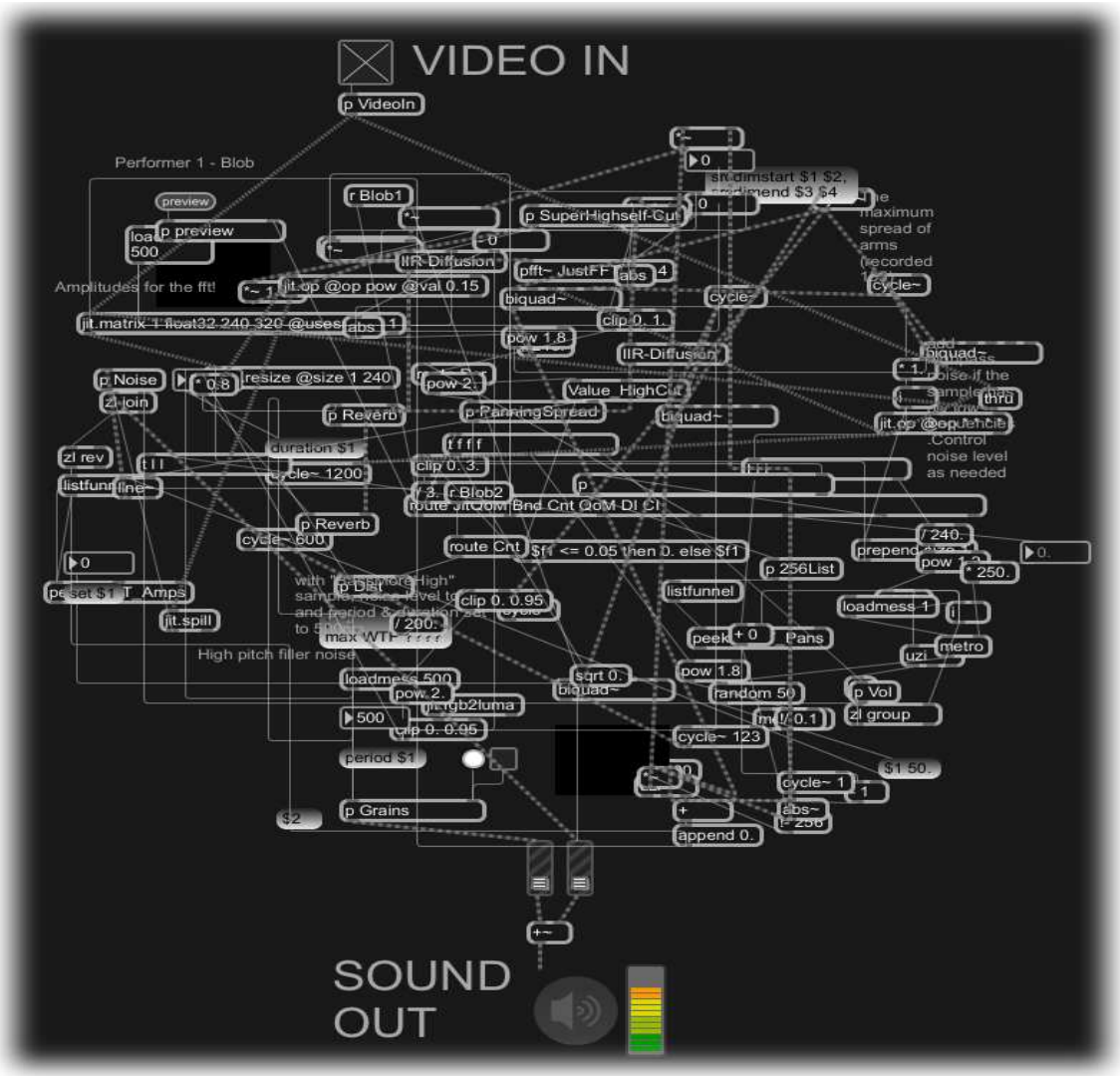
# ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Τμήμα Μουσικής Τεχνολογίας και Ακουστικής




**Πτυχιακή εργασία**

# "Ανάπτυξη διαδραστικού συστήματος ομαδικής μουσικής εκτέλεσης"

Νικόλας Κωνσταντακόπουλος

*Επιβλέπων:* *Στέλλα Πασχαλίδου*

*Καθηγήτρια Εφαρμογών*

Ρέθυμνο 2012

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΟΥΣΙΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΑΚΟΥΣΤΙΚΗΣ

**Πτυχιακή Εργασία**

**"Ανάπτυξη διαδραστικού συστήματος ομαδικής μουσικής εκτέλεσης"**

**του**

**Νικόλα Κωνσταντακόπουλου**

Επιβλέπων Καθηγήτρια: Στέλλα Πασχαλίδου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……./……/……..

…… …… ……

Καθηγητής Επίκουρος Καθηγητής, ηγητής

Ρέθυμνο Νοέμβριος 2012

## Πνευματικά δικαιώματα



Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μουσικής Τεχνολογίας και Ακουστικής του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

**Ευχαριστίες**

Η παρούσα εργασία αποτελεί μέρος της εκπλήρωσης των υποχρεώσεων του σπουδαστή προς το ΤΕΙ Κρήτης/Τμήμα Μουσικής Τεχνολογίας και Ακουστικής.

Επιθυμώ να εκφράσω τις θερμές μου ευχαριστίες στην κα Στέλλα Πασχαλίδου, για την ανάθεση και στη συνέχεια την επιστημονική καθοδήγηση, την υποστήριξη και τις πολύτιμες συμβουλές της, στην υλοποίηση της εργασίας, καθώς επίσης και για την εμπιστοσύνη με την οποία με περιέβαλε.

Ευχαριστίες οφείλονται επίσης στους Κοραλία Παλιεράκι, Κυριακή Παλιεράκι, Έλενα Χατζηπαναγιώτη, Χριστίνα Σαντακά, Μαρία Τερζοπούλου, Αντιγόνη Σταυροπούλου και Βαλεντίνο Σταματιάδη για την βοήθειά τους στην πραγματοποίηση του πρακτικού μέρους, και την υπομονή τους.

Ομοίως ευχαριστώ τους Γεώργιο Κωνσταντακόπουλο και Φρειδερίκη Πλατή.

# ΠΕΡΙΛΗΨΗ

Ο σκοπός της εργασίας αυτής είναι η δημιουργία ενός ομαδικού διαδραστικού μουσικού συστήματος, το οποίο θα μεταφράζει την κίνηση και την εκφραστικότητα των ολόσωμων κινήσεων των χρηστών του σε ήχο και μουσική. Εκτός από την απευθείας διάδραση του κάθε χρήστη με το ψηφιακό σύστημα η εφαρμογή ευελπιστεί να ενθαρρύνει και την μεταξύ των χρηστών αλληλεπίδραση. Για το σκοπό αυτό χρησιμοποιείται στην εφαρμογή ένα επιπλέον άτομο, ως άτομο αναφοράς. Ο κάθε χρήστης/εκτελεστής της ομάδας ελέγχει ένα ξεχωριστό μουσικό ηλεκτρονικό όργανο με της κινήσεις του σώματός του, ενώ ο τρόπος που σχετίζονται μεταξύ τους στο χώρο σε συνδυασμό με τις κινήσεις του "χρήστη αναφοράς" επιτρέπει τον έλεγχο στο συνολικό ηχητικό αποτέλεσμα.

Στη συγκεκριμένη εφαρμογή έχει δοθεί βαρύτητα στην ελαχιστοποίηση παρεμβολών από τη χρήση αισθητήρων απευθείας στο σώμα των χρηστών, οι οποίοι συχνά παρεμποδίζουν την κίνησή τους λόγω καλωδίων και άλλων εξαρτημάτων. Για το λόγο αυτό γίνεται χρήση βιντεοκάμερας τοποθετημένης απέναντι από τη σκηνή, η οποία και οριοθετεί την περιοχή λειτουργίας της εγκατάστασης. Στη συνέχεια, η μετάφραση των κινήσεων των χρηστών σε παραμέτρους εκφραστικότητας γίνεται χρησιμοποιώντας τεχνικές ψηφιακής ανάλυσης δισδιάστατης (κινούμενης) εικόνας βίντεο σε πραγματικό χρόνο. Η υλοποίηση της εργασίας αφορά στην ανάπτυξη ενός προγράμματος με την ονομασία “Bread”, που αποτελείται από τους αλγορίθμους υπολογισμού ενός πλήθους χαρακτηριστικών μεγεθών κίνησης (σε πραγματικό χρόνο) που αντιπροσωπεύουν στοιχεία εκφραστικότητας των χρηστών, τους αλγορίθμους σύνθεσης των πρωτότυπων παραγόμενων ήχων, καθώς και το αντίστοιχο γραφικό περιβάλλον χρήσης του προγράμματος.

**Λέξεις Κλειδιά:**

Ηχητικές/μουσικές διαδραστικές εφαρμογές, ηλεκτρονικά μουσικά όργανα, διάδραση, αλληλεπίδραση σε πραγματικό χρόνο, διεπαφή, ψηφιακή επεξεργασία εικόνας, αφαίρεση φόντου, ανίχνευση κίνησης / κηλίδας, αντιστοίχηση παραμέτρων, γεννήτρια ήχου

# ABSTRACT

The current dissertation aims to the creation of a collaborative interactive music system, which will translate motion and the expressional content of its users' full body movements to sound and music. Beside the direct interaction between each individual user and the computer, the system aims to encourage group interaction too, i.e. the interaction between the users themselves. This is facilitated by the use of an additional user/performer, who acts as a reference. Thus, each user of the group controls an electronic instrument with his individual motion, while an additional user, the "reference user", will have control over the total sound outcome.

In this application we have paid attention on minimizing the interference problems related to placing sensors directly onto the users' bodies, which often impede their movement due to cables and other accessories. For this reason we choose the use of a video camera placed across the scene, which thus frames the working space of the installation. Consequently, the translation of the users' movements to parameters of artistic expression is succeeded by using real-time two-dimensional digital image analysis methods from streaming video. The implementation in this project involves the development of a software called "Bread", which includes algorithms for the calculation of a number of movements parameters (in real-time) which represent the users' expressional elements, the algorithms for original audio synthesis, as well as the appropriate G.U.I. for handling the program.

**Keywords:**

Audio/musical interactive applications, electronic musical instruments, real-time interaction, interface, interface, digital image processing, background subtraction, blob tracking, parameter mapping, sound generator

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

# ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

# ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

# ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

| | |
|---|---|
| Blob | **B**inary **L**arge **Ob**ject |
| H.C.I. | Human Computer Interaction |
| M.I.D.I | Musical Instrument Digital Interface |
| O.S.C. | Open Sound Control |
| T.C.P | Transmission Control Protocol |
| U.D.P | User Datagram Protocol |
| “Bread” | Διαδραστικό σύστημα ομαδικής μουσικής εκτέλεσης |

# ΑΠΟΔΟΣΗ ΟΡΩΝ

| | |
|---|---|
| Data | Δεδομένα |
| Raw Data | Δεδομένου που δεν έχουν υποστεί επεξεργασία |
| Blob | Κηλίδα / Μεγάλο δυαδικό αντικείμενο |
| Background | Φόντο εικόνας / Παρασκήνιο |
| Foreground | Προσκήνιο εικόνας |
| User Interface | Διεπαφή / Διαπροσωπείο χρήστη |
| Mapping | Αντιστοίχηση Παραμέτρων |
| Pixel | Εικονοστοιχείο |
| Hardware | Φυσικό εξάρτημα υπολογιστή |
| Performer | Εκτελεστής |
| Threshold | Κατωφλίωση / τιμή κατωφλίου |
| Feedback | Ανατροφοδότηση πληροφορίας |
| Bread | Ψωμί… |

# Εισαγωγή

Στην εποχή μας η ψηφιακή τεχνολογία βρίσκεται στην καθημερινότητα του καθενός. Η εξέλιξη των ηλεκτρονικών υπολογιστών οδήγησε στην αύξηση της υπολογιστικής ισχύος που διαθέτουν και αντιστρόφως μείωση του μεγέθους τους, καταλήγοντας σε πολύ φορητά και πολύ δυνατά εργαλεία για κάθε σκοπό. Οι υπολογιστές έχουν αναμειχθεί στην τέχνη εδώ και πολλά χρόνια, είτε ως μέσο δημιουργίας είτε ακόμα και ως στοιχείο ερμηνείας. Ένας από τους τομείς της τέχνης είναι η διάδραση, τόσο μεταξύ ερμηνευτών, όσο και μεταξύ ερμηνευτή και υπολογιστή. Η ύπαρξη υπολογιστή, ως συστατικού στοιχείου, εντός ενός έργου τέχνης δίνει δυνατότητες στον καλλιτέχνη τις οποίες αλλιώς δεν θα μπορούσε να έχει. Ειδικά σε οπτικό-ακουστικά έργα τέχνης η δυνατότητα της διάδρασης μεταξύ ενός έργου και του παρατηρητή αυτού, είναι ένα δυνατό χαρτί στα χέρια του καλλιτέχνη. Μπορεί κανείς πλέον να δημιουργήσει διαδραστικά συστήματα τα οποία ανταποκρίνονται στις ενέργειες/κινήσεις ενός χρήστη/εκτελεστή παράγοντας σε (σχεδόν) πραγματικό χρόνο πρωτότυπους ήχους και εικόνες. Επειδή στην παρούσα εργασία ενδιαφερόμαστε για την παραγωγή πρωτότυπου συνθετικού μουσικού υλικού, από εδώ και πέρα θα αναφερόμαστε σε τέτοιου είδους μουσικά διαδραστικά συστήματα και με τον όρο ηλεκτρονικά μουσικά όργανα.

Ο χειρισμός ενός ηλεκτρονικού υπολογιστή απαιτεί ορισμένες “θυσίες” όσον αφορά τον τρόπο ερμηνείας ενός καλλιτέχνη. Ο χειρισμός ενός υπολογιστή συνήθως (για εργασίες γραφείου) επιτυγχάνεται μέσω του “ποντικιού” και του πληκτρολογίου. Όμως η χρήση αυτών των συσκευών απαιτεί συγκεκριμένες κινήσεις, η οποίες δεν αντιπροσωπεύουν τις αυθόρμητες καλλιτεχνικές κινήσεις ενός ατόμου. Η διάδραση μεταξύ υπολογιστή και καλλιτέχνη μπορεί να επιτρέψει στον καλλιτέχνη την απόδοση του έργου του με ελευθερία κινήσεων, μεγαλύτερη και πιο εκφραστική από εκείνη που του προσφέρει ο “συμβατικός” τρόπος χειρισμού των ηλεκτρονικών υπολογιστών.

Στο θεωρητικό μέρος της εργασίας που ακολουθεί θα γίνει αναφορά στα δομικά στοιχεία που απαρτίζουν τα διαδραστικά συστήματα και τις εφαρμογές τους και θα δοθούν αντίστοιχα παραδείγματα. Θα αφιερωθούν δύο κεφάλαια στην θεωρία. Το πρώτο αναφέρεται σε θεωρία γενικού υπόβαθρου και το δεύτερο σε εξειδικευμένη θεωρία απαραίτητη για την κατανόηση και διεξαγωγή του πειραματικού μέρους. Μετά την ολοκλήρωση της θεωρίας θα παρουσιαστεί το

πειραματικό μέρος της εργασίας από πλευρά που αφορά τον χρήστη αλλά και από την σκοπιά του προγραμματιστή εμβαθύνοντας στην ανάλυση των προγραμματιστικών διαδικασιών. Στο τελικά κεφάλαιο που ακολουθεί το πειραματικό μέρος, παρουσιάζονται συμπεράσματα, προβλήματα και μελλοντικές βελτιώσεις.

# 1. ΓΕΝΙΚΆ ΣΤΟΙΧΕΊΑ ΘΕΩΡΊΑΣ

Στο κεφάλαιο αυτό θα παρουσιαστούν γενικά θεωρητικά στοιχεία γύρω από τις βασικές έννοιες πάνω στις οποίες στηρίζεται η παρούσα εργασία.

Θα ξεκινήσουμε εξηγώντας δύο από τις βασικότερες έννοιες των διαδραστικών συστημάτων, τη διάδραση και τη διεπαφή.

## 1.1. Διάδραση μεταξύ ανθρώπου και υπολογιστή – HCI

Στο υποκεφάλαιο αυτό θα δοθεί μια θεωρητική βάση για τον ορισμό της διάδρασης και της διεπαφής. Η διάδραση εξετάζεται ως φαινόμενο, ενώ η διεπαφή ως ο απαραίτητος μεσάζοντας μεταξύ δύο αλληλεπιδρώντων συστημάτων.

### 1.1.1. Διάδραση

Με τον όρο διάδραση αναφερόμαστε στο φαινόμενο της συνεχούς αλληλεπίδρασης μεταξύ δύο ή περισσότερων παραγόντων. Ο όρος συναντιέται σε πολλά επιστημονικά πεδία, φυσική, βιολογία κ.α. αλλά στην παρούσα περίπτωση μας απασχολεί η διάδραση μεταξύ ανθρώπου και υπολογιστή (Human-Computer Interaction - HCI).

*"Αλληλεπίδραση ανθρώπου-υπολογιστή είναι μια επιστήμη που ασχολείται με το σχεδιασμό, την αξιολόγηση και την υλοποίηση διαδραστικών υπολογιστικών συστημάτων που προορίζονται για ανθρώπινη χρήση και με τη μελέτη των μεγάλων φαινομένων γύρω τους" (Hewett, και συν., 1996 ).*

*"Η διαδραστικότητα αναφέρεται γενικά στη μουσική διάδραση ανθρώπου-υπολογιστή, ή στη διάδραση ανθρώπου-ανθρώπου η οποία επιτυγχάνεται μέσω υπολογιστή. Συχνά σε μια διαδραστική παράσταση, σύνθεση ή αυτοσχεδιασμό συμπεριλαμβάνεται η δημιουργία προγραμματισμού ή λογισμικού το οποίο αντιδρά σε προκαθορισμένες πτυχές της ζωντανής παράστασης. Το λογισμικό με τη σειρά του θα καθορίζει άλλες πτυχές της μουσικής, είτε δημιουργώντας συνθετικό ήχο ή τροποποιώντας με κάποιο τρόπο κάποιο μέρος ή όλο τον "ζωντανό" ήχο….. Η διαδραστική μουσική*

*σύνθεση συχνά θολώνει τα όριο μεταξύ μουσικής σύνθεσης και μουσικού αυτοσχεδιασμού…." (Ears : Interactive Instruments ).*

## Τύποι Διάδρασης

Ο τομέας αυτός, της διάδρασης ανθρώπου – υπολογιστή ως πεδίο έρευνας ασχολείται με τη δημιουργία και βελτιστοποίηση του τρόπου της διάδρασης. Με την ανάπτυξη νέων τεχνολογιών και δυνατότερων υπολογιστών, εξελίσσονται και οι τρόποι με τους οποίους μπορεί να επιτευχθεί ευκολότερη και πιο άμεση επικοινωνία μεταξύ ανθρώπου και υπολογιστή.

Διακρίνουμε δύο κύριους τύπους διάδρασης μεταξύ Ανθρώπου και Υπολογιστή (Hershey, 2009 )

**Μονόπλευρη Διάδραση:** Οι τρόποι επικοινωνίας παλαιότερα ήταν περιορισμένοι στο ποντίκι, στο πληκτρολόγιο, στην οθόνη και στον ήχο. Η επικοινωνία στις περιπτώσεις αυτές όμως είναι μονόπλευρη, καθώς ο ήχος και η οθόνη (εικόνα) αφορούν ροή πληροφορίας αποκλειστικά από τον υπολογιστή προς τον χρήστη, ενώ το ποντίκι και το πληκτρολόγιο μόνο από τον χρήστη προς τον υπολογιστή.

**Πολυτροπική Διάδραση :** Οι πολυτροπικές διαδραστικές εφαρμογές αποτελούν το τρέχον πεδίο έρευνας της διάδρασης ανθρώπου – υπολογιστή. Η επικοινωνία επιτυγχάνεται πλέον μέσω ανάλυσης της κίνησης, άρα αναγνώρισης χειρονομιών, αναγνώρισης φωνής, αναγνώρισης προσώπων κ.α. Ο λόγος δημιουργίας μιας πολυτροπικής διεπαφής είναι η φυσικότερη και συνεπώς ευκολότερη επικοινωνία μεταξύ των δύο μέσων. Ο σκοπός είναι οι διαφορετικοί τρόποι να χρησιμοποιούνται και να συνδυάζονται, έτσι ώστε ο ένας να καλύπτει τα ελαττώματα του άλλου και κατά συνέπεια η συνολική επικοινωνία να διευκολύνεται και να είναι πιο ακριβής.

Μπορεί εύκολα ένα τέτοιο σύστημα χωρίς σωστό σχεδιασμό εντέλει να μην επιτυγχάνει τον σκοπό του, χρησιμοποιώντας περισσότερους τρόπους επικοινωνίας απ' ότι πραγματικά χρειάζεται.

Παραδείγματος χάριν, έστω ένα σύστημα που χρησιμοποιεί το ποντίκι και ταυτόχρονα εντοπίζει τη θέση του χεριού του χρήστη. Και οι δύο τρόποι εισαγωγής δεδομένων από τον χρήστη έχουν αρκετά όμοια δεδομένα, καθιστώντας πιθανόν τον ένα από τους δύο περιττό, άρα η ύπαρξη και των δύο τρόπων εισαγωγής στο σύστημα αυτό, πιθανόν να επιβαρύνει το σύστημα και δεν ωφελεί, αλλά περιπλέκει, τον τρόπο επικοινωνίας – διάδρασης.

Οι τρόποι αυτοί βασίζονται ως επί το πλείστον σε μεθόδους ανάλυσης, οι οποίοι είναι πολύ δεκτικοί σε λάθη. Π.χ. η αναγνώριση φωνής χρειάζεται αρκετά ιδανικές συνθήκες, δεδομένου ότι είναι πολύ ευαίσθητη στον θόρυβο του ήχου, στην προφορά του λόγου, στην ποιότητα του ίδιου του σήματος. Αντίστοιχα η ανάλυση εικόνας απαιτεί καλή ποιότητα εικόνας, αφού έστω και λίγος θόρυβος μπορεί να διαστρεβλώσει τα αποτελέσματα της ανάλυσης η και να καταστήσει την ανάλυση αδύνατη.

Η έρευνα στην πολυτροπική διάδραση και την διάδραση ανθρώπου - υπολογιστή ασχολείται με την βελτίωση των τρόπων διάδρασης μεμονωμένα, ώστε να επιτυγχάνεται με μεγαλύτερη ακρίβεια η επικοινωνία από τον χρήστη στον υπολογιστή και το αντίστροφο, αλλά και με τη βελτιστοποίηση του συνδυασμού των τρόπων, ώστε να επιτευχθεί όσο το δυνατόν ακριβής και άμεση διάδραση.

### 1.1.2. Διεπαφή

*"Η διεπαφή αναφέρεται είτε σε λογισμικό ή hardware, το οποίο χρησιμεύει ως ο ενδιάμεσος δύο διαφορετικών συστημάτων που επιτρέπει την ανταλλαγή πληροφοριών μεταξύ τους"* ( Ears : Interface ).

Στην συγκεκριμένη περίπτωση των διαδραστικών ΜΟΥΣΙΚΩΝ συστημάτων, όπου η ανταλλαγή πληροφοριών γίνεται μεταξύ ανθρώπου και υπολογιστή, ο ρόλος της διεπαφής είναι η δυνατότητα ελέγχου του εκάστοτε συστήματος από τον χρήστη, καθώς και η ενημέρωση του χρήστη για τα αντίστοιχα αποτελέσματα (δεδομένα εξόδου) του συστήματος. Σημαντικά στοιχεία για μια επιτυχημένη διεπαφή μεταξύ υπολογιστή και ανθρώπου, προσανατολισμένη στης διαδραστικές μουσικές εφαρμογές, είναι τα παρακάτω (Hunt & Hermann, 2004 ):

- Να υπάρχει ακουστική ανατροφοδότηση σε πραγματικό χρόνο
- Να επιδιώκεται σύστημα με αυξημένο χρόνο εκμάθησης, εφόσον έχει διαπιστωθεί πως ο αυξημένος χρόνος εκμάθησης αποφέρει περισσότερη "λεπτομέρεια" και πολυπλοκότητα στην εκτέλεση
- Η διεπαφή να αντιδρά με έναν "φυσικό" τρόπο

- Η αντιστοίχηση των εισόδων ελέγχου με την γεννήτρια εξόδου να επιτρέπει στον έμπειρο χρήστη να εισέλθει σε κατάσταση εκτέλεσης (performance mode), κατά την οποία υπάρχει η αίσθηση της ροής (flow, δόκιμος όρος στην ψυχολογία)
- Να υπάρχει συνοχή στην παροχή πληροφοριών και ο σωστός διαμοιρασμός τους σε διάφορους τρόπους διάδρασης (modalities)

## 1.2. Μουσικά διαδραστικά συστήματα

Στο παρόν κεφάλαιο θα γίνει αναφορά στα μουσικά διαδραστικά συστήματα. Αρχικά θα γίνει μια σύγκριση μεταξύ συμβατικών (φυσικών) και ηλεκτρονικών μουσικών οργάνων. Έπειτα θα παρουσιαστούν εν συντομία μερικές επιτυχημένες μουσικές διαδραστικές εφαρμογές και συστήματα, και τέλος θα αναφερθούν τα βασικότερα μέρη ενός τέτοιου συστήματος μαζί με ορισμένα παραδείγματα.

### 1.2.1. Φυσικό έναντι ηλεκτρονικού μουσικού οργάνου – η προβληματική

Αξίζει να σημειωθεί η εξής διαφορά μεταξύ των φυσικών μουσικών οργάνων (βιολί, κιθάρα) και των ηλεκτρονικών οργάνων:

Σε ένα φυσικό μουσικό όργανο η διεπαφή είναι το ίδιο το μέσον που παράγει και το ηχητικό αποτέλεσμα μέσα από τον τρόπο δόνησής του. Ο σχεδιασμός της διεπαφής εμπίπτει αναγκαστικά σε φυσικούς περιορισμούς οι οποίοι ταυτόχρονα καθορίζουν αλλά και περιορίζουν το επιδιωκόμενο ηχοχρωματικό αποτέλεσμα. Το σχήμα, το υλικό κατασκευής και ο τρόπος διέγερσης του σώματος ενός οργάνου όπως η κιθάρα ή το βιολί είναι απαραίτητα για να παράξουν τα ηχοχρώματα και το χαρακτήρα του ήχου που γνωρίζουμε. Κατά συνέπεια, και ο τρόπος αλληλεπίδρασης του εκτελεστή με το όργανο καθορίζεται ανάλογα.

Σε αντίθεση με τα φυσικά όργανα, το διάμεσο (δηλαδή το αντικείμενο που χειρίζεται ο εκτελεστής) δεν ταυτίζεται με τη μηχανή παραγωγής του ήχου. Αυτά αποτελούν δύο ξεχωριστά τμήματα ενός ηλεκτρονικού μουσικού οργάνου με πλήρη ανεξαρτησία μεταξύ τους και το γεγονός αυτό επιτρέπει την δημιουργία διεπαφών χωρίς περιορισμό από την φύση του παραγόμενου ήχου. Αυτή η διαπίστωση αποτελεί ταυτόχρονα το πλεονέκτημα, αλλά και το μειονέκτημα των ηλεκτρονικών μουσικών οργάνων. Εν συντομία, από τη μία η ιδιότητα αυτή επιτρέπει τον σχεδιασμό διάμεσων χωρίς ιδιαίτερους περιοριστικούς όρους (άρα δίνει μεγάλες ελευθερίες στον σχεδιαστή), από την άλλη όμως έχει διαπιστωθεί μια μεγάλη δυσκολία στην επίτευξη μιας αντιστοίχησης η οποία θα δίνει την αίσθηση στον εκτελεστή ότι η αλληλεπίδραση γίνεται με κάποιον ‘φυσικό’ τρόπο. Σύμφωνα με τους Castagné & Cadoz (2005 σελ. 2), «το πιο σημαντικό χαρακτηριστικό για ένα [συνθετικό] μουσικό ήχο δεν είναι να αντιστοιχεί σε κάποιον ήχο που προέρχεται από πραγματικά αίτια (Castagné & Cadoz, 2005 ), αλλά να παρουσιάζει ένα σύνολο από ανεπαίσθητες δυναμικές

διακυμάνσεις στις διάφορες αντιληπτικές παραμέτρους, οι οποίες δημιουργούν στον ακροατή την αίσθηση ότι κάποια φυσική διεργασία συμμετείχε στην παραγωγή του (ήχου)». Ακολουθούν ορισμένα παραδείγματα διαδραστικών εφαρμογών.

## 1.3. Ατομικές/ομαδικές διαδραστικές εφαρμογές

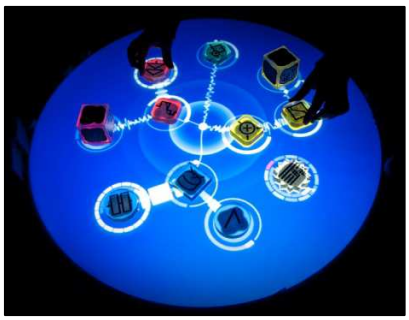
Οι διαδραστικές εφαρμογές σχεδιάζονται είτε για μεμονωμένους χρήστες είτε για ομάδες. Οι ομαδικές διαδραστικές εφαρμογές μπορούν να σχεδιαστούν με τέτοιο τρόπο ώστε το σύστημα να ανταποκρίνεται είτε μόνο στην απευθείας διάδραση μεταξύ χρήστη και συστήματος είτε και στην απαιτούμενη συνεργασία/διάδραση μεταξύ των χρηστών.

Σε μία εφαρμογή, όπου ο χώρος για δημιουργία είναι μεγάλος, δύο άτομα μπορούν να "παίξουν" ανεξάρτητα το ένα από το άλλο. Ο σκοπός όμως συνήθως είναι η συνεργασία μεταξύ των χρηστών για την επίτευξη ενός κοινού, ομαδικού καλλιτεχνικού αποτελέσματος. Όπως και σε ένα κλασικό μουσικό σχήμα ή μια ορχήστρα, έτσι και ηλεκτρονικές διαδραστικές εφαρμογές θεωρούμε ότι κάθε εκτελεστής μπορεί να παίξει μόνος του, αλλά το αποτέλεσμα είναι σαφώς καλύτερο όταν υπάρχει η συνεργασία μεταξύ των μουσικών.

Επειδή στο παρόν εγχείρημα σκοπός είναι η παρότρυνση της μεταξύ των εκτελεστών συνεργασίας, θα δούμε εδώ παραδείγματα ορισμένων επιτυχημένων ομαδικών διαδραστικών εφαρμογών.

**ReacTable**

Το ReacTable είναι ένα πρότυπο ηλεκτρονικό μουσικό όργανο. Η σύλληψη και ανάπτυξη της ιδέας ξεκίνησε το 2003 στο πανεπιστήμιο Pompeu Fabra από τους Marcos Alonso, Sergi Jordà, Gunter Geiger και Martin Kaltenbrunner. Το reacTable είναι ένα "τραπέζι" πάνω στο οποίο ο χρήστης ή οι χρήστες, μεταχειρίζονται κάποια αντικείμενα (Εικόνα 1).



ΕΙΚΌΝΑ 1 REACTABLE

ΕΙΚΟΝΑ 2 REACTABLE - ΠΟΛΛΟΙ ΧΡΗΣΤΕΣ

Τα αντικείμενα αυτά αποτελούν τα εργαλεία χειρισμού και καθορισμού των παραμέτρων του ήχου. Λόγω του τρόπου διάδρασης του ηλεκτρονικού αυτού οργάνου μπορεί να χρησιμοποιηθεί ατομικά ή ομαδικά. Ο περιορισμός στον αριθμό των χρηστών φαίνεται να εξαρτάται μόνο από το πόσα άτομα μπορούν να χωρέσουν γύρω από την επιφάνεια του ReacTable ταυτόχρονα (Εικόνα 2). Η άνεση και ευκολία της ταυτόχρονης χρήσης του ReacTable από πληθώρα χρηστών είναι ιδανική για να προωθεί τη δημιουργική συνεργασία μεταξύ τους (Reactable , 2009 ).

**Shiftables**

Στο MIT Media lab, ο David Merrill, με την συνεργασία του Jeevan Kalanithi, δημιούργησαν τα Shiftables. Τα Shiftables είναι μικροί υπολογιστές, περίπου 4x4 εκατοστά, εφοδιασμένοι με μια μικρή LCD οθόνη, ένα αισθητήρα επιτάχυνσης και έναν ασύρματο πομπό και δέκτη.

ΕΙΚΌΝΑ 3 SIFTABLES

Τα shiftables δημιουργήθηκαν με αυτές τις προδιαγραφές ώστε να μπορούν να χρησιμοποιηθούν όχι όπως ένας υπολογιστής, αλλά ως απτά καθημερινά αντικείμενα.

ΕΙΚΌΝΑ 5 ΜΙΞΗ ΧΡΩΜΑΤΩΝ

ΕΙΚΌΝΑ 4 ΜΙΞΗ ΧΡΩΜΑΤΩΝ - SIFTABLES

Στις παραπάνω εικόνες, βλέπουμε πως συνδυάζουμε και αναμιγνύουμε χρώματα ώστε να πετύχουμε την επιθυμητή απόχρωση με την χρήση δύο φυσικών αντικειμένων (2 ποτήρια, Εικόνα 4) και, πως με την ίδια λογική αναμιγνύουμε τα χρώματα με την χρήση των shiftables (Εικόνα 5).

Η μεταχείριση των αντικειμένων και στις δύο περιπτώσεις έχει την ίδια λογική και αίσθηση, στην περίπτωση όμως των shiftables, αν ρίξουμε παραπάνω μπογιά από ένα χρώμα, μπορούμε να την αφαιρέσουμε από το μείγμα των χρωμάτων χωρίς κανένα πρόβλημα.

Τα shiftables είναι διαδραστικά όχι μόνο με τον χρήστη, αλλά και μεταξύ τους. Στις παραπάνω εικόνες φαίνεται η εξάρτηση του κάθε shiftable με τα δίπλα του, αλλά ο χρήστης επιλέγει ποιά shiftable θα χρησιμοποιήσει μαζί. Στην Εικόνα 8 βλέπουμε πως με δύο shiftables με αριθμητικές


ΕΙΚΌΝΑ 7 ΠΡΟΣΘΕΣΗ ΜΕ SIFTABLES


ΕΙΚΌΝΑ 6 SCRABBLE ΜΕ SIFTABLES

τιμές, και δύο με αριθμητικές πράξεις, υπολογίζεται το 3+5= 8 , τα οποία τρία πλέον νούμερα μπορούν να χρησιμοποιηθούν σε διαφορετική σειρά για νέες πράξεις. Αντίστοιχα στην εικόνα 9, η λέξη “DAD” κυκλώνεται μόλις ο χρήστης την σχηματίσει σε ένα πρωτόγονο “scramble” (Merrill, Kalanithi, & Maes).

**Beatbugs**

Τα Beatbugs, είναι ένα δίκτυο από οκτώ κρουστά όργανα χειρός, τα οποία είναι σχεδιασμένα να ενθαρρύνουν τον κόσμο να παίζει μουσική ομαδικά. Οι χρήστες μπορούν να εισάγουν απλά ρυθμικά μοτίβα, να τα παραποιήσουν, να τα εξελίξουν και να συνεργαστούν μεταξύ τους με σκοπό να δημιουργήσουν και να εκτελέσουν μουσική (Weinberg, 2003).

ΕΙΚΌΝΑ 8 BEATBUG

Τα Beatbugs, αποτελούνται από οκτώ "χειριστήρια" στα οποία ενσωματώνονται διάφοροι αισθητήρες. Το κάθε χειριστήριο επικοινωνεί ασύρματα με ένα κεντρικό υπολογιστή ο οποίος δέχεται τα μηνύματα των beatbugs και παράλληλα εκτελεί τον αλγόριθμο της γεννήτριας ήχου. Το ηχητικό αποτέλεσμα προκύπτει από τον έλεγχο και των οκτώ χρηστών.

Τα beatbugs μπορούν να λειτουργήσουν με τρεις τρόπους :

**Free Play:** Σύμφωνα με τον τρόπο αυτόν οι χρήστες παίζουν ανεξάρτητα ο καθένας από τους υπόλοιπους και ο έλεγχος επεξεργασίας των μουσικών μοτίβων είναι απενεργοποιημένος. Σε αυτή την λειτουργία ο χρήστης εξοικειώνεται με το beatbug.

**Drum Circle :** Στη λειτουργία αυτή κάποιος από τους οκτώ χρήστες παίρνει τον ρόλο του αρχηγού, δίνοντας το αρχικό τέμπο και ρυθμικό μοτίβο. Στη συνέχεια το μοτίβο αυτό μεταβάλλεται από τον δημιουργό του και έπειτα με τη σειρά τους όλοι οι χρήστες προσθέτουν τα δικά τους στοιχεία στην ομαδική σύνθεση. Υπό την "προσταγή" του αρχηγού, ο οποίος κατέχει ένα ρόλο μαέστρου, όλοι οι χρήστες μαζί επεξεργάζονται με έναν αυτοσχεδιαστικό τρόπο τα ρυθμικά τους μοτίβα. Τέλος, με ένα συγχρονισμένο σήμα το μουσικό κομμάτι σταματάει.

**Snake :** Στη λειτουργία αυτή ο αρχηγός δίνει ένα ρυθμικό μοτίβο το οποίο στέλνεται κατευθείαν στον επόμενο χρήστη, χωρίς την δυνατότητα να τροποποιηθεί από τον δημιουργό του. Κάθε χρήστης που λαμβάνει ένα μοτίβο από τον προηγούμενο έχει την επιλογή να τροποποιήσει το εισερχόμενο ρυθμικό μοτίβο και να το στείλει στον επόμενο ή να δημιουργήσει ένα νέο δικό του. Όταν κλείσει ο κύκλος αυτός, οι χρήστες προκαλούνται σε μία αυτοσχεδιαστική μουσική διαμάχη. Το κομμάτι κλείνει με κάποιες συγχρονισμένες κινήσεις από τους οκτώ χρήστες.

Όπως φαίνεται τα beatbugs, στοχεύουν στην ομαδική μουσική σύνθεση και την συνεργασία μεταξύ των οκτώ εκάστοτε χρηστών.

Μια λειτουργία που παρατηρούμε στα beatbugs, είναι πως η συμπεριφορά του κάθε ενός από τους οκτώ χρήστες είναι εύπλαστη όταν ο πρώτος στην σειρά το θελήσει. Η ομάδα των οκτώ ατόμων έχει πάντα κάποιον σε αρχηγικό ρόλο και είναι ευάλωτη στη δική του θέληση. Αντιθέτως στο ReacTable δεν υπάρχει κάποια ηγετική φιγούρα για τον έλεγχο του συνολικού ήχου.

### 1.3.1. Χρήστης αναφοράς

Στα Beatbugs παρατηρούμε πως, εκτός της λειτουργίας free play κατά την οποίο δεν ενθαρρύνεται η ομαδικότητα και η συνεργασία, στις υπόλοιπες λειτουργίες απαιτείται ο ορισμός ενός αρχηγικού προσώπου. Το άτομο αυτό, έχει έλεγχο επάνω σε όλα τα υπόλοιπα μέλη της ομάδας, ενώ ο ίδιος έχει απόλυτη ελευθερία στις δικές του επιλογές. Ο ρόλος τους διαφέρει ξεκάθαρα.

Παρατηρούμε πως σε ορισμένες ομαδικές διαδραστικές εφαρμογές ένα από τα άτομα τις ομάδας δέχεται διαφορετικό ρόλο. Όπως στα Beatbugs, έτσι και στο έργο “Precarious Balance” της Susan Bickford οι κινήσεις μίας ομάδας επηρεάζουν τον ήχο από την αφήγηση ενός ατόμου. Φαίνεται καθαρά ο διαφορετικός ρόλος του αφηγητή, από τα υπόλοιπα μέλη. Μια αντίστοιχη διαφοροποίηση ενός ατόμου από το σύνολο επιχειρήθηκε και στην παρούσα πτυχιακή εργασία.

Στην εφαρμογή Bread ένα από τα άτομα επί σκηνής, λαμβάνει τον ρόλο του χρήστη αναφοράς. Οι κινήσεις του επηρεάζουν το σύνολο του παραγόμενου ήχου ανεξάρτητα από τις κινήσεις των υπολοίπων μελών. Επίσης οι παράμετροι που επηρεάζει έχουν μεγάλη βαρύτητα στον χαρακτήρα του ήχου, όπως η συνολική ένταση. Παρότι ο ίδιος κινείται χωρίς να επηρεάζεται από τους υπόλοιπους, τα γύρω μέλη επηρεάζονται από την απόστασή τους σε προς αυτόν. Περισσότερη ανάλυση για τον χρήστη αναφορά και την προσέγγιση της ιδέας ενός τέτοιου χρήστη θα γίνει στο κεφάλαιο του πειραματικού μέρος , στο υποκεφάλαιο “3.2.5 MAESTROS”

Στη συνέχεια θα παρουσιαστεί το μοντέλο δομής των διαδραστικών εφαρμογών, καθώς και της προσέγγισης η οποία χρησιμοποιήθηκε στο πειραματικό μέρος της εργασίας. Πρέπει να ληφθεί υπόψη ο προσανατολισμός της παρούσας εργασίας στα μουσικά διαδραστικά συστήματα και εφαρμογές.

### 1.3.2. Δομή μουσικών διαδραστικών εφαρμογών / συστημάτων

Στο κεφάλαιο αυτό θα παρουσιαστούν δύο μοντέλα όσον αφορά στη δομή μιας μουσικής διαδραστικής εφαρμογής ή συστήματος. Τα μοντέλα αυτά προσδιορίστηκαν από τους Robert Rowe(1993)και Todd Wilker(1998). Η οπτική του Wilker είναι μία συνέχεια, μια προσπάθεια βελτίωσης, της οπτικής του Rowe.

#### 1.3.2.1. Κατά Rowe

Σύμφωνα με τον Robert Rowe, ένα διαδραστικό σύστημα χωρίζεται σε τρία διαδικαστικά στάδια (Rowe, 1993 ).

- Sensing - Ανίχνευση
- Processing - Επεξεργασία
- Response – Ανταπόκριση

**Sensing– Ανίχνευση**

Στην διαδικασία ανίχνευσης το σύστημα συλλέγει δεδομένα από τον χρήστη. Ο τρόπος εισαγωγής στοιχείων του χρήστη στο σύστημα μπορεί να συμπεριλαμβάνει συσκευές ελέγχου midi ή προσαρμοσμένους αισθητήρες ανά περίπτωση , για τα οποία θα μιλήσουμε παρακάτω.

**Processing – Επεξεργασία**

Το στάδιο της επεξεργασίας περιέχει την ανάγνωση και ερμηνεία των πληροφοριών από τις εισόδους του συστήματος . Περιέχει αλγόριθμους οι οποίοι οδηγούν το υποσύστημα εξόδου θέτοντας τις παραμέτρους του.

**Response – Ανταπόκριση**

Στο τελικό στάδιο, δημιουργείται το ηχητικό αποτέλεσμα και εξάγεται. Στην διαδικασία δημιουργίας του αποτελέσματος συμπεριλαμβάνονται αλγόριθμοι δημιουργίας ήχου σε ζωντανό

χρόνο με χρήση λογισμικού ή έλεγχος εξωτερικών (hardware) συσκευών δημιουργίας ήχου, ακόμα και έλεγχος ρομποτικών εκτελεστών.

Στην οπτική του Rowe παρατηρείται μία ασάφεια στα όρια μεταξύ των σταδίων της ανίχνευσης και της επεξεργασίας, εφόσον απαιτείται ορισμένη επεξεργασία για την ανάγνωση και κυρίως την ερμηνεία των τιμών εισόδου.

#### 1.3.2.2. Κατά Wilker

Ο Todd Wilker απαντάει στα θολά αυτά όρια χωρίζοντας περαιτέρω την διαδικασία της επεξεργασίας (Wilker, 1998 ).

- Human input - Είσοδος από τον χρήστη
- Computer listening, performance analysis–ανάλυση δεδομένων εκτέλεσης
- Interpretation - Ερμηνεία δεδομένων
- Computer composition - σύνθεση "παρτιτούρας"
- Sound generation and output, performance–Δημιουργία ήχου και έξοδος, εκτέλεση

Στο μοντέλο αυτό ο Wilker χωρίζει την διαδικασία της "Επεξεργασίας", κατά Rowe, σε τρείς ξεχωριστές διαδικασίες.

Η διαδικασία ανάλυσης δέχεται τα δεδομένα από τις συσκευές εισόδου και τα αναλύει σε ολοκληρωμένες χειρονομίες. Οι χειρονομίες στην συνέχεια ερμηνεύονται σε μουσικά χαρακτηριστικά και αποστέλλονται στην διαδικασία σύνθεσης κατά την οποία τα προηγούμενα στοιχεία αντιστοιχίζονται στα στοιχεία την μουσικής εκτέλεσης του υπολογιστή.

Το μοντέλο του Wilkerδιαχωρίζει την είσοδο ή σύλληψη τιμών μέσω συσκευών (hardware), από την ανάλυσή τους. Όμως, αφήνει αφηρημένες τις διαφορές μεταξύ των διαδικασιών ανάλυσης των δεδομένων (Computer Listening) και ερμηνείας (Interpretation) (Drummond, 2009 ).

#### 1.3.2.3. Ακολουθούμενη πρακτική

Στο πειραματικό μέρος αυτής της εργασίας η δομή που χρησιμοποιήθηκε προσεγγίζει περισσότερη το μοντέλο του Rowe. Ο διαχωρισμός των διαδικασιών καταλήγει σε τρείς ξεχωριστές διαδικασίες.

- Είσοδος και ανάλυση δεδομένων του χρήστη
- Ερμηνεία δεδομένων/ αντιστοίχηση - mapping
- Γεννήτρια / Δημιουργία ήχου - Ανταπόκριση

Στην πρώτη διαδικασία περιλαμβάνονται η είσοδος των φυσικών δεδομένων (πρώτης τάξης), έτσι όπως αυτά ανιχνεύονται από τις συσκευές εισόδου, καθώς και η ανάλυσή τους σε δεύτερης τάξης δεδομένα (παράμετροι χαρακτηριστικές της κίνησης). Παρότι, όπως θα δούμε στο κεφάλαιο του πειραματικού μέρους, η διαδικασία αυτή αποτελεί ένα μεγάλο μέρος του συστήματος, ο ρόλος της δεν εισάγει καμία μουσική ερμηνεία.

Στη διαδικασία της ερμηνείας περιλαμβάνονται η αναγνώριση των χαρακτηριστικών παραμέτρων της κίνησης που σχετίζονται έντονα με τη μουσική πρόθεση του εκτελεστή, καθώς και η αντιστοίχηση αυτών με τις παραμέτρους της γεννήτριας ήχου. Ο όρος αντιστοίχηση θα αναλυθεί παρακάτω στο κεφάλαιο “1.4.5”.

Στην τελευταία διαδικασία περιέχεται ο αλγόριθμος σύνθεσης του ήχου. Το ηχητικό αποτέλεσμα είναι η ανταπόκριση του συστήματος στα εξωτερικά ερεθίσματα, που είναι στην προκειμένη περίπτωση οι ενέργειες/κινήσεις του χρήστη/μουσικού εκτελεστή.

Παρακάτω θα γίνει μία συζήτηση ως προς κάποια σημαντικά στοιχεία που αφορούν τα παραπάνω στάδια. Συγκεκριμένα θα δοθούν παραδείγματα συσκευών εισόδου (από τον χρήστη), θα γίνει μια εκτενέστερη αναφορά σε τρόπους αντιστοίχησης παραμέτρων (mapping) και θα δοθούν κάποιες γενικές πληροφορίες γύρω από την παραγωγή συνθετικών ήχων.

### 1.3.3. Συσκευές εισόδου

Ως συσκευές εισόδου θεωρούμαι τις συσκευές οι οποίες μπορούν να μεταφέρουν στοιχεία κινήσεων του χρήστη στο περιβάλλον του διαδραστικού συστήματος. Τέτοιες συσκευές είναι οι αισθητήρες, συσκευές ελέγχου τύπου MIDI ή OSC, κάμερες και άλλα.

Στο παρόν υποκεφάλαιο θα παρουσιαστούν μερικά παραδείγματα αισθητήρων, και των συσκευών που χρειάζονται για την επιτυχημένη σύνδεσή τους με τον υπολογιστή, καθώς και ορισμένες συσκευές MIDI ή OSC, ενώ η βιντεοκάμερες θα παρουσιαστούν στο κεφάλαιο της εξειδικευμένης θεωρίας εφόσον χρησιμοποιούνται στο πειραματικό μέρος.

#### 1.3.3.1. Αισθητήρες

Τα στοιχεία εκείνα που επιτρέπουν την εισαγωγή δεδομένων στο σύστημα ονομάζονται αισθητήρες. Υπάρχουν διάφορα είδη αισθητήρων, που διαφέρουν στον τρόπο ανίχνευσης της κίνησης, ενώ με τον συνδυασμό τους μπορούμε να αναπτύξουμε πρωτότυπους ολοκληρωμένους ελεγκτές. Οι αισθητήρες μπορούν να δώσουν δύο ειδών σήματα :

- Συνεχόμενα σήματα όπου το εύρος των τιμών τους είναι συνεχόμενο (Continuous),
- Σήματα τύπου “διακόπτη” όπου το σήμα αντιστοιχεί σε on/off και το φάσμα των τιμών δεν είναι συνεχές (Switch).

##### Αισθητήρες Συνεχούς Σήματος (Continuous SENSORS)

Οι κατηγορία αισθητήρων με συνεχές σήμα περιλαμβάνει αισθητήρες (I-CubeX : Sensors ):

**Πίεσης**, όπου η τιμή του αισθητήρα αποδίδει την δύναμη με την οποία πιέζεται η επιφάνεια του αισθητήρα.

**Bend**, όπου ο αισθητήρας σηματοδοτεί το πόσο πολύ τον λυγίζει ο χρήστης.

**Slide**, ο οποίος δίνει μια τιμή για την θέση του χεριού πάνω στον αισθητήρα.

**G-force**, ανιχνεύει την επιτάχυνση που προκαλείται στον αισθητήρα. Στην περίπτωση του αισθητήρα επιτάχυνσης, αναλόγως το μοντέλο, ανιχνεύει και μετράει την επιτάχυνση είτε σε 2 άξονες είτε σε 3 άξονες, αποδίδοντας αντίστοιχα την επιτάχυνση του αισθητήρα σε 2 ή 3 διαστάσεις. Μερικοί αισθητήρες G-force, μετράνε και την γωνία κλήσης στην οποία βρίσκονται.

Οι παραπάνω αισθητήρες είναι ένα κλάσμα μόνο από την πληθώρα αισθητήρων που υπάρχουν στο εμπόριο, και αντιπροσωπεύουν λίγες από τις καταστάσεις που μπορούν να μετρηθούν μέσω τέτοιων εργαλείων. Στη διάθεση του καταναλωτή υπάρχουν αισθητήρες για την μέτρηση της έντασης του διάχυτου φωτός, την αυξομείωση της θερμοκρασίας και της υγρασίας ή ακόμα την ποσότητα και ένταση εγκεφαλικής ή μυϊκής δραστηριότητας.

### Αισθητήρες τύπου διακόπτη (Switch Sensors)

Ας δούμε τώρα ορισμένους αισθητήρες με μη συνεχόμενο σήμα, τύπου switch (I-CubeX : Sensors ).

**Detection.** Ο αισθητήρας αυτός ανιχνεύει αν υπάρχει κίνηση εντός του οπτικού του πεδίου. Επιστρέφει τιμή 0 (off) όσο δεν υπάρχει κίνηση και τιμή 127 (on) όσο υπάρχει κίνηση. Η τιμή αυτή δεν μεταβάλλεται σε σχέση με την ποσότητα της κίνησης.

**Movement**. Ο αισθητήρας αυτός, ανιχνεύει αν ο ίδιος κινείται. Όσο παραμένει ακίνητος επιστρέφει τιμή 0, και όσο κινείται 127. Η τιμή αυτή δεν μεταβάλλεται όπως στον G-force όπου όσο μεγαλύτερη επιτάχυνση ανιχνεύει, τόσο μεγαλύτερη τιμή επιστρέφει.

**Footswitch/Switch**. Αυτού του τύπου αισθητήρες, είναι ίδιοι στην φιλοσοφία τους με ένα κοινό διακόπτη έχοντας δύο καταστάσεις 0 (off) και 127 (on). Υπάρχουν σε μορφή κοινού διακόπτη – κουμπιού ή footswitch για χρήση με το πόδι.

**Interactive Carpet**. Ένας αισθητήρας σε μορφή χαλιού όπου ανιχνεύει αν κάποιος το πατήσει!

Επίσης και σε αυτή την κατηγορία υπάρχουν πολύ ακόμα αισθητήρες. Γενικά οι περισσότεροι αισθητήρες συνεχούς σήματος, μπορούν να χρησιμοποιηθούν και ως switch αισθητήρες με κάποια διαχείριση στις τιμές που αποδίδουν.

Στην περίπτωση χρήσης αισθητήρων ή βιντεοκαμερών με αναλογική έξοδο, είναι απαραίτητη η χρήση ενός μετατροπέα του αναλογικού σήματος σε ψηφιακό. Η διαδικασία αυτή γίνεται με την χρήση των “digitizers” για τους οποίους θα μιλήσουμε παρακάτω.

#### 1.3.3.2. Μετατροπείς σήματος

Οι αισθητήρες, σαν συσκευές δεν είναι παρά ηλεκτρικά κυκλώματα τα οποία σύμφωνα με την χρήση τους, αυξομειώνουν την ηλεκτρική τους αντίσταση, μεταβάλλοντας έτσι την τιμή του ρεύματος που τα διαπερνά. Οι αισθητήρες χρειάζονται ρεύμα ισχύος 5V.

Για τον λόγο αυτό, υπάρχουν ολοκληρωμένα κυκλώματα, μετατροπείς σήματος, όπου τροφοδοτούν τους συνδεδεμένους αισθητήρες, διαβάζουν την τιμή του ρεύματος που επιστρέφουν και την μετατρέπουν σε ψηφιακό σήμα. Ορισμένες από τις συσκευές μετατροπής τέτοιου τύπου είναι οι εξής.

**Eo-body.**

Μια συσκευή από την εταιρία Eo-wave, η οποία επιτρέπει, αναλόγως την έκδοσή της, να συνδεθούν από 1 ως 16 αισθητήρες, τους οποίους τροφοδοτεί και ψηφιοποιεί. Τα δεδομένα των αισθητήρων μπορούν να σταλούν είτε μέσω USB σε κάποιον υπολογιστή, μέσω του συνοδευτικού λογισμικού της συσκευής, είτε μέσω θύρας midi-out σε μία συσκευή με αντίστοιχη midi-in θύρα (EoWave - interfaces ).

**I-cube Digitizerv4.**

Ένα αντίστοιχο προϊόν από την I-cubeX, με δυνατότητα σύνδεσης ως 32 αισθητήρες. Επίσης, αναλόγως την έκδοσή του, συνδέεται με υπολογιστή είτε μέσω USB, Midi-out ή ασύρματα μέσω Bluetooth, καθώς και με συσκευές, synthesizers, που έχουν θύρα Midi-in (I-CubeX : Digitizers ).

Εκτός των αναλογικών συσκευών ελέγχου, υπάρχουν συσκευές των οποίων οι τιμές εξόδου είναι ήδη ψηφιοποιημένες. Παρακάτω θα γίνει αναφορά στον τρόπο επικοινωνίας συσκευών τέτοιου είδους με έναν αλγόριθμο.

### 1.3.4. Αντιστοίχηση – Πρωτόκολλα επικοινωνίας

Όπως αναφέρθηκε πριν, μεταξύ των συσκευών, αισθητήρων και του υπολογιστή στον οποίο συνδέονται, μεσολαβούν οι ψηφιοποιητές (digitizers). Τα δεδομένα τα οποία στέλνει η συσκευή ψηφιοποίησης είναι μηνύματα τα οποία ακολουθούν τους κανόνες κάποιου πρωτοκόλλου επικοινωνίας και μορφοποιούνται βάση αυτού. Τα πρωτόκολλα αυτά χρησιμοποιούνται για την αποστολή μηνυμάτων μεταξύ υπολογιστών, synthesizers και κάθε συσκευής, συμβατής με το ανάλογο πρωτόκολλο.

#### 1.3.4.1. Πρωτόκολλα άμεσης επικοινωνίας

Τα πιο συνηθισμένα μουσικά πρωτόκολλα άμεσης επικοινωνίας είναι τα MIDI και OSC.

**Πρωτόκολλο MIDI**

Το πρωτόκολλο MIDI (**M**usical **I**nstrument **D**igital **I**nterface), πρώτο-ανακοινώθηκε το 1982, και τον Ιανουάριο του 1983 εμφανίστηκε ενσωματωμένο σε ένα μουσικό όργανο.

Τότε, οι προδιαγραφές του συμπεριελάμβαναν μόνο τις πιο βασικές οδηγίες για τα μηνύματα που ήταν πιθανόν να αποσταλούν μεταξύ δύο synthesizers, όπως, μηνύματα για ποιά νότα να παιχτεί ή την συνολική ένταση εξόδου.

Ο τρόπος λειτουργίας του MIDI πρωτοκόλλου έχει παραμείνει σχεδόν ο ίδιος από την εμφάνισή του ως σήμερα, παρότι οι ικανότητές του και το πλήθος των MIDI μηνυμάτων έχουν αυξηθεί σε τεράστιο βαθμό.

Οι αρχικές προδιαγραφές του MIDI πρωτοκόλλου, όριζαν και την φυσική σύνδεση και τη διαμόρφωση των μηνυμάτων για την σύνδεση με τις συσκευές και τον έλεγχό τους, σε πραγματικό χρόνο (History of MIDI ).

Το MIDI έχει τρεις πτυχές οι οποίες είναι οι εξής :

- Τα MIDI μηνύματα
- Η φυσική σύνδεση και,

- Τα Standard MIDI αρχεία.

Τα μηνύματα, αποτελούν το κυριότερο μέρος του πρωτοκόλλου MIDI, και αποτελούνται είτε από 8 ή 16 bits, αναλόγως του περιεχομένου του μηνύματος.

Η φυσική σύνδεση, των MIDI συσκευών αρχικά μπορούσε να επιτευχθεί μόνο με το καλώδιο MIDIDIN, όπως αυτό οριζόταν από το MIDI πρωτόκολλο. Με την εμφάνιση και την εξάπλωση νέων μέσων μεταφοράς ψηφιακών δεδομένων, το MIDI ανεξαρτητοποιήθηκε από το καλώδιο αυτό και μπορεί να λειτουργήσει μέσω οποιουδήποτε πρωτοκόλλου μεταφοράς ψηφιακής πληροφορίας (The Technology of MIDI ).

Τα αρχεία MIDI, είναι το μέσο αποθήκευσης των MIDI μηνυμάτων και της ανάγνωσής τους σε μη πραγματικό χρόνο. Η δυνατότητα αυτή αποθήκευσης των MIDI μηνυμάτων δεν υπήρχε μέχρι και το 1991.

**Πρωτόκολλο OSC**

Το πρωτόκολλο OSC (**O**pen **S**ound **C**ontrol) δημιουργήθηκε στο τμήμα CNMAT, του πανεπιστημίου του Berkeley. Ο στόχος του είναι να υπερισχύσει του πρωτοκόλλου MIDI. Ο τρόπος λειτουργίας του βασίστηκε στην αυξημένη υπολογιστική ισχύ και την ταχύτητα μετάδοσης μέσω δικτύου.

Αντίθετα με τον περιορισμό του MIDI στον τύπο δεδομένων, το OSC μπορεί να μεταδώσει δεδομένα τύπου integer, float, String, blob, Boolean, Null, Impulse. Χρησιμοποιεί την λογική των πακέτων, όπου αποστέλλει δεδομένα σε περισσότερες από μία παραμέτρους, σε μία ομαδοποιημένη εντολή τύπου OSCBundle.

Οι παράμετροι που χειρίζεται ο χρήστης έχουν διεύθυνση τύπου String οργανωμένες ιεραρχικά, με εντολές που μπορούν να ρυθμίζουν ομαδοποιημένες παραμέτρους με μία μόνο εντολή. Λόγω της μορφής των διευθύνσεων που χρησιμοποιεί αυτό το πρωτόκολλο, το πλήθος των διευθύνσεων είναι άπειρο εν αντιθέσει με τον περιορισμό του MIDI σε 16 κανάλια (OSC ) (Fraietta, 2008 ).

Μέχρι στιγμής δεν έχει υπερισχύσει κάποιο από τα δύο πρωτόκολλα, εφόσον και τα δύο χρησιμοποιούνται ισάξια. Το καθένα έχει τους δικούς του περιορισμούς και πλεονεκτήματα. Οι

σχέσεις των ομάδων έρευνας και των δύο πρωτοκόλλων, εκτυλίσσονται εντός κλίματος ευγενούς άμιλλας.

Τα πρωτόκολλα MIDI και OSC δύναται να χρησιμοποιηθούν και μεταξύ απομακρυσμένων συσκευών, μέσω δικτύου. Τα συνηθέστερα πρωτόκολλα επικοινωνίας μέσω δικτύου αναφέρονται παρακάτω.

#### 1.3.4.2.Πρωτόκολα επικοινωνίας μέσω δικτύου

**Πρωτόκολλο UDP**

Το πρωτόκολλο επικοινωνίας UDP (**U**ser **D**atagram **P**rotocol) δημιουργήθηκε από τον David Reed το 1980. Ο σκοπός του ήταν να δημιουργηθεί ένα πρωτόκολλο το οποίο θα χρησιμοποιούσε όσο λιγότερη δικτυακή "γραφειοκρατία" ήταν δυνατό, καθιστώντας το ένα πρωτόκολλο ταχείας μεταφοράς δεδομένων.

Λόγω του προσανατολισμού του προς την ταχύτητα της μετάδοσης, το UDP δεν μπορεί να εγγυηθεί, την διανομή των πακέτων δεδομένων, τα σφάλματα σε αυτά λόγω θορύβου δικτύου ή τις διπλές αποστολές και στηρίζεται στην ύπαρξη ελέγχου των πακέτων δεδομένων στο εκάστοτε λογισμικό που τα δέχεται (Reed & Postel, 1978)

**Πρωτόκολλο TCP**

Τον Μάιο του 1974, από τους Vint Cerf και Bob Kahn εμφανίστηκε το πρωτόκολλο TCP (**T**ransmission **C**ontrol **P**rotocol). Η έρευνά του ξεκίνησε κυρίως για δίκτυα του στρατού, τα οποία βασίζονταν περισσότερο στην αξιοπιστία της μετάδοσης παρά στην ταχύτητα. Το πρωτόκολλο αυτό χρησιμοποιεί αριθμημένα πακέτα δεδομένων και συμπεριλαμβάνει έλεγχο αλλοιωμένων ή διπλά απεσταλμένων πακέτων ώστε να εγγυάται την σίγουρη συνοχή των πακέτων δεδομένων. Για τον λόγο αυτό, δίνοντας προτεραιότητα στην αξιοπιστία της μετάδοσης, το πρωτόκολλο αυτό δεν μπορεί να εγγυηθεί την ταχεία μεταφορά δεδομένων (Cerf & Kahn, 1974).

Σε εφαρμογές με απομακρυσμένη αποστολή δεδομένων όπου χρειάζεται η άμεση επικοινωνία μεταξύ αποστολέα και λήπτη, όπως εφαρμογές και συστήματα διάδρασης, συνηθίζεται να χρησιμοποιείται το πρωτόκολλο UDP λόγω της ταχύτερης μεταφοράς του. Σε τέτοιες εφαρμογές πραγματικού χρόνου, προτιμάται να χαθεί κάποια τιμή παρά να καθυστερήσει η παράδοσή της στο

σύστημα, εφόσον, αν η εφαρμογή δεν μπορεί να λειτουργήσει σε πραγματικό χρόνο, χάνει την λειτουργικότητά της.

Μετά την επιτυχημένη εισαγωγή των τιμώ ελέγχου του χρήστη, στο ψηφιακό περιβάλλον του εκάστοτε διαδραστικού συστήματος, ακολουθεί η δρομολόγηση τους. Η σωστή σύνδεση των τιμών από τις συσκευές ελέγχου, με τις παραμέτρους της γεννήτριας ήχου επιτυγχάνεται μέσω της αντιστοίχησης διασυνδέσεων (Mapping), διαδικασία την οποία θα αναλύσουμε παρακάτω.

### 1.3.5. Αντιστοίχηση παραμέτρων (Mapping)

Η σύνδεση των κινήσεων του χρήστη με διαδικασίες εντός ενός διαδραστικού συστήματος και των διαδικασιών με την ανταπόκριση του συστήματος προς τον χρήστη ονομάζεται αντιστοίχηση παραμέτρων (mapping) (Miranda&Wanderlay, 2006). Εντός των πλαισίων των μουσικών διαδραστικών εφαρμογών, αντιστοίχηση παραμέτρων εφαρμόζεται μεταξύ όλων των σταδίων και διαδικασιών του εκάστοτε συστήματος (Drummond, 2009 ).

Σημαντικό στοιχείο για την εκφραστικότητα μιας μουσικής διαδραστικής εφαρμογής, εκτός της ακριβούς σύλληψης των κινήσεων του χρήστη, είναι ο τρόπος με τον οποίο εφαρμόζεται η αντιστοίχηση των δεδομένων των κινήσεων με τις παραμέτρους του αλγορίθμου του ήχου. Συνηθίζεται να χρησιμοποιείται αντιστοίχηση μίας τιμής εισόδου, σε μία παράμετρο του αλγορίθμου. Ο τρόπος αυτός παρότι άμεσος, δεν είναι εκφραστικά αποδοτικός επειδή δεν εκμεταλλεύεται την υψηλού επιπέδου (high level) σύνδεση κίνησης του χρήστη με τον έλεγχο του αλγορίθμου δημιουργίας ήχου (Rovan, Wanderley, Dubnov, & Depalle, 1997 ).

Προτείνεται ο διαχωρισμός των τρόπων αντιστοίχησης σε τρείς περιπτώσεις (Rovan, Wanderley, Dubnov, & Depalle, 1997 ) (Hunt & Kirk, 2000 ).

**One-to-One / ένα προς ένα :** Στην πρώτη κατηγορία όπως και το όνομά της υποδεικνύει, κάθε μία παράμετρος ελέγχου αντιστοιχεί σε μία παράμετρο του αλγορίθμου σύνθεσης ήχου. Αυτή είναι η πιο άμεση τεχνική αντιστοίχησης εφόσον μεταξύ της παραμέτρου ελέγχου και αλγορίθμου σύνθεσης, υπάρχει ελάχιστη μέχρι και καθόλου επεξεργασία.

**One-to-Many/ διιστάμενη αντιστοίχηση**: Στην περίπτωση της διιστάμενης αντιστοίχησης, μία παράμετρος ελέγχου αντιστοιχείται σε περισσότερες της μίας παραμέτρους στην δημιουργία του

ήχου. Η επιλογή αυτής της λογικής αντιστοίχησης διασυνδέσεων, μπορεί να προσφέρει έναν πιο ολοκληρωτικό έλεγχο στο σύστημα αλλά λόγω της φύσης της, όταν χρησιμοποιείται μόνη της μπορεί να αποβεί ελλιπής στον πιο λεπτομερειακό έλεγχο του εκάστοτε συστήματος.

**Many-to-One / Συγκλίνουσα αντιστοίχηση**: Η τρίτη κατηγορία αντιστοίχησης, συγκλίνουσα αντιστοίχηση, λειτουργεί συνδυάζοντας περισσότερες από μια παραμέτρους ελέγχου αντιστοιχίζοντας το αποτέλεσμα του συνδυασμού αυτού σε μια παράμετρο του αλγορίθμου σύνθεσης. Η χρήση αυτού του τρόπου mapping, απαιτεί εμπειρία στην χρήση του συστήματος στο οποίο εφαρμόζεται λόγω της μεγαλύτερης πολυπλοκότητάς της. Όμως, σε σύγκριση με τους άλλους δύο τρόπους, η αντιστοίχηση αυτή μπορεί να αποδώσει την περισσότερη εκφραστικότητα του χρήστη αυτού του διαδραστικού συστήματος.

Ο άνθρωπος στην καθημερινότητά του συναντάει διιστάμενες και συγκλίνουσες διασυνδέσεις, παρόλα αυτά, ο χειρισμός των εφαρμογών σχεδιάζεται ως επί το πλείστον με την χρήση της πρώτης κατηγορίας αντιστοίχησης, ένα προς ένα. Εάν δούμε τον τρόπο χρήσης ενός βιολιού για παράδειγμα, θα παρατηρήσουμε πως ο έλεγχος για την τονικότητα του θα μπορούσε να έχει την εξής συνάρτηση:

Τονικότητα = ((κύριος συντελεστής) * θέση δαχτύλου) + ((δευτερεύων συντελεστής) * πίεση δοξαριού).

Είναι ξεκάθαρο πως ο τρόπος ελέγχου εμπίπτει στην συγκλίνουσα αντιστοίχηση. Παράλληλα όμως η πίεση του δοξαριού δεν επηρεάζει και την ένταση του ήχου που παράγεται, δημιουργώντας μία διιστάμενη αντιστοίχηση (Hunt & Kirk, 2000 ).

Εφόσον μιλήσαμε για συσκευές εισόδου σε κάποιο διαδραστικό σύστημα και τους τρόπους αντιστοίχησης των δεδομένων τους, παρακάτω υποκεφάλαιο θα αναφερθούν στοιχεία για τις γεννήτριες ήχου και ορισμένες βασικές τεχνικές σύνθεσης ήχου.

### 1.3.6. Γεννήτριες ήχου

Η γεννήτρια ήχου σε ένα ηχητικό έργο, μπορεί να προέρχεται είτε από την δημιουργία ενός λογισμικού ή από μια "hardware" συσκευή στην οποία ενσωματώνεται ένα ολοκληρωμένο ψηφιακό ή αναλογικό σύστημα. Ο σχεδιασμός ενός αλγορίθμου ήχου μπορεί να επιτευχθεί μέσω διαφόρων τρόπων προσέγγισης.

Σε πρώτο επίπεδο ο καλλιτέχνης μπορεί να σχεδιάσει έναν αλγόριθμο με βήμα-βήμα οδηγίες για την παραγωγή του κάθε sample. Αυτός ο τρόπος όμως καθιστά σχεδόν αδύνατη την τροποποίηση του αλγορίθμου για την επίτευξη ενός συγκεκριμένου ήχου.

Στο δεύτερο επίπεδο απαιτείται η κατάτμηση του συνολικού αλγορίθμου σε μικρότερες ολοκληρωμένες ρουτίνες. Οι ρουτίνες αυτές ονομάζονται "unit generators" και η καθεμία ευθύνεται για μία λειτουργία και μόνο. Έχει συγκεκριμένες εισόδους ελέγχου και τουλάχιστον μία έξοδο πληροφοριών. Οι μονάδες αυτές αποτελούν τα δομικά στοιχεία για έναν αλγόριθμο. Έτσι ο χρήστης ελαχιστοποιεί την απαραίτητη γνώση για τις διαδικασίες που είναι κριμένες πίσω από το κάθε στοιχείο.

Το τρίτο επίπεδο φέρνει τον χρήστη απέναντι από ολοκληρωμένα συστήματα, δομημένα με "unit generators", τα οποία υλοποιούν έτοιμες τεχνικές σύνθεσης ήχου. Στην περίπτωση αυτή ο χρήστης δεν έχει παρά μόνο να προσδιορίσει τις παραμέτρους της τεχνικής που επιλέγει (Dodge & Jerse, 1997 ).

Στο επόμενο κεφάλαιο θα παρουσιαστούν κάποια βασικά στοιχεία και πληροφορίες για το ψηφιακό βίντεο καθώς και στο τέλος του κεφαλαίου θα αναφερθούν ορισμένες τεχνικές σύνθεσης ήχου.

# 2. ΕΞΕΙΔΙΚΕΥΜΈΝΗ ΘΕΩΡΊΑ

Στο παρόν κεφάλαιο παρουσιάζονται κάποια στοιχεία θεωρίας τα οποία συνδέονται άμεσα με την εκπόνηση του πειραματικού μέρους. Εφόσον στην αναπτυχθείσα εφαρμογή η συσκευή εισόδου είναι μια κάμερα βίντεο, θα παρουσιάσουμε κάποια στοιχειώδη θεωρία για τις τεχνικές ψηφιακής ανάλυσης εικόνας βίντεο. Τέλος, θα συμπεριλάβουμε και κάποια στοιχεία τεχνικών σύνθεσης ψηφιακού ήχου.

Πρώτα θα μιλήσουμε για τις τεχνικές ψηφιακής ανάλυσης (κινούμενης) εικόνας βίντεο

## 2.1. Ψηφιακό βίντεο

Μια ψηφιακή εικόνα αποτελείται από ένα δισδιάστατο πίνακα τιμών, οι οποίες αντιπροσωπεύουν την ένταση του φωτός. Κάθε εικόνα είναι μια συνάρτηση f(x,y) όπου f είναι η φωτεινότητα και x,y είναι οι χωρικές συντεταγμένες του στοιχείου της εικόνας, ή αλλιώς του εικονοστοιχείου / pixel.

Κατά σύμβαση το εικονοστοιχείο με συντεταγμένες (0,0) ορίζεται στην πάνω αριστερά γωνία μιας εικόνας. Ο x (οριζόντιος) άξονας αυξάνεται από αριστερά προς δεξιά, ενώ ο y (κατακόρυφος) άξονας αυξάνεται από πάνω προς κάτω.

Η ανάλυση της εικόνας καθορίζεται από τον αριθμό στηλών και γραμμών από εικονοστοιχεία. Μία εικόνα με πλήθος m στύλων, και n γραμμών, έχει ανάλυση m*n.

Ένα σημαντικό στοιχείο σε μία ψηφιακή εικόνα είναι το “bit depth” και ορίζει το πόσα bit χρησιμοποιούνται για την απόδοση της έντασης φωτός σε κάθε εικονοστοιχείο. Οι πιθανές τιμές που μπορεί να λάβει ένα εικονοστοιχείο, σε μια εικόνα με “bit depth N”, είναι $2^N$. Για παράδειγμα, με 8 bit, κάθε pixelμπορεί να πάρει τιμές από 0 ως 255, σε σύνολο 256 τιμών. Με 16 bits, οι πιθανές τιμές είναι από 0 ως 65535.

Οι τιμές του bit depth, που χρησιμοποιούνται είναι:

- 8bit, 16bit, 32bit για απεικόνιση εικόνων σε αποχρώσεις του γκρι.
- 32bitγια έγχρωμη RGB ή HSL εικόνα
- 64bitγια εικόνες που περιέχουν τιμές μιγαδικών αριθμών.

Κάθε εικόνα αποτελείται από έναν ή περισσότερους δισδιάστατους πίνακες (συνήθως τέσσερεις). Στην περίπτωση της έγχρωμης εικόνας χρησιμοποιούνται 4 πίνακες. Οι τρείς πίνακες περιέχουν την ένταση για κάθε ένα από τα βασικά χρώματα, κόκκινο (Red), πράσινο (Green), μπλε (Blue), και ο τέταρτος πίνακας αποθηκεύει τις alpha τιμές.

Οι τιμές "alpha" χρησιμοποιούνται για να καθορίσουν το ποσοστό διαφάνειας κάθε εικονοστοιχείου όταν γίνονται πράξεις μεταξύ δύο εικόνων. Οι τιμές που δέχεται είναι 0 ως 1. Όταν οι τιμή του καναλιού τείνει στο 0, τότε το εικονοστοιχείο θεωρείται διάφανο, ενώ όταν η τιμή είναι 1, το εικονοστοιχείο θεωρείται αδιαφανές. Θα μπορούσε να θεωρηθεί ως συντελεστής βαρύτητας στην μίξη εικονοστοιχείων . Στην περίπτωση που χρησιμοποιείται μία μοναδική ροή βίντεο, το κανάλι "alpha" παραλείπεται, θεωρώντας πως περιέχει μόνο την τιμή 1.

Οπότε τα 32bitσε κάθε εικονοστοιχείου αντιστοιχούν σε 4 πίνακες των 8 bit σε κάθε εικονοστοιχείο, κάθε ένας για ένα από τα βασικά χρώματα και το κανάλι alpha.

Για τις εικόνες γκρίζων αποχρώσεων χρησιμοποιείται ένας πίνακας μόνο, των 8, 16 ή 32bitγια κάθε εικονοστοιχείο. Οι τιμές αυτές αντιπροσωπεύουν τη φωτεινότητα σε κάθε εικονοστοιχείο, με 0 να ισοδυναμεί σε μαύρο εικονοστοιχείο, και 255 (8bit integer) ή 1 (16 & 32bit float) ισοδυναμεί σε άσπρο.

Οι εικόνες των οποίων οι πίνακες περιέχουν μιγαδικές τιμές αποτελούνται από δύο πίνακες. Ο πρώτος περιέχει το πραγματικό μέρος του μιγαδικού αριθμού και ο δεύτερος το φανταστικό μέρος. Οι τιμές αυτές προέρχονται από την εφαρμογή FFT (Fast Fourier Transform) σε εικόνες γκρι αποχρώσεων (National_Instruments, 2003).

## 2.2. Ανάλυση εικόνας και στοιχεία “blob”

Στο κεφάλαιο αυτό αναφέρονται κάποια στοιχεία για την ανάλυση της εικόνας βίντεο. Αρχικά θα γίνει αναφορά στο μέσο που χρησιμοποιήθηκε στην παρούσα εργασία για την σύλληψη των κινήσεων του χρήστη, τη βιντεοκάμερα, και στην συνέχεια θα παρουσιαστούν ορισμένες τεχνικές οι οποίες είναι απαραίτητες στην αφαίρεση φόντου και ανάλυση της κίνησης.

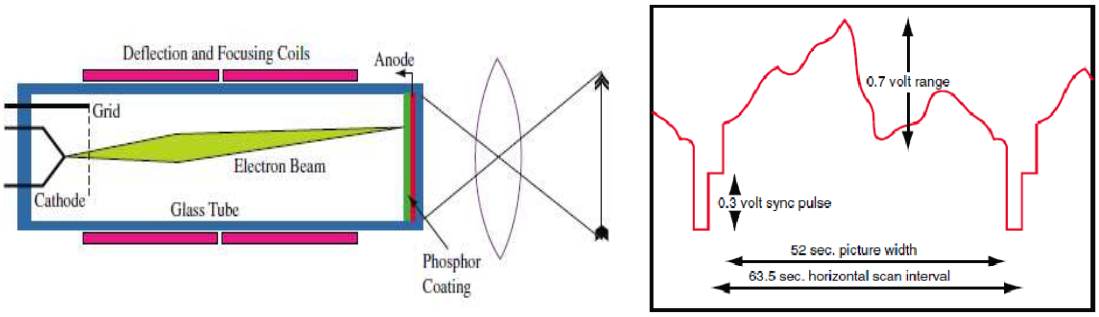
### 2.2.1. Βιντεοκάμερα

Πολλές εφαρμογές χρησιμοποιούν βιντεοκάμερες ως συσκευές εισόδου, για τη σύλληψη δεδομένων κινήσεων και τον έλεγχο παραμέτρων εξόδου της διαδραστικής εφαρμογής. Μπορούμε λοιπόν να θεωρήσουμε την κάμερα ως μεσολαβητή μεταξύ του φυσικού κόσμου και της ψηφιακής εφαρμογής, σαν ένα τύπο αισθητήρα. Οι κάμερες χρησιμοποιούνται ολοένα και περισσότερο σε διαδραστικές εφαρμογές εφόσον από το βίντεο μπορούν να εξαχθούν πολλών τύπων παράμετροι, ανάλογα με τις εκάστοτε απαιτήσεις.

**Κάμερα τύπου σωλήνα (Tube type Cameras)**

Οι πρώτες ευρέως χρησιμοποιούμενες κάμερες ήταν οι κάμερες τύπου σωλήνα. Οι κάμερες αυτές, λειτουργούν ως εξής :

Το φως πέφτει πάνω σε ένα στρώμα από φώσφορο. Αναλόγως με την ένταση του φωτός σε κάθε σημείο της επιφάνειας, η αντίσταση του φωσφόρου αλλάζει. Μία συγκεντρωμένη ακτίνα ηλεκτρονίων σαρώνει την επιφάνεια του φωσφόρου και αναλόγως την αντίσταση στο κάθε σημείο σάρωσης, περνάει περισσότερο ή λιγότερο ρεύμα αποδίδοντας έτσι μια συνεχόμενη αυξομειούμενη τάση που αντιστοιχεί στην ένταση του φωτός στο σημείο αυτό (Εικόνα 11).




Εικόνα 11

Η επιφάνεια του φωσφόρου σαρώνεται σε συγκεκριμένο αριθμό σειρών. Εφόσον η εξερχόμενη τάση είναι συνεχόμενη, για τον συγχρονισμό και σηματοδότηση της έναρξης κάθε σειράς χρησιμοποιείται ένα σήμα-παλμός 0.3 Volt ανά τακτά χρονικά διαστήματα, με διάρκεια όση απαιτείται για τον χρόνο σάρωσης της κάθε γραμμής (Ross, 2011 ).

**Κάμερα CCD (Charge Coupled Device)**

Οι κάμερες τύπου σωλήνα αντικαταστάθηκαν από τις κάμερες CCD. Σε αυτές τις κάμερες η φωτοευαίσθητη περιοχή που χρησιμοποιείται για τη σύλληψη της εικόνας αποτελείται από μια συστοιχία πυκνωτών, καθένας από τους οποίους λειτουργεί ως συσσωρευτής ηλεκτρονίων αναλόγως με την ένταση του φωτός στο σημείο που βρίσκεται.

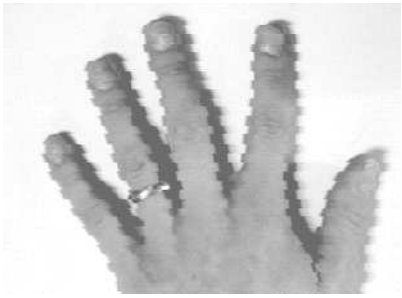
Λόγω του πλήθους των πυκνωτών, που κυμαίνεται από αρκετές χιλιάδες ως εκατομμύρια, δεν είναι πρακτικό να υπάρξει καλωδίωση για την μέτρηση του φορτίου σε κάθε πυκνωτή. Μόλις αποτυπωθεί η εικόνα στην συστοιχία των πυκνωτών, με το σήμα από ένα ρολόι, κάθε σειρά πυκνωτών μεταφέρει το φορτίο της στην επόμενη. Η τελευταία σειρά πυκνωτών στέλνει το φορτίο του κάθε πυκνωτή της με την σειρά, σε ένα ενισχυτή όπου αυτός παράγει ένα σήμα αναλογικής τάσης το οποίο με την σειρά του είτε οδηγείται στην έξοδο της κάμερας ως αναλογικό είτε ψηφιοποιείται πρώτα (Ross, 2011 ) .

**Πεπλεγμένη και προοδευτική σάρωση – Interlaced and progressive scan**

Υπάρχουν δύο τρόποι για την σάρωση των δεδομένων από τον αισθητήρα της κάμερας.

Ο πρώτος ονομάζεται "πεπλεγμένη σάρωση" / "interlaced scan". Η μέθοδος αυτή χωρίζει το καρέ της εικόνας σε δύο πεδία. Το ένα πεδίο περιέχει όλες τις ζυγές σειρές του καρέ, ενώ το δεύτερο περιέχει όλες τις μονές σειρές του καρέ. Σε κάθε νέο καρέ καταγράφεται είτε το πεδίο των μονών σειρών, είτε των ζυγών εναλλάξ. Αυτό σημαίνει πως η πλήρης ανανέωση των σειρών της εικόνας απαιτεί τον χρόνο δύο διαδοχικών καρέ. Η μέθοδος αυτή χρησιμοποιείται στην μετάδοση εικόνας βίντεο ώστε να μειωθεί η ποσότητα των δεδομένων.

Η δεύτερη τεχνική, προοδευτικής σάρωσης "διαβάζει" ολόκληρο κάθε νέο καρέ (Taylor, 1998 ).



*"Interlaced"*

Όπως φαίνεται και στην παραπάνω εικόνα, η μέθοδος "interlaced" δεν είναι ικανοποιητική στην σύλληψη εικόνας, και ο σκοπός της είναι η ευκολότερη μετάδοση εικόνας. Για την καθαρή σύλληψη κίνησης είναι απαραίτητη η χρήση της προοδευτικής μεθόδου.

**Πλεονεκτήματα και Μειονεκτήματα**

Και οι δύο τύποι κάμερας έχουν τα πλεονεκτήματα και μειονεκτήματα τους.

Στις κάμερες τύπου σωλήνα, η φασματική τους απόκριση είναι πολύ κοντινή στην φασματική απόκριση του ανθρώπινου ματιού. Επίσης η ανάλυση της εικόνας περιορίζεται μόνο από το μέγεθος των κόκκων του φωσφόρου στον αισθητήρα και το μέγεθος της συγκεντρωμένης ακτίνας ηλεκτρονίων.

Ένα ακόμα πλεονέκτημα είναι η δυνατότητα λήψης σε συνθήκες πολύ χαμηλού φωτισμού λόγω της ευαισθησίας του φωσφόρου. Όμως, οι κάμερες αυτές έχουν προβλήματα εστίασης στα γύρω μέρη της εικόνας, καθώς και παραμόρφωση στο σύνολό τους. Επίσης η μετάβαση από μια φωτεινή σε μία σκοτεινή εικόνα είναι αργή, αφήνοντας φωτεινά σημεία στην επόμενη εικόνα μεγαλύτερα από τις πραγματικές τους διαστάσεις, τα λεγόμενα "comet trails".

Οι κάμερες CCD, λόγω της κατασκευής τους δεν παραμορφώνουν την εικόνα όμως η φασματική τους απόκριση δεν είναι γραμμική. Οι CCD κάμερες, είναι πολύ ευαίσθητες στο ερυθρό φως, ακόμα και στο υπέρυθρο. Για τον λόγο αυτό χρησιμοποιούν φίλτρα που αφαιρούν την υπέρυθρη περιοχή του φωτός, το οποίο παρότι δεν είναι ορατό προκαλεί θολά σημεία στην εικόνα. Αντίστροφα η ευαισθησία τους στο μπλε φως είναι πολύ μικρή (Ross, 2011 ).

Παρακάτω παρουσιάζονται ορισμένες τεχνικές, οι οποίες χρησιμοποιούνται ως αυτόνομες λειτουργίες σε τεχνικές αφαίρεσης φόντου.

### 2.2.2. Βασικές διαδικασίες επεξεργασίας εικόνας

Στο υποκεφάλαιο αυτό θα γίνει η επεξήγηση των διαδικασιών επεξεργασίας της ψηφιακής εικόνας, οι οποίες απαιτούνται κατά την αφαίρεση φόντου και την ανίχνευση κηλίδων (blob). Παρουσιάζονται με την σειρά που συνήθως συναντώνται.

**Μετατροπή έγχρωμης εικόνας σε ασπρόμαυρη – RGB to luminance**

Η μετατροπή της έγχρωμης εικόνας σε ασπρόμαυρη γίνεται με τη μίξη των τριών καναλιών χρώματος της εικόνας. Κάθε κανάλι χρώματος έχει διαφορετικό συντελεστή. Οι συνάρτηση που χρησιμοποιείται συνήθως από λογισμικά είναι η εξής (Cycling74) (MathWorks):

$$L = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

όπου R είναι το κανάλι του κόκκινου χρώματος, G του πράσινου και B του μπλε.

Άλλες τεχνικές υπολογίζουν το ισόβαρο μέσο όρο των καναλιών ή το μέσο όρο των ακραίων τιμών μεταξύ των τριών καναλιών (Cook, 2009 ).

**Median filter**

Το φίλτρο μέσου, αντιστοιχεί σε κάθε pixel τη μεσαία τιμή των γειτονικών του pixels. Από τον χρήστη ορίζεται η απόσταση σε pixel που θεωρείται γειτονική περιοχή (συνήθως 3 ή 5) και ο αλγόριθμος βρίσκει τη μέση τιμή τους, και την αποθηκεύει στο συγκεκριμένο pixel. Ο τρόπος που λειτουργεί το φίλτρο αυτό είναι ο εξής: Καταρχήν επιλέγεται το pixel που θα υποστεί τη διαδικασία της αλλοίωσης. Στη συνέχεια, συλλέγονται οι τιμές των γειτονικών pixels, το πλήθος των οποίων έχει επιλεγεί νωρίτερα από τον χρήστη (εντός κάποιας απόστασης από το κεντρικό). Εφόσον μαζευτούν οι τιμές αυτές σε ένα πίνακα, ταξινομούνται σε αύξουσα σειρά. Στην περίπτωση που το μέγεθος του πίνακα είναι μονός αριθμός, τότε το pixel παίρνει την τιμή ο οποία περιέχεται στο μεσαίο κελί του πίνακα, αλλιώς, αν το μέγεθος του είναι ζυγό, τότε το pixel παίρνει την τιμή του μέσου όρου των τιμών των δύο μεσαίων κελιών του πίνακα (National_Instruments, 2003).

**Κατωφλίωση εικόνας - threshold**

Η εφαρμογή μιας τιμής κατωφλίου σε μια γκρι εικόνα έχει ως αποτέλεσμα τη μετατροπή της σε δυαδική εικόνα. Κάθε εικονοστοιχείο με τιμή μικρότερη της τιμής κατωφλίου θα πάρει την τιμή 0, ενώ κάθε εικονοστοιχείο με τιμή μεγαλύτερη θα πάρει την τιμή 1. Η τεχνική αυτή επιτρέπει την επιλογή των εικονοστοιχείων μιας εικόνας τα οποία θεωρούνται αντικείμενα ενδιαφέροντος, ενώ τα υπόλοιπα απορρίπτονται ως μαύρα (θεωρούμενα ως φόντο της εικόνας).

Εκτός της απλής κατωφλίωσης, που χωρίζει τις τιμές σε δύο περιοχές, υπάρχει και η τεχνική της διπλής κατωφλίωσης. Στην περίπτωση αυτή ορίζονται δύο τιμές κατωφλίου. Τα εικονοστοιχεία με τιμή εκτός των ορίων που θέτουν οι δύο τιμές θεωρούνται φόντο και τους δίνεται η τιμή 0, ενώ εκείνα με τιμές εντός των τιμών των κατωφλίων γίνονται δεκτά ως περιοχή ενδιαφέροντος (τους δίνεται τιμή 1) (National_Instruments, 2003).

Θα ακολουθήσει η παρουσίαση τεχνικών για την επεξεργασία της ψηφιακής εικόνας με σκοπό την αφαίρεση φόντου, καθώς και τεχνικές για την μέτρηση παραμέτρων των κηλίδων (blob), τα οποία ανιχνεύονται στη συνέχεια.

### 2.2.3. Αφαίρεση φόντου, ανίχνευση και ανάλυση“blob”

Η διαδικασία ανάλυσης της κίνησης διαφόρων στοιχείων σε ένα βίντεο απαιτεί τα παρακάτω βήματα :

- Αφαίρεση φόντου
- Κατωφλίωση εικόνας
- Ανίχνευση / ονομασία κηλίδων (blob)
- Διαχωρισμός, αναγνώριση και ανάλυση ξεχωριστών κηλίδων (blob)

Η κάθε μία των παραπάνω διαδικασιών περιέχει ορισμένες επιμέρους αυτόνομες διαδικασίες. Παρακάτω ακολουθεί η ανάλυση κάθε βήματος και των εσωτερικών του διαδικασιών.

## Αφαίρεση φόντου

Η τεχνική της αφαίρεσης φόντου είναι ευρέως διαδεδομένη. Χρησιμοποιείται από ιατρικά μηχανήματα μέχρι συστήματα ασφαλείας. Έχουν αναπτυχθεί πολλές μέθοδοι για την διαδικασία αυτή, για περιπτώσεις στατικού φόντου όπως συναντάμε στην παρούσα εργασία, και για κινούμενο φόντο. Οι κύριες διαφορές μεταξύ των διαφόρων τεχνικών αφαίρεσης φόντου εντοπίζονται στον τρόπο που υπολογίζεται το μοντέλο του φόντου και την ανανέωση ή προσαρμογή του μοντέλου αυτού καθώς και στον τρόπο σύγκρισης του μοντελοποιημένου φόντου με το τρέχον καρέ.

Οι απλούστερες τεχνικές μετατρέπουν την εικόνα σε ασπρόμαυρη πριν την ανάλυσή της, ενώ ορισμένες βελτιωμένες και πολυπλοκότερες, εκμεταλλεύονται και τα τρία κανάλια χρώματος.

Τα βήματα για την αφαίρεση του φόντου σε κάθε τεχνική είναι:

- Μετατροπή εικόνας σε ασπρόμαυρη (σε ορισμένες τεχνικές το βήμα αυτό παραλείπεται)
- Μοντελοποίηση του φόντου
- Σύγκριση τρέχοντος καρέ με το μοντέλο φόντου
- Κατωφλίωση εικόνας

Ακολουθεί η παρουσίαση των δύο τεχνικών οι οποίες προσεγγίστηκαν κατά τον σχεδιασμό και εκπόνηση του πρακτικού μέρους της εργασίας.

**Απλή αφαίρεση φόντου – Διαφορά φόντου**

Η απλούστερη τεχνική για την αφαίρεση του φόντου υλοποιεί τα βήματα ως εξής:

Το πρώτο βήμα στην απλή τεχνική είναι η μετατροπή της εικόνας από έγχρωμη σε ασπρόμαυρη. Για την δημιουργία ενός μοντέλου για το φόντο, χρησιμοποιείται απλά ένα καρέ, στο οποίο θεωρείται πως απεικονίζεται μόνο το φόντο (λαμβάνεται μια εικόνα του χώρου ενδιαφέροντος χωρίς τα κινούμενα αντικείμενα). Το βήμα σύγκρισης του μοντέλου φόντου με το τρέχων καρέ γίνεται υπολογίζοντας το απόλυτο της διαφοράς μεταξύ των δύο. Τέλος, η εικόνα συγκρίνεται με μία τιμή κατωφλίου. Η τιμή αυτή υπολογίζεται πειραματικά ανά περίπτωση και συμβάλει στην μείωση θορύβου της εικόνας (Eyesweb).

Η συνάρτηση η οποία εκφράζει την διαδικασία της απόλυτης διαφοράς είναι η εξής:

$$F_t = |I_t - B|$$

Όπου:

- $I_t$, η τιμή του κάθε pixel την στιγμή t
- B, το φόντο
- $F_t$, εικόνα προσκηνίου την στιγμή t

Η τεχνική αυτή είναι απολύτως ευάλωτη στον θερμικό θόρυβο που δημιουργείται στην εικόνα από την ίδια της κάμερα καθώς και η ύπαρξη σκιών επηρεάζει πολύ το αποτέλεσμά. Η απλή διαφορά φόντου μπορεί να αποδώσει επαρκές αποτέλεσμα μόνο υπό ιδανικές συνθήκες φωτισμού, θερμικού θορύβου κάμερας και απολύτως σταθερού φόντου.

**Αφαίρεση φόντου με χρήση τυπικής απόκλισης**

Η υλοποίηση αυτού του τρόπου αφαίρεσης φόντου απαιτεί τον υπολογισμό του Gaussian μέσου όρου των τιμών για κάθε εικονοστοιχείο της εικόνας, για ορισμένο πλήθος καρέ. Επίσης, παράλληλα υπολογίζεται και η τυπική απόκλιση για κάθε εικονοστοιχείο. Οι δύο αυτές τιμές αποτελούν το μοντέλο του φόντου για κάθε pixel.

Η συνάρτηση υπολογισμού του Gaussian μέσου όρου είναι:

$$\mu_t = a*I_t + (1-a)*\mu_{t-1}$$

Ο υπολογισμός της τυπικής απόκλισης γίνεται με της συνάρτηση:

$$\sigma = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(I_t - \mu)^2}$$

Τα στοιχεία των συναρτήσεων είναι:

- $\mu_t$, η τιμή του Gaussian μέσου όρου της στιγμή t
- $I_t$, η τιμή του κάθε pixel την στιγμή t
- a, ο συντελεστής βαρύτητας του τρέχοντος καρέ στον υπολογισμό του Gaussian μέσου όρου
- "σ", η τυπική απόκλιση
- N, το πλήθος των καρέ για τον υπολογισμό της τυπικής απόκλισης

Το πρώτο βήμα είναι η μετατροπή του έγχρωμου βίντεο σε ασπρόμαυρο. Στην συνέχεια ορίζεται μια διάρκεια, σε πλήθος καρέ, κατά την οποία υπολογίζονται το Gaussian μέσο όρο και η τυπική απόκλιση του φόντου. Με τον υπολογισμό των δύο τιμών ολοκληρώνεται η δημιουργία του μοντέλου του φόντου. Κάθε επόμενο καρέ συγκρίνεται με το μοντέλο ως εξής:

Πρώτα, για το νέο καρέ υπολογίζεται το απόλυτο της διαφοράς μεταξύ αυτού με το υπολογισμένο Gaussian μέσο όρο, $|\Delta_t|$. Για την κατωφλίωση της εικόνας, ως τιμή κατωφλίου χρησιμοποιείται η τιμή τις τυπικής απόκλισης πολλαπλασιασμένη με αν συντελεστή K, όπου K ακέραιος και μεγαλύτερος του 0. Ο συντελεστής K υπολογίζεται πειραματικά κατά περίπτωση (Piccardi, 2004 ).

Η τεχνική αυτή, είναι λιγότερο ευάλωτη στον θερμικό θόρυβο της κάμερας, και μπορεί να αποδώσει θεμιτά αποτελέσματα και υπό την ύπαρξη ελαφριά κινούμενων απαλών σκιών στην εικόνα. Η τεχνική αυτή χρησιμοποιήθηκε στο πειραματικό μέρος της παρούσας εργασίας. Στο κεφάλαιο του πειραματικού μέρους θα παρουσιαστεί η υλοποίησή της.

Αξίζει να αναφερθεί πως σε κάποιες περιπτώσεις πριν το στάδιο μοντελοποίησης του φόντου στην εικόνα εφαρμόζεται ένα φίλτρο μέσου (median). Η εφαρμογή του φίλτρου έχει ως αποτέλεσμα την αισθητή μείωση θορύβου στην εικόνα αφού εξαλείφει τις απομονωμένες ακραίες τιμές. Επίσης σε εικόνες χαμηλής ευκρίνειας η ποιότητα της εικόνας εξομαλύνεται καθιστώντας την εικόνα, συνήθως, καταλληλότερη για την αφαίρεση φόντου.

Εφόσον χρησιμοποιηθεί κάποια τεχνική για τον διαχωρισμό φόντου και προσκηνίου, στο επόμενο βήμα γίνεται η ανίχνευση των ανεξάρτητων στοιχείων του προσκηνίου και η ονομασία τους.

**Ανίχνευση / ονομασία κηλίδων (Blob)**

Η ανίχνευση των κηλίδων χρησιμοποιεί την δυαδική εικόνα η οποία προκύπτει από την κατωφλίωση του αποτελέσματος της διαδικασίας αφαίρεσης φόντου. Ο ρόλος της διαδικασίας αυτής είναι η ονομασία των απομονωμένων κηλίδων στην εικόνα έτσι ώστε να μπορούν να χρησιμοποιηθούν ανεξάρτητα. Οι πληροφορίες που θα παρουσιαστούν στο κεφάλαιο αυτό προέρχονται από τον πηγαίο κώδικα του αντικειμένου [cv.jit.label] το οποίο χρησιμοποιείται στο περιβάλλον της Max για τον σκοπό αυτό, και αποτελεί μία προσέγγιση στην ονομασία κηλίδων.

Η διαδικασία αποτελείται από δύο επιμέρους διαδικασίες, τις Search_for_blobs() και fill_blob().

Αρχικά καλείται η διαδικασία Search_for_blobs(). Η διαδικασία αυτή κάνει χρήση δύο πινάκων, του πίνακα εισόδου ο οποίος αποτελείται από τις τιμές του εισερχόμενου καρέ, και του πίνακα εξόδου ο οποίος θα επιστραφεί στην έξοδο της διαδικασίας.

Ξεκινώντας από το πρώτο εικονοστοιχείο του καρέ (πάνω αριστερά) ελέγχει αν η τιμή του εικονοστοιχείου είναι διάφορη του μηδενός στους πίνακες εισόδου και εξόδου. Αν βρεθεί ένα εικονοστοιχείο με τιμή 1 στον πίνακα εισόδου και 0 στον πίνακα εξόδου, σημαίνει πως ανήκει σε μία κηλίδα η οποία παραμένει χωρίς όνομα. Τότε ξεκινάει η διαδικασία fill_blob(). Η διαδικασία αυτή, αρχικά θέτει το αντίστοιχο εικονοστοιχείο στον πίνακα εξόδου με ένα νούμερο το οποίο

αντιστοιχεί στο όνομα της κηλίδας. Έπειτα ελέγχει κάθε γειτονικό εικονοστοιχείο, και όποιο έχει τιμή 1 θέτει αντίστοιχα το εικονοστοιχείο στον πίνακα εξόδου με το ίδιο νούμερο / όνομα κηλίδας. Η διαδικασία αυτή συνεχίζει να ελέγχει όλα τα γειτονικά εικονοστοιχεία από κάθε ένα το οποίο έχει τιμή 1, μέχρι να μη βρεθεί κανένα άλλο με τιμή 1. Τότε, καλείται η ξανά διαδικασία Search_for_blobs().

Αν η Search_for_blobs() βρει ένα εικονοστοιχείο στον πίνακα εισόδου με τιμή 1, και το αντίστοιχο εικονοστοιχείο στον πίνακα εξόδου έχει τιμή διάφορη του μηδενός, σημαίνει πως ανήκει σε κάποια ονομασμένη κηλίδα και προχωράει στο επόμενο εικονοστοιχείο.

Και οι δύο διαδικασίες διαδέχονται η μία την άλλη μέχρι να ελεγχθεί και το τελευταίο εικονοστοιχείο (κάτω δεξιά) από την Search_for_blobs().

Ακολουθεί ένα μικρό τμήμα του κώδικα σε μορφή ψευδοκώδικα:

```
PRODECURE Search_for_blobs()
 FOR i=0; i < height; i++
   FOR j=0; j< width; j++
     IF (cout[j]=0) AND (cin[j] <> 0) THEN       //Βρήκαμε την αρχή ενός νέου blob

          ndx += 1;                              //Αύξηση του μετρητή των blobs κατά 1

       Blob[ndx].index := ndx;                    //Ενημέρωση του Α/Α του Blob στον πίνακα
                                                  //πληροφοριών για τα εντοπιζόμενα blobs

          Blob[ndx].size = fill_blobs();          //Καλούμε την διαδικασία επεξεργασίας του
                                                  //εντοπισθέντος blob, η οποία επιστρέφει το
                                                  //μέγεθος του blob (το πλήθος των pixels
                                                  //που αποτελούν το blob)

   END_IF
 END_FOR

 Cout += step;                                    // Προχωρούμε το cout και το cin κατά μία
```

```
 Cin += step;                          // οριζόντια γραμμή μπροστά (στην επόμενη row)

END_FOR
END_Search_for_blobs;
```

Στο ΠΑΡΑΡΤΗΜΑ Α παρουσιάζεται ο πλήρης αλγόριθμος της διαδικασίας σε μορφή ψευδοκώδικα (Pelletier).

Εφόσον επιτευχθεί η ονομασία των κηλίδων, μπορούν πλέον να πραγματοποιηθούν μετρήσεις διαφόρων παραμέτρων σε κάθε κηλίδα ανεξάρτητα. Παρακάτω παρουσιάζονται οι παράμετροι που μετρήθηκαν κατά την εκπόνηση της εργασίας.

**Ανάλυση κηλίδων**

Οι παράμετροι των κηλίδων προέρχονται είτε άμεσα από μετρήσεις ακατέργαστων τιμών (raw data), ή έμμεσα από την ανάλυση των ακατέργαστων τιμών.

Οι παράμετροι των ακατέργαστων τιμών περιέχουν:

- Μάζα κηλίδας– blob mass
- Σιλουέτα κηλίδας – blob silhouette
- Περιβάλλον ορθογώνιο – bounding rectangle
- Ύψος και πλάτος του κηλίδας –blob height & width

Ενώ παράμετροι από την ανάλυση ή συνδυασμό των παραπάνω είναι οι εξής:

- Ποσότητα κίνησης
- Δείκτης συστολής
- Κέντρο βάρους
- Δείκτης κατευθυντικότητας

Ακολουθεί μια αναφορά σε κάθε μία από τις παραπάνω παραμέτρους.

**Μάζα κηλίδας – blob mass**

Η επιφάνεια της κηλίδας υπολογίζεται ως το πλήθος των εικονοστοιχείων τα οποία αποτελούν την κηλίδα την δεδομένη χρονική στιγμή.

**Σιλουέτα κηλίδας – blob silhouette**

Η σιλουέτα της κηλίδας αποτελείται από τις συντεταγμένες των εικονοστοιχείων της κηλίδας. Ουσιαστικά, απεικονίζει το σχήμα του blob (Camurri, Mazzarino, & Volpe, 2003).

**Περιβάλλον ορθογώνιο – bounding rectangle**

Το περιβάλλον ορθογώνιο, είναι το μικρότερο ορθογώνιο το οποίο περιβάλει την κηλίδα. Υπολογίζεται από τις συντεταγμένες των τεσσάρων ακραίων εικονοστοιχείων στα κάθετα και οριζόντια όρια. Προβάλλεται ως δύο ζεύγη τιμών συντεταγμένων (x,y) τα οποία αφορούν τα πάνω-δεξιά και κάτω-αριστερά σημεία του ορθογωνίου. Οι τέσσερεις τιμές επί της ουσίας αφορούν την ανώτερη και κατώτερη τιμή συντεταγμένων στον άξονα x, και αντίστοιχα στον άξονα y (Camurri, Mazzarino, & Volpe, 2003).

**Ύψος και πλάτος του κηλίδας**

Οι τιμές του ύψους και του πλάτους υπολογίζονται από τις ίδιες τιμές που χρησιμοποιούνται για το περιβάλλον ορθογώνιο. Το ύψος υπολογίζεται ως η απόλυτη διαφορά μεταξύ της μικρότερη και μεγαλύτερης τιμή συντεταγμένων στον άξονα y, ενώ το πλάτος υπολογίζεται ως η απόλυτη διαφορά μεταξύ της μικρότερης και μεγαλύτερης τιμής συντεταγμένων στον άξονα x.

**Ποσότητα κίνησης – Quantity of motion**

Η ποσότητα κίνησης υπολογίζεται χρησιμοποιώντας την σιλουέτα της κηλίδας. Απαιτείται η χρήση της σιλουέτας $S_t$ και $S_{t-n}$, όπου $S_t$ η σιλουέτα της χρονική στιγμή tκαι $S_{t-n}$ η σιλουέτα την χρονική

στιγμή t-n. Η τιμή της ποσότητας της κίνησης ορίζεται ως το πλήθος των διαφορετικών συντεταγμένων μεταξύ των δύο σιλουετών. Η χρονική διαφορά, n, μεταξύ των επιλεγμένων σιλουετών επηρεάζει την ευαισθησία στο αποτέλεσμα του υπολογισμού (Camurri, Mazzarino, & Volpe, 2003).

**Δείκτης συστολής**

Ο δείκτης συστολής υπολογίζεται από την διαίρεση της μάζας της κηλίδας προς το εμβαδόν του περιβάλλοντος ορθογωνίου της. Η τιμή του δείκτη συστολής ορίζεται μεταξύ των τιμών 0 και 1 και υποδεικνύει τον τρόπο που η κηλίδα καταλαμβάνει το γύρο χώρο. Η τιμές κοντά στην μονάδα υποδεικνύουν μια συμπαγή μορφή, ενώ τιμές που προσεγγίζουν το μηδέν υποδεικνύουν αραιή κατάληψη του γύρο χώρου (Camurri, Mazzarino, & Volpe, 2003).

**Κέντρο βάρους - Baricenter**

Το κέντρο βάρους υπολογίζεται από την σιλουέτα της κηλίδας. Ένας αλγόριθμος για τον υπολογισμό του κέντρου βάρους αναφέρεται στο "Παράρτημα Α – cv.jit.centroids" (Pelletier).

**Δείκτης κατευθυντικότητας – Directivity index**

Ο δείκτης κατευθυντικότητας υποδεικνύει το πόσο η τροχιά του κέντρου βάρους μιας κηλίδας προσεγγίζει την ευθεία. Ο υπολογισμός του δείκτη χρησιμοποιεί εκτός των τρεχόντων τιμών συντεταγμένων και ένα πλήθος παρελθοντικών. Το πλήθος των παρελθοντικών τιμών καθορίζεται ανά περίπτωση εμπειρικά, σύμφωνα με τον ρυθμό μεταβολής της πορείας του κέντρου βάρους. Οι τιμές του δείκτη ορίζονται μεταξύ των τιμών 0 και 1. Όσο περισσότερο ο δείκτης κατευθυντικότητας προσεγγίζει την μονάδα, τόσο η τροχιά του κέντρου βάρους προσεγγίζει την ευθεία (Camurri, Mazzarino, & Volpe, 2003).

Οι παραπάνω παράμετροι είναι κάποιες από μια πληθώρα μετρήσιμων παραμέτρων που χρησιμοποιούνται στην μηχανική όραση και υπολογιστική όραση.

Μέχρι το σημείο αυτό, έχουν αναφερθεί τα απαραίτητα στοιχεία θεωρίας τα οποία χρησιμοποιήθηκαν στην διεξαγωγή του πειραματικού μέρους από πλευράς επεξεργασίας και διαχείρισης εικόνας. Ακολουθεί ένα υποκεφάλαιο το οποίο παρουσιάζει ορισμένες τεχνικές σύνθεσης ήχου, κάποιες εκ των οποίων χρησιμοποιήθηκαν επίσης κατά το πειραματικό μέρος.

## 2.3. Τεχνικές σύνθεσης ήχου

Στον ψηφιακό ήχο, κάποιες από τις βασικότερες και συνηθέστερες τεχνικές σύνθεσης ήχου είναι οι παρακάτω:

- Προσθετική και αφαιρετική σύνθεση
- Διαμόρφωση πλάτους και συχνότητας
- Προσέγγιση φυσικού μοντέλου (physical modeling).
- Sampling & Re-sampling
- Fast Fourier Transform

**Προσθετική και αφαιρετική σύνθεση**

Η λογική στην προσθετική σύνθεση είναι η εξής. Ο τελικός ήχος δημιουργείται από την πρόσθεση των εξόδων πολλών ταλαντωτών (γεννητριών ημιτονοειδών). Κάθε ταλαντωτής έχει την δική του συχνότητα, το δικό του πλάτος και την δική του χρονική εξέλιξη. Ένα παράδειγμα χρήσης της τεχνικής αυτής είναι οι "Καμπάνες" του Risset, ο οποίος με 11 ημιτονοειδείς γεννήτριες προσομοίωσε τον ήχο του χτύπου μιας καμπάνας.

Η αφαιρετική σύνθεση όπως προδίδει και το όνομά της, στηρίζεται στην αφαίρεση στοιχείων από ένα ήχο. Η τεχνική επιτυγχάνεται κυρίως με την χρήση φίλτρων. Οποιοσδήποτε ήχος μπορεί να χρησιμοποιηθεί ως αρχικό υλικό στην αφαιρετική σύνθεση, όμως τα καλύτερα αποτελέσματα συμβαίνουν όταν χρησιμοποιείται πλούσιο, φασματικά, ηχητικό υλικό. (Dodge & Jerse, 1997 )

**Απλή διαμόρφωση πλάτους και συχνότητας**

Διαμόρφωση πλάτους ονομάζεται η τεχνική σύνθεσης κατά την οποία, το πλάτος ενός ταλαντωτή, διαμορφώνεται από έναν δεύτερο ταλαντωτή. Κάθε ένας από αυτούς, έχει το δικό τους πλάτος, και συχνότητα. Ο πρώτος ονομάζεται "φορέας" και ο δεύτερος "διαμορφωτής". Η έξοδος του διαμορφωτή προστίθεται με την τιμή πλάτους τους φορέα. Η διαμόρφωση πλάτους, στην περίπτωση χρήσης απλών ημιτονοειδών σημάτων, έχει ως αποτέλεσμα, στην έξοδο του ταλαντωτή φορέα, την ύπαρξη τριών συχνοτήτων. Η μία είναι η συχνότητα του ίδιου του ταλαντωτή, fc, και οι δύο νέες είναι fc+fm και fc-fm, όπου fm είναι η συχνότητα του ταλαντωτή διαμορφωτή. Οι δύο νέες συχνότητες εμφανίζονται έχοντας μισό πλάτος από εκείνο του ταλαντωτή διαμορφωτή. Στην περίπτωση χρήσης πολυπλοκότερων αρμονικά σημάτων, με πολλές αρμονικές συχνότητες σε κάνε ημιτονοειδή ταλαντωτή, το αποτέλεσμα θα είναι σαφώς πλουσιότερο αρμονικά

Ομοίως με την διαμόρφωση πλάτους, η διαμόρφωση συχνότητας χρησιμοποιεί δύο ταλαντωτές, ένα φορέα και ένα διαμορφωτή. Η συνδεσμολογία μεταξύ των ταλαντωτών είναι παρόμοια με την συνδεσμολογία στην διαμόρφωση πλάτους, με την διαφορά πως ο διαμορφωτής πολλαπλασιάζεται με την τιμή της συχνότητας του φορέα. Με την χρήση απλών ημιτονοειδών ως συναρτήσεις των ταλαντωτών, το αποτέλεσμα στην έξοδο είναι συχνότητες από fc+d, ως fc-d, όπου fc η συχνότητα του ταλαντωτή φορέα και d το πλάτος του διαμορφωτή. Η διαμόρφωση συχνότητας έχει ως τεχνική την δυνατότητα παραγωγής ιδιαίτερα πλούσιου συχνοτικού φάσματος με χρήση απλών ημιτονοειδών ταλαντώσεων (Dodge & Jerse, 1997 ).

Όπως στην προσθετική και αφαιρετική σύνθεση, έτσι και στην διαμόρφωση συχνότητας και πλάτους, η παραγωγή ήχων με σταθερές τιμές ελέγχου καταλήγει σε σταθερούς αφύσικους ήχους. Με τον δυναμικό έλεγχο των παραμέτρων εισόδου των ταλαντωτών, μπορούν να επιτευχθούν ενδιαφέροντες και εξελισσόμενοι ήχοι.

Οι παραπάνω τεχνικές είναι μερικές από ένα πλήθος διαθέσιμων τεχνικών σύνθεσης. Μερικές από τις πιο σύνθετες τεχνικές περιέχουν ανάλυση φάσματος και συναρτήσεις πολυωνύμων. Πολύ μεγάλο ενδιαφέρον παρουσιάζει η προσέγγιση φυσικών μοντέλων ήχου, όπου ο αλγόριθμος προσομοιώνει την πορεία του ήχου μέσα από ένα φυσικό αντηχείο (resonator), επιτυγχάνοντας την προσομοίωση κρουστών ήχων, πνευστών οργάνων μέχρι και της ανθρώπινης φωνής.

**Karplus-Strong – Φυσικό μοντέλο νυκτής χορδής**

Ένας από τους πιο γνωστούς και βασικούς αλγορίθμους προσομοίωσης φυσικών μοντέλων είναι η προσομοίωση νυκτής χορδής.

Η προσομοίωση της νυκτής χορδής απαιτεί μία γεννήτρια θορύβου, μια μονάδα καθυστέρησης και ένα χαμηλοπερατό φίλτρο. Το φίλτρο στην πρώτη έκδοση του αλγορίθμου αυτού ήταν ο υπολογισμός του μέσου όρου μεταξύ της παρούσας τιμής εξόδου και της αμέσως προηγούμενης. Μερικοί από τους περιορισμούς του αλγορίθμου αυτού αφορούν τις επιτρεπόμενες συχνότητες που μπορούν να παραχθούν. Η γεννήτρια θορύβου χρειάζεται να παράγει πλήθος τυχαίων τιμών ίσο με πλήθος θέσεων της μονάδας καθυστέρησης, όπου ισοδυναμεί με την διάρκεια της περιόδου του επιθυμητού τονικού ύψους. Ως ψηφιακό κύκλωμα η μονάδα καθυστέρησης μπορεί να εφαρμόσει καθυστέρηση σε τιμές ακεραίων πολλαπλάσιων μιας βασικής μονάδας. Σε χαμηλά τονικά ύψη με μεγάλη περίοδο, ο περιορισμός αυτός δεν επιφέρει μεγάλα προβλήματα, ενώ αντίθετα στα υψηλά τονικά ύψη, το κενό μεταξύ των διαθέσιμων συχνοτήτων είναι πολύ ευδιάκριτο και συνεπώς περιοριστικό. Για παράδειγμα μια χαμηλή συχνότητα 23hz με περίοδο 43,479 ms, χρειάζεται μία μονάδα καθυστέρησης 1917,4 samples (βασικές μονάδες ήχου), γεγονός αδύνατο. Η πλησιέστερη διαθέσιμη τιμή καθυστέρησης είναι 1917 samples, με αποτέλεσμα μια συχνότητα 23,0047hz. Η διαφορά αυτή δεν γίνεται αντιληπτή από το ανθρώπινο αυτί. Το ίδιο πρόβλημα όμως σε μια υψηλή συχνότητα δεν έχει τα ίδια αποτελέσματα. Για παράδειγμα αν η επιθυμητή συχνότητα είναι 10000Hz, με περίοδο 0.1ms, η απαιτούμενη καθυστέρηση είναι 4,41 samples. Στην περίπτωση αυτή η πλησιέστερη ακέραια τιμή είναι τα 4 samples, με αποτέλεσμα μια συχνότητα 11025Hz, ή επιλέγοντας τα 5 samples, μια συχνότητα 8820Hz. Σαφώς η διαφορά των +1025Hz ή των -1180Hz είναι ξεκάθαρη στο αυτί.

Μία λύση για το πρόβλημα αυτό είναι η αύξηση του ρυθμού δειγματοληψίας. Τα παραπάνω παραδείγματα υπολογίστηκαν με ρυθμό δειγματοληψίας 44.1 kHz (Karplus & Strong, 1983 ).

**Sampling&Re-sampling**

Η τεχνική αυτή αφορά την χρήση προ-ηχογραφιμένων ήχων, είτε την ζωντανή ηχογράφηση τους και μετέπειτα αναπαραγωγή τους. Στηρίζεται στους διάφορους τρόπους επεξεργασίας του προϋπάρχοντος ήχου, είτε σε πραγματικό χρόνο, είτε σε επεξεργασία που μεσολαβεί μεταξύ της ηχογράφησης και της νέας αναπαραγωγής του. Στην εφαρμογή της τεχνικής με αναλογικά μέσα, μια από τις πιο βασικές επεξεργασίες του ήχου είναι η τμημάτιση του και αναπαραγωγή των

τμημάτων σε διαφορετική σειρά με την χρήση των “mellotrons”. Η έννοια αντιπροσωπεύει ένα τύπο οργάνου το οποίο χρησιμοποιούσε μαγνητικές ταινίες μικρού μήκους για την αναπαραγωγή μικρών τμημάτων από κάποιο ηχογραφημένο ήχο. Η βασική ιδέα είναι η σύνδεση μικρών τμημάτων ενός μεγαλύτερου ήχου, σε κάθε ένα από τα κλειδιά ενός πληκτρολογίου (πιάνο). Η εναλλαγή τονικότητας αναπαραγωγής ήταν και παραμένει στοιχείο της τεχνικής αυτής. Αρχικά επιτυγχανόταν με την αλλαγή της ταχύτητας αναπαραγωγής στα κασετόφωνα, επηρεάζοντας μοιραία την διάρκεια του ήχου. Με την χρήση ψηφιακών μέσων ανάλυσης φάσματος, η διάρκεια ανεξαρτητοποιείται από την τονικότητα (Russ, 1996 ).

**Fast Fourier Transform / ΜετασχηματισμόςFourier**

Ο μετασχηματισμός Fourier, μεταφέρει το ηχητικό σήμα από το πεδίου του χρόνου στο πεδίο συχνοτήτων και χρησιμοποιείται ευρέως για φασματική ανάλυση. Η λεπτομέρεια που μπορεί να ληφθεί από ένα δείγμα ήχου, είναι αντιστρόφως ανάλογη προς το μέγεθος του δείγματος ήχου. Έτσι, χρησιμοποιώντας μικρά τμήματα ήχου ο μετασχηματισμός θα επιστρέψει χοντροκομμένα αποτελέσματα, ενώ σε μεγάλα δείγματα ήχου θα επιστραφούν λεπτομερή αποτελέσματα. Βέβαια, το μέγεθος του δείγματος αντιστοιχεί και στον χρόνο, καθυστέρηση, που απαιτείται για τον μετασχηματισμό. Έτσι, πιθανόν να απαιτούνται θυσίες, ιδιαιτέρως σε ζωντανού χρόνου χρήση της τεχνικής, όσον αφορά την λεπτομέρεια της συχνοτικής ανάλυσης. Ο μετασχηματισμός Fourier, μπορεί να χρησιμοποιήσει τμήματα ήχου τα οποία υπερκαλύπτονται μεταξύ τους ώστε να επιστρέψει μεγαλύτερης λεπτομέρειας αποτελέσματα (Russ, 1996 ).

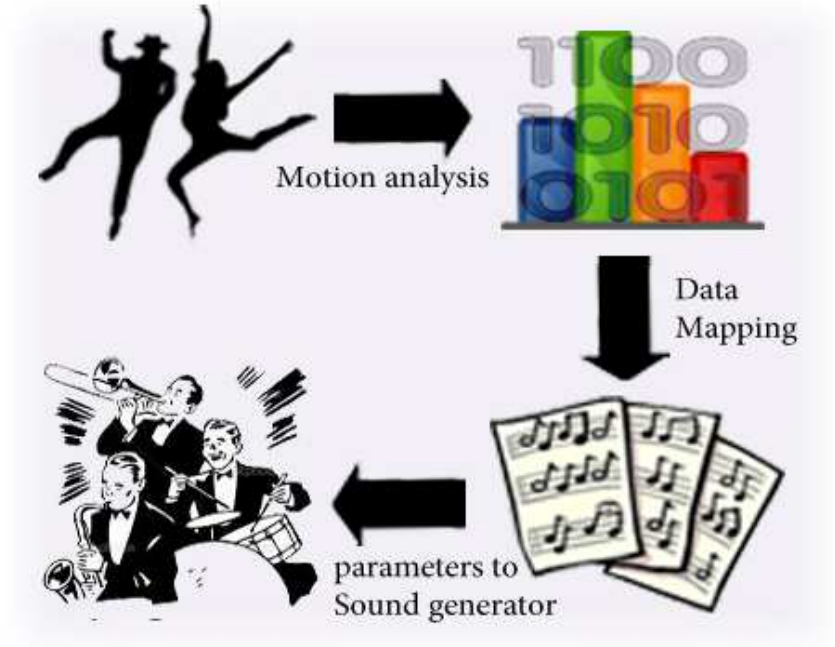
Στο επόμενο κεφάλαιο ακολουθεί η παρουσίαση του πειραματικού μέρους. Εντός του κεφαλαίου αυτού, θα παρουσιαστεί το διαδραστικό σύστημα το οποίο δημιουργήθηκε και θα αναλυθεί από δύο σκοπιές. Αρχικά θα αναλυθεί ως προ του τρόπο με τον οποίο μπορεί να χρησιμοποιηθεί και τα απαραίτητα βήματα ώστε να λειτουργήσει, και θα ακολουθήσει μία τεχνική παρουσίαση των επιμέρους διαδικασιών του στο περιβάλλον της Max.

# 3. ΠΕΙΡΑΜΑΤΙΚΌ ΜΈΡΟΣ

Στο πειραματικό μέρος της εργασίας, υλοποιήθηκε η ανάπτυξη ενός ομαδικού διαδραστικού συστήματος. Ο σκοπός του συστήματος είναι η ελαχιστοποίηση του περιορισμού στην κίνηση των χρηστών, και η προώθηση της διάδρασης τόσο ανάμεσα στους μεμονωμένους χρήστες και τον υπολογιστή, όσο και μεταξύ των χρηστών. Η ανάπτυξη του συστήματος περιλαμβάνει το γραφικό περιβάλλον με το οποίο γίνεται εν μέρει ο χειρισμός από τον χρήστη, και την ανάπτυξη ορισμένων αλγορίθμων οι οποίοι συμβάλουν στην ανάλυση της εικόνας, με το όνομα “Bread”.

Πρόκειται για ένα διαδραστικό σύστημα το οποίο παράγει ήχο βάσει των κινήσεων ορισμένων εκτελεστών οι οποίοι βρίσκονται επί σκηνής. Ο ένας από αυτούς έχει τον ρόλο του χρήστη αναφοράς (πληροφορίες στο κεφάλαιο “1.3.1 Χρήστης Αναφοράς” ). Η διαφορά του χρήστη αναφοράς με τους υπόλοιπους εκτελεστές έγκειται κυρίως στην διαφορά ελέγχου. Οι υπόλοιποι εκτελεστές αναφέρονται ως “Instrument” διότι ελέγχουν την δημιουργία του ήχου, ενώ ο χρήστης αναφοράς ονομάζεται “Maestro” και επηρεάζει τον συνολικό ήχο με χρήση εφέ καθώς και την συνολική ένταση. Πιο συγκεκριμένα ο έλεγχος του χρήστη αναφορά προς τους υπόλοιπους θα αναφερθεί στα υποκεφάλαια “3.1.1.5” και “3.2.5”. Κατά τη θεωρητική σύλληψη του συστήματος, ο σχεδιασμός του προσανατολίστηκε προς ομάδα τριών ατόμων. Λόγω περιορισμών, όπως η έκταση της σκηνής, το σύστημα κατέληξε να υποστηρίζει μια ομάδα δύο ατόμων. Ο ένας εκ των δύο αντιστοιχεί στο “Instrument” και ο δεύτερος στο “Maestro”.

Οι κινήσεις των χρηστών εισέρχονται στο Bread με την χρήση μίας κάμερας, όπου αντιστοιχούνται στις διάφορες παραμέτρους της γεννήτριας ήχου. Στην εικόνα 9, φαίνεται η ροή της πληροφορίας στα διάφορα στάδια του Bread.




ΕΙΚΟΝΑ 9 ΡΟΗ ΔΕΔΟΜΕΝΩΝ

Αρχικά η κάμερα καταγράφει τις κινήσεις των δύο εκτελεστών και τις εισάγει ως δυσδιάστατη εικόνα στο περιβάλλον της εφαρμογής. Στην συνέχεια, για κάθε χρήστη υπολογίζονται κάποιες παράμετροι ξεχωριστά. Οι παράμετροι εισάγονται στην γεννήτρια ήχου και τέλος δημιουργείται και διαμορφώνεται ο ήχος.

Η πρώτη διαδικασία συμπεριλαμβάνει τον υπολογισμό των παραμέτρων από την κίνηση των χρηστών. Για την πραγματοποίηση της διαδικασίας αυτής χρειάζονται τα εξής βήματα:

- Λήψη / εισαγωγή βίντεο στην εφαρμογή
- Απομόνωση σιλουέτων (κηλίδων) των χρηστών από την υπόλοιπη εικόνα
- Μέτρηση παραμέτρων της κίνησης του χρήστη.

Το αποτέλεσμα της παραπάνω διαδικασίας είναι η μέτρηση ορισμένων παραμέτρων των κηλίδων, τις οποίες θεωρούμε αντιπροσωπευτικές της κίνηση των εκτελεστών. Συγκεκριμένα οι παράμετροι αυτοί είναι:

1. Κέντρο βάρους - Baricenter

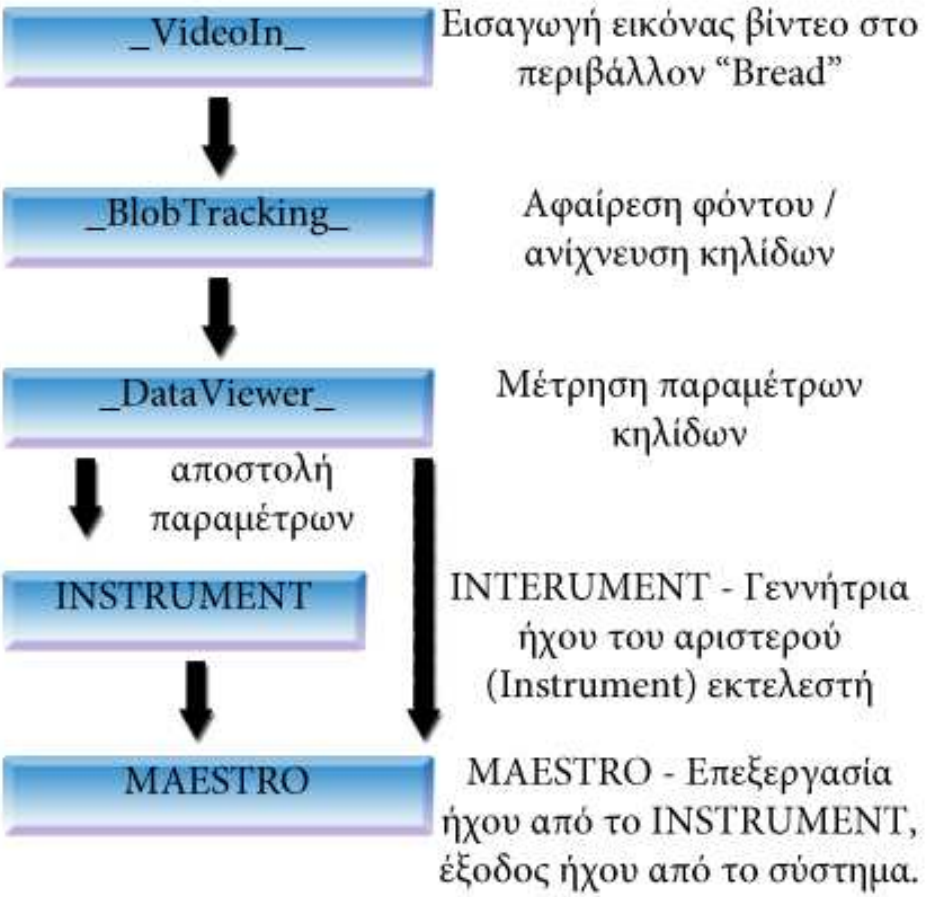
2. Δείκτης κατευθυντικότητας – Directivity index
3. Δείκτης συστολής
4. Ποσότητα κίνησης – Quantity of motion
5. Ύψος και πλάτος του κηλίδας
6. Περιβάλλον ορθογώνιο – bounding rectangle

Για πληροφορίες όσον αφορά τις παραμέτρους, ο αναγνώστης μπορεί να ανατρέξει στο κεφάλαιο "2.2.3. Αφαίρεση φόντου, ανίχνευση και ανάλυση"blob" / ανάλυση κηλίδων".

Εντός της εφαρμογής Bread οι παραπάνω διαδικασίες υλοποιούνται στα εξής υποπρογράμματα:

Λήψη / εισαγωγή βίντεο στην εφαρμογή ➔ VideoIn
Απομόνωση σιλουέτων (κηλίδων) των χρηστών ➔ BlobTracking
Μέτρηση παραμέτρων της κίνησης του χρήστη. ➔ DataViewer

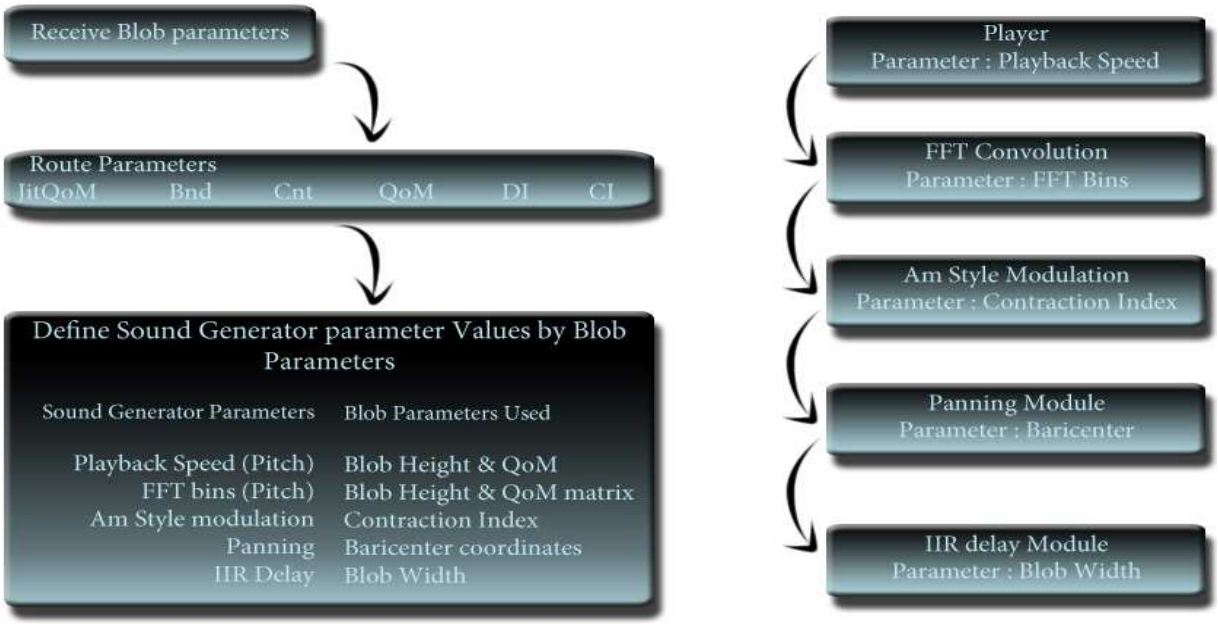
Για την τεχνική αφαίρεσης φόντου, οι πληροφορίες υπάρχουν στο κεφάλαιο "2.2.3. Αφαίρεση Φόντου". Εντός του κεφαλαίο 2.1.1 υπάρχουν όλες οι απαραίτητες πληροφορίες για το θεωρητικό υπόβαθρο και τα βήματα για την ανάλυση του βίντεο, συμπεριλαμβανομένου και των τεχνικών που χρησιμοποιήθηκαν στην υλοποίηση του πειραματικού μέρους της παρούσας εργασίας. Στην εικόνα 10 φαίνεται το διάγραμμα ροής των υποπρογραμμάτων που χρησιμοποιήθηκαν με την λειτουργία που εκτελούν.




ΕΙΚΟΝΑ 10 ΓΕΝΙΚΟ ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ

Εφόσον οι παράμετροι για κάθε χρήστη υπολογιστούν στη συνέχεια στέλνονται στο αντίστοιχο υποπρόγραμμα. Στο υποπρόγραμμα "INSTRUMENT" στέλνονται οι μετρήσεις από τον ένα εκτελεστή, και στο υποπρόγραμμα "MAESTRO" οι μετρήσεις από τον χρήστη αναφοράς. Η εφαρμογή Bread είναι ρυθμισμένη ώστε να αναγνωρίζει ως χρήστη αναφοράς, το άτομο που στέκεται δεξιά στην σκηνή, από την οπτική του παρατηρητή.

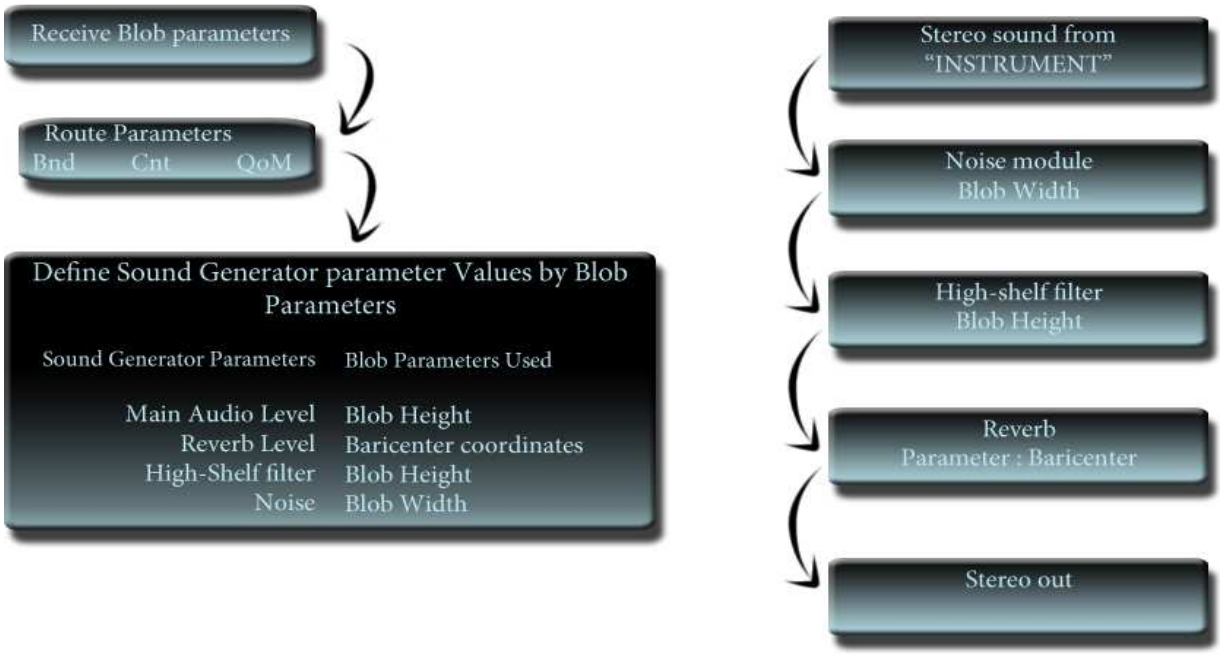
Παρακάτω (εικόνα 11) φαίνεται το διάγραμμα ροής των παραμέτρων στο υποπρόγραμμα "INSTRUMENT".




ΕΙΚΟΝΑ 11 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΤΟΥ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "INSTRUMENT"

Όπως φαίνεται στο παραπάνω διάγραμμα, εντός του υποπρογράμματος πραγματοποιούνται δύο διαδικασίες. Η πρώτη (αριστερά) δέχεται τις παραμέτρους της κηλίδας και κάνει την αντιστοίχηση παραμέτρων (mapping). Η άλλη διαδικασία (αριστερά) που φαίνεται στο διάγραμμα, αφορά τους παράγοντες της γεννήτριας ήχου του “INSTRUMENT” και σε κάθε παράγονται φαίνεται ποια παράμετρος της κηλίδας χρησιμοποιείται.

Επόμενο φαίνεται το διάγραμμα ροής του υποπρογράμματος “MAESTRO” (σχήμα 12).




ΕΙΚΟΝΑ 12 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ "MAESTRO"

Ομοίως με το πρώτο διάγραμμα, το διάγραμμα ροής του υποπρογράμματος “MAESTRO” δείχνει τις διαδικασίες της αντιστοίχησης και γεννήτριας ήχου. Η αντιστοίχηση των παραμέτρων της κηλίδας στο υποπρόγραμμα αυτό είναι μικρότερης κλίμακας, εφόσον δεν παράγεται νέος ήχος, αλλά γίνεται μόνο μια μικρή επεξεργασία στον δημιουργημένο ήχο από τον υποπρόγραμμα “INSTRUMENT”.

Τα παραπάνω διαγράμματα απεικονίζουν την γενική πορεία των δεδομένων εντός των υποπρογραμμάτων. Παρακάτω θα ακολουθήσει η επεξήγηση των υποπρογραμμάτων ώστε να μπορούν να χρησιμοποιηθούν από κάποιο πρόσωπο σωστά. Η επεξήγηση των υποπρογραμμάτων θα γίνει από δύο σκοπιές. Αρχικά θα παρουσιαστεί ο τρόπος χρήσης των υποπρογραμμάτων, οι παράμετροι που μπορούν να ελεγχθούν στο κάθε ένα καθώς και τα βήματα που απαιτούνται για το στήσιμο και την λειτουργία του Bread. Έπειτα θα παρουσιαστεί αναλυτικά ο τρόπος υλοποίησης των επιμέρους διαδικασιών οι οποίες εκτελούνται στης εφαρμογή Bread.

## 3.1. Περιγραφή για τον χρήστη και οδηγίες χρήσης

Η εφαρμογή αποτελείται από πέντε ξεχωριστά υποπρογράμματα. Τα τρία, _VideoIn_, _BlobTracking_, _DataViewer_, είναι υπεύθυνα για την εισαγωγή βίντεο, αναγνώριση των blob και μέτρηση των χαρακτηριστικών τους. Τα υπόλοιπα δύο, _Instrument_ και _Maestro_, υλοποιούν την αντιστοίχηση των παραμέτρων των κηλίδων με τις παραμέτρους την γεννήτριας ήχου και την δημιουργία του ηχητικού αποτελέσματος.

**Μια γρήγορη ματιά**

Συγκεκριμένα,

- στο _VideoIn_, γίνεται η εισαγωγή της εικόνας από την βιντεοκάμερα στο περιβάλλον της εφαρμογής.
- Το _BlobTracking_ είναι υπεύθυνο για την αφαίρεση φόντου και ανίχνευση των blob.
- Το _DataViewer_ υλοποιεί τις μετρήσεις των παραμέτρων των blob.
- Το _Instrument_ διαχειρίζεται τις διασυνδέσεις παραμέτρων και την δημιουργία ήχου βάσει ενός εκτελεστή.
- Το _Maestro_ διαχειρίζεται τις διασυνδέσεις των παραμέτρων με την ηχητική του αλλοίωση βάσει του χρήστη αναφοράς.

**Βήματα εκκίνησης**

Αρχικά ο χρήστης χρειάζεται να τοποθετήσει την κάμερα σε ένα σταθερό σημείο, τρίποδο κατά προτίμηση, απέναντι από τον χώρο κίνησης των εκτελεστών. Η κάμερα χρειάζεται να συνδεθεί με τον υπολογιστή είτε μέσω θύρας USB ή FireWire. Επίσης η τοποθέτηση των ηχείων έτσι ώστε να αποδίδουν στεροσκοπικά. Εφόσον τα ηχεία και η κάρτα ήχου συνδεθούν με τον υπολογιστή, πρέπει να ανοίξουν τα εξής υποπρογράμματα :

- _VideoIn_
- _BlobTracking_
- _DataViewer_
- INSTRUMENT

- MAESTRO

Στο σημείο αυτό, τα παρακάτω βήματα που παρουσιάζονται πιθανόν να μην μπορούν να ακολουθηθούν χωρίς να ανατρέξουμε παρακάτω, εντός του υποκεφαλαίου, στις εξηγήσεις της λειτουργίας του κάθε υποπρογράμματος.

Για την σωστή χρήση, απαιτείται η όσο το δυνατόν καλύτερη αφαίρεση φόντου. Σε μια περίπτωση όπως την δική μας, η διαδικασία αυτή δε συναντάει κάποια δυσκολία, υπό τις εξής συνθήκες.

- Κατά την δειγματοληψία του φόντου, να μην υπάρχει στην σκηνή τίποτα που δε θεωρείται φόντο
- Προτείνεται σταθερός και επαρκής διάχυτος φωτισμός, για όσο το δυνατόν λιγότερες σκιές.

Εφόσον φωτιστεί καλά ο χώρος, και αδειάσει η σκηνή, πρέπει να γίνει η επιλογή της κάμερας που επιθυμούμε από το _VideoIn_.

Στην συνέχεια, γίνεται η δειγματοληψία του φόντου με το πλήκτρο “capture” στο υποπρόγραμμα _BlobTracking_.

Από το σημείο αυτό και μετά, χρειάζεται μόνο η επιλογή του “Data Visualization is “Open”” και η διαδικασίες ανίχνευσης και ανάλυσης των blob λειτουργούν. Σε κάθε ένα βήμα από τα παραπάνω αναφέρονται οι απολύτως απαραίτητες επιλογές για την λειτουργία της εφαρμογής, ενώ υπάρχουν περισσότερες επιλογές σε κάθε σημείο, οι οποίες στοχεύουν στην βελτιστοποίησή της.

Τα δύο υποπρογράμματα τα οποία δημιουργούν τον ήχο και τον διαχειρίζονται, έχουν περισσότερο παθητικό χαρακτήρα στον χειρισμό τους. Στο “Instrument” είναι απαραίτητη η επιλογή και φόρτωση ενός ήχου-πηγή, ενώ στο “Maestro” η μόνη απαραίτητη επιλογή είναι η ενεργοποίηση και απενεργοποίηση της δημιουργίας του ήχου (DSP). Και στα δύο υποπρογράμματα ο έλεγχος από τον χρήστη δεν έχει καμία επαφή με την τροποποίηση παραμέτρων στην αντιστοίχηση ή αλλαγή συντελεστών στην γεννήτρια του ήχου. Ο έλεγχος αυτών των διαδικασιών επιτυγχάνεται μέσω της ανάλυσης του βίντεο και μόνο.

Χρειάζεται να αναφέρουμε πως ο οραματισμός της εφαρμογής περιείχε τρία άτομα επί σκηνής, ενώ λόγο χωρικού περιορισμού η εφαρμογή υλοποιήθηκε προβλέποντας για δύο άτομα, ένα εκτελεστή

και ένα άτομο αναφοράς. Η λειτουργία με τρία άτομα είναι δυνατή σε κατάλληλο χώρο, χωρίς όμως εγγυημένα αποτελέσματα.

Ακολουθεί η παρουσίαση των παραμέτρων των υποπρογραμμάτων και ο τρόπος χρήσης τους.

### 3.1.1. Περιγραφή περιβάλλοντος για τον χρήστη

Στο παρόν υποκεφάλαιο παρουσιάζονται τα υποπρογράμματα με την σειρά που συναντώνται στην ροή των δεδομένων.

#### 3.1.1.1.VideoIn

Το υποπρόγραμμα "_VideoIn_" αναλαμβάνει την εισαγωγή της εικόνας από την κάμερα στο περιβάλλον της εφαρμογής. Προσφέρεται η δυνατότητα εισαγωγής της εικόνας από την εκάστοτε βιντεοκάμερα ή η αναπαραγωγή προγεγραμμένου βίντεο. Η επιλογή προέλευσης της εικόνας, γίνεται από τα πλήκτρα "Camera" και "Playback".

**CAMERA**


ΕΙΚΟΝΑ 13 _VIDEOIN_ ΕΠΙΛΟΓΗ "CAMERA"

Στην περίπτωση που ο χρήστης επιλέξει να χρησιμοποιήσει βίντεο σε πραγματικό χρόνο, δηλαδή από τις δύο επιλογές επιλέγει το πλήκτρο “Camera”, έχει στην διάθεσή του τα εξής πλήκτρα για τον έλεγχο του βίντεο.

- Πλήκτρο “**Camera On / Off**” για την εκκίνηση ή διακοπή του βίντεο.
- Μενού “**Video Dimension**”, επιλογές μεταξύ των συνηθέστερων διαστάσεων για βίντεο.
- Μενού “**Input Devices List**”, περιέχει τις συνδεδεμένες, με τον υπολογιστή, συσκευές καταγραφής βίντεο.
- Μενού “**Digitizer List**”, περιέχει τα πρωτόκολλα ψηφιοποίησης/συμπίεσης της εικόνας της κάμερας.

Στο σημείο αυτό, ο έλεγχος όσον αφορά την εικόνα είναι ο απολύτως βασικός!

**PLAYBACK**


ΕΙΚΟΝΑ 14 _VIDEOIN_ - ΕΠΙΛΟΓΗ "PLAYBACK"

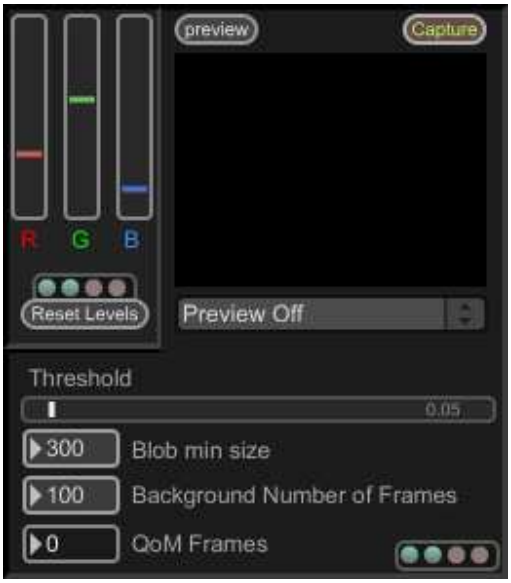
Για την αναπαραγωγή προγεγραμμένου αρχείου βίντεο, ο χρήστης έχει στην επιλογή του τα εξής πλήκτρα χειρισμού.

- “**LoadFile**”. Επιλέγει το αρχείο βίντεο για αναπαραγωγή
- “**Play**”. Εκκίνηση αναπαραγωγής.

- “**Pause**”. Προσωρινή διακοπή αναπαραγωγής. Σε περίπτωση επιλογής “Play” το βίντεο συνεχίζει από το σημείο στο οποίο σταμάτησε.
- “**Stop**”. Διακοπή του βίντεο. Σε περίπτωση επιλογής “Play” το βίντεο αρχίζει από την αρχή.
- “**to Start**”. Αν το βίντεο είναι σταματημένο, ο κέρσορας του βίντεο μεταφέρεται στην αρχή. Αν το βίντεο αναπαράγεται, ξεκινάει από την αρχή.
- “**to End**”. Ισχύει ότι και για το “toStart”, αλλά με τον κέρσορα να μεταφέρεται στο τελευταίο καρέ του βίντεο.
- “**Loop On/Off**”. Ενεργοποιεί ή απενεργοποιεί τη λειτουργία της κυκλικής αναπαραγωγής του βίντεο.
- “**Reset Loop**”. Θέτει τα σημεία αρχής και τέλους της κυκλικής αναπαραγωγής να είναι το πρώτο και το τελευταίο καρέ του βίντεο αντίστοιχα.
- **Loop slider**. Δεν υπάρχει πλήκτρο με το όνομα αυτό, αλλά αναφέρομαι στο slider που βρίσκεται κάτω από τα δύο προαναφερθέντα πλήκτρα. Η λειτουργία του είναι να ορίζει το αρχικό και το τελικό σημείο για την κυκλική αναπαραγωγή.
- “**Frame/Scrub**”. Το “numberbox” και το “slider” επιτρέπουν στον χρήστη να κυλίσει το βίντεο στο χρονικό σημείο που επιθυμεί.

Το πλήκτρο “Preview” είναι κοινό και στις δύο περιπτώσεις και η λειτουργία του είναι να ενεργοποιεί ή να απενεργοποιεί την προβολή του βίντεο που εισάγεται ή αναπαράγεται αντίστοιχα σε κάθε περίπτωση.

### 3.1.1.2.BlobTracking




ΕΙΚΟΝΑ 15 ΔΙΕΠΑΦΗ ΤΟΥ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "_BLOBTRACKING+"

Το υποπρόγραμμα "_BlobTracking_" αναλαμβάνει την απομόνωση της σιλουέτας των εκτελεστών από την υπόλοιπη εικόνα του βίντεο (φόντο), καθώς και τη βασική ανάλυσή της (παράμετροι 1ου βαθμού). Ο διαθέσιμος χειρισμός για τον χρήστη περιλαμβάνει τα παρακάτω.

- "**Preview**". Λειτουργεί για τον ίδιο σκοπό, όπως και στο υποπρόγραμμα "_VideoIn_".
- Μενού "**Preview Off**". Περιέχει επιλογές για διαφορετικούς τρόπους προεπισκόπησης του βίντεο στο σημείο αυτό.
- Πλήκτρο "**Preview Off**". Κλείνει την προεπισκόπηση.
    - "**Grayscale**". Φαίνεται το βίντεο μετά την μετατροπή του σε ασπρόμαυρο.
    - "**Foreground Image**". Ο χρήστης βλέπει τα απομονωμένα στοιχεία της εικόνας, που θεωρούνται προσκήνιο ("foreground").
    - "**Blobs with IDs**". Φαίνονται τα blobs, τα οποία διακρίνονται σύμφωνα με το χρώμα και το όνομα/νούμερό τους.
- **RGB**, (**Red**, **Green**, **Blue**) αναλογίες για μετατροπή σε ασπρόμαυρο.
  Τρία slider για τον έλεγχο της ισορροπίας μεταξύ των καναλιών RGB,που από δεξιά προς τα αριστερά είναι κόκκινο, πράσινο, μπλε.

- **Reset Levels**. Επιστρέφει τις αναλογίες στις αρχικές/προκαθορισμένες, που χρησιμοποιεί το περιβάλλον του Max/Msp.
- Slider "**Threshold**". Η τιμή του slider τίθεται ως τιμή κατωφλίου (threshold) για τη διαλογή μεταξύ των στοιχείων του φόντου και των στοιχείων προσκηνίου (foreground).
- "**Blob min size**" numberbox. Θέτει την τιμή, σύμφωνα με την οποία οι κηλίδες με μέγεθος μικρότερο από αυτήν απορρίπτονται.
- "**Background Number of Frames**". Ορίζει το πλήθος των καρέ από τα οποία υπολογίζεται το φόντο.
- "**QoM Frames**". Ορίζει το πλήθος των καρέ μεταξύ των οποίον υπολογίζεται η ποσότητα κίνησης (Quantity of Motion, QoM) των blobs.

### 3.1.1.3. Παράμετροι κηλίδων


ΕΙΚΟΝΑ 16 ΔΙΕΠΑΦΗ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "_DATAVIEWER_"

Χρειάζεται να σημειωθεί στο σημείο αυτό πως η σχεδίαση του παρόντος υποπρογράμματος προβλέπει την αναγνώριση τριών κηλίδων. Λόγω των δυσκολιών στην πραγματοποίηση του συστήματος με τρία άτομα, χρησιμοποιούνται μόνο οι δύο πρώτες κηλίδες. Στην παρακάτω παρουσίαση του υποπρογράμματος, η αναφορές για τρίτη κηλίδα ή “blob 3” μπορούν να παραληφθούν εφόσον δεν υποστηρίζεται η χρήση της.

Αφού διαχωριστεί ο κάθε εκτελεστής από το φόντο, μένει η ανάλυση της αντίστοιχης σιλουέτας και της κίνησής της. Κάποιες πρωτογενείς παράμετροι εξάγονται στο προηγούμενο υποπρόγραμμα και η διαδικασία αυτή ολοκληρώνεται στο υποπρόγραμμα “_DataViewer_”. Το υποπρόγραμμα αυτό αποτελείται από τρεις ίδιες επιφάνειες, κάθε μία για ένα από τα τρία blob που προβλέπονται στην εφαρμογή. Ο χρήστης έχει τα εξής πλήκτρα στην διάθεσή του.

Μενού δρομολόγησης blob.

- Menu “**Data Visualization is “Open/Closed**”. Επιτρέπει την διέλευση των δεδομένων από το προηγούμενο υποπρόγραμμα στο ισχύων.
- Menu “**Blob ID**”. Στο μενού αυτό ο χρήστης μπορεί να επιλέξει ποιό από τα ανιχνευμένα blobsθα πάρει τον ρόλο του “Blob 1”, “Blob 2” ή “Blob 3”. Τα Blobs 1&3, αντιστοιχούν σε πιθανά “instruments”,ενώ το blob 2 είναι ο χρήστης αναφοράς.

Η τιμή ποσότητας κίνησης μπορεί να αποδοθεί με δύο πιθανούς τρόπους.

- Τιμή“**Quantity of Motion in** “**Pixels/Percent**”

Η επιλογή του τρόπου γίνεται κάνοντας “click” επάνω στο numberbox το οποίο περιέχει την τιμή της ποσότητας της κίνησης.

Στην επιλογή **Pixels** η τιμή που παρουσιάζεται, αναφέρεται στο πλήθος των εικονοστοιχείων τα οποία κινούνται. Στην επιλογή **Percent**, η τιμή αφορά το πλήθος των εικονοστοιχείων ως ποσοστό επί τοις εκατό βάση μιας αρχικής τιμής. Αυτή η αρχική τιμή αντιπροσωπεύει το πλήθος των εικονοστοιχείων τα οποία αποτελούν την σιλουέτα του χρήστη, όταν εκείνος είναι σε στάση “ανάπαυσης”, **Starting Blob Size**. Ο λόγος απόδοσης της ποσότητας της κίνησης ως ποσοστό, είναι η διαφορές μεγέθους του σώματος μεταξύ των χρηστών. Με τη χρήση μιας τιμής αρχικού όγκου για κάθε χρήστη ξεχωριστά, επιτυγχάνεται μια ισορροπία μεταξύ μεγαλόσωμων και μικρόσωμων εκτελεστών.

- Τιμή “**Starting Blob Size**”.

Για τη σύλληψη της τιμής ο χρήστης καλείται να κάνει “click” επάνω στο numberbox της τιμής.

Μία ακόμα μέτρηση είναι ο δείκτης κατευθυντικότητας **DI** (**Directivity Index**). Για τον υπολογισμό της τιμής του δείκτη, απαιτείται ένα πλήθος καρέ, το οποίο δίνεται από τον χρήστη στο αντίστοιχο numberbox:

- Number Box “**Number of Frames for DI**”

Επίσης, όπως σε κάθε οπτική ανατροφοδότηση στο χρήστη:

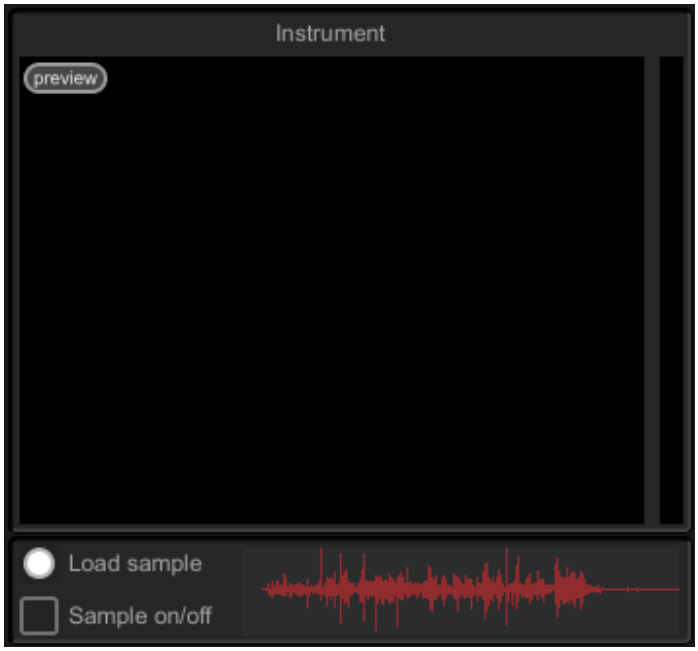
- Δύο πλήκτρα “**preview**”. Η λειτουργία τους είναι η ίδια με των προηγούμενων υποπρογραμμάτων.

Εκτός των πλήκτρων ελέγχου το συγκεκριμένο υποπρόγραμμα παρέχει και οπτική πληροφορία (visual feedback) για τις υπολογισμένες παραμέτρους.

- Στην πρώτη οθόνη (πάνω) φαίνονται τα εικονοστοιχεία “κίνησης”.
- Στην δεύτερη οθόνη (κάτω) φαίνονται
    - “**Bounding Rectangle**” (Περιβάλλον παραλληλόγραμμο)
    - **“Baricenter**” (Κέντρο Βάρους)
- Numberbox “**Contraction Index**”. Δείχνει την τιμή του “**δείκτη συστολής**”.

- Numberbox“**Baricenter Directivity**”. Εδώ φαίνεται ο βαθμός κατευθυντικότητας της μεταβολής των συντεταγμένων του κέντρου βάρους.

### 3.1.1.4.Instrument




ΕΙΚΟΝΑ 17 ΔΙΕΠΑΦΗ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "IMSTRUMENT"

Το υποπρόγραμμα “INSTRUMENT” υλοποιεί την αντιστοίχηση των παραμέτρων από την κίνηση του εκτελεστή με την γεννήτρια ήχου και πραγματοποιεί την δημιουργία του ήχου. Από τα δύο προβλεπόμενα άτομα στην σκηνή τον ο έλεγχος του “INSTRUMENT” γίνεται από το άτομο στα αριστερά ως προς τον ακροατή.

Ο έλεγχος που παρέχεται στον χρήστη μέσω του γραφικού περιβάλλοντος είναι:

- Επιλογή ήχου – πηγής, πατώντας το “**Button**”

Το υπόλοιπο μέρος του υποπρογράμματος προσφέρει οπτική ανατροφοδότηση στον χρήστη μέσω των παρακάτω:

- **Οθόνη**. Στην οθόνη φαίνονται τα pixelτης κίνησης του εκτελεστή.
- **Στήλη pixel δεξιά της οθόνης,** δείχνει την αντιστοιχία του χρήστη με ορισμένες ομάδες συχνοτήτων. Η αντιστοιχία αυτή εξηγείται σε βάθος στο κεφάλαιο “εξήγηση ανάπτυξης”
- **Sample On/off**, δείχνει την κατάσταση αναπαραγωγής του ήχου – πηγής.

Περαιτέρω χειρισμός επιτυγχάνεται από τις κινήσεις του εκτελεστή.

Το **ύψος του εκτελεστή** επηρεάζει το **τονικό ύψος** του ήχου. Η τιμή που παίρνει το ύψος του blobπου αντιστοιχεί στην εκτελεστή, οδηγεί βάση εκθετικής αναλογίας την τονική μετατόπιση του ήχου. Στην ποσότητα της τονικής μετατόπισης χρησιμοποιείται ως γραμμικός συντελεστής η ποσότητα της κίνησης.

Το **πλάτος του εκτελεστή** σε συνδυασμό με τον **δείκτη συστολής** καθορίζει την ποσότητα δύο τύπων εφέ. Το πρώτο εφέ είναι διαμόρφωση πλάτους, και δεύτερο εφέ είναι μια σειρά καθυστέρηση τύπου IIR. Στην σειρά καθυστέρησης το πλάτος του εκτελεστή οδηγεί βάση εκθετικής αναλογίας την ανατροφοδότηση, ενώ σε αντίστροφη αναλογία τον χρόνο καθυστέρησης.

Η **κατεύθυνση του εκτελεστή** σε συνδυασμό με την ποσότητα κίνησης καθορίζει την χωροτοποθέτηση του παραγόμενου ήχου στερεοσκοπικά. Η κατεύθυνση του εκτελεστή προδιαθέτει την θέση του ήχου στην αντίστοιχη κατεύθυνση και η ποσότητα της κίνησης μετατοπίζει ανάλογα τον ήχο από την κεντρική θέση. Παραδείγματος χάριν, όταν ο εκτελεστής κάνει μια μεγάλη κίνηση προς τα δεξιά, η στερεοσκοπική θέση του ήχου θα μετατοπιστεί πολύ δεξιά, αντίστοιχα όταν η ποσότητα της κίνησης είναι μικρή, η μετατόπιση θα είναι λίγη.

Η **απόσταση από το χρήστη αναφοράς** καθορίζει την ποσότητα εφέ τύπου reverb. Όσο μεγαλύτερη η απόσταση μεταξύ των δύο χρηστών, τόσο περισσότερο ακούγεται το εφέ σε σχέση με τον αρχικό ήχου.

#### 3.1.1.5. Maestro


ΕΙΚΟΝΑ 18 ΔΙΕΠΑΦΗ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "MAESTRO"

Στο υποπρόγραμμα “Maestro” ο μοναδικός διαθέσιμος έλεγχος για τον χρήστη είναι η τελική στάθμη έντασης εξόδου, με τη χρήση ενός ροοστάτη (fader). Ο εκτελεστής που αναλαμβάνει τον ρόλο του “χρήστη αναφοράς” είναι εκείνος ο οποίος στέκεται στα δεξιά, από την οπτική του ακροατή, όταν στη σκηνή υπάρχουν δύο άτομα. Στην περίπτωση τριών ατόμων ο χρήστης αναφοράς είναι το άτομο στην μέση, αλλά όπως αναφέρθηκε στην αρχή του κεφαλαίου η εφαρμογή αναπτύχθηκε για δύο άτομα και δεν υποστηρίζει σε όλες της τις λειτουργίες παραπάνω από δύο, επίσης κάθε επεξήγηση για την εφαρμογή αναφέρεται στο σενάριο δύο ατόμων.

Ο σκοπός του υποπρογράμματος αυτού είναι η διαμόρφωση του ήχου ο οποίος δημιουργείται στο INSTRUMENT. Ο χρήστης αναφοράς ελέγχει το υποπρόγραμμα ως εξής:

Αρχικά θέτουμε μία τιμή κατωφλίου η οποία χωρίζει τις τιμές ύψους του χρήστη αναφορές σε δύο ομάδες, της **χαμηλές τιμές** και τις **υψηλές τιμές**.

Οι **χαμηλές τιμές** επηρεάζουν την συνολική ένταση. Η τιμή της έντασης αλλάζει ανάλογα με την τιμή ύψους.

Οι **υψηλές τιμές** θέτουν την στάθμη έντασης στο μέγιστο, και επηρεάζουν ένα φίλτρο τύπου low-shelf. Ανάλογα με το ύψος του εκτελεστή το ύψος μειώνεται η απόκριση του φίλτρου στις χαμηλές συχνότητες, δημιουργώντας έναν φιλτραρισμένο υψίσυχνο ήχο.

Το **πλάτος του χρήστη αναφοράς** εισάγει θόρυβο στον ήχο. Όσο μεγαλώνει το πλάτος του χρήστη αναφοράς, μεγαλώνει το εύρος ζώνης του θορύβου γύρο από κάθε συχνότητα του εισερχόμενου σήματος.

Η **απόσταση από τον εκτελεστή** είναι η ίδια τιμή με την **απόσταση από τον χρήστη αναφοράς** στο υποπρόγραμμα INSTRUMENT, και αναφέρεται μόνο λόγω της αλληλένδετης ύπαρξης των δύο τιμών.

Στο επόμενο υποκεφάλαιο θα γίνει μια εκτενέστερη ανάλυση των υποπρογραμμάτων. Η ανάλυση αυτή θα επικεντρωθεί στον τρόπο με τον οποίο υλοποιούνται οι διαδικασίες στο περιβάλλον της Max και αντίστοιχα στην εφαρμογή Bread.

## 3.2. Ανάλυση υποπρογραμμάτων και διαδικασιών

Το κεφάλαιο αυτό στοχεύει στην βαθύτερη ανάλυση και επεξήγηση των υποπρογραμμάτων που δημιουργήθηκαν για το πειραματικό μέρος. Τα υποπρογράμματα θα αναλυθούν με την σειρά που συναντώνται στη ροή των δεδομένων. Οι όροι που εξηγούνται στο κεφάλαιο "2. Εξειδικευμένη θεωρία" θα θεωρηθούν γνωστοί και δεν θα λάβουν περεταίρω εξήγηση εντός του κεφαλαίου.

### 3.2.1. VideoIn

Όπως αναφέρθηκε προηγουμένως, η μόνη λειτουργία του συγκεκριμένου υποπρόγραμμα είναι η εισαγωγή βίντεο εντός του περιβάλλοντος της Max και ο βασικός χειρισμός αναπαραγωγής ή αιχμαλώτισης (capturing) του βίντεο αυτού.

**Camera**

Στην περίπτωση κατά την οποία χρησιμοποιείται ζωντανή βιντεοσκόπηση, το αντικείμενο της Maxγια την επίτευξη της αιχμαλωσίας του βίντεο, είναι το [jit.qt.grab] για το περιβάλλον των Macintosh, και το [jit.dx.grab] για το περιβάλλον των Windows. Το πρώτο στηρίζεται στην βιβλιοθήκη του QuickTime, ενώ το δεύτερο στο DirectX.

Το αντικείμενο ενδέχεται να πάρει μερικές βασικές παραμέτρους αρχικοποίησης (Εικόνα 19) :


ΕΙΚΟΝΑ 19 [JIT.QT.GRAB] ΜΕ ΟΡΙΣΜΑΤΑ ΑΡΧΙΚΟΠΟΙΗΣΗΣ

- 640 480. Τα δύο νούμερα ακολουθούν αμέσως μετά το όνομα του αντικειμένου και ορίζουν τα διαστάσεις του βίντεο, την ανάλυσή του. Αν η εικόνα της κάμερας έχεις διαφορετικές διστάσεις, προσαρμόζονται με την χρήση κλιμάκωσης (scaling).
- @vmode 2. Η παράμετρος "vmode" ορίζει τον τρόπο και την ποιότητα της ψηφιοποίησης της εικόνας. Επιλογή @vmode 2, σημαίνει "DV high quality mode"

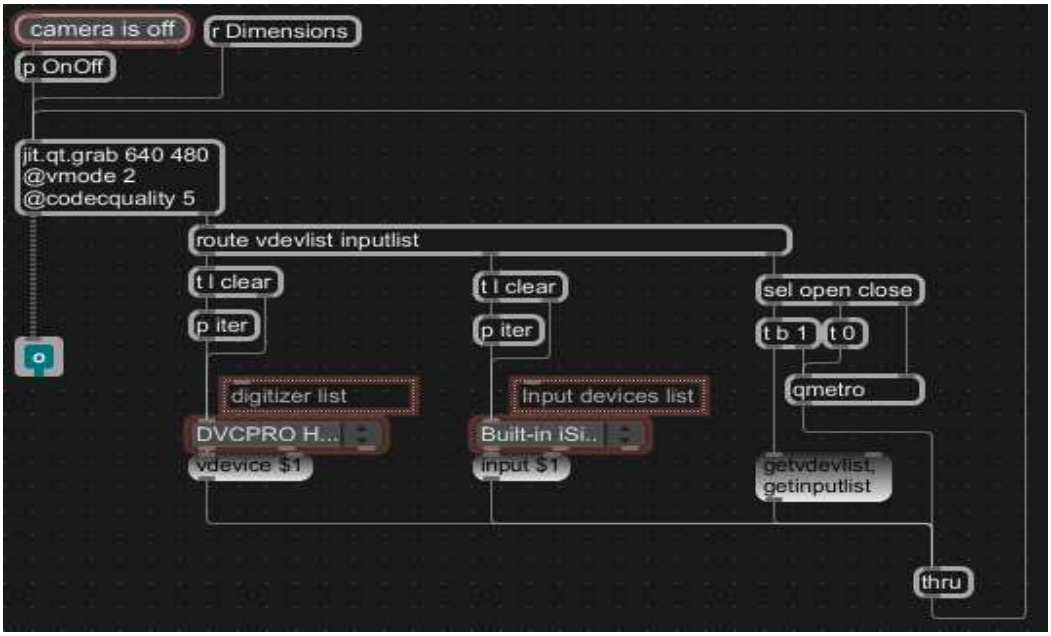
- @codecquality 5. Η παράμετρος αυτή ορίζει το πρωτόκολλο και την ποιότητα της συμπίεσης της εικόνας από την κάμερα. Η τιμή 5 σημαίνει “lossless”, δηλαδή καμία συμπίεση. Η εικόνα μεταφέρεται ως “raw” δεδομένα.

Επιπλέον χειρισμός στο αντικείμενο γίνεται μέσω του πλήκτρου “Camera is On/Off”. Όταν το πλήκτρο βρεθεί στην κατάσταση “Camera is On”, στέλνει το μήνυμα “open” στο [jit.qt.grab] το οποίο ξεκινάει έτσι την ψηφιοποίηση της εικόνας. Αντίθετα, στην κατάσταση “Camera is Off”, στέλνει το μήνυμα “close” στο αντικείμενο κάνοντάς το να σταματάει οποιαδήποτε διαδικασία εκτελεί (Εικόνα 20).


ΕΙΚΟΝΑ 20

Κάθε φορά που το αντικείμενο [**jit.qt.grab**] λαμβάνει το μήνυμα “**open**”, στέλνει από την δεξιά του έξοδο το μήνυμα “open”, σηματοδοτώντας έτσι ότι η λειτουργία του εκτελείται σωστά. Το μήνυμα αυτό, που προέρχεται από το [jit.qt.grab] με την σειρά του στέλνει τα μηνύματα “**getvdevlist**” και “**getinputlist**” πίσω στο αντικείμενο. Η διαδικασία αυτή εξασφαλίζει ότι κάθε φορά που “ανοίγει” το [jit.qt.grab], αφού ξεκινήσει η βασική λειτουργία του, δέχεται τα μηνύματα “getvdevlist, getinputlist”, με τα οποία ελέγχει ποιές συσκευές λήψης είναι συνδεδεμένες στον υπολογιστή και τα πρωτόκολλα ψηφιοποίησης. Έτσι δημιουργούνται οι λίστες “**Input Devices List**” και “**Digitizer List**”,οι οποίες αναφέρονται στο προηγούμενο κεφάλαιο.

Τα περιεχόμενά τους αλλάζουν αναλόγως με τον υπολογιστή και τις συσκευές με της οποίες είναι συνδεδεμένος. Όταν από κάποια λίστα επιλέγεται κάποιο στοιχείο, αποστέλλεται στο [jit.qt.grab] το μήνυμα “**input $1**” για την λίστα “**Input Devices list**”, και “**vdevice $1**” για την λίστα “**Digitizer List**”, όπου το “$1” αντικαθιστάται από τον δείκτη (index) που αντιστοιχεί στο επιλεγμένο στοιχείο. Ο αλγόριθμος φαίνεται Εικόνα 21.




ΕΙΚΟΝΑ 21

Η τελευταία διαθέσιμη παράμετρος ελέγχου είναι στην λίστα "**Dimensions**". Η λίστα αυτή περιέχει μερικά από τα πιο πιθανά ζεύγη διαστάσεων για συσκευές βίντεο. Οι διαστάσεις που περιλαμβάνονται στην λίστα είναι διαστάσεις λόγου 4:3 από "120*160" ως "800*600", καθώς και η διάσταση "720 * 576", η οποία είναι η πρότυπη διάσταση του DV βίντεο.

Προτείνεται η χρήση διαστάσεων ίδιων με εκείνες της κάμερας ανά περίπτωση. Η αλλαγή διαστάσεων του εισαγόμενου βίντεο από το αντικείμενο [jit.qt.grab] επιφέρει εμφανείς απόλυες στην διαθέσιμη επεξεργαστική ισχύ του εκάστοτε υπολογιστή. Στον πίνακα παρακάτω φαίνεται χρησιμοποιούμενο ποσοστό πόρων του επεξεργαστή σε ορισμένες επιλογές διαστάσεων.

| Διαστάσεις βίντεο, από κάμερα | Επιλογή διαστάσεων στο υποπρόγραμμα _VideoIn_ | Ποσοστό κατάληψης πόρων επεξεργαστή |
|---|---|---|
| 720 * 576 | 640 * 480 | 60 % |
| 720 * 576 | 320 * 240 | 45 % |
| 720 * 576 | 720 * 576 | 45 % |

ΠΙΝΑΚΑΣ 1 ΣΥΓΚΡΙΣΗ ΑΛΛΑΓΗΣ ΔΙΑΣΤΑΣΕΩΝ ΣΕ ΣΧΕΣΗ ΜΕ ΤΟΥΣ ΑΠΑΙΤΟΥΜΕΝΟΥΣ ΥΠΟΛΟΓΙΣΤΙΚΟΥΣ ΠΟΡΟΥΣ

Παρατηρούμε πως η διαστάσεις του εισερχόμενου βίντεο, χρειάζεται να υποδιπλασιαστούν τουλάχιστον ώστε να ξεκινήσει να υπάρχει κέρδος στην ποσότητα απαιτούμενης υπολογιστικής ισχύος, σε σχέση με της φυσικές διαστάσεις του βίντεο από την κάμερα. Στην περίπτωση αυτή η λεπτομέρεια της εικόνας έχει μειωθεί σε μεγάλο βαθμό και πιθανόν να μην είναι πλέον επαρκής για την χρήση του βίντεο. Τα δεδομένα του πίνακα, δεν αποτελούν ακριβείς μετρήσεις, αλλά απεικονίζουν τιμές που μετρήθηκαν επανειλημμένα σε ένα πλήθος τυχαίων στιγμών και αφορούν αποκλειστικά τις απαιτήσεις πόρων για τα υποπρογράμματα _VideoIn_, _BlobTracking_ και _DataViewer_, τα οποία είναι υπεύθυνα για την ανάλυση της εικόνας. Ο αλγόριθμος της γεννήτριας ήχου εφόσον σχεδιάζεται ανεξάρτητα από την ανάλυση του βίντεο, έχει ανεξάρτητες απαιτήσεις, οι οποίες μπορούν να κυμαίνονται από ελάχιστες ως και απαγορευτικές!

Η κάμερα που χρησιμοποιήθηκε καταγράφει το βίντεο με μορφή "DV - Pal", του οποίου η ανάλυση είναι 720 * 576 εικονοστοιχεία. Όταν ο χρήστης επιλέξει ένα ζεύγος διαστάσεων, τα δύο νούμερα εξέρχονται από την λίστα και λαμβάνουν το πρόθεμα "dim". Το πρόθεμα αυτό, γνωστοποιεί στο αντικείμενο [jit.qt.grab] ότι τα δύο αυτά νούμερα αντιστοιχούν στις επιθυμητές διαστάσεις.

**PLAYBACK**

Το αντικείμενο, υπεύθυνο για την αναπαραγωγή προγεγραμμένου βίντεο, είναι το [jit.qt.movie] (Εικόνα 22). Τα αρχικά ορίσματα που επιλέχθηκαν στο αντικείμενο αυτό είναι τα εξής:

- 320 240. Τα πρώτα ορίσματα θέτουν τις αρχικές διαστάσεις του βίντεο.
- @autostart 0. Η παράμετρος "@autostart" ορίζει αν θα αρχίσει να αναπαράγεται το αρχείο βίντεο, μόλις επιλεχθεί για αναπαραγωγή. Η τιμή "0", σημαίνει ότι το βίντεο θα αρχίσει να αναπαράγεται μόνο μέσω εξωτερικού μηνύματος.
- @vol 0. Η παράμετρος "@vol", ελέγχει την ένταση ήχου του βίντεο. Στην περίπτωση της εργασίας, ο συνολικός ήχος παράγεται μέσω αλγορίθμων, οπότε θέτω την παράμετρο στο "0" ώστε να μην έρχεται μέσα στο υποπρόγραμμα κανένας εξωτερικός ήχος.


ΕΙΚΟΝΑ 22 [JIT.QT.MOVIE] ΜΕ ΟΡΙΣΜΑΤΑ ΑΡΧΙΚΟΠΟΙΗΣΗΣ

Ο επιπλέον χειρισμός στην περίπτωση του προγεγραμμένου βίντεο είναι περισσότερος σε σχέση με τον διαθέσιμο χειρισμό του ζωντανού βίντεο.

Τα πλήκτρα ελέγχου αφορούν την διαδικασία αναπαραγωγής του βίντεο. Λόγω του πλήθους των, ακολουθεί ένα πίνακας με το όνομα του πλήκτρου στην αριστερή στήλη και το μήνυμα που στέλνει κάθε πλήκτρο στην δεξιά.

| Load file | Read |
|---|---|
| Play | Start |
| Stop | Stop, frame 0. |
| Pause | Stop |
| To Start | Frame 0 |
| To End | Frame $1. Το $1 αντικαθιστάται με τον αριθμό του τελευταίου καρέ μόλις φορτωθεί το βίντεο. |
| Reset loop | Θέτει ως αρχή και τέλος της κυκλικής αναπαραγωγής τις τιμές των frames που έχουν τα “toEnd” και “toStart” πλήκτρα. |
| Scrub slider/number | Και τα δύο “χειριστήρια” στέλνουν έναν αριθμό που αντιστοιχεί σε κάποιο καρέ. Το μήνυμα είναι “frame $1”&“stop”. Το $1 αντικαθιστάται με τον αριθμό από το slider ή το numberbox. Το “stop” ακολουθεί ώστε το βίντεο να παραμείνει σε αυτό το καρέ. |
| Loop slider | Looppoints $1 $2. Οι τιμές $1 και $2 αντικαθιστούνται από τις δύο τιμές που βγάζει αυτό το slider. Η μικρότερη αντικαθιστά το $1 και η μεγαλύτερη το $2. |

ΠΙΝΑΚΑΣ 2 ΜΗΝΥΜΑΤΑ ΠΛΗΚΤΡΩΝ ΤΟΥ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "_VIDEOIN_ - CAPTURE"

Με την φόρτωση ενός αρχείου, το αντικείμενο [jit.qt.movie] στέλνει το μήνυμα read, το οποίο με την σειρά του ξεκινάει μια διαδικασία η οποία στέλνει στο αντικείμενο τα μηνύματα **getfps**, **gettimescale**, **getduration**. Αυτά τα μηνύματα προκαλούν το [jit.qt.movie] να στείλει κάποιες πληροφορίες για το αρχείο που μόλις φορτώθηκε.

- **Getfps**, στέλνει το πλήθος των καρέ ανά δευτερόλεπτο.

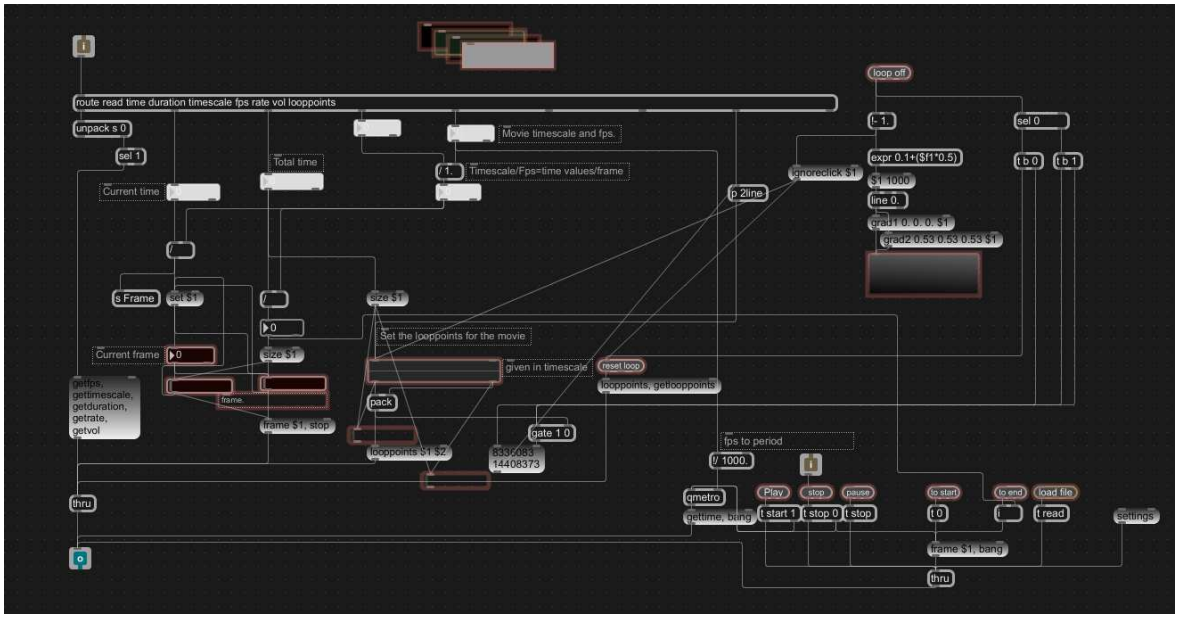
- **Gettimescale**, στέλνει το “timescale” του αρχείου. Η τιμή αυτή χρησιμοποιείται για την αναγνώριση της χρονικής θέσης ενός καρέ μέσα στο αρχείο.
- **Getduration**, στέλνει την διάρκεια του αρχείου σε μονάδες timescale.

Ισχύει το παρακάτω

Timescale / Fps = time values / frame index

Με κάθε νέο καρέ, στο αντικείμενο στέλνεται και το μήνυμα “gettime”. Έτσι για κάθε νέο καρέ υπολογίζεται ο δείκτης (index) του μέσα στο βίντεο. Η τιμή του δείκτη χρησιμοποιείται για να “οδηγήσει” το scrub slider, ώστε να παρέχει στον χρήστη ένα οπτικό feedback για το χρονικό σημείο αναπαραγωγής που βρίσκεται το βίντεο.

Η πολυπλοκότητα του υποπρογράμματος, προερχόμενη από τον τρόπο “προγραμματισμού” στο περιβάλλον της max, δεν επιτρέπει μια ευκρινή εικόνα, για της παραπάνω διαδικασία, όπως φαίνεται στην εικόνα 23.




ΕΙΚΟΝΑ 23ΜΙΚΡΟΓΡΑΦΙΑ ΕΛΕΓΧΟΥ ΤΟΥ ΑΝΤΙΚΕΙΜΕΝΟΥ [JIT.QT.MOVIE]

Λόγω σχεδιαστικής απροσεξίας, η λίστα διαστάσεων στην επιλογή “Camera”, είναι η λίστα η οποία θέτει τις διαστάσεις και στο “Playback”. Οπότε ο χρήστης χρειάζεται να αλλάξει τις διαστάσεις του βίντεο μέσω της επιλογής “Camera”.

### 3.2.2. Αφαίρεση φόντου – ανίχνευση κηλίδων

**RBG to Grayscale**

Η πρώτη διαδικασία που πραγματοποιεί το υποπρόγραμμα αυτό, είναι η μετατροπή της εικόνας από έγχρωμη σε ασπρόμαυρη. Ο τρόπος που χρησιμοποιείται, όντας και ο πιο συνηθισμένος, είναι η πρόσθεση των τριών καναλιών χρώματος, σε διαφορετικές αναλογίες. Για την διαδικασία αυτή η max περιέχει το αντικείμενο [**jit.rgb2luma**]. Το αντικείμενο αυτό χρησιμοποιεί τις αναλογίες **Gray = 0.2989 * R + 0.5870 * G + 0.1140 * B**, όπως αναφέρεται και σε προηγούμενο κεφάλαιο.

Στο υποπρόγραμμα δίνεται στον χρήστη η δυνατότητα να δοκιμάσει τις δικές του αναλογίες μέσω του υποπρόγραμμα "V2.1.rgb2bw.maxpat" (εικόνα 24). Κάθε ένα από τα τρία slider αντιστοιχεί στον συντελεστή ενός από τα τρία κανάλια χρώματος.

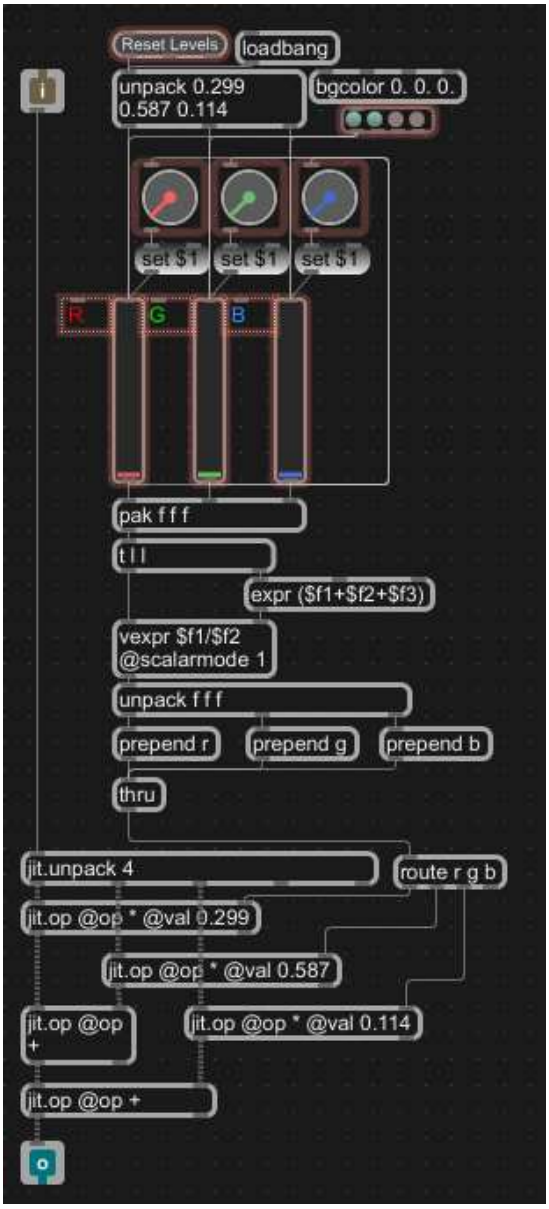

ΕΙΚΟΝΑ 24 RGB2GRAY

Με το αντικείμενο [**jit.unpack 4**], η ενιαία εικόνα χωρίζεται στα τρία κανάλια χρώματος. Κάθε ένα οδηγείται σε ένα διαφορετικό αντικείμενο [**jit.op @op ***], στο οποίο γίνεται ο πολλαπλασιασμός με τον συντελεστή. Στην συνέχεια τα τρία κανάλια προστίθενται με μια σειρά αντικειμένων [**jit.op @op +**], και το αποτέλεσμα είναι μια εικόνα με ένα κανάλι χρώματος. Οι συντελεστές έχουν πάντα συμπληρωματική αξία ως προς την μονάδα, έτσι διατηρούνται σωστές αναλογίες. Ο συνολικός αλγόριθμος φαίνεται στην εικόνα 26.

Ο λόγος που χρησιμοποιήθηκε αυτή η υλοποίηση είναι η πιθανή βελτίωση της διαφοράς της αντίθεσης μεταξύ του φόντου της εικόνας και των αντικειμένων. Όπως φαίνεται στην παρακάτω εικόνα 25, υπάρχει διαφορά μεταξύ των τιμών τις maxκαι των πειραματικών τιμών που τελικά χρησιμοποιήθηκαν. Με τις τιμές που τελικά επιλέχθηκαν, στην περίπτωση αυτή, υπάρχει μεγαλύτερη αντίθεση μεταξύ των performersκαι του φόντου. Το γεγονός αυτό αφήνει μεγαλύτερα περιθώρια για υψηλότερες τιμές κατωφλίου σε μετέπειτα στάδιο, με αποτέλεσμα την καθαρότερη binaryεικόνα.

Ο χρήστης έχει στην διάθεσή του το πλήκτρο "**reset levels**", με την χρήση του οποίου, οι αναλογίες επιστρέφουν σε εκείνες της max.

ΕΙΚΟΝΑ 25 ΑΝΤΙΘΕΣΗ ΜΕ ΔΙΑΦΟΡΕΤΙΚΕΣ ΣΤΑΘΜΕΣ RGB




ΕΙΚΟΝΑ 26 RGB2GRAY – ΥΛΟΠΟΙΗΣΗ

### Αφαίρεση φόντου

Για τον καθορισμό των κριτηρίων σύμφωνα με τα οποία τα εικονοστοιχεία κατατάσσονται ως φόντο ή προσκήνιο υλοποιήθηκε η μέθοδος χρήσης τυπικής απόκλισης.

**Background standard deviation – τυπική απόκλιση**

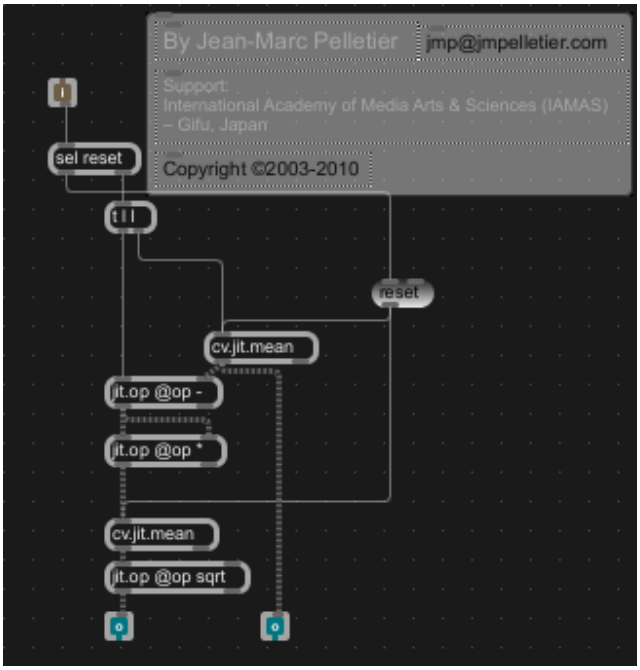
Σε αυτή την μέθοδο χρησιμοποιείται το πρόγραμμα της βιβλιοθήκης cv.jit, [cv.jit.stddev]. Το πρόγραμμα αυτό (όπου φαίνεται στην εικόνα 27) υπολογίζει:

- το μέσο όρο των τιμών της εικόνας και
- την τυπική απόκλισή τους από αυτό.

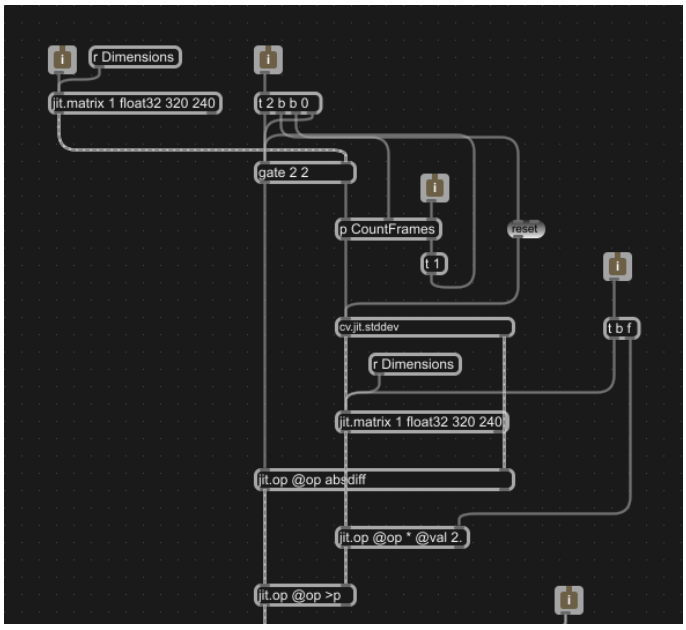
Οι τιμές αυτές υπολογίζονται από κάποιο συγκεκριμένο αριθμό καρέ, στα οποία το φόντο-όπως και στην πρώτη περίπτωση- πρέπει να είναι απαλλαγμένο από κινούμενα ή άλλα αντικείμενα στο μπροστινό μέρος της εικόνας (που δεν θεωρείται φόντο).

Η τρέχουσα εικόνα συγκρίνεται και αφαιρείται από το μέσο όρο που έχει υπολογιστεί και στην συνέχεια το αποτέλεσμα συγκρίνεται με την τυπική απόκλιση. Αν το απόλυτο αποτέλεσμα της αφαίρεσης είναι μεγαλύτερο από διπλάσιο της τυπικής απόκλισης, τότε το pixel θεωρείται ότι ανήκει στο προσκήνιο ('foreground'), αλλιώς θεωρείται φόντο. Ο αλγόριθμος που δημιουργήθηκε για την διαδικασία αυτή φαίνεται στην εικόνα 28.

Η μέθοδος αυτή, αφήνει λιγότερο θόρυβο στο τελικό αποτέλεσμα, και επηρεάζεται λιγότερο από τις σκιές των ατόμων επί σκηνής.




ΕΙΚΌΝΑ 27 [CV.JIT.STDDEV]




ΕΙΚΟΝΑ 28 ΑΦΑΙΡΕΣΗ ΦΟΝΤΟΥ

## Ανίχνευση και ονομασία των BLOBS

Η ανίχνευση και η ονομασία των κινούμενων αντικειμένων (blobs) γίνεται εξ' ολοκλήρου από το αντικείμενο [cv.jit.label] (Εικόνα 29).


ΕΙΚΟΝΑ 29 CV.JIT.LABEL] ΜΕ ΟΡΙΣΜΑΤΑ ΑΡΧΙΚΟΠΟΙΗΣΗΣ

Όπως τα προηγούμενα κύρια αντικείμενα των υποπρογραμμάτων, έτσι και το [cv.jit.label] έχει δύο ορίσματα αρχικοποίησης. Το αντικείμενο αυτό έχει μερικούς διαφορετικούς τρόπους λειτουργίας οι οποίοι ορίζονται μέσω δύο παραμέτρων, των "charmode" και "mode".

Το "charmode" δέχεται τα ορίσματα 1 ή 0. Το πρώτο, "charmode 1, δηλώνει ότι το jitter matrix που εξάγεται από το αντικείμενο είναι τύπο character, δηλαδή κάθε κελί του περιέχει μία 8bitακέραια τιμή. Σε αυτή την περίπτωση αν το "mode", το οποίο επίσης δέχεται τα ορίσματα 1 ή 0 έχει την τιμή 1, τότε τα blobs ταξινομούνται και ονομάζονται σύμφωνα με το μέγεθός τους. Δηλαδή το μεγαλύτερο blob έχει το όνομα 1, το δεύτερο μεγαλύτερο έχει το όνομα 2 και ούτω καθ' εξής. Στην περίπτωση όπου το mode έχει την τιμή 0, τα blobs ταξινομούνται και ονομάζονται σύμφωνα με την θέση τους στον κατακόρυφο άξονα (y), και στην περίπτωση κατά της οποία δύο blobs βρεθούν στο ίδιο σημείο του άξονα, προηγείται στην αρίθμηση εκείνο του οποίου η θέση έχει την μικρότερη τιμή στο οριζόντιο άξονα (x).

Στην περίπτωση όπου το charmode είναι 0, τότε το jitter matrix στην έξοδο του αντικειμένου είναι τύπου "long". Αυτό σημαίνει ότι το mode 1 ονομάζει τα blobs με το σύνολο των pixels που τα αποτελούν, π.χ. αν ένα blob αποτελείται από 1500 εικονοστοιχεία, το όνομά του είναι 1500. Στην περίπτωση του mode 0, το αντικείμενο συμπεριφέρεται όπως και στην περίπτωση του charmode 1, mode 0.

Στο υποπρόγραμμα, το αντικείμενο ορίζεται ως "charmode 1" και "mode 0", το οποίο αν δεν δηλωθεί διαφορετικά είναι η προεπιλεγμένη κατάσταση. Το δεύτερο όρισμα που φαίνεται, "@threshold 1000" δίνει το ελάχιστο αποδεκτό μέγεθος των blobs. Στην περίπτωση που βλέπουμε κάθε blob με μέγεθος μικρότερο των χιλίων εικονοστοιχείων, απορρίπτεται πριν καν ξεκινήσει η διαδικασία της ονομασίας.

Ένα θέμα που προκύπτει στην διαδικασία είναι, η ασταθής ονομασία των blobs. Το αντικείμενο δεν κρατάει “ιστορικό” των blobs, οπότε κάθε καρέ υπόκειται στην ίδια διαδικασία ανίχνευσης και ονομασίας αυτών από τον αρχή και ανεξάρτητα από τα προηγούμενα. Λόγω του ότι η διαδικασία ονομασίας δίνει “προτεραιότητα” στη θέση του blob κατά μήκος του άξονα y, δύο blob τα οποία κινούνται σε σταθερή θέση στον άξονα x αλλά “ανταλλάσουν” την τιμή τους στον άξονα y, θα ανταλλάσουν και όνομα. Ίσως σε κάποιες περιπτώσεις αυτό να μην αποτελεί πρόβλημα, αλλά στην δική μας είναι απαραίτητο κάθε μουσικός να διατηρεί την ταυτότητά του κατά τη διάρκεια της παράστασης, κατά συνέπεια η απρόοπτη αλλαγή του ονόματος κάποιου blob θεωρείται μη αποδεκτή.

Μια λύση δίνει η ίδια η βιβλιοθήκη cv.jit με το αντικείμενο [cv.jit.blobs.sort], του οποίου η λειτουργία είναι να βοηθάει στη διατήρηση των ονομάτων των blobs ανεξαρτήτως της θέσης τους. Το αντικείμενο [cv.jit.blobs.sort] δέχεται ως είσοδο μια λίστα, σε μορφή jittermatrix, με τα κέντρα βάρους των blobs, και βάση αυτών δημιουργεί έναν οδηγό ανακατάταξης – μετονομασίας των blobs. Παρότι έτσι υποθετικά εξασφαλίζεται η σταθερή ονομασία, στην πράξη δε συμβαίνει αυτό. Το συγκεκριμένο αντικείμενο έχει μια αστάθεια στην περίπτωση που ο αριθμός των blobs μειωθεί και αυξηθεί πάλι, δηλαδή σε μια διαδοχική εξαφάνιση/εμφάνιση κάποιου blob. Π.χ. αν υπάρχουν 3 blobs με ονόματα 1,2,3, και το 2 σβήσει και ανάψει, πιθανότερο είναι τα νέα ονόματα να είναι 1,3,4.

Είναι φανερό πως η προσέγγιση αυτή δεν είναι επαρκής. Η εναλλακτική λύση που βρέθηκε, παρότι χαρακτηρίζεται παράδοξη , στην περίπτωση αυτή, αποδείχθηκε εύστοχη και κάλυψε της ανάγκες της εφαρμογής. Εφόσον το πρόβλημα προέρχεται από την θέση στον άξονα y, και αφού οι performersέχουν οριοθετημένες περιοχές στον άξονα x, το κάθε καρέ, πριν εισέλθει στο [cv.jit.label], περιστρέφεται γύρο από τον διαγώνιο άξονά του με το αντικείμενο [jit.transpose].Η ακριβής διαδικασία μεταφέρει κάθε σειρά της εικόνας (row) και το περιεχόμενό της, σε στήλη (column), και το ανάποδο. Έτσι οι συντεταγμένες x μετατρέπονται σε y, και το ανάποδο. Κατά συνέπεια, η κίνηση ενός μουσικού κατά τον (πραγματικό) κατακόρυφο άξονα αναγνωρίζεται ως οριζόντια κίνηση στο οριζοντιωμένο καρέ, με αποτέλεσμα να μην επηρεάζει την ονομασία και ανίχνευση της εκάστοτε κηλίδας. Λόγω του τρόπου κατά τον οποίο το [jit.transpose] περιστρέφει τον πίνακα, με την χρήση του αντικειμένου δύο φορές, ο πίνακας επιστρέφει στον αρχικό του προσανατολισμό (Εικόνα 30). Είναι προφανές, ότι στο τέλος κάνουμε αντίστροφη χρήση του [jit.transpose] στο jittermatrix που προκύπτει στην έξοδο του [cv.jit.label], για να επαναφέρουμε

την εικόνα στον αρχικό της προσανατολισμό. Ο αλγόριθμος που εκτελείται μέσω του αντικειμένου [cv.jit.label] αναφέρεται στο ΠΑΡΑΡΤΗΜΑ Α.

Μετά την παραπάνω διαδικασία το jitter matrix που περιέχει τα (σταθερά) ονομασμένα blobs είναι έτοιμο για ανάλυση.


ΕΙΚΟΝΑ 30 ΧΡΗΣΗ ΤΟΥ [JIT.TRANSPOSE] ΓΙΑ ΠΕΡΙΣΤΟΦΗ ΠΙΝΑΚΑ, ΚΑΙ ΕΠΑΝΑΦΟΡΑ

### 3.2.3. Ανάλυση κηλίδων

Σε κάθε blob υπολογίζονται οι εξής παράμετροι:

- Quantity of motion, QoM (1 καρέ, 5 καρέ)
- Mass (Μέγεθος σε pixels)
- Bounding Rectangle ( left – top – right – bottom )
- Baricenter (x, y)
- Contraction Index
- Directivity Index

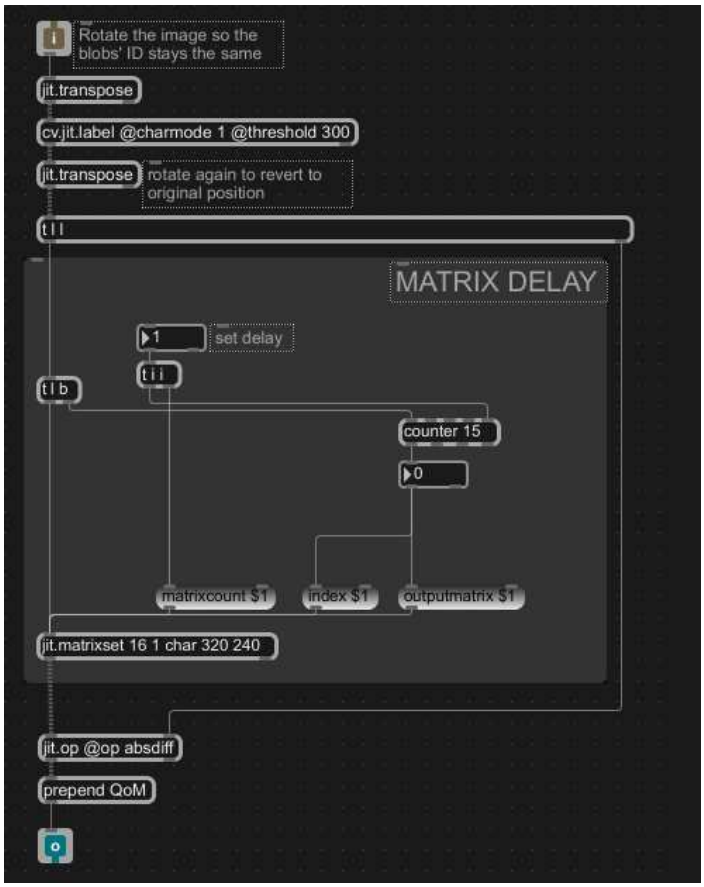
Επίσης, με βάση το κέντρο βάρους του κάθε blob υπολογίζονται οι αποστάσεις μεταξύ των μουσικών/εκτελεστών.

Για πληροφορίες σχετικά με της παραμέτρους, ο αναγνώστης μπορεί να ανατρέξει στο κεφάλαιο "2.2.3.. Αφαίρεση φόντου, ανίχνευση και ανάλυση"blob"".

**Quantity of motion**

Η παράμετρος "quantity of motion", αντιστοιχεί στο μέγεθος της κίνησης του blob, δηλαδή την έκταση και ταχύτητα μετατόπισης του Blob στον χώρο. Ο τρόπος υπολογισμού βασίζεται στην απόλυτη διαφορά μεταξύ δύο, ή και περισσότερων, καρέ του blob. Η χρονική διαφορά μεταξύ των δύο επιλεγμένων καρέ (μπορεί να είναι δύο διαδοχικά ή με μεγαλύτερη απόσταση μεταξύ τους) επηρεάζει το μέγεθος της κίνησης. Η διαδικασία εντός του περιβάλλοντος του Max/Msp φαίνεται στην εικόνα 31. Απαιτείται η εισαγωγή μιας καθυστέρησης στο jittermatrix , ώστε να δημιουργηθεί η χρονική διαφορά μεταξύ των καρέ που αφαιρούνται. Για το σκοπό αυτό και επειδή το Max/Msp δεν περιέχει αντικείμενο κατάλληλο για αυτή την εργασία, σχεδιάστηκε το απλό υποπρόγραμμα [MatrixDelay] (φαίνεται εντός του γκρι πλαισίου στην εικόνα 31) το οποίο δέχεται έναν ακέραιο θετικό αριθμό, ο οποίος συμβολίζει τον χρόνο καθυστέρησης, σε καρέ, του εισερχόμενου jittermatrix αντίστοιχα. Το υποπρόγραμμα αυτό λειτουργεί για πίνακες με 1 κανάλι (plane). Για αλλαγή, σε πραγματικό χρόνο, του πλήθους των καναλιών του εισερχόμενου πίνακα, απαιτείται έλεγχος στο [jit.matrixset] εντός του [MatrixDelay]. Το jittermatrix που περιέχει τα

blobs έχει μόνο ένα κανάλι (plane), και για τον λόγο αυτό, η προαναφερθείσα λειτουργία δεν υποστηρίζεται προς το παρόν.

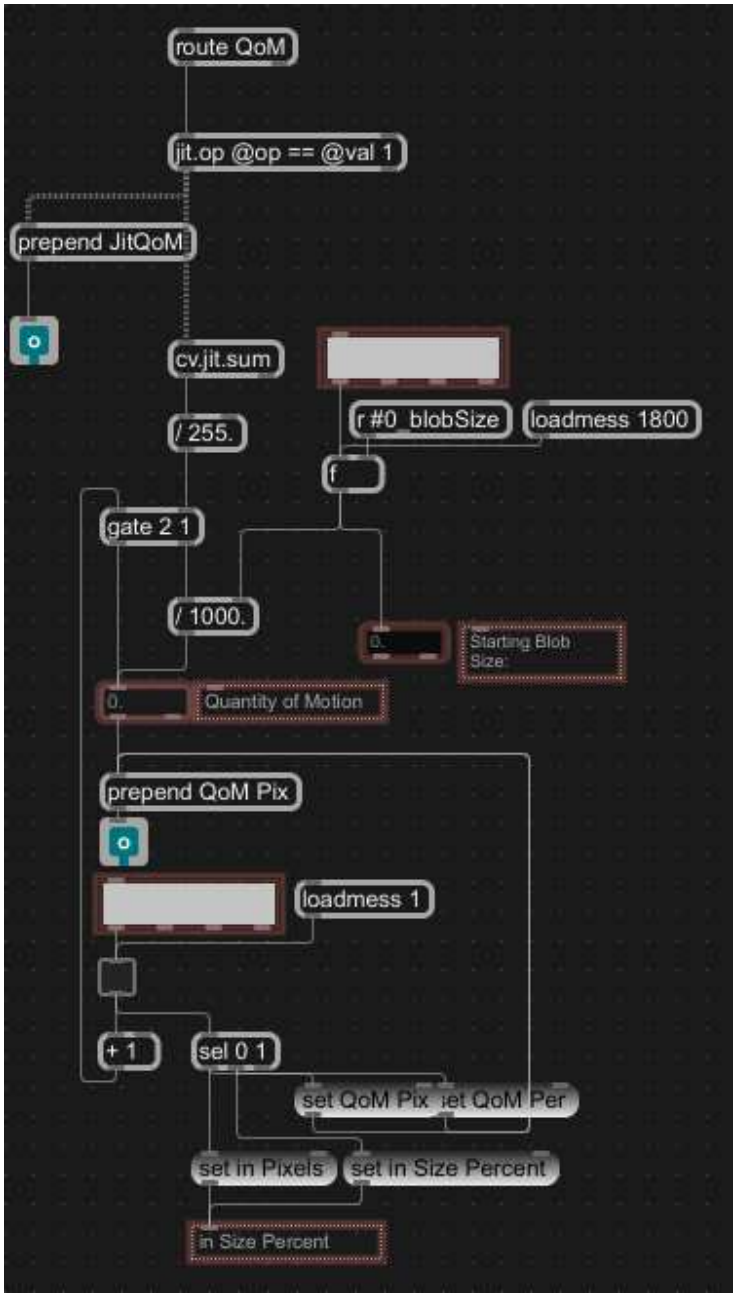



ΕΙΚΟΝΑ 31 ΥΠΟΛΟΓΟΣΜΟΣ QOM

Η απόλυτη διαφορά μεταξύ των δύο καρέ επιστρέφει μόνο τα εικονοστοιχεία που μετακινήθηκαν από κάθε blob (δηλαδή την επιφάνεια των blob που μετακινήθηκε μέσα σε αυτόν τον χρόνο). Το πλήθος αυτών των εικονοστοιχείων είναι η τιμή του quantity of motion.

Στην υλοποίηση αυτής της μέτρησης προστέθηκε ένα βήμα παραπάνω, κατά το οποίο η παράμετρος QoM κανονικοποιείται (normalization) ως προς τις σωματικές διαστάσεις του κάθε μουσικού/εκτελεστή. Συγκεκριμένα, πριν την έναρξη του υπολογισμού των παραμέτρων των blob αποθηκεύεται η επιφάνεια που καλύπτει ο κάθε μουσικός/εκτελεστής στην εικόνα σε πλήθος pixels καθώς αυτοί στέκονται σε όρθια θέση και με τα χέρια κατεβασμένα. Στη συνέχεια, ως "quantity of motion" ορίζεται ο λόγος των κινούμενων pixel, προς αυτό το αρχικό μέγεθος του εκάστοτε μουσικού/εκτελεστή (Εικόνα 32).

Με αυτόν τον τρόπο η παράμετρος αυτή (QoM) θεωρείται πιο αντιπροσωπευτική για κάθε μουσικό/εκτελεστή.



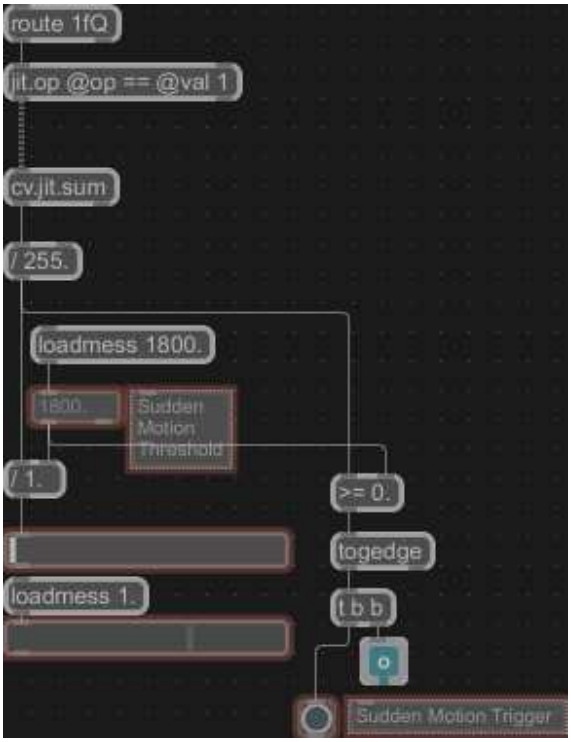

ΕΙΚΟΝΑ 32 QOM ΤΟΙΣ ΕΚΑΤΟ, ΩΣ ΠΡΟΣ ΤΗΝ ΑΡΧΙΚΗ ΤΙΜΗ ΜΕΓΕΘΟΥΣ ΤΟΥ BLOB

**Quantity of motion 1 Frame**

Παράλληλα, και ανεξάρτητα, με το μέγεθος της κίνησης μεταξύ 5 καρέ, υπολογίζεται και το μέγεθος της κίνησης μεταξύ διαδοχικών καρέ. Ο λόγος αυτού του υπολογισμού είναι η προσπάθεια μιας αφηρημένης προσέγγισης στην ανίχνευση της αυθόρμητης κίνησης του performer. Η ονομασία του αποτελέσματος ορίστηκε ως "1fQ" (1 frame Quantity (of Motion)).

Ο τύπος του δείκτη 1fQ δεν είναι κάποια αριθμητική τιμή. Λόγο της φύσης της αυθόρμητης κίνησης, απότομη αυξομείωση του μεγέθους της κίνησης, μικρή διάρκεια, θεωρήθηκε πιο αντιπροσωπευτικό η αποστολή ενός σήματος "trigger event", εντός της max.Το μήνυμα αυτό είναι το "bang".

Το μέρος του υποπρογράμματος που υπολογίζει το 1fQ φαίνεται στο σχήμα 30. Το αριστερό μέρος του υποπρόγραμμα, μπορεί να παραβλεφθεί εφόσον υπάρχει μόνο για οπτική πληροφορία μέσω της οθόνης. Στα δεξιά φαίνεται πως όταν ξεπεραστεί ένα "threshold", στην εικόνα η τιμή αυτή είναι 1800, αποστέλλεται το "bang" από την έξοδο. Η σύγκριση της τιμής της κίνησης με την τιμή "threshold" γίνεται από το αντικείμενο [>= ], όπου αν η αριστερή είσοδος (τιμή κίνησης) είναι μεγαλύτερη από την δεξιά είσοδο (τιμή threshold) τότε στην έξοδο του στέλνει τον αριθμό 1, αλλιώς το 0. Για την αποφυγή πολλαπλής ανίχνευσης της ίδιας κίνησης η έξοδος του [>= ] αποστέλλεται στο αντικείμενο [togedge] το οποίο ανιχνεύει μεταβάσεις σε σχέση με το 0. Δηλαδή, από την αριστερή του έξοδο, η οποία και χρησιμοποιείται στην παρούσα εργασία, στέλνει το μήνυμα "bang" μόνο όταν στην είσοδό του υπάρξει μια μετάβαση από έναν αριθμό μικρότερου ή ίσου του μηδενός, σε έναν αριθμό μεγαλύτερο του μηδενός. Έτσι στην περίπτωση που στην είσοδό του εισέλθουν δύο ή περισσότεροι διαδοχικοί αριθμοί μεγαλύτεροι του μηδενός, ως αποτέλεσμα θα υπάρξει ένα bang, το οποίο θα προκληθεί μόνο από τον πρώτο. Για να αποσταλεί και πάλι το μήνυμα bang, θα πρέπει πρώτα να υπάρξει ένας τουλάχιστον ακέραιος ίσος ή μικρότερος του μηδενός, άρα θα πρέπει η ποσότητα της κίνησης να ελαττωθεί κάτω από την τιμή του threshold.




ΕΙΚΟΝΑ 33 ΥΠΟΛΟΓΙΣΤΟΣ 1FQ

**Mass**

Η μάζα ενός blob, η αλλιώς το μέγεθός του, υπολογίζεται από το σύνολο των pixels που το αποτελούν. Εφόσον σε ένα καρέ, τα pixels ενός blob έχουν τιμή τύπου on/off ή 1/0, αθροίζοντας τα pixels του καρέ παίρνουμε ως αποτέλεσμα το μέγεθος του. Στο υποπρόγραμμα, η διαδικασία αυτή γίνεται με την χρήση του αντικειμένου [cv.jit.sum]. Για των υπολογισμό των τιμών QoM, και 1fQ, συμπεριλαμβάνεται η ίδια τεχνική.

**Bounding rectangle – περιβάλλων ορθογώνιο**

Το Bounding Rectangle, περιβάλλον ορθογώνιο, του κάθε blob, αποτελείται από τις συντεταγμένες για τα σημεία του blob που βρίσκονται στις ακραίες, άνω αριστερή, και κάτω δεξιά,

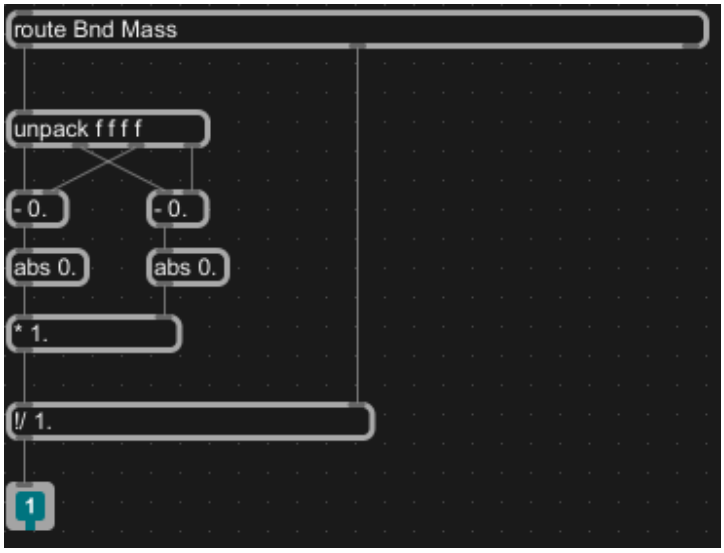
θέσεις του. Στα αριστερά και δεξιά σημεία, χρησιμοποιείται μόνο η x συντεταγμένη, ενώ στα πάνω και κάτω μόνο η y, έτσι μπορούν να συνδυαστούν ως δύο ζεύγη συντεταγμένων x,y όπου αντιστοιχούν στην αριστερά-πάνω γωνία και την δεξιά-κάτω γωνία του περιβάλλοντος ορθογωνίου. Η συντεταγμένες αυτές υπολογίζονται με το αντικείμενο [cv.jit.blobs.bounds]. ο αλγόριθμος που χρησιμοποιεί το αντικείμενο για την εύρεση των συντεταγμένων αναφέρεται στο ΠΑΡΑΡΤΗΜΑ Α.

**Baricenter – Κέντρο Βάρους**

Το κέντρο βάρους αποτελείτε από ένα ζεύγος συντεταγμένων x,y και προσδιορίζει το σημείο του blob στο οποίο "ισορροπεί" και υπολογίζεται με το αντικείμενο [cv.jit.centroids]. Επί της ουσίας οι δύο τιμές που αποτελούν τις συντεταγμένες του κέντρου βάρους υπολογίζονται από την βαρύτητα κάθε γραμμής του blob, δηλαδή το πλήθος των pixel στην γραμμή αυτή που αποτελούν το blob για την x συντεταγμένη, και αντίστοιχα την βαρύτητα κάθε στήλης για την y συντεταγμένη. Ο αλγόριθμος που εκτελείται από το αντικείμενο παρατίθεται στο ΠΑΡΑΡΤΗΜΑ Α.

**Contraction Index – Δείκτης συστολής**

Το contraction index, είναι ένας δείκτης όπου προσπαθεί να εκφράσει τον τρόπο που κάποιος καταλαμβάνει ένα χώρο. Στο υποπρόγραμμα ο δείκτης υπολογίζεται με την βοήθεια του [cv.jit.blobs.bounds]. Εφόσον βρεθούν οι ακραίες συντεταγμένες του blob, στην συνέχεια υπολογίζονται από αυτές, το ύψος και το πλάτος του, και από αυτά υπολογίζεται ο όγκος του περιβάλλοντος ορθογωνίου. Η τιμή του όγκου του blob διαιρείται με την τιμή του όγκου του ορθογωνίου με ελάχιστο αποτέλεσμα να τείνει στο 0, εφόσον είναι αδύνατο το μέγεθος του ορθογωνίου να φτάνει το άπειρο, και μέγιστο, θεωρητικά, 1. Στην εικόνα 34 φαίνεται η υλοποίηση του παραπάνω υπολογισμού, στο περιβάλλον της max.




ΕΙΚΟΝΑ 34 ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΔΙΕΚΤΗ "CONTRACTION", ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΩΝ ΑΚΡΑΙΩΝ ΣΥΝΤΕΤΑΓΜΕΝΩΝ (ΛΙΣΤΑ BND) ΚΑΙ ΟΓΚΟΥ (MASS)

**Directivity Index – Δείκτης κατευθυντικότητας**

Το directivity index είναι ένα δείκτης όπου αντιπροσωπεύει την κατευθυντικότητα του blob. Για το κάθε blob πρέπει να οριστεί ένα σημείο του οποίου η κατεύθυνση της κίνησης θα αντιπροσωπεύει την συνολική κατεύθυνση του blob. Για αυτό, επιλέχθηκε το κέντρο βάρους εφόσον η κίνησή του εξαρτάται και από την συνολική κατεύθυνση του performer αλλά και τη στάση του σώματός του η οποία καθορίζει επίσης την κατεύθυνση του blob.

Ο τρόπος που υπολογίζεται είναι ο εξής:

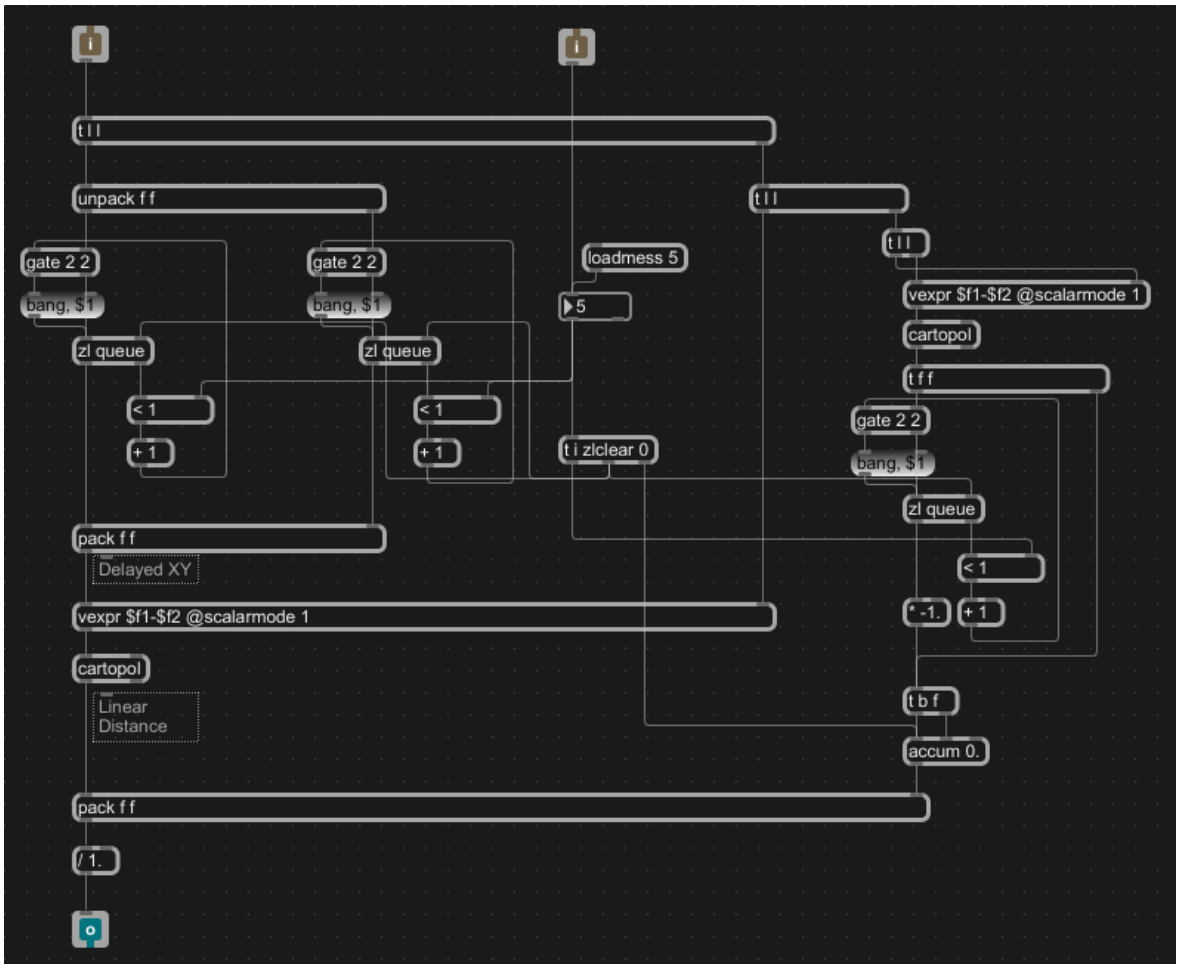
Για κάποια προκαθορισμένη διάρκεια, αριθμό καρέ, υπολογίζονται δύο αριθμοί:

- Η συνολική απόσταση που διένυσε το επιλεγμένο σημείο
- Η απόσταση της ευθείας μεταξύ της πρώτης και της τελευταίας θέσης του σημείου.

Στην συνέχεια βρίσκεται ο λόγος της ευθείας προς την συνολική απόσταση. Θέτοντας ως βάση την συνολική απόσταση ο δείκτης δε μπορεί να ξεπεράσει την μονάδα, αφού δεν μπορεί να βρεθεί πιο μικρή απόσταση από την ευθεία μεταξύ δύο σημείων. Άρα, οι ακραίες τιμές του δείκτη είναι:

- Ως μέγιστο, η μονάδα όπου σηματοδοτεί την πλήρως κατευθυντηκή κίνηση του σημείου.
- Ως ελάχιστο, η μακρινή προσέγγιση του μηδενός, αφού ο δείκτης υπολογίζεται εντός πεπερασμένου χρόνου και η συνολική απόσταση δεν μπορεί να είναι άπειρη, σηματοδοτεί την πλήρως τυχαία και χαοτική κίνηση.

Η υλοποίηση της παραπάνω διαδικασίας εντός της max, φαίνεται στην εικόνα 34.




ΕΙΚΟΝΑ 35 ΑΓΛΟΡΙΘΜΟΣ DIRECTIVITY INDEX – ΔΕΙΚΤΗΣ ΚΑΤΕΥΘΥΝΤΙΚΟΤΗΤΑΣ

Από την είσοδο του υποπρόγραμμα εισέρχεται το ζεύγος συντεταγμένων x,y του κέντρου βάρους του blob και μέσω των δύο [t l l] αποστέλλεται σε τρία σημεία. Στα πρώτα δύο, δεξιά του υποπρόγραμμα, υπολογίζεται η απόσταση των συντεταγμένων μεταξύ διαδοχικών θέσεων, και στο τρίτο στην συνέχεια, τροφοδοτεί τα [accume 0.] και [zl queue] αντικείμενα.

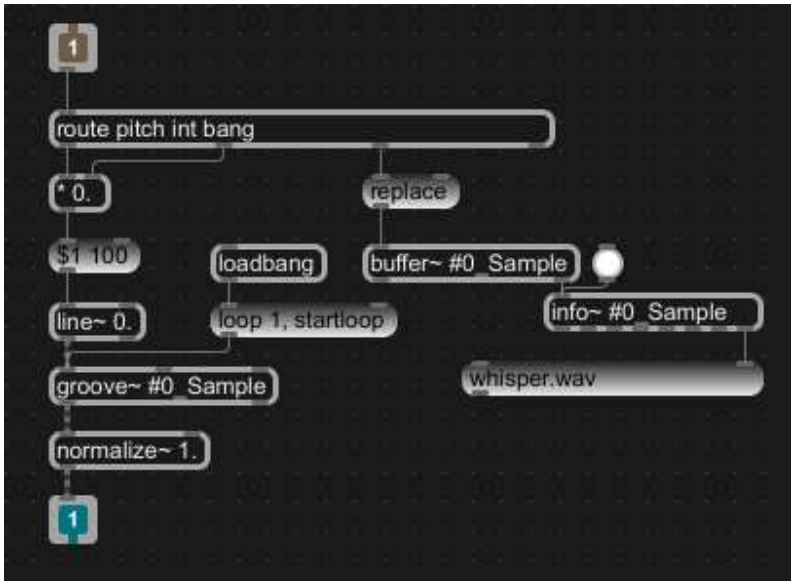
- Το [accume 0.] προσθέτει την τιμή κατευθείαν στην τρέχουσα τιμή της απόστασης.
- Το [zl queue] αποθηκεύει μηνύματα με τον τρόπο FIFO (First In First Out).

Κάθε τιμή που έρχεται στην είσοδό του αποθηκεύεται σε ένα “queue” (ουρά) τιμών. Όταν έρθει το μήνυμα bang, η πρώτη τιμή αποστέλλεται στην έξοδο και διαγράφεται από το queue, ώστε η επόμενη τιμή να είναι πλέον η πρώτη και ούτω καθεξής. Από την δεξιά έξοδο, το [zl queue], με κάθε αλλαγή στο queue, είτε εισχώρηση είτε αποστολή, αναφέρει το τρέχον μέγεθος του. Μόλις το μέγεθος του queue ισοδυναμεί με το πλήθος των καρέ που έχει τεθεί για τον υπολογισμό του δείκτη κατευθυντικότητας, κάθε νέα τιμή που εισέρχεται προηγείται από ένα μήνυμα bang. Με τον τρόπο αυτό δημιουργείται μια σειρά καθυστέρησης στις εισερχόμενες τιμές. Οι τιμές αυτές, αφού ήδη υπάρχουν στο τρέχον μέγεθος της απόστασης, τώρα αφαιρούνται από αυτό. Έτσι το τρέχον μέγεθος της απόστασης αντιπροσωπεύει τα καρέ που θέσαμε για τον υπολογισμό του δείκτη κατευθυντικότητας.

### 3.2.4. “Instrument”

Σε αυτό το υπό-κεφάλαιο θα εξηγήσουμε τις βασικές λειτουργίες του υποπρόγραμμα “Instrument”, και την υλοποίηση των στοιχείων ελέγχου που περιγράφονται στον πίνακα του κεφαλαίου “5.2.1.4 Instrument”.

**Sample Playback**




ΕΙΚΟΝΑ 36 ΥΠΟΠΡΟΓΡΑΜΜΑ "P PLAYER"

Η αναπαραγωγή του επιλεγμένουsample, γίνεται εντός του υποπρόγραμμα “p Player” (εικόνα 36). Το υποπρόγραμμα δέχεται τα εξής μηνύματα:

- Μηνύματα “Bang” μέσω ενός [button].
- Τους ακέραιους 1 ή 0, μέσω του αντικείμενου [toggle].
- Pitch $1, όπου το “$1” αντικαθίσταται από ένα δεκαδικό αριθμό.

Το μήνυμα “bang” προκαλεί την αποστολή του μηνύματος “replace” σε ένα αντικείμενο [buffer~]. Με το μήνυμα αυτό, το αντικείμενο [buffer] ανοίγει ένα “παράθυρο” μέσω του οποίου ο χρήστης επιλέγει το sampletης αρεσκείας του.

Οι ακέραιοι 0 &1, χρησιμοποιούνται ως συντελεστές στην ταχύτητα αναπαραγωγής του sample. Αν ο ακέραιος είναι 0, ο πολλαπλασιασμός του με την τρέχουσα ταχύτητα έχει ως αποτέλεσμα η αναπαραγωγή του ήχου να γίνεται με μηδενική ταχύτητα, άρα να σταματάει. Αντίστοιχα ο πολλαπλασιασμός της ταχύτητας με συντελεστή 1, επαναφέρει τον ρυθμό αναπαραγωγής, στην τρέχουσα τιμή. Όπως είναι φανερό, οι συντελεστές 0 & 1, χρησιμοποιούνται ως off&on, αντίστοιχα, για το υποπρόγραμμα "p Player".

Το μήνυμα "pitch $1" ορίζεται μέσω των κινήσεων του performer, και επηρεάζει την ταχύτητα αναπαραγωγής του ήχου, με σκοπό την τονική μετακίνηση του. Ο τρόπος ορισμού της τιμής "$1" θα εξηγηθεί παρακάτω.

Η αναπαραγωγή του ήχου γίνεται με την χρήση του αντικειμένου [groove~]. Ανοίγοντας το υποπρόγραμμα, στέλνονται αυτόματα στο αντικείμενο [groove~] τα μηνύματα "loop 1" και "startloop". Τα μηνύματα αυτά αρχικοποιούν το αντικείμενο ώστε να κάνει κυκλική αναπαραγωγή του ήχου. Ο πολλαπλασιασμός των προαναφερθέντων ακεραίων αριθμών, με την τιμή του "pitch $1" μετατρέπονται από τιμές ελέγχου σε σήμα audio, μέσω του αντικειμένου [line~], το οποίο αποστέλλεται στην πρώτη είσοδο του [groove~]. Με τον τρόπο αυτό πραγματοποιείται ο έλεγχος της ταχύτητας αναπαραγωγής στο αντικείμενο αυτό, εφόσον η τιμή του σήματος audio στην είσοδο, θέτει το βήμα αναπαραγωγής.

Πριν την έξοδο του ήχου από το υποπρόγραμμα, το σήμα περνάει από το αντικείμενο [normalize~ 1.], το οποίο ελέγχει την στάθμη έντασης του σήματος και το πολλαπλασιάζει με ένα συντελεστή τέτοιο, ώστε να φτάσει την μονάδα. Με τον τρόπο αυτό εξομαλύνονται οι περισσότερες δυναμικές και το συνολικό επίπεδο έντασης προσεγγίζει το ιδανικό για την επεξεργασία του ήχου.

**Εισερχόμενες παράμετροι από το BLOB**

Οι παράμετροι οι οποίοι εισέρχονται στο υποπρόγραμμα "Instrument" είναι οι εξής:

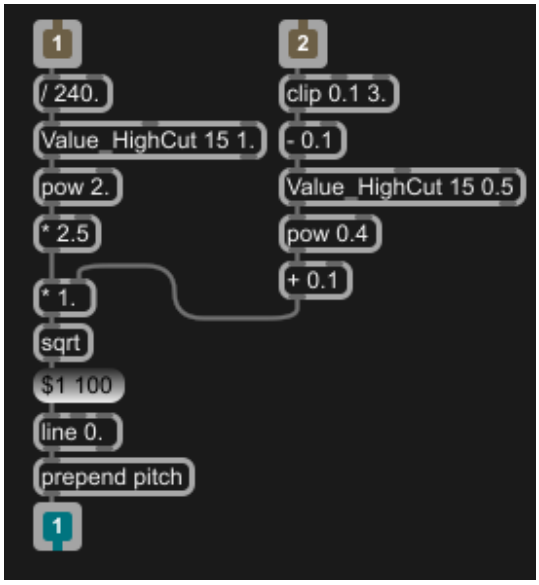
- jitQoM, πίνακας ο οποίος περιέχει τα pixelstης κίνησης.
- Bnd, λίστα τεσσάρων αριθμών, οι οποίοι αποτελούν τις συντεταγμένες σε pixels, του περιβάλλοντος ορθογωνίου.

- Cnt, λίστα δύο αριθμών, οι οποίοι αποτελούν τις συντεταγμένες σε pixels, του κέντρου βάρους.
- QoM, η τιμή ποσότητας της κίνησης, σε ποσοστό επί τοις εκατό προς την αρχική μάζα του blob.
- DI, η τιμή του δείκτη κατευθυντικότητας.
- CI, η τιμή τουContraction Index.
- Οι παράμετροι του ήχου που επηρεάζονται από τα παραπάνω είναι οι εξής.
- Sample pitch
    - Pitch-shiftλόγω ταχύτητας αναπαραγωγής
    - FFT filter
- Almost Simple Amplitude Modulation
- IIR Delay line
    - Χρόνος IIR Delay
    - Συντελεστής ανατροφοδότησης
- Χωροτοποθέτηση ήχου
- Reverb
    - στάθμη dry/wet
    - στάθμες“size”, “decay”και “duration”.

**Pitch – Τονικό ύψος**

Για διαμόρφωση του τονικού ύψους λειτουργούν δύο υποπρογράμματα.

***Ταχύτητα αναπαραγωγής***




ΕΙΚΟΝΑ 37 ΥΠΟΠΡΟΓΡΑΜΜΑ ΕΛΕΓΧΟΥ ΤΑΧΥΤΗΤΑΣ ΑΝΑΠΑΡΑΓΩΓΗΣ

Το πρώτο υπολογίζει την τιμή της παραμέτρου “pitch $1”, που αναφέρθηκε στο προηγούμενο υποκεφάλαιο “Sample Playback”, και ονομάζεται “p QoMnHeighttoPitch”. Από το όνομά του υποπρόγραμμα, προδίδονται οι δύο παράμετροι βάσει των οποίων υπολογίζεται η τιμή “pitch $1”, οι οποίες είναι:

- Quantity of motion (QoM)
- Ύψος σε pixels.

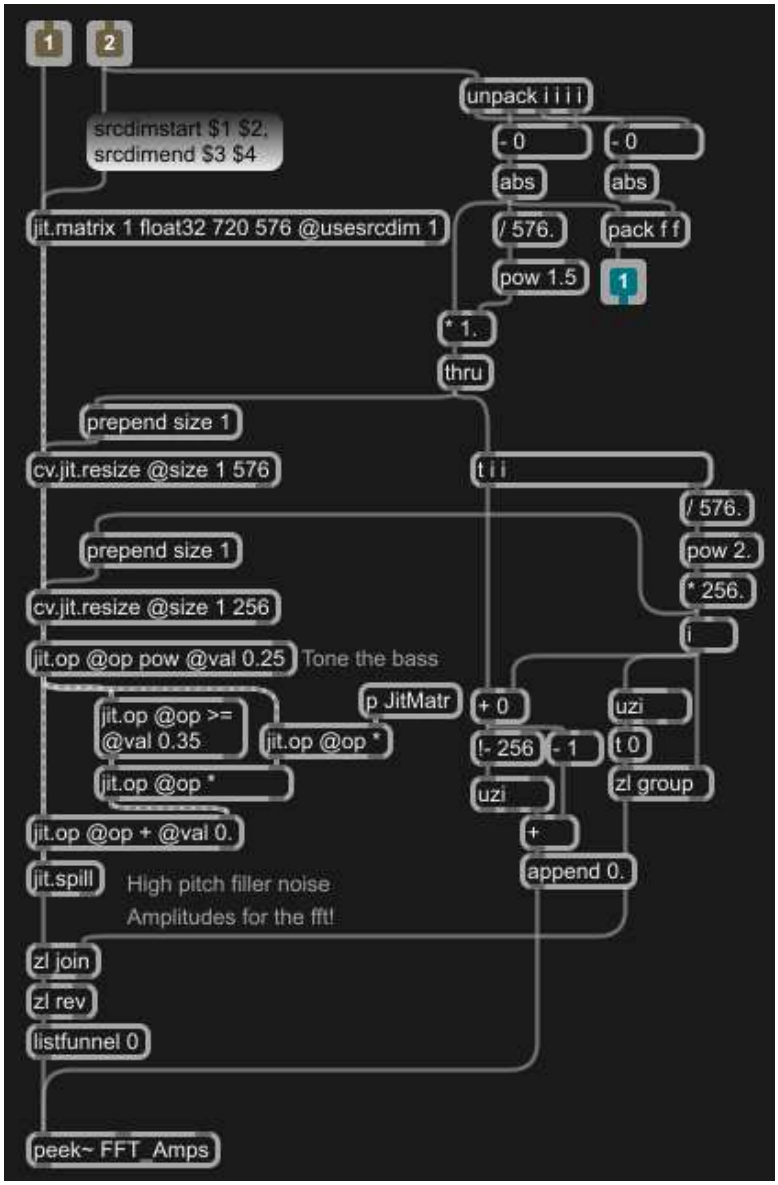
Η τιμή του δείκτη QoM, κυμαίνεται από 0 έως 3. Οι τιμές αυτές βρέθηκαν μέσω χρήσης του αλγορίθμου υπολογισμού της τιμής του δείκτη και παρακολούθησης των αποτελεσμάτων. Για λόγους ασφάλειας οι τιμές εκτός του πεδίου [0,3] απορρίπτονται και αντικαθίστανται από το πλησιέστερο όριο του πεδίου. Έπειτα οι τιμές υψώνονται στη δύναμη “0.4” μεταφέροντας το πεδίο των αριθμών στο [0, 1.55]. Τέλος προστίθεται μια σταθερά , 0.1, ώστε να αποφευχθεί η τιμή 0.

Παράλληλα, η τιμή του ύψους έχει παρατηρηθεί εντός του πεδίου τιμών [30, 150]. Οι τιμές αφορούν ποσότητα σε pixels. Σε στάση αναμονής ο performerπαρατηρήθηκε να έχει ύψος περίπου 100 pixels. Για τον λόγο αυτό η τιμή ύψους διαιρείται με το 100. Οι προβλεπόμενες τιμές κυμαίνονται μεταξύ 0.3 και 1.5. Στην συνέχεια η τιμή υψώνεται στο τετράγωνο και πολλαπλασιάζεται με το 2.5.

Τέλος, τα δύο αποτελέσματα πολλαπλασιάζονται μεταξύ τους.

Οι παραπάνω τιμές είναι πειραματικές, βρέθηκαν μέσω πειραματισμού και επιλέχθηκαν βάσει εικαστικής υποκειμενικής άποψης. Με την παραπάνω συνδυαστική διαδικασία όσο αυξάνονται οι δύο τιμές, τόσο αυξάνεται και η ταχύτητα αναπαραγωγής, και το αντίθετο. Ο σκοπός είναι η αύξηση ή μείωση του τονικού ύψους σε αντιστοιχία με το ύψος και, σε μικρότερο βαθμό, το QoMτου blob.

***FFT φίλτρο***




ΕΙΚΟΝΑ 38 ΥΠΟΠΡΟΓΡΑΜΜΑ "BLOBTOSPECTRALBINAMPS"

Το δεύτερο υποπρόγραμμα, υπεύθυνο για την τονικότητα, είναι το “p BlobToSpectralBinAmps”. Η λειτουργία του υποπρόγραμμα αυτού είναι να μεταφέρει, με βάση το ύψος του performer, την σιλουέτα του από δισδιάστατο πίνακα σε μονοδιάστατο, ο οποίος θα χρησιμοποιηθεί ως πίνακας με πλάτη FFT bins για cross-synthesisμε το σήμα του sample.

Το υποπρόγραμμα χρησιμοποιεί ως ορίσματα εισόδου τα “jitQoM” και “Bnd”.

Η παράμετρος Bnd αποτελείται από μία λίστα τεσσάρων τιμών. Οι τιμές αυτές αντιστοιχούν στις ακραίες συντεταγμένες ενός νοητού κατακόρυφου ορθογωνίου το οποίο περιβάλει το blob. Με την σειρά, οι τιμές αυτές εκφράζουν το ακραία σημεία, πάνω αριστερά και κάτω δεξιά. Εντός του υποπρόγραμμα χρησιμοποιούνται για να υπολογιστεί το ύψος του blob και για να απομονωθεί η εικόνα του περιβάλλοντος ορθογωνίου από τον πίνακα jitQoM.

Ο υπολογισμός του ύψους γίνεται υπολογίζοντας την απόλυτη διαφορά μεταξύ των δύο τιμών “πάνω” και “κάτω”.

Για την απομόνωση του ορθογωνίου του blob από τον πίνακα jitQoM, χρησιμοποιείται το αντικείμενο [jit.matrix]. Στο αντικείμενο τίθεται το αρχικό όρισμα “@usesrcdim 1”. Το όρισμα αυτό, ενημερώνει το αντικείμενο πως παρά τις διαστάσεις του εισερχόμενου πίνακα, στην έξοδό του θα εξάγει ένα μέρος αυτού, με διαστάσεις από τον χρήστη. Οι συντεταγμένες από την λίστα “Bnd” αποστέλλονται επίσης στο μήνυμα “srcdimstart $1 $2, srcdimend $3 $4”, μέσω του οποίου ορίζεται το τμήμα του εισερχόμενου πίνακα το οποίο θα βρεθεί στην έξοδο του [jit.matrix], το οποίο είναι πλέον το περιβάλλον ορθογώνιο του blob.

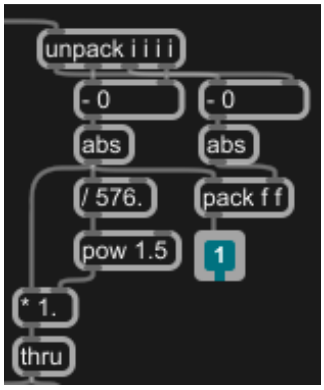
Αφού απομονωθεί το τμήμα αυτό, μετατρέπεται από δυσδιάστατο σε μονοδιάστατο. Η μετατροπή επιτυγχάνεται με το αντικείμενο [cv.jit.resize]. το οποίο για κάθε γραμμή του πίνακα, υπολογίζει το μέσω όρο της. Χρησιμοποιώντας την μέθοδο αυτή, ο νέος μονοδιάστατος πίνακας περιέχει πληροφορία σε κάθε κελί του, η οποία υποδεικνύει το μέγεθος της κίνησης στην αντίστοιχη γραμμή του αρχικού πίνακα. Έπειτα το μέγεθος του πίνακα κλιμακώνεται με ένα δεύτερο αντικείμενο [cv.jit.resize], σε μέγεθος ανάλογο εκείνου που θα είχε, αν ξεκινούσε με διαστάσεις “1 * 256”. Ο λόγος αυτής της μείωσης στον αριθμό των κελιών του πίνακα είναι το μέγεθος του παραθύρου που χρησιμοποιείται στον μετασχηματισμό Fourier, στην περίπτωση 256 δείγματα. Αν το τελικό μέγεθος του πίνακα είναι μικρότερο από 256, τότε συμπληρώνεται με τόσα κελιά, τα οποία περιέχουν την τιμή “0”, ώστε να φτάσει την τιμή 256.

Έπειτα, ο πίνακας αυτός, από μορφή jitter_matrix, με την χρήση του αντικειμένου [jit.spill] στέλνεται ως λίστα 256 τιμών, στο αντικείμενο [listfunnel]. Το αντικείμενο [listfunnel] χωρίζει την λίστα σε μεμονωμένες τιμές και παραθέτει τον αριθμό της θέσης που καταλάμβαναν στην λίστα. Τα πακέτα αυτά, στέλνονται στο αντικείμενο [peek~ FFT_Amps], και καταγράφονται στο αντίστοιχο [buffer~ FFT_Amps.]

Με την παραπάνω διαδικασία, το ύψος του performer, ελέγχει άμεσα τις συχνότητες οι οποίες επιτρέπεται να περάσουν από την συνέλιξη του σήματος με την προαναφερθείσα λίστα. Ένα πρόβλημα που παρατηρήθηκε, προκύπτει από την γραμμική συχνοτική κατάταξη στα fft bins. Από τα 256 bins, λιγότερα από τα πρώτα μισά είναι εκείνα που περιέχουν περισσότερες οκτάβες από τα όλα τα υπόλοιπα. Για τον λόγο αυτό, το ύψος του εκτελεστή επηρεάζει τον δημιουργημένο μονοδιάστατο πίνακα, με εκθετικό τρόπο.

Αρχικά το ύψος διαιρείται με τον αριθμό 576, ο οποίος αντιστοιχεί στο μέγιστο ύψος που μπορεί να έχει κάποιο blob, στην συνέχεια υψώνεται στην δύναμη 1.5 και έπειτα πολλαπλασιάζεται με την αρχική τιμή του ύψους. Η συνάρτηση που χρησιμοποιείται είναι H = h*((h/576)^1.5) , όπου h το αρχικό ύψος του blob, και H το τελικό.

Ο αλγόριθμος φαίνεται στο σχήμα 39.

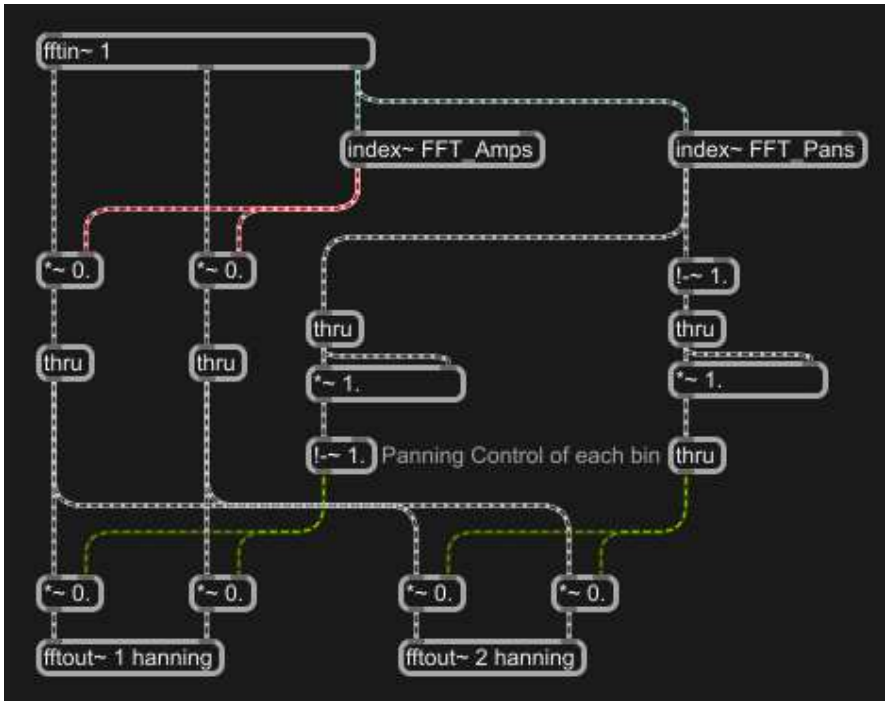



ΕΙΚΟΝΑ 39

Η συνέλιξη μεταξύ του σήματος της εξόδου του “p Player”, και του πίνακα, επιτυγχάνεται μέσα στο υποπρόγραμμα [pfft~ justFFT 256 4].Τα τρία ορίσματα στο αντικείμενο [pfft~ ] ορίζουν:

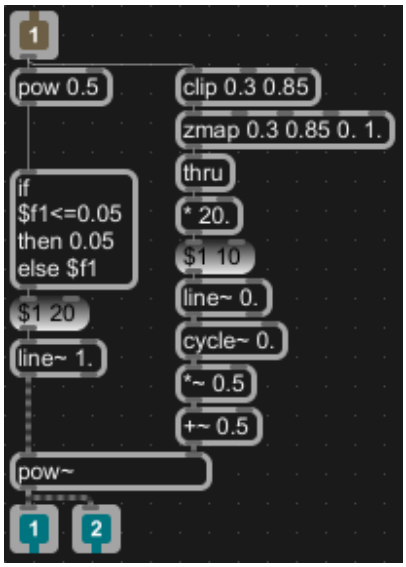
- 1ο όρισμα,“justFFT”, το όνομα του υποπρογράμματος που χρησιμοποιείται μέσα στο αντικείμενο [pfft~]
- 2ο όρισμα, 256, το μέγεθος του fft window.
- 3ο όρισμα, 4, ο αριθμός των αλληλεπικαλυπτόμενων fft windows.

Το υποπρόγραμμα φαίνεται στο παρακάτω σχήμα 40.




ΕΙΚΌΝΑ 40 ΥΠΟΠΡΟΓΡΑΜΜΑ "JUSTFFT"

**Almost Simple Amplitude Modulation**



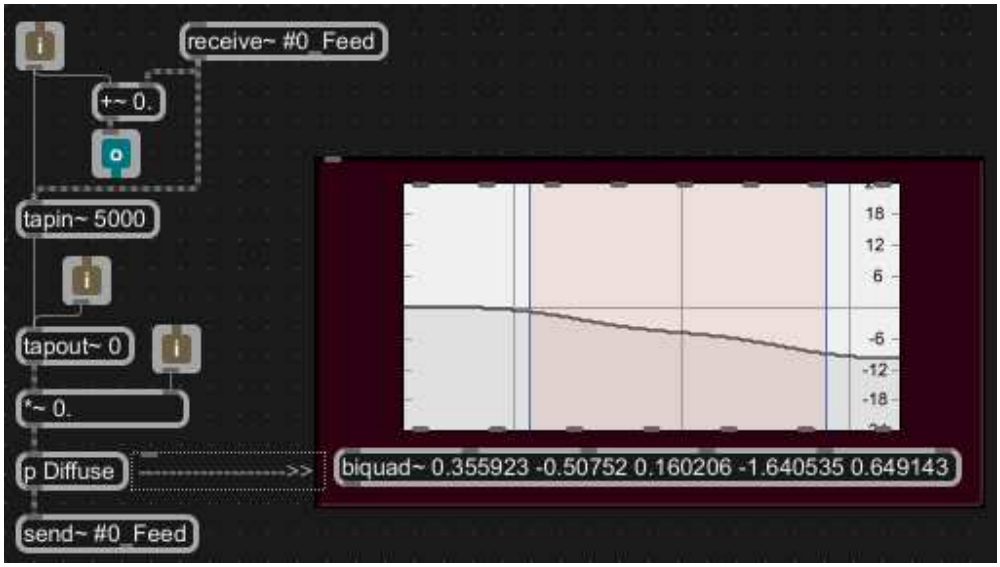

ΕΙΚΌΝΑ 41 ΥΠΟΠΡΟΓΡΑΜΜΑ "ALMOSTSIMPLEAMPLITUDEMODULATION"

Η έξοδος του αντικειμένου [pfft~ justFFT 256 4], πολλαπλασιάζεται με την έξοδο του υποπρόγραμμα “p AlmostAMstyle”. Το υποπρόγραμμα “pAlmostAMstyle” υλοποιεί έναν διαμορφωτή διαμόρφωσης πλάτους. Ο αλγόριθμος ελέγχεται από τον δείκτη “ContractionIndex”, CI. Η τιμή “CI” ελέγχει την συχνότητα μίας ημιτονοειδούς γεννήτριας και την τιμή της δύναμης στην οποία υψώνεται η έξοδος της γεννήτριας. Η συχνότητα της γεννήτριας υπολογίζεται με την συνάρτηση f = CI’ * 20, όπου CI’ η τιμή του δείκτη CI, διαμορφωμένη ώστε οι τιμές εκτός του πεδίου [0.3, 0.85] αντικαθίστανται από την τιμή στο πλησιέστερο όριο του πεδίου. Στην συνέχεια οι τιμές αντιστοιχίζονται στο πεδίο [0,1]. Ο λόγος του περιορισμού των διαθέσιμων τιμών, είναι ο εξής : ο δείκτης, λόγω φυσικών περιορισμών, δεν μπορεί να φτάσει σε τιμές κοντινές των 0 και 1. Επίσης το πεδίο [0.3, 0.85], είναι περισσότερο αντιπροσωπευτικό για την κίνηση του performer.

Η τιμή της δύναμης, στην οποία θα υψωθεί η έξοδος της γεννήτριας, ορίζεται από την τιμή της τετραγωνικής ρίζας του δείκτηCI. Οι τιμές τις τετραγωνικής ρίζας μικρότερες του 0.05, αντικαθίστανται από την τιμή 0.05.

Οι παραπάνω συναρτήσεις και τιμές, επιλέχθηκαν μέσω πειραματισμού και σαστισμού.

**IIR Delay line**




ΕΙΚΟΝΑ 42 ΥΠΟΠΡΟΓΡΑΜΜΑ "IIR-DIFFUSION"

Το υποπρόγραμμα που υλοποιεί την γραμμή καθυστέρησης είναι το “IIR-Diffusion” (σχήμα 42). Στο συγκεκριμένο υποπρόγραμμα, στην αλυσίδα ανατροφοδότησης παρεμβάλλεται ένα χαμηλοπερατό φίλτρο το οποίο φθίνει της υψηλές συχνότητες σε κάθε κύκλο ανατροφοδότησης του σήματος. Στο υποπρόγραμμα, υπάρχουν δύο παράμετροι ελέγχου :

- IIR delay time, η τιμή καθυστέρησης σε milliseconds
- IIR feedback, ο συντελεστής ανατροφοδότησης. Ορίζεται στο πεδίο [0,1).

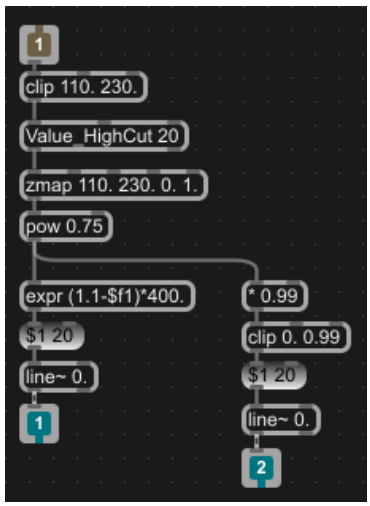
Οι δύο παράμετροι ελέγχονται από το πλάτος του performer, το οποίο υπολογίζεται από την λίστα “Bnd”. Οι τιμές πλάτους του blob του performeréχουν παρατηρηθεί εντός του πεδίου [110, 230]. Οι τιμές ανώτερες του 230, απορρίπτονται και αντικαθίστανται από την τιμή 230. Οι τιμές στην συνέχεια υπόκεινται στην εξίσωση “w’ = (w-110)/120”, όπου w η τιμή του πλάτους και w’ η αντίστοιχη τιμή μεταφερμένη στο πεδίο [0,1]. Έπειτα οι τιμές υψώνονται στην δύναμη “0.75”.

Η παράμετρος “IIR feedback”, υπολογίζεται πολλαπλασιάζοντας το αποτέλεσμα της ύψωσης στην δύναμη 0.75, με τον συντελεστή 0.99. Για λόγους ασφαλείας οι τιμές εκτός του πεδίου [0,0.99], απορρίπτονται.

Η παράμετρος “IIR delay time” υπολογίζεται με την συνάρτηση delay = (1.1–W)*400, όπου Wείναι η έξοδος από την ύψωση στην δύναμη 0.75, και delayη τιμή της καθυστέρησης σε milliseconds.

Στο υποπρόγραμμα αυτό, οι τιμές των δύο παραμέτρων είναι αντιστρόφως ανάλογες. Με την αύξηση του πλάτους του εκτελεστή (performer width), αυξάνεται ο συντελεστής feedback, ενώ μειώνεται ο χρόνος καθυστέρησης, προκαλώντας ένα εφέ “μεταλλικής” χροιάς.

Ο αλγόριθμος υπολογισμού των δύο παραμέτρων φαίνεται στο σχήμα 43.



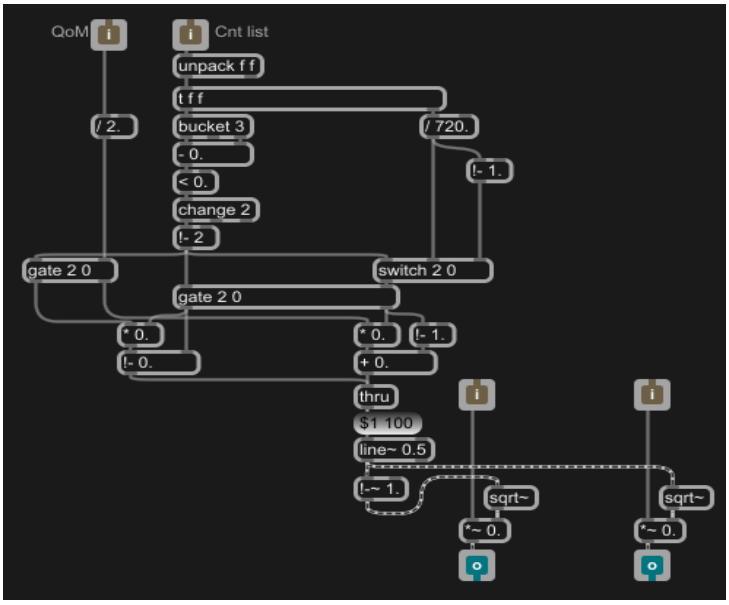

ΕΙΚΟΝΑ 43 ΥΠΟΛΟΓΙΣΜΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ ΥΠΟΠΡΟΓΡΑΜΜΑΤΟΣ "IIR-DIFFUSION"

**Χωροτοποθέτηση ήχου**

Η Χωροτοποθέτηση του ήχου γίνεται στο υποπρόγραμμα “PanningModule”. Οι παράμετροι που χρησιμοποιούνται για τον καθορισμό της στερεοσκοπικής θέσης του ήχου, είναι το QoM και Cnt. Από την λίστα Cnt, υπολογίζεται η κατεύθυνση του performerστο άξονα x. Η τιμή του δείκτη QoM, ορίζει την απόσταση, στον άξονα x, από τον performer, που θα λάβει η θέση του ήχου. Η κατεύθυνση ορίζει αν η απόσταση θα είναι αρνητική ή θετική.

Όταν ο performer κινείται με δεξιά φορά, ο ήχος μπορεί να λάβει όλες τις πιθανές θέσεις δεξιότερα του performer. Όσο μεγαλύτερη τιμή QoM έχει ο performer, τόσο δεξιότερα θα οριστεί η θέση του ήχου. Αντίστοιχα ισχύουν προς την αριστερή κατεύθυνση.

Ο αλγόριθμος φαίνεται στο σχήμα 44.




ΕΙΚΟΝΑ 44 ΥΠΟΠΡΟΓΡΑΜΜΑ "PANNINGMODULE"

**Reverb**

Η ποσότητα του εφέ Reverb, ελέγχεται από την απόσταση του performerμε τον “maestro”. Ο αλγόριθμος και ο έλεγχος του εφέ, υλοποιείται στο υποπρόγραμμα “MAESTROS” και θα εξηγηθούν στο επόμενο κεφάλαιο.

### 3.2.5. “Maestro”

Το υποπρόγραμμα “MAESTROS” χρησιμοποιεί τέσσερεις εισόδους. Οι δύο από τις εισόδους υπάρχουν για τα δύο κανάλια του ήχου από το υποπρόγραμμα “INSTRUMENT”. Τα υπόλοιπα δύο ορίσματα εισόδου είναι η λίστα Bnd, και η τιμή “Distance”.

Η λίστα Bnd περιέχει, όπως και στο υποπρόγραμμα “INSTRUMENT”, της τέσσερεις ακραίες συντεταγμένες του περιβάλλοντος ορθογωνίου του blob, από τις οποίες υπολογίζονται το ύψος και το πλάτος του performer, σε pixels. Η τιμή “Distance” υπολογίζεται στο υποπρόγραμμα “INSTRUMENT”, χρησιμοποιώντας τις λίστες Cnt, του performerκαι του μαέστρου, και αντιστοιχεί στην απόσταση μεταξύ των κέντρων βάρους αυτών, έπειτα αποστέλλεται στο υποπρόγραμμα “MAESTROS”.

Ο χειρισμός του ήχου μέσω του υποπρόγραμμα “MAESTROS” είναι ο εξής :

- Master Volume
- Low Shelving Filter
- Noise
- Reverb
    - Dry / Wet
    - Size

**Master Volume**

Η στάθμη της συνολικής έντασης, ελέγχεται από το ύψος του μαέστρου, όσο παραμένει χαμηλότερο μίας τιμής threshold. Στην περίπτωση κατά την οποία το ύψος του μαέστρου ξεπερνάει την τιμή του threshold, η στάθμη παραμένει στην μέγιστη δυνατή τιμή.

Η τιμή κατωφλίου είναι 300. Η τιμή αυτή παρατηρήθηκε ότι αντιπροσωπεύει το ύψος του μαέστρου σε στάση ανάπαυσης. Υψηλότερες τιμές ύψους, σημαίνουν πως ο μαέστρος έχει εκτεταμένα άκρα, και η περίπτωση αυτή αναλαμβάνεται στο υποπρόγραμμα “p LowShelving ” το οποίο θα αναλυθεί παρακάτω .

Η τιμή ύψους κυμαίνεται στο πεδίο [100,300]. Για να μεταφερθεί σε ένα πεδίο κατάλληλο για τιμή στάθμη έντασης, χρησιμοποιείται η συνάρτηση L = (h-100)/200. Η συνάρτηση μεταφέρει τις τιμές ύψους στο πεδίο [0,1]. Για λόγους ασφαλείας οι τιμές εκτός του πεδίου απορρίπτονται και αντικαθίστανται από την τιμή του πλησιέστερου ορίου.

Εφόσον υπολογιστεί η στάθμη της έντασης , εφαρμόζεται στα δύο κανάλια ήχου από το υποπρόγραμμα "INSTRUMENT".

**Χαμηλοπερατό φίλτρο / Low Shelving Filter**

Στην περίπτωση που το ύψος του μαέστρου ξεπερνάει την τιμή 300, μπαίνει σε λειτουργία το υποπρόγραμμα"p LowShelving" . Οι τιμές ύψους εντός του πεδίου [300, 480]εισέρχονται στην εξίσωση :

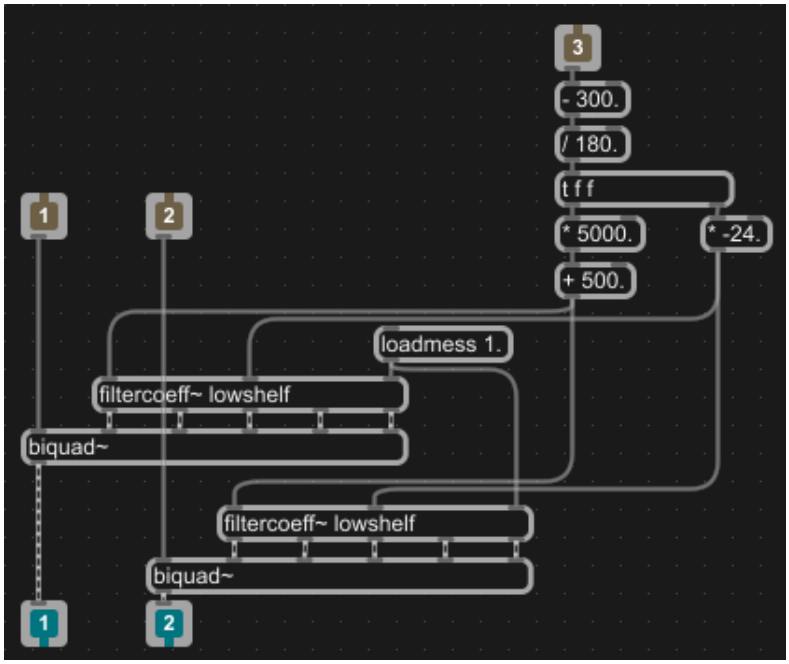
H=(h-300)/180, όπου h το ύψος του μαέστρου και "H" οι τιμές μεταφερμένες στο πεδίο [0, 1].Ο σκοπός του υποπρόγραμμα είναι η μείωση των χαμηλών συχνοτήτων, ανάλογα με την αύξηση του ύψους του μαέστρου. Οι τιμές "H" πολλαπλασιάζονται με το νούμερο -24. Η τιμή αυτή χρησιμοποιείται ως τιμή dB σε ένα low-shelf φίλτρο. Με την διαδικασία αυτή, όταν το ύψος του μαέστρου είναι 300, η τιμή gain είναι 0, ενώ όταν είναι 480, η τιμή gain είναι -24dB.

Η κεντρική συχνότητα του φίλτρου επίσης αλλάζει με την τιμή του ύψους. Με την συνάρτηση f = 1500+((h-300)/180)*5000, όπου h το ύψος του μαέστρου και f η κεντρική συχνότητα του φίλτρου, το πεδίο των τιμών μεταφέρεται εντός των ορίων[1500,6500].

Το φίλτρο υλοποιείται με τα αντικείμενα [biquad~ ] και [filtercoeff~ lowself].Το αντικείμενο [filtercoeff~ ] αρχικοποιείται με το όρισμα "lowshelf". Το όρισμά αυτό, ενημερώνει το αντικείμενο πως θα υπολογίσει συντελεστές lowshelf φίλτρου. Το αντικείμενο έχει τρεις εισόδους :

- Center Frequency
- Gain
- Q / S (slope)

Η τιμή S, παραμένει σταθερή, με τιμή 1. Οι άλλες δύο είσοδοι λαμβάνουν τις υπολογισμένες τιμές , που αναφέρθηκαν παραπάνω, αντίστοιχα. Στο σχήμα __ φαίνεται ο αλγόριθμος υπολογισμού των παραμέτρων του φίλτρου.




ΕΙΚΟΝΑ 45 ΥΠΟΠΡΟΓΡΑΜΜΑ "P LOWSHELF"

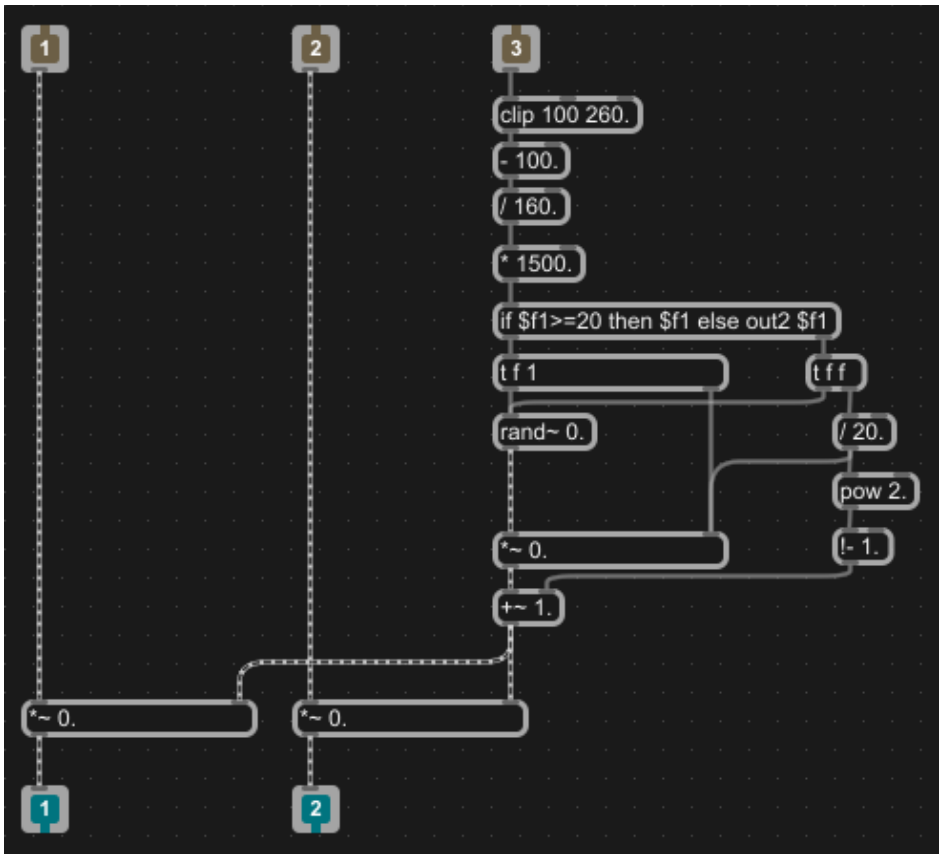
**Noise**

Το υποπρόγραμμα “p Noise” ευθύνεται για την πρόσθεση θορύβου στο εισερχόμενο σήμα. Ο θόρυβος δημιουργείται πολλαπλασιάζοντας το εισερχόμενο σήμα με την έξοδο από τι αντικείμενο [rand~ ].

Το αντικείμενο [rand~ ] δημιουργεί τυχαίες τιμές στο πεδίο [-1, 1], και χρησιμοποιεί έναν αριθμό ως παράμετρο ο οποίος θέτει την συχνότητα με την οποία παράγονται οι τυχαίες τιμές. Για την μετάβαση μεταξύ των παραγόμενων τιμών χρησιμοποιείται η μέθοδος “linear interpolation” - “γραμμική παρεμβολή”. Η μέθοδος αυτή παρεμβάλει μία γραμμική μετάβαση μεταξύ δύο τιμών.

Η τιμή του πλάτους του χρήστη αναφοράς αντιστοιχίζεται στην συχνότητα του αντικειμένου [rand~ ] με την συνάρτηση $f_{rand}$ = W/. Η τιμή της συχνότητας στην συνέχεια επηρεάζει την στάθμη σήματος του αντικειμένου [rand~ ]. Όσο η συχνότητα παραμένει μικρότερη από ακουστό φάσμα, $f_{rand}$ < 20hz, επηρεάζει ανάλογα την στάθμη του σήματος του αντικειμένου [rand~ ], $A_{rand}$. Όταν $f_{rand}$ = 0hz, τότε $A_{rand}$ = 0. Ενώ όταν $f_{rand}$ >= 20hz τότε $A_{rand}$= 1. Για την μετάβαση μεταξύ των δύο ακραίων περιπτώσεων Η αναλογία αυτή δημιουργήθηκε ώστε να μειωθεί το φαινόμενο/εφφε

“tremolo”, το οποίο στηρίζεται στην αισθητή αυξομείωση της στάθμης ενός σήματος. Στην εικόνα 46 φαίνεται το υποπρόγραμμα “p Noise”.




ΕΙΚΟΝΑ 46 ΥΠΟΠΡΟΓΡΑΜΜΑ "P NOISE"

**Reverb**

Το εφέ Reverb, επιτυγχάνεται με την χρήση του αντικειμένου [yarf]. Το αντικείμενο περιέχει έναν έτοιμο αλγόριθμο reverb. Στο αντικείμενο επηρεάζονται και οι τρείς παράμετροι, καθώς και εξωτερικά η στάθμη του εφέ, από την τιμή της απόστασης, “distance”. Όσο αυξάνεται η τιμή “distance” τόσο αυξάνονται οι τιμές των παραμέτρων στο αντικείμενο, και παράλληλα και η στάθμη έντασής του.

Χρησιμοποιείται η συνάρτηση :

R = d/560, όπου d η τιμή “distance” και R ο δείκτης Reverb, ο οποίος χρησιμοποιείται ως συντελεστής στις παραμέτρους του αντικειμένου [yafr], και ως τιμή έντασης. Οι τιμές του δείκτη εκτός του πεδίου [0,1] απορρίπτονται και αντικαθίστανται με το πλησιέστερο όριο του πεδίου.

Τα ορίσματα του αντικειμένου [yafr] είναι :

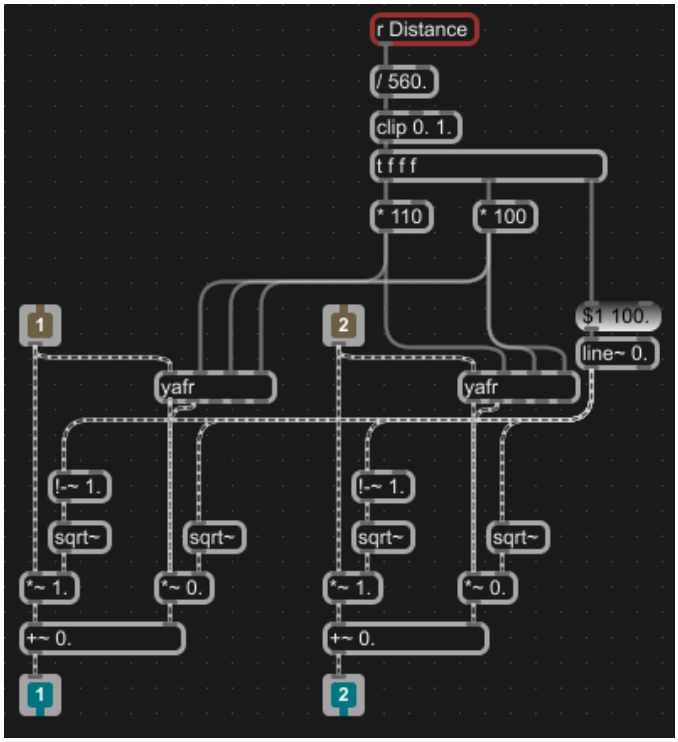
- Reverb time
- Early reflections
- Liveness

Οι τιμές των παραπάνω ορισμάτων ορίζονται στο πεδίο [0, 127]. Με την αύξηση του δείκτη R, μεταβάλλονται ανάλογα και οι παραπάνω παράμετροι, δημιουργώντας ένα βαθύτερο reverb, μεγαλύτερης διάρκειας. Παράλληλα αυξάνεται η στάθμη έντασής του. Η διαδικασία είναι η παρακάτω.

Η τιμή R, χρησιμοποιείται ως συντελεστής της μέγιστης επιθυμητής τιμής, σε κάθε παράμετρο. Οι επιθυμητές τιμές για τις παραμέτρους Reverb time, Early reflections, Liveness, είναι αντίστοιχα 110, 100, 100. Οι τιμές αυτές δημιουργούν ένα βαθύ reverb μεγάλης διάρκειας, όχι όμως σε τέτοιο βαθμό ώστε το αποτέλεσμα να είναι μουντό.

Επειδή τα κανάλια εισόδου ήχου είναι δύο, η ίδια διαδικασία συμβαίνει δύο φορές. Μία για το αριστερό κανάλι ήχου, και μια για το δεξί κανάλι.

Η διαδικασία φαίνεται στην εικόνα 47.




ΕΙΚΟΝΑ 47 ΥΠΟΠΡΟΓΡΑΜΜΑ "P REVERB"

Με την ανάλυση του υποπρογράμματος “MAESTRO” ολοκληρώνεται η παρουσίαση της πειραματικής διαδικασίας η οποία διεξάχθηκε στα πλαίσια της εργασίας. Ακολουθούν τα συμπεράσματα από την εκπόνηση του πειραματικού μέρους και προτάσεις για μελλοντική βελτίωση ή εξέλιξη του μουσικού διαδραστικού συστήματος “BREAD”.

# 4. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ ΒΕΛΤΙΩΣΗΣ ΚΑΙ ΕΞΕΛΙΞΗΣ

Το παρόν κεφάλαιο αφιερώνεται στην αναφορά προβλημάτων που προέκυψαν κατά τη διεξαγωγή της πειραματικής διαδικασίας. Επίσης θα παρουσιαστούν ορισμένες ιδέες, οι οποίες αφορούν στη βελτίωση του συστήματος.

Ένα από τα βασικότερα προβλήματα εμφανίστηκε κατά την θεωρητική σχεδίαση του συστήματος. Το πλήθος των διαθέσιμων παραμέτρων των blob είναι τέτοιο ώστε απαιτείται πολύ οριοθετημένη σχεδίαση για την αποφυγή σύγχυσης. Το γεγονός αυτό, παράλληλα με τη μεγάλη ελευθερία όσον αφορά στην αντιστοίχηση των παραμέτρων αυτών και τον σχεδιασμό της γεννήτριας ήχου, καταλήγει εύκολα στην αυθαίρετη σχεδίαση αλγορίθμων και αντιστοιχήσεων τιμών, οι οποίες είναι κάθε άλλο παρά πρακτικές και λειτουργικές.

Πρακτικές δυσκολίες παρουσιάστηκαν και κατά την υλοποίηση της διαδικασίας αφαίρεσης φόντου. Ο αλγόριθμος που χρησιμοποιήθηκε, αν και το αποτέλεσμά του κρίθηκε επαρκές, χρειάστηκε αρκετά ιδανικές συνθήκες όπως:

- Πολύ καλός διάχυτος, αμετάβλητος φωτισμός
- Αντίθεση των ρούχων των εκτελεστών σε σχέση με το φόντο
- Προσεκτικό “κούρδισμα” παραμέτρων του αλγορίθμου κατά περίπτωση

Επίσης ένα θέμα που προέκυψε και εμπόδισε την λειτουργία του συστήματος με μεγαλύτερη ομάδα ατόμων, είναι η μικρή γωνία λήψης του φακού της κάμερας.

Τα παραπάνω προβλήματα δύναται να επιλυθούν με:

- Χρήση καλύτερου αλγορίθμου αφαίρεσης φόντου, ο οποίος να προβλέπει κινούμενο φόντο.
- Χρήση ευρυγώνιου φακού κάμερας για μεγαλύτερη ελευθερία στο μέγεθος του χώρου, και συνεπώς δυνατότητα μεγαλύτερης ομάδας χρηστών.
- Χρήση κάμερας με χαμηλότερο θερμικό θόρυβο, θα επέφερε καλύτερα αποτελέσματα στην αφαίρεση φόντο με τον παρόν αλγόριθμο.

Οι παραπάνω προτάσεις αφορούν την βελτίωση του αποτελέσματος του παρόντος συστήματος, “Bread”. Παρακάτω ακολουθούν ορισμένες ιδέες βελτίωσης του συστήματος, οι οποίος προσανατολίζονται στην εξέλιξη του συστήματος και την αύξηση των δυνατοτήτων του.

Από τις πρώτες ιδέες για βελτίωση του “Bread”, είναι η χρήση της συσκευής Kinect, ως κάμερα λήψης. Η συσκευή είναι εφοδιασμένη με δύο κάμερες, μία κανονική έγχρωμη κάμερα και μία κάμερα βάθους. Η κάμερα βάθους μπορεί να χρησιμοποιηθεί για την ανάλυση της κίνησης των εκτελεστών σε τρείς διαστάσεις.

Χρησιμοποιώντας την τρισδιάστατη απεικόνιση του εκτελεστή, είναι δυνατή η δημιουργία ενός ανθρωπίνου μοντέλου και από αυτή η αναγνώριση της ακριβούς πόζας στου εκτελεστή στον τρισδιάστατο χώρο. (Kohli & Shotton, 2012). Επίσης, με το Kinect, είναι δυνατή η αναγνώριση κηλίδων οι οποίες επικαλύπτονται όσον αφορά την σιλουέτα τους στους άξονες x και y. Στην δισδιάστατη αναγνώριση, η επικάλυψη θα είχε ως αποτέλεσμα την δημιουργία μίας ενιαίας κηλίδας, ενώ στης τρισδιάστατη αναγνώριση κηλίδων, θα αναγνωρίζονταν ως διαφορετικές.

Μια ιδέα η οποία δεν υλοποιήθηκε λόγω δυσκολιών, είναι η τοποθέτηση μιας κάμερας σε σημείο τέτοιο ώστε να “κοιτάει” την σκηνή των εκτελεστών κάθετα. Στο σενάριο αυτό, οι εκτελεστές φοράνε καπέλα, διαφορετικών χρωμάτων για την σταθερή ταυτοποίηση τους με την τεχνική ανίχνευσης χρώματος, “color tracking”, με σταγονοειδές σχήμα. Από το σχήμα αυτό είναι δυνατός ο υπολογισμός της κατεύθυνσης της “ματιάς” του εκτελεστή. Η πληροφορία αυτή μπορεί να δημιουργήσει πολλές νέες σχέσεις μεταξύ των εκτελεστών. Ένα απλό παράδειγμα χρήσης θα ήταν η συνέλιξη των ήχων που δημιουργούνται από δύο εκτελεστές, όταν εκείνοι κοιτιούνται μεταξύ τους.

Μια ακόμα ιδέα είναι η τοποθέτηση διακριτικών αισθητήρων ίσως σε κάποια αντικείμενα τα οποία θα διαδρούν με τον χρήστη ο οποίος τα χειρίζεται την εκάστοτε στιγμή.

Συνοψίζοντας, ένα μουσικό διαδραστικό σύστημα περιορίζεται από την επιλογή των επιμέρους παραγόντων που το αποτελούν. Έχοντας διαθέσιμα τα εργαλεία ανάπτυξης τα οποία προσφέρουν σχεδόν απόλυτη ελευθερία το κυριότερο πρόβλημα βρίσκεται στην σύλληψη της ιδέας παρά στην υλοποίηση.

# ΠΑΡΑΡΤΗΜΑ Ά – Περιγραφή λειτουργίας των cv.jit αντικειμένων

### Α.1. Επεξεργασία της Εικόνας και Αναγνώριση των Blobs

Κάθε εικόνα του βίντεο, υπόκειται σε επεξεργασία προκειμένου να αναγνωρισθούν τα αντικείμενα-άτομα που υπάρχουν σε αυτή (Blobs), να ταξινομηθούν και να διαπιστωθούν ορισμένα χαρακτηριστικά τους, τα οποία στην συνέχεια θα χρησιμοποιηθούν ως παράμετροι στην γεννήτρια ήχου.

Για τον σκοπό αυτόν χρησιμοποιούνται ορισμένα προγράμματα ανοικτού κώδικα (Open source), από την τοποθεσία http://jmpelletier.com/cvjit/,λ η οποία περιέχει την συλλογή CV.JIT. Η συλλογή αυτή, παρέχει ένα σύνολο εργαλείων προγραμματισμού σε γλώσσα C++ τα οποία είναι προσανατολισμένα στην MAX/MSP, σε σχέση με τις εφαρμογές αναγνώρισης εικόνας μέσω υπολογιστή.

Η συγκεκριμένη συλλογή, παρέχει τους ορισμούς και την αφαιρετική περιγραφή των δεδομένων εικόνας, στοιχεία που είναι απαραίτητα για εργασίες όπως η κατάτμηση εικόνας, η μορφή και η αναγνώριση χειρονομίας, καθώς και η ανίχνευση κινήσεων.

Από το σύνολο των προγραμμάτων που παρέχει η προαναφερόμενη τοποθεσία, στην παρούσα εργασία χρησιμοποιούνται τα αναφερόμενα στον κατωτέρω πίνακα, για την εκτέλεση των εργασιών που αντιστοιχούν στο καθένα.

| **Πρόγραμμα** | **Εργασία που πραγματοποιεί** |
|---|---|
| **cv.jit.label** | Αναγνωρίζει τα Blobs επί της εικόνας, τα οριοθετεί, τα ονομάζει, και υπολογίζει το μέγεθός τους σε pixels |
| **cv.jit.blobs.centroids** | Υπολογίζει την μάζα (πλήθος pixels) και το κέντρο βάρους κάθε Blob |
| **cv.jit.blobs.bounds** | Υπολογίζει τις συντεταγμένες του ιδεατού ορθογωνίου |

| | |
|---|---|
| | παραλληλογράμμου το οποίο περικλείει κάθε blob. |

Στην γενική τους λειτουργία τα προγράμματα αυτά, επεξεργάζονται μία εικόνα (από βίντεο) και εξάγουν από αυτή ορισμένες πληροφορίες για τα αντικείμενα που η εικόνα αυτή περιέχει. Κάθε αντικείμενο επί της εικόνας, αναγνωρίζεται μόνο αν το μέγεθός του είναι μεγαλύτερο από ένα προκαθορισμένο μέγεθος.

Ακολουθεί, μία γενική περιγραφή της λειτουργίας κάθε προγράμματος. Ο ψευδοκώδικας που αναπαριστά την λογική των προγραμμάτων, παρατίθεται στην συνέχεια του Παραρτήματος.

### A.1.1. cv.jit.label (Αναγνώριση των Blobs, Ονοματοδοσία και επισήμανσή τους)

**Input :** Δέχεται πίνακα δύο διαστάσεων τύπου char. Οι διαστάσεις αντιστοιχούν στην ανάλυση του καρέ (π.χ. 800χ600 ή 1024χ768 κ.λπ.). Κάθε θέση του πίνακα αποτελείται από ένα byte (8-bit).

Κάθε byte αντιπροσωπεύει ένα pixel. Η binary τιμή κάθε Byte είναι είτε 00000000 είτε 11111111, ήτοι περιέχει την δεκαδική τιμή 0 ή 255. Αν το δούμε αυτό υπό άλλο πρίσμα, ο πίνακας περιέχει μία ασπρόμαυρη εκδοχή του καρέ σε δύο μόνο χρώματα : Μαύρο (0) και Λευκό (255).

**Επεξεργασία :**

- Εντοπίζει ποια pixel αντιστοιχούν σε κάθε blob. Απορρίπτει αυτά που είναι μικρότερα από ένα συγκεκριμένο μέγεθος.
- Ονομάζει τα αποδεκτά blobs με έναν αύξοντα αριθμό 1,2,…χ, σύμφωνα με την σειρά που τα εντοπίζει από άνω-αριστερά προς κάτω-δεξιά. Μπορεί να αναγνωρίσει μεχρι 255 Blobs

**Output :** Εξάγει τον ίδιο πίνακα που δέχθηκε ως είσοδο, τροποποιημένο ως εξής :

- Όλα τα bytes-pixels που ανήκουν σε ένα Blob και περιέχουν την τιμή 255, αντικαθιστά αυτή την τιμή με τον α/α του blob, όπως τον εντόπισε κατά την επεξεργασία.
- Τα bytes με τιμή 0, δεν υφίστανται καμία μεταβολή.

### A.1.2. cv.jit.blobs.centroids (Υπολογισμός μάζας και κέντρου βάρους)

**Input :** Δέχεται τον πίνακα που έχει παραχθεί από το πρόγραμμα **Labels**.

**Επεξεργασία :**

Υπολογίζει :

- Την μάζα κάθε blob, δηλαδή το πλήθος των pixels από τα οποία αποτελείται.
- Το κέντρο βάρους κάθε blob

**Output :** Εξάγει την ακόλουθη δομή, υπό μορφή πίνακα :

```
Structure
    {
        t_object        ob;
        long            mass[256];
        float           m10[256];
        float           m01[256];
        char            maxval;
    }
```

Όπου :

- **Mass :** Η μάζα του Blob, ήτοι το πλήθος των pixels από τα οποία αποτελείται.
- **m10, m01** : Οι συντεταγμένες που αντιστοιχούν στο κέντρο βάρους του Blob.
- **maxval** : Το πλήθος των blobs που περιέχει ο πίνακας.
-

  Η ταυτότητα κάθε blob (ο α/α που αποδόθηκε από το Labels), δεν περιλαμβάνεται στους εξαγόμενους πίνακες. Η σειρά με την οποία τοποθετούνται τα δεδομένα στους πίνακες εξόδου, αντιστοιχεί στον α/α κάθε blob.

### A.1.3. cv.jit.blobs.bounds (Υπολογισμός ιδεατού παραλληλογράμμου)

**Input :** Δέχεται τον πίνακα που έχει παραχθεί από το πρόγραμμα **Labels**

**Επεξεργασία:** Υπολογίζει τις συντεταγμένες του ιδεατού ορθογωνίου παραλληλογράμμου το οποίο περικλείει κάθε blob.

**Output :** Εξάγει την ακόλουθη δομή, υπό μορφή πίνακα :

```
structure
  {
      t_object          ob;
      long              left[256];
      long              top[256];
      long              right[256];
      long              bottom[256];
      char              maxval;
  }
```

Όπου :

- **left, top** : Οι άνω αριστερά συντεταγμένες του ορθογωνίου.
- **right, bottom** : Οι κάτω αριστερά συντεταγμένες του ορθογωνίου.
- **maxval** : Το πλήθος των blobs που περιέχει ο πίνακας.
-

Η ταυτότητα κάθε blob (ο α/α που αποδόθηκε από την label), δεν περιλαμβάνεται στους εξαγόμενους πίνακες. Η σειρά με την οποία τοποθετούνται τα δεδομένα στους πίνακες εξόδου, αντιστοιχεί στον α/α κάθε blob.

Α.2. Ψευδοκώδικας προγραμμάτων της πλατφόρμας CV.JIT

### Α.2.1. cv.jit.label (Αναγνώριση των Blobs, ονοματοδοσία και επισήμανσή τους)

```
PUBLIC VARIABLES
 char  in[i,j];          //Πίνακας εισόδου (καρέ πρός επεξεργασία)
 char  out[i,j];         //Πίνακας εξόδου (επεξεργασμένο καρέ)
 int      step;                  //Πλήθος χαρακτήρων (pixels) μιάς οριζόντιας
                    γραμμής της εικόνας

 int      width,height;          //Πλάτος και ύψος του καρέ σε pixels
 long     i, j;                  // Ακέραιοι γιά την επεξεργασία των πινάκων
 long     min_blob;                  //Πλήθος Pixels που ορίζουν το μικρότερο
                          αποδεκτό Blob
 int      ndx;              // Ακέραιος γιά την αύξουσα αρίθμηση των Blobs
 long cin                 // Δείκτης στην αρχή του πίνακα εισόδου (in)
 long cout                // Δείκτης στην αρχή του πίνακα εξόδου (out)

 structure                //Δομή γιά την αποθήκευση των πληροφοριών που
 {                   // αφορούν κάθε Blob
  int index[256];         // Index : Α/Α του Blob
  long size[256];         // size : Πλήθος Pixels που απαρτίζουν το blob
 }Blob;

PRODECURE Search_for_blobs()
/*-----------------------------------------------------------------------
/ Διαδικασία εντοπισμού των blobs στον πίνακα εισόδου in
/ Εξετάζουμε κάθε θέση του πίνακα εισόδου σε συνδυασμό με την αντίστοιχη θέση του
/  πίνακα εξόδου. Εάν η εξεταζόμενη θέση του πίνακα εισόδου <> 0 και η αντίστοιχη
/  θέση του πίνακα εξόδου = 0, τότε, έχουμε βρεί την αρχή ενός blob.
/  Στο σημείο αυτό, καλούμε την διαδικασία fill_blob η οποία, «γεμίζει» τις θέσεις
/  του πίνακα εξόδου κατ' αντιστοιχία των θέσεων του πίνακα εισόδου, για το εντοπισμένο
```

```
/  blob και επιστρέφει το πλήθος των pixels που απαρτίζουν το blob.
----------------------------------------------------------------------------*/

/* Αρχικοποίηση πίνακα Blobs;

FOR i=0; i<256; i++
 Blob.index := 0;
 Blob.size := 0;
END_FOR

/* Αρχικοποίηση μεταβλητών */

Cin := in;              // Αποδίδουμε στο cin την διεύθυνση αρχής του in
Cout := out;            // Αποδίδουμε στο cout την διεύθυνση αρχής του out
Ndx:= 0;                // Αρχικοποίηση του μετρητή των blobs
i:=0, j:=0;

FOR i=0; i < height; i++
   FOR j=0; j< width; j++
     IF (cout[j]=0) AND (cin[j] <> 0) THEN   //Βρήκαμε την αρχή ενός νέου blob
            ndx += 1;                  //Αύξηση του μετρητή των blobs κατά 1
       Blob[ndx].index := ndx;          //Ενημέρωση του Α/Α του Blob στον πίνακα
                                        Πληροφοριών για τα εντοπιζόμενα blobs
          Blob[ndx].size = fill_blobs();     //Καλούμε την διαδικασία επεξεργασίας του
                          εντοπισθέντος blob, η οποία επιστρέφει το
                            μέγεθος του blob (το πλήθος των pixels που
                          αποτελούν το blob)
   END_IF
 END_FOR
 Cout += step;                  // Προχωρούμε το cout και το cin κατά μία
 Cin += step;                    οριζόντια γραμμή μπροστά (στην επόμενη row)
END_FOR
```

```
END_Search_for_blobs;
/*-----------------------------------------
/  Τέλος κύριας Διαδικασίας
-----------------------------------------*/
PROCEDURE fill_blob()
/*-------------------------------------------------------------------------
/ Διαδικασία επεξεργασίας του εντοπισμένου Blob.
/ Θεωρούμε ότι κάθε blob, διαχωρίζεται από τα γειτονικά του, με μία τουλάχιστον κενή
/ θέση γύρω από το περίγραμμά του.
/ Στην παρούσα διαδικασία, εξετάζουμε διαδοχικά, τις θέσεις γύρω από την αρχική θέση
/ του blob (δεξιά, αριστερά, πάνω και κάτω) η οποία εντοπίσθηκε από
/ την διαδικασία Search_for_blobs.
/ Για κάθε θέση που δεν είναι κενή, εξετάζουμε επίσης τις θέσεις γύρω από αυτήν.
/ Η διαδικασία τελειώνει όταν δεν υπάρχουν άλλες γειτονικές θέσεις που να ανήκουν στο
ίδιο blob.
/ Η παρούσα διαδικασία, αποτελεί μία τυπική τεχνική χρήσης της στοίβας (stack),
/ στην οποία αποθηκεύουμε την κάθε τρέχουσα θέση προκειμένου να εργασθούμε με την
επόμενη,
/ έως ότου δεν θα υπάρχει άλλη θέση προς αποθήκευση.
/ Στην συνέχεια, επεξεργαζόμαστε τις αποθηκευμένες θέσεις με την αντίστροφη σειρά και
/ η διαδικασία τελειώνει όταν θα έχουμε επεξεργασθεί και την τελευταία θέση.
/ Στο τέλος αυτής της διαδικασίας, οι θέσεις του πίνακα OUT που αντιστοιχούν στις θέσεις
/ του τρέχοντος Blob στον πίνακα IN, θα έχουν γεμίσει με τον Α/Α του τρέχοντος Blob.
/*-------------------------------------------------------------------------
LOCAL VARIABLES
 long row, col;        // Για την επεξεργασία των πινάκων, από την θέση που έχει
                       εντοπισθεί το τρέχον blob
 long Stin             // Το πλήθος των τρεχουσών θέσεων που αποθηκεύονται στο Stack
 long RI, LE, YC       // RI για την επεξεργασία προς τα δεξιά της τρέχουσας θέσης
                       LE για την επεξεργασία προς αριστερά της τρέχουσας θέσης
                       YC για την μετακίνηση στην προηγούμενη ή επόμενη row
 float Direction       // Δείχνει την επόμενη κατεύθυνση (άνω ή κάτω) εξέτασης σε
```

```
                σχέση με την τρέχουσα row
 long i,j          // Για την κίνηση εντός των πινάκων
 float UP := -1;
 float DOWN := 1;     // Flag για την επισήμανση της επόμενης κίνησης εντός των rows




/* Αρχικοποίηση των τοπικών μεταβλητών

row := i;            //Η τρέχουσα θέση αρχής του blob
col := j;
RI := LE := col;
YC := row;

/* Ερευνούμε αν υπάρχουν γειτονικά pixels προς τα δεξιά της τρέχουσας θέσης */

WHILE (RI < width – 1) AND (in[RI +1] <> 0)
  Col++;
  Out[col] = ndx;
  Size++;
END_WHILE

/* Ερευνούμε αν υπάρχουν γειτονικά pixels προς τα αριστερά της τρέχουσας θέσης */

WHILE (LE <> 0) AND (in[LE -1] <> 0)
  LE--;
  Out[LE] = ndx;
  Size++;
END_WHILE

IF YC <> height        // Εάν η row που αρχίσαμε δεν είναι η τελευταία, η επόμενη
```

```
  Direction := UP;          κίνησή μας θα είναι προς τα πάνω (-1 row, αλλοιώς
ELSE                        θα είναι προς τα κάτω (+1 row)
  Direction := DOWN
END_IF

PUSH in Stin(Stin++, YC, Col, Direction)  //Προσθήκη στο stack, της τρέχουσας θέσης

/* Επαναληπτική διαδικασία εξέτασης γειτονικών θέσεων

WHILE Stin <> 0                     // Όσο υπάρχουν αντικείμενα στο stack

 POP from Stin(YC, col, Direction, Stin--)  // Ανάκτηση ενός αντικειμένου από το stack

 IF (YC – direction < height) AND (YC - direction >= 0)
   Cout += step;
   FOR i=LE; i< RI +1; i++              // Εξετάζουμε από αριστερά προς τα δεξιά
    IF cin[i] <> 0 AND cout[i] = 0      // Αν το pixel είναι ON, έχουμε έναν κόμβο
      j := i;                   // θα επεξεργασθούμε τα γειτονικά Pixels
      cout[j] := val;
      size++
      WHILE j <> 0 AND cin[j-1] <> 0    // Όσο δεν είμαστε στο αριστερό άκρο της row
        Cout[j-1] = val;            // και το αριστερότερο Pixel είναι ON,
        J++                     // περιλαμβάνουμε αυτό το pixel στο blob
        Size++                    // και πάμε μία θέση αριστερότερα
      END_WHILE
      WHILE (I < width -1) AND (cin[i+1] <> 0) // Κάνουμε το ίδιο και προς τα δεξιά
       i++;
       cout[i] := val;
       size++;
    END_WHILE

    PUSH in Stin(Stin++, YC, Col, Direction);   // Προσθήκη τρέχουσας θέσης στο stack
```

```
    END_IF
   END_FOR
  END_IF

  Cin = step + YC + Direction;    // Τοποθετούμε τους δείκτες στην αρχή της
                                  λογικώς επόμενης row, δηλαδή
  Cout = step + YC + direction;     // στην προηγούμενη row, αν η κατεύθυνση είναι UP,ή
στην
                                //επόμενη row, αν η κατεύθυνση είναι DOWN
  FOR i=LE; i < RI+1; i++         // Ερευνάμε από την τρέχουσα αριστερή άκρη μέχρι την
   IF (cin[i] <> 0 AND cout[i] = 0)     // προηγούμενη αριστερή άκρη. Αν το pixel είναι
    j:=i;                        // ON, έχουμε έναν νέο κόμβο.
    cout[i] := ndx;
    size++
    WHILE (j<>0 AND cout[j-1] <> 0)   // Προχωράμε αριστερά μαρκάροντας τα γειτονικά
     j--;                       // pixels που είναι ON
     cout[j] := val;
     size++;
    END_WHILE
    WHILE (i < width AND cout[i+1] <> 0) // Προχωράμε δεξιά, μαρκάροντας τα γειτονικά
     I++;                         // pixels που είναι ON
     cout[i] := val;
     size++;
    END_WHILE
    PUSH in Stin(Stin++, YC + Direction, j, i++); // Στο stack, τις διαμορφωμένες τιμές
   END_IF                                  της row και col
  END_FOR
  FOR i=RI-1; i < RI+1; i++          // Εδώ, αρχίζουμε από την προηγούμενη δεξιά άκρη και
  IF cin[i] <> 0 AND cout[i] = 0    // πάμε προς την τρέχουσα δεξιά άκρη. Αν το Pixel
   j:=i;                        // είναι ON, έχουμε έναν νέο κόμβο.
   cout:= val;
   size++;
```

```
   WHILE (j <> 0 AND cin[j-1] <> 0)  // Κάνουμε την ίδια διαδικασία όπως και πρίν
    j++;
    cout[j] := val;
    size++
   END_WHILE
   WHILE (i < width AND cin[i+1] <> 0)
    i++;
    cout[i] := val;
    size++
   END_WHILE
  PUSH in Stin(Stin++, YC+direction, j, i++);
  END_IF
 END_FOR
END_WHILE
END fill_blog   // Τέλος της διαδικασίας συμπλήρωσης του τρέχοντος blob
```

### A.2.2. cv.jit.blobs.centroids (Υπολογισμός μάζας και κέντρου βάρους)

```
PUBLIC VARIABLES
 char  in[i,j];          //Πίνακας εισόδου (καρέ πρός επεξεργασία)
 int      step;                  //Πλήθος χαρακτήρων (pixels) μιάς οριζόντιας
                  γραμμής της εικόνας
 int      width,height;          //Πλάτος και ύψος του καρέ σε pixels
 long    i, j;                  // Ακέραιοι γιά την επεξεργασία των πινάκων
 long cin              // Δείκτης στην αρχή του πίνακα εισόδου (in)
 int cbl              // Δείκτης στο τρέχον blob

 Structure
  {
t_object          ob;
long              mass[256];
```

```
float           m10[256];
float           m01[256];
char            maxval;
    } blob

/* Αρχικοποίηση μεταβλητών */

Cin := in;              // Αποδίδουμε στο cin την διεύθυνση αρχής του in
Cout := out;            // Αποδίδουμε στο cout την διεύθυνση αρχής του out
i:=0, j:=0;

Blob.maxval := 256;
FOR i=0; i< blob.maxval; i++
  Blob[i].mass := 0;
  Blob[i].m10 := 0;
  Blob[i].m01 := 0;
END_FOR

PRODECURE centroids_calculate()
/*-------------------------------------------------------------------
/ Εξετάζουμε κάθε θέση του πίνακα εισόδου in. Οι θέσεις του IN, περιέχουν
/ τον α/α του blob, του οποίου τα pixels καταλαμβάνουν αυτές τις θέσεις.
/ Χρησιμοποιούμε τον α/α, για την αποθήκευση των στοιχείων κάθε Blob
/ στον πίνακα της structure Blob.
-------------------------------------------------------------------*/
FOR j:=0 ; j < height; j++
  Cin := in + j *step;      // Τοποθετούμε τον δείκτη στην αρχή της τρέχουσας
                            Γραμμής του πίνακα εισόδου IN
  FOR i:=0; i < width; i++
    Cbl := in[i];         // Το cbl περιέχει τώρα τον α/α του τρέχοντος blob

   IF cbl > 0
```

```
    Blob[cbl].mass++;       // Προσθέτουμε 1 στο πλήθος των pixels από τα
                            Οποία αποτελείται το τρέχον Blob
    Blob[cbl].m10+=i;       // Προσθέτουμε την τιμή της τρέχουσας col
    Blob[cbl].m01+=j;       // Προσθέτουμε την τιμή της τρέχουσας row
  END_IF
END_FOR
END_FOR

/* Υπολογισμός των συντεταγμένων του κέντρου βάρους κάθε blob

FOR i:=0; i < blob.maxval; i++
        Blob[i].m10 := Blob[i].m10 / Blob[i].mass;
        Blob[i].m01 := Blob[i].m01 / Blob[i].mass;
END_FOR

END_PROCEDURE centroids_calculate
```

### A.2.3. cv.jit.blobs.bounds (Υπολογισμός ιδεατού παραλληλογράμμου)

```
PUBLIC VARIABLES
 char  in[i,j];         //Πίνακας εισόδου (καρέ πρός επεξεργασία)
 int     step;                  //Πλήθος χαρακτήρων (pixels) μιάς οριζόντιας
                        γραμμής της εικόνας
 int     width,height;          //Πλάτος και ύψος του καρέ σε pixels
 long    i, j;                  // Ακέραιοι γιά την επεξεργασία των πινάκων
 long cin               // Δείκτης στην αρχή του πίνακα εισόδου (in)
```

```
 int cbl              // Δείκτης στο τρέχον blob
 structure
  {
     t_object      ob;
long               left[256];
long               top[256];
long               right[256];
long               bottom[256];
char               maxval;
} blob

/* Αρχικοποίηση μεταβλητών */

Cin := in;              // Αποδίδουμε στο cin την διεύθυνση αρχής του in
Cout := out;            // Αποδίδουμε στο cout την διεύθυνση αρχής του out
i:=0, j:=0;

Blob.maxval := 256;

FOR i=0; i< blob.maxval; i++
  Blob[i].left := 0x7FFFFFFF;      // Αρχικοποιούμε με την μέγιστη τιμή τις
  Blob[i].top := 0x7FFFFFF;        // άνω αριστερά συντεταγμένες
  Blob[i].right := 0;             //Αρχικοποιούμε με την κατώτατη τιμή
  Blob[i].bottom := 0;            // τις κάτω δεξιά συντεταγμένες
END_FOR


PRODECURE bounds_calculate()
/*-----------------------------------------------------------------
/ Εξετάζουμε κάθε θέση του πίνακα εισόδου in. Οι θέσεις του IN, περιέχουν
/ τον α/α του blob, του οποίου τα pixels καταλαμβάνουν αυτές τις θέσεις.
/ Για κάθε θέση η οποία περιέχει ενεργό blob (α/α > 0), αποθηκεύουμε τις
```

```
/ συντεταγμένες στις αντίστοιχες μεταβλητές της structure blob
-------------------------------------------------------------------------*/
FOR j:=0 ; j < height; j++
  Cin := in + j *step;      // Τοποθετούμε τον δείκτη στην αρχή της τρέχουσας
                            Γραμμής του πίνακα εισόδου IN
  FOR i:=0; i < width; i++
    Cbl := in[i];         // Το cbl περιέχει τώρα τον α/α του τρέχοντος blob

   IF cbl > 0
     IF i < blob[cbl].left
          blob[cbl].left := i
     END_IF
     IF i > blob[cbl].right
          blob[cbl].right := i
     END_IF
     IF j < blob[cbl].top
          blob[cbl].top := j
     END_IF
     IF j > blob[cbl].bottom
          blob[cbl].bottom := j
     END_IF
   END_IF
END_FOR
END_FOR

END_PROCEDURE bounds_calculate
```

# 5. ΒΙΒΛΙΟΓΡΑΦΙΑ


*Ears : Interface* . (n.d.). Retrieved 09 29, 2012, from Ears ElectroAcoustic Resourse Site: http://www.ears.dmu.ac.uk/spip.php?rubrique274

Camurri, A., Mazzarino, B., & Volpe, G. (2003). *Analysis of Expressive Gesture: The EyesWeb Expressive Gesture Processing Library.* Genova, Italy.

Castagné, N., & Cadoz, C. (2005 ). A Goals-Based Review of Physical Modelling . *International Computer Music Conference ICMC05* , (p. 2 ). Barcelona, Spain .

Cerf, V. G., & Kahn, R. E. (1974). A Protocol for Packet Network Intercommunication. *IEEE Trans on Comms* .

Cook, J. D. (2009 , 8 24 ). *Three algorithms for converting color to grayscale* . Retrieved 10 14 , 2012 , from The Endeavour: http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

Cycling74. (n.d.). *Jit.rgb2luma* . Retrieved 10 14 , 2012 , from http://cycling74.com/docs/max5/refpages/jit-ref/jit.rgb2luma.html

Dodge, C., & Jerse, T. A. (1997 ). *Computer Music - Synthesis Composition and Performance .* Macmillan Library Reference .

Drummond, J. (2009 ). Understanding Interactive Systems . *Organised Sound , 14* (2), 130-131 .

*Ears : Interactive Instruments* . (n.d.). Retrieved 9 26 , 2012 , from Ears : ElectroAcoustic Resource Site: http://www.ears.dmu.ac.uk/spip.php?rubrique40

*EoWave - interfaces* . (n.d.). Retrieved 9 28 , 2012 , from EoWave - interactive sensor tools: http://www.eowave.com/shop/products.php?prod=918

Eyesweb. (n.d.). Examples - Simple Background Subtraction .

Fraietta, A. (2008 ). Open Sound Control: Constraints and Limitations . *NIME .*

Hershey. (2009 ). An Overview of Multimodal Interaction Techniques and Applications . In *Human Computer Interaction : Concept, Methodologies, Tools and Applications* (p. Chapter 1.8 ). New York .

Hewett, Baecker, Card, Carey, Gasen, Mantei, et al. (1996 ). *2. Definition and Overview of Human-Computer Interaction .* Retrieved 9 27 , 2012 , from ACM SIGCHI Curricula for Human-Computer Interaction : http://old.sigchi.org/cdg/cdg2.html

*History of MIDI* . (n.d.). Retrieved 9 29 , 2012 , from MIDI Manufacturers Association - The official source of information about MIDI: http://www.midi.org/aboutmidi/tut_history.php

Hunt, A., & Hermann, T. (2004 ). THE IMPORTANCE OF INTERACTION IN SONIFICATION . *ICAD 04-Tenth Meeting of the International Conference on Auditory Display* , (p. 4 ). Sydney, Australia .

Hunt, A., & Kirk, R. (2000 ). Mapping Strategies for Musical Performance .

*I-CubeX : Digitizers* . (n.d.). Retrieved 9 28 , 2012 , from I-CubeX Online Store : Motion Sensors for Digital Media Control: I-CubeX Online Store : Motion Sensors for Digital Media Control

*I-CubeX : Sensors* . (n.d.). Retrieved 9 28 , 2012 , from I-CubeX Online Store : Motion Sensors for Digital Media Control: http://infusionsystems.com/catalog/index.php/cPath/24

Karplus, K., & Strong, A. (1983 ). Digital Synthesis of Plucked-String and Drum Timbres . *Computer Music Journal , 7* (2), 43-55 .

Kester, W., & Bryant, J. (2003 ). SAMPLED DATA SYSTEMS . In W. Kester, *MIXED-SIGNAL AND DSP* (p. Chapter 2.1 ). Newnes .

Kohli, P., & Shotton, J. (2012). Key Developments in Human Pose Estimation for Kinect.

MathWorks. (n.d.). *Convert RGB image or colormap to grayscale - MATLAB* . Retrieved 10 14 , 2012 , from http://www.mathworks.com/help/images/ref/rgb2gray.html

Merrill, D. (2008 ). *David Merrill* . Retrieved 9 29 , 2012 , from MIT Media Laboratory: http://alumni.media.mit.edu/~dmerrill/siftables.html

Merrill, D., Kalanithi, J., & Maes, P. *Siftables: Towards Sensor Network User Interfaces .* Cambridge, MA : MIT Media Laboratory .

Miranda, E. R., & Wanderlay, M. M. (2006 ). Control and Interaction Beyond the Keyboard . *New Digital Musical Instruments* .

National_Instruments. (2003). *IMAQ Vision Concepts Manual.*

*OSC* . (n.d.). Retrieved 9 29 , 2012 , from opensoundcontrol.org: http://opensoundcontrol.org/

Pelletier, J.-M. (n.d.). *Computer Vision for Jitter* . Retrieved 10 5 , 2012 , from Computer Vision for Jitter: http://jmpelletier.com/cvjit/

Piccardi, M. (2004 ). Background subtraction techniques: a review . *International Conference on Systems, Man and Cybernetics* .

Postel, J. (1980 , August 28 ). *RCF768 .* Retrieved 9 29 , 2012 , from The Internet Engineering Task Force : http://www.ietf.org/rfc/rfc768.txt

Postel, J. (1981 , 9 ). *RCF793 .* Retrieved 9 29 , 2012 , from The Internet Engineering Task Force : http://www.ietf.org/rfc/rfc793.txt

*Reactable* . (2009 ). Retrieved 9 29 , 2012 , from http://www.reactable.com/

Reed, D. P., & Postel, J. (1978, January 21). User Datagram Protocol.

Ross, J. C. (2011 ). Aquiring Images . In J. C. Ross, *The Image Prossesing Handbook* (pp. 6-20 ). Raleigh, North Carolina : CRC Press .

Rovan, J. B., Wanderley, M. M., Dubnov, S., & Depalle, P. (1997 ). Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance . *KANSEI - The Technology of Emotion .* ITS Berkeley publications .

Rowe, R. (1993 ). Machine Lerning and Composing . *Interactive Music Systems* .

Russ, M. (1996 ). *Sound Synthesis and Sampling .* Burlington, MA : Focal Press .

Taylor, S. A. (1998 ). *CCD and CMOS Imaging Array Technologies:Technology Review .* Xerox Limited.

*The Technology of MIDI* . (n.d.). Retrieved 9 29 , 2012 , from MIDI Manufacturers Association - The official source of information about MIDI: http://www.midi.org/aboutmidi/tut_techomidi.php

Weinberg, G. (n.d.). *Beatbugs* . Retrieved 9 29 , 2012 , from MIT Media Lab: http://web.media.mit.edu/~roberto/beatbugs2.html

Weinberg, G. (2003). *Interconnected Musical Networks - Bringing Expression and Thoughtfulness to Collaborative Group Playing.* Massachusetts.

Wilker, T. (1998 ). Techniques and Ideas Using Max . *Composing Interactive Music* .